Part I

Overview

Version 4.6.1
I Overview – Contents

Preface \hspace{1cm} 4

Typographical Conventions \hspace{1cm} 6

Contents of All MpCCI Manuals \hspace{1cm} 26
Preface

"MpCCI Coupling-Environment" is the standard for simulation code coupling.
In this manual MpCCI will be used as abbreviation for "MpCCI Coupling-Environment".
MpCCI has been developed at the Fraunhofer Institute SCAI in order to provide an application independent interface for the coupling of different simulation codes.

Codes Supported by MpCCI

MpCCI enables a direct communication between the coupled codes by providing adapters for a growing number of commercial codes. These code adapters make use of the already existing application programming interfaces (APIs) of the simulation tools. This technique allows for an easy installation of MpCCI at the end user's site without changing the standard installation of the simulation codes.
A list of currently supported codes is given in the Release Notes.

Internal Architecture

The MpCCI environment consists of several components:
- MpCCI Code Adapter allow to adapt MpCCI to commercial codes through their standard code APIs without any changes in the source of the simulation code.
- The MpCCI Graphical User Interface provides a comfortable way to define the coupling setup and to start the simulation - independent of the codes involved in the coupled application.
- The MpCCI Coupling Server is the "heart" of the MpCCI system. Environment handling, communication between the codes, neighborhood computation and interpolation are part of this kernel.

Standardized Quantities

One major advantage of having compatible code adapters for all codes supported by MpCCI is the standardization of coupling parameters and procedures independent from the used code pairing. MpCCI provides unified quantity definitions for
- Global quantities: time, iteration, residuals
- Mass source and sink: production species
- Momentum sources: e.g. Lorentz forces
- Energy sources: e.g. joule heat
- Material properties: e.g. electrical conductivity
- Boundary condition values: e.g. temperature or pressure
- Boundary condition gradients: e.g. heat flux density
- Grid data: nodal positions or displacements
- And chemical components: e.g. for reaction kinetics

MpCCI Manuals

The MpCCI documentation is split up into several manuals which are also called parts. Each part aims at a special kind of readers.
Release Notes The Release Notes contain information on changes versus prior versions of MpCCI. They are thus interesting for users, who have some experience with earlier versions of MpCCI.

Installation Guide The Installation Guide describes how to install MpCCI. It also contains information about the licensing.

Getting Started is intended for new users of MpCCI. The most important features of MpCCI are described by following a typical setup of a coupled simulation.

User Manual The User Manual contains a complete overview of the functions and features of MpCCI. This includes information on code coupling, command line options and functions of the MpCCI GUI.

Codes Manual The Codes Manual contains code-specific information. For each code which can be coupled with MpCCI a section is included.

Tutorial is a collection of examples which are explained in detail.

Programmers Guide The Programmers Guide is intended for users who want to write their own code adapters.

How To is a collection of best practice cases.

FSIMapper The MpCCI FSIMapper Guide is intended for users who want to transfer quantity values from the CFD to the FEM mesh as one-way file based coupling.
Typographical Conventions

This manual adheres to a set of typographical conventions so that you can recognize actions and items. The following list illustrates each of the conventions:

- Text you enter from the keyboard or outputs is written in a typewriter font and surrounded by a gray box, e.g.: `mpcci gui`
- Filenames are enclosed in quotation marks, "example.txt". All paths are given with a slash (/) as directory separator, e.g. "mpcci/doc/pdf". On Windows systems this must be replaced by a backslash (\).
- Meta variables represent values and are enclosed in angle brackets as in `<MpCCI home>`. They can appear everywhere and always should be replaced by appropriate values.
- Environment variables are always written in uppercase typewriter letters, like `VARIABLE`.
- Buttons in the MpCCI GUI look like buttons, e.g. `Next`.
- Entries of the menu have a colored background, sub-menus are separated by an arrow. E.g. `File → Open Project` means the submenu `Open Project` of the `File` menu.
- Other options which can be selected are written like `Option`.
- Names of software are written in a sans-serif font, like `MpCCI`.
- Links can be clicked directly in the PDF version of the manual and are marked blue there, this applies to links within the manual like `IV-2 Setting up a Coupled Simulation` or to web pages `www.mpcci.de`. 
# Contents of All MpCCI Manuals

## I Overview

- Preface  
- Typographical Conventions  
- Contents of All MpCCI Manuals

## II Release Notes

1. **Introduction MpCCI 4.6**  
2. **Changes and New Features in MpCCI 4.6**  
   - 2.1 MpCCI 4.6.1-1  
   - 2.1.1 MpCCI FSIMapper  
   - 2.1.2 Bug fixes  
   - 2.1.3 Code Specific Changes  
   - 2.1.4 MpCCI Tutorial  
   - 2.2 MpCCI 4.6.0-2  
   - 2.2.1 Bug fixes  
   - 2.3 MpCCI 4.6.0-1  
   - 2.3.1 MpCCI Licensing  
   - 2.3.2 New Features  
   - 2.3.3 Further Enhancements of Existing Features

3. **Changes and New Features in the Earlier Releases**  
   - 3.1 MpCCI 4.5.2-1  
   - 3.1.1 New Features  
   - 3.1.2 Further Enhancements of Existing Features  
   - 3.2 MpCCI 4.5.1-1  
   - 3.2.1 New Features  
   - 3.2.2 Further Enhancements of Existing Features  
   - 3.3 MpCCI 4.5.0-1  
   - 3.3.1 MpCCI Licensing  
   - 3.3.2 MpCCI Platforms  
   - 3.3.3 New Features  
   - 3.4 MpCCI 4.4.2-1  
   - 3.4.1 Further Enhancements of Existing Features  
   - 3.5 MpCCI 4.4.1-1  
   - 3.5.1 Further Enhancements of Existing Features  
   - 3.6 MpCCI 4.4.0-1  
   - 3.6.1 New Features  
   - 3.6.2 Further Enhancements of Existing Features

4. **Prerequisites for MpCCI Installation**

5. **Supported Platforms in MpCCI 4.6**  
   - 5.1 Platforms Supported by the MpCCI 4.6 Server
5.2 Codes Supported by MpCCI 4.6 on Different Platforms 34

6 Third Party License Information 35
6.1 OpenJDK Java 35
6.2 LAPACK 40
6.3 JDOM 41
6.4 LOG4J 41

III Installation Guide 1
1 Installation Overview 5
2 Before the Installation 7
2.1 Downloading MpCCI 7
2.2 Where to Install 7
2.3 The Perl Interpreter 9
2.4 The Java Runtime Environment 9
2.5 OpenSSH for Windows 9
2.6 MpCCI-RSH for Windows 10
3 Installation of the MpCCI Software 11
3.1 Multi-Platform for Linux and Windows 11
3.2 Local Windows Installation with the MSI 12
4 Immediately After the Installation - Quick Installation Tests without a License 14
4.1 Your Home Directory under Windows 14
4.2 Testing the MpCCI Working Environment and Perl 14
4.3 Testing whether MpCCI Finds Your Simulation Codes 15
5 Licensing 17
5.1 Request for a License File 17
5.2 Installing and Activating a License 18
5.2.1 Troubleshooting License Start on Linux 19
5.2.2 Troubleshooting Getting License on Linux from a Windows System 19
5.2.3 Configure a License Manager as Linux Service 20
5.2.4 Configure a License Manager as Windows Service 20
5.3 Defining the License Server 22
5.4 Multiple License Servers 23
5.5 Testing the License Server 23
6 Configuring the MpCCI Users Environment 24
6.1 Accessing Remote Hosts 24
6.2 Configuring MpCCI via Environment Variables 25
7 Testing the MpCCI Installation and Communication 27
8 Troubleshooting 28
8.1 Secure Shell in General 28
8.2 OpenSSH under Windows 28
8.3 rsh, rcp and rlogin under Windows 28
9 Installing Perl 32
IV Getting Started

1 Multiphysics Computation with MpCCI
   1.1 Multiphysics 
   1.2 Solution of Coupled Problems 
   1.3 Code Coupling with MpCCI 

2 Setting up a Coupled Simulation
   2.1 A Simple Example 
   2.2 Model Preparation 
   2.2.1 CFD Model 
   2.2.2 FE Model 
   2.3 Starting the MpCCI GUI 
   2.3.1 Parameter Notes in the MpCCI GUI 
   2.3.2 Navigating through the MpCCI GUI 
   2.4 Models Step – Choosing Codes and Model Files 
   2.5 Algorithm Step – Defining the Coupling Algorithm 
   2.6 Regions Step – Defining Coupling Regions and Quantities 
   2.6.1 Build Coupling Regions 
   2.6.2 Define Quantities to be Exchanged 
   2.6.3 Assign Defined Quantities to Built Coupling Regions 
   2.7 Settings Step – Specifying Further Coupling Options 
   2.8 Go Step – Configuring the Application Startup and Running the Coupled Simulation 
   2.8.1 Setting Runtime Parameters 
   2.8.2 Creating a Project File 
   2.8.3 Checking the Application Configuration 
   2.8.4 Starting the Coupled Simulation 
   2.8.5 Interrupting the Computation 

3 Checking the Results
   3.1 The MpCCI Visualizer 
   3.2 Post-Processing 

V User Manual

1 Introduction 
   1.1 Basic Structure of MpCCI 

2 The MpCCI Software Package
   2.1 Introduction 
   2.2 The MpCCI Home Directory 
   2.3 Environment and Environment Variables 
   2.3.1 MPCCI_ARCH - Architecture Tokens 
   2.3.2 MPCCI_DEBUG - for Debugging 
   2.4 The MpCCI Resource Directory 

contents of all mpcci manuals
3 Code Coupling

3.1 Multiphysics

3.2 Mesh Checks

3.3 Data Exchange

3.3.1 Association

3.3.2 Interpolation

3.3.3 Quantity Relaxation

3.3.4 Quantity Operators

3.3.5 Quantity Transformation for Mesh Motion

3.3.6 Quantity Transformation for Cyclic Symmetric Meshes

3.3.7 Operation Workflow for a Quantity

3.4 Coupling Process

3.4.1 Coupling Algorithm

3.4.2 Coupling Schemes

3.4.3 Coupling with Subcyling

3.4.4 Coupling with Delayed Boundary Updates

3.4.5 Coupling with Transient Analysis Type

3.4.6 Coupling with Mixed Analysis Types

3.4.7 Restarting a Coupled Simulation

3.5 Running MpCCI in a Network

3.5.1 Client-Server Structure of MpCCI

3.5.2 Hostlist File

3.5.3 Remote Shell and Remote Copy

3.6 Coupled Analysis in Batch Mode

3.6.1 General Approach

3.6.2 Job Scheduler Environment

3.7 MpCCI Project and Output Files

3.7.1 MpCCI Project Files

3.7.2 MpCCI Server Input Files

3.7.3 Log Files

3.7.4 Tracefile

4 Graphical User Interface

4.1 Starting and Exiting MpCCI GUI

4.1.1 Starting MpCCI GUI

4.1.2 Exiting MpCCI GUI

4.2 MpCCI GUI Properties

4.3 MpCCI GUI Menus

4.3.1 File Menu
I Overview

4.3.2 Edit Menu .................................................. 78
4.3.3 Batch Menu .................................................. 79
4.3.4 License Menu ............................................... 79
4.3.5 Tools Menu .................................................. 80
4.3.6 Codes Menu .................................................. 80
4.3.7 Help Menu .................................................... 80
4.4 Models Step ................................................... 81
4.4.1 Code Parameters .......................................... 81
4.4.2 Requirements ................................................. 81
4.4.3 Changing the Model - Implications for the Setup ....... 81
4.5 Algorithm Step .................................................. 83
4.5.1 Common Basic Algorithm Settings ....................... 83
4.5.2 Code Specific Algorithm Settings ....................... 86
4.6 Regions Step ..................................................... 88
4.6.1 Global Variables ............................................. 88
4.6.2 Editing Components ....................................... 89
4.6.3 Coupling Components with Different Dimensions ...... 93
4.6.4 Simplified Selection ....................................... 93
4.6.5 Automatic Generation of Regions by Rules ............. 94
4.6.6 Applying Region Properties ................................ 98
4.6.7 Quantity Specifications ................................... 100
4.6.8 Assigning Preconfigured Quantity Settings .......... 102
4.6.9 Overview of the Coupled Regions ....................... 105
4.6.10 Requirements .............................................. 106
4.7 Monitors Step .................................................. 106
4.8 Settings Step .................................................... 107
4.8.1 Monitor ...................................................... 107
4.8.2 Job .......................................................... 108
4.8.3 Relation Search ............................................ 111
4.8.4 Output ....................................................... 111
4.9 Go Step .......................................................... 113
4.9.1 Configuring the MpCCI Coupling Server ............... 113
4.9.2 Checking the Configuration ................................ 114
4.9.3 Starting the Coupled Simulation ....................... 114
4.9.4 Status of the Simulation ................................ 115
4.10 Remote File Browser ....................................... 115
4.10.1 File Browser Handling .................................. 115
4.10.2 How to Mount a New File System ..................... 117

5 Command Line Interface ...................................... 119
5.1 Using the Command Line Interface ......................... 119
5.2 Overview of All Subcommands ................................ 120
5.3 Starting MpCCI ................................................. 122
5.3.1 mpcci fsimapper ........................................... 123
5.3.2 mpcci gui .................................................... 124
5.3.3 mpcci logviewer ............................................ 125
5.3.4 mpcci monitor .............................................. 126
5.3.5 mpcci visualize ............................................. 127
5.4 MpCCI Tools ..................................................... 129
5.4.1 mpcci ccvxcat ............................................... 130
5.4.2 mpcci cosimpre ............................................. 131
5.4.3 mpcci morpher .............................................. 132

MpCCI 4.6.1-1 I 11
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.2.4</td>
<td>Panels</td>
<td>184</td>
</tr>
<tr>
<td>7.2.5</td>
<td>Viewport Area</td>
<td>190</td>
</tr>
<tr>
<td>7.2.6</td>
<td>Preferences Viewer Dialog</td>
<td>193</td>
</tr>
<tr>
<td>7.2.7</td>
<td>Preferences Server Dialog</td>
<td>195</td>
</tr>
<tr>
<td>7.2.8</td>
<td>Store Animated Files Dialog</td>
<td>195</td>
</tr>
<tr>
<td>7.2.9</td>
<td>Error Dialog</td>
<td>197</td>
</tr>
<tr>
<td>7.2.10</td>
<td>Command Line Parameters</td>
<td>198</td>
</tr>
<tr>
<td>7.3</td>
<td>Frequently Asked Questions</td>
<td>199</td>
</tr>
<tr>
<td>8</td>
<td>MpCCI Grid Morpher</td>
<td>200</td>
</tr>
<tr>
<td>8.1</td>
<td>Using the MpCCI Grid Morpher</td>
<td>200</td>
</tr>
<tr>
<td>8.1.1</td>
<td>Options Description</td>
<td>200</td>
</tr>
<tr>
<td>VI</td>
<td>Codes Manual</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Overview</td>
<td>11</td>
</tr>
<tr>
<td>1.1</td>
<td>Common MpCCI Subcommands for Simulation Codes</td>
<td>12</td>
</tr>
<tr>
<td>1.2</td>
<td>Unit Systems</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>Abaqus</td>
<td>15</td>
</tr>
<tr>
<td>2.1</td>
<td>Quick Information</td>
<td>15</td>
</tr>
<tr>
<td>2.1.1</td>
<td>Supported Coupling Methods</td>
<td>15</td>
</tr>
<tr>
<td>2.1.2</td>
<td>Supported Platforms and Versions</td>
<td>15</td>
</tr>
<tr>
<td>2.1.3</td>
<td>References</td>
<td>15</td>
</tr>
<tr>
<td>2.1.4</td>
<td>Adapter Description</td>
<td>16</td>
</tr>
<tr>
<td>2.1.5</td>
<td>Prerequisites for a Coupled Simulation</td>
<td>16</td>
</tr>
<tr>
<td>2.2</td>
<td>Coupling Process</td>
<td>16</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Model Preparation</td>
<td>16</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Restart</td>
<td>17</td>
</tr>
<tr>
<td>2.2.3</td>
<td>Models Step</td>
<td>17</td>
</tr>
<tr>
<td>2.2.4</td>
<td>Algorithm Step</td>
<td>18</td>
</tr>
<tr>
<td>2.2.5</td>
<td>Regions Step</td>
<td>18</td>
</tr>
<tr>
<td>2.2.6</td>
<td>Go Step</td>
<td>19</td>
</tr>
<tr>
<td>2.2.7</td>
<td>Running the Computation</td>
<td>20</td>
</tr>
<tr>
<td>2.2.8</td>
<td>Post-Processing</td>
<td>21</td>
</tr>
<tr>
<td>2.3</td>
<td>Code-Specific MpCCI Commands</td>
<td>22</td>
</tr>
<tr>
<td>2.4</td>
<td>Code Adapter Reference</td>
<td>23</td>
</tr>
<tr>
<td>2.4.1</td>
<td>Patched Input File</td>
<td>23</td>
</tr>
<tr>
<td>2.4.2</td>
<td>SIMULIA’s Co-Simulation Engine (CSE)</td>
<td>23</td>
</tr>
<tr>
<td>2.5</td>
<td>Co-Simulation Restart</td>
<td>23</td>
</tr>
<tr>
<td>2.6</td>
<td>Known Limitations</td>
<td>24</td>
</tr>
<tr>
<td>2.7</td>
<td>Trouble Shooting, Open Issues and Known Bugs</td>
<td>25</td>
</tr>
<tr>
<td>3</td>
<td>ANSYS</td>
<td>26</td>
</tr>
<tr>
<td>3.1</td>
<td>Quick Information</td>
<td>26</td>
</tr>
<tr>
<td>3.1.1</td>
<td>Supported Coupling Methods</td>
<td>26</td>
</tr>
<tr>
<td>3.1.2</td>
<td>Supported Platforms and Versions</td>
<td>26</td>
</tr>
<tr>
<td>3.1.3</td>
<td>References</td>
<td>26</td>
</tr>
<tr>
<td>3.1.4</td>
<td>Adapter Description</td>
<td>26</td>
</tr>
<tr>
<td>3.1.5</td>
<td>Prerequisites for a Coupled Simulation</td>
<td>27</td>
</tr>
<tr>
<td>3.1.6</td>
<td>Supported ANSYS Product Variable</td>
<td>27</td>
</tr>
</tbody>
</table>
## Contents of All MpCCI Manuals

### I Overview

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2</td>
<td>Coupling Process</td>
<td>29</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Model Preparation</td>
<td>29</td>
</tr>
<tr>
<td>3.2.2</td>
<td>APDL Script</td>
<td>32</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Models Step</td>
<td>36</td>
</tr>
<tr>
<td>3.2.4</td>
<td>Algorithm Step</td>
<td>36</td>
</tr>
<tr>
<td>3.2.5</td>
<td>Regions Step</td>
<td>36</td>
</tr>
<tr>
<td>3.2.6</td>
<td>Go Step</td>
<td>39</td>
</tr>
<tr>
<td>3.2.7</td>
<td>Running the Computation</td>
<td>40</td>
</tr>
<tr>
<td>3.3</td>
<td>Code-Specific MpCCI Commands</td>
<td>41</td>
</tr>
<tr>
<td>3.4</td>
<td>Code Adapter Reference</td>
<td>43</td>
</tr>
<tr>
<td>3.5</td>
<td>Frequently Asked Questions</td>
<td>43</td>
</tr>
</tbody>
</table>

### 4 ANSYS Icepak

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Quick Information</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Supported Coupling Methods</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Supported Platforms and Versions</td>
</tr>
<tr>
<td>4.1.3</td>
<td>Supported Quantities</td>
</tr>
<tr>
<td>4.2</td>
<td>Code-Specific MpCCI Commands</td>
</tr>
</tbody>
</table>

### 5 FINE/Open

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Quick Information</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Supported Coupling Methods</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Supported Platforms and Versions</td>
</tr>
<tr>
<td>5.1.3</td>
<td>References</td>
</tr>
<tr>
<td>5.1.4</td>
<td>Adapter Description</td>
</tr>
<tr>
<td>5.1.5</td>
<td>Prerequisites for a Coupled Simulation</td>
</tr>
<tr>
<td>5.2</td>
<td>Coupling Process</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Model Preparation</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Models Step</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Algorithm Step</td>
</tr>
<tr>
<td>5.2.4</td>
<td>Regions Step</td>
</tr>
<tr>
<td>5.2.5</td>
<td>Go Step</td>
</tr>
<tr>
<td>5.2.6</td>
<td>Running the Computation</td>
</tr>
<tr>
<td>5.2.7</td>
<td>Post-Processing</td>
</tr>
<tr>
<td>5.3</td>
<td>Code-Specific MpCCI Commands</td>
</tr>
<tr>
<td>5.4</td>
<td>Code Adapter Reference</td>
</tr>
<tr>
<td>5.5</td>
<td>Limitations</td>
</tr>
</tbody>
</table>

### 6 FINE/Turbo

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Quick Information</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Supported Coupling Methods</td>
</tr>
<tr>
<td>6.1.2</td>
<td>Supported Platforms and Versions</td>
</tr>
<tr>
<td>6.1.3</td>
<td>References</td>
</tr>
<tr>
<td>6.1.4</td>
<td>Adapter Description</td>
</tr>
<tr>
<td>6.1.5</td>
<td>Prerequisites for a Coupled Simulation</td>
</tr>
<tr>
<td>6.2</td>
<td>Coupling Process</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Model Preparation</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Models Step</td>
</tr>
<tr>
<td>6.2.3</td>
<td>Algorithm Step</td>
</tr>
<tr>
<td>6.2.4</td>
<td>Regions Step</td>
</tr>
<tr>
<td>6.2.5</td>
<td>Go Step</td>
</tr>
<tr>
<td>6.2.6</td>
<td>Running the Computation</td>
</tr>
</tbody>
</table>

---

I MpCCI 4.6.1-1
I Overview

17.2.3 Algorithm Step .................................................. 188
17.2.4 Regions Step ...................................................... 189
17.2.5 Go Step ............................................................ 190
17.2.6 Checking the Computation ....................................... 192
17.2.7 Running the Computation .......................................... 193
17.2.8 Post-Processing .................................................... 195

17.3 Code-Specific MpCCI Commands .................................... 196

17.4 Code Adapter Reference ................................................ 198

17.4.1 Quantity Handling ................................................ 198

VII Tutorial

1 Introduction ............................................................. 8

2 Vortex-Induced Vibration of a Thin-Walled Structure ................. 11

2.1 Problem Description .................................................. 11
2.2 Model Preparation ................................................... 11
2.2.1 Fluid Model ...................................................... 12
2.2.2 Solid Model ..................................................... 13
2.3 Setting Up the Coupled Simulation with MpCCI GUI .................. 14
2.3.1 Models Step ..................................................... 14
2.3.2 Algorithm Step .................................................. 17
2.3.3 Regions Step ................................................... 18
2.3.4 Go Step .......................................................... 19
2.4 Running the Computation ............................................. 21
2.5 Discussion of Results ............................................... 21

3 Driven Cavity ............................................................ 23

3.1 Problem Description .................................................. 23
3.2 Model Preparation ................................................... 23
3.2.1 Fluid Model ...................................................... 24
3.2.2 Solid Model ..................................................... 24
3.3 Setting Up the Coupled Simulation with MpCCI GUI ................. 24
3.3.1 Models Step ..................................................... 25
3.3.2 Algorithm Step .................................................. 26
3.3.3 Regions Step ................................................... 27
3.3.4 Settings Step ................................................... 29
3.3.5 Go Step .......................................................... 29
3.4 Running the Computation ............................................. 30
3.5 Discussion of Results ............................................... 30
3.5.1 Coupling with Adaptive Time Step .............................. 32

4 Elastic Flap in a Duct ................................................... 34

4.1 Problem Description .................................................. 34
4.2 Model Preparation ................................................... 34
4.2.1 Solid Model ...................................................... 34
4.2.2 Fluid Model ..................................................... 36
4.3 Setting Up the Coupled Simulation with MpCCI GUI ................. 37
4.3.1 Models Step ..................................................... 37
4.3.2 Algorithm Step .................................................. 40
4.3.3 Regions Step ................................................... 41
### Contents of All MpCCI Manuals

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3.4</td>
<td></td>
<td>Go Step</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>Running the Computation</td>
</tr>
<tr>
<td>4.1</td>
<td></td>
<td>Starting the Simulation</td>
</tr>
<tr>
<td>4.2</td>
<td></td>
<td>End of the Simulation</td>
</tr>
<tr>
<td>4.5</td>
<td></td>
<td>Discussion of Results</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>Exhaust Manifold</td>
</tr>
<tr>
<td>5.1</td>
<td></td>
<td>Problem Description</td>
</tr>
<tr>
<td>5.2</td>
<td></td>
<td>Model Preparation</td>
</tr>
<tr>
<td>5.2.1</td>
<td></td>
<td>Solid Model</td>
</tr>
<tr>
<td>5.2.2</td>
<td></td>
<td>Fluid Model</td>
</tr>
<tr>
<td>5.2.3</td>
<td></td>
<td>Uncoupled Flow Simulation</td>
</tr>
<tr>
<td>5.4</td>
<td></td>
<td>Prepare Models for Coupled Simulation</td>
</tr>
<tr>
<td>5.3</td>
<td></td>
<td>Setting Up the Coupled Simulation with MpCCI GUI</td>
</tr>
<tr>
<td>5.3.1</td>
<td></td>
<td>Models Step</td>
</tr>
<tr>
<td>5.3.2</td>
<td></td>
<td>Algorithm Step</td>
</tr>
<tr>
<td>5.3.3</td>
<td></td>
<td>Regions Step</td>
</tr>
<tr>
<td>5.3.4</td>
<td></td>
<td>Go Step</td>
</tr>
<tr>
<td>5.4</td>
<td></td>
<td>Running the Computation</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>Busbar System</td>
</tr>
<tr>
<td>6.1</td>
<td></td>
<td>Problem Description</td>
</tr>
<tr>
<td>6.2</td>
<td></td>
<td>Model Preparation</td>
</tr>
<tr>
<td>6.2.1</td>
<td></td>
<td>Fluid Model</td>
</tr>
<tr>
<td>6.2.2</td>
<td></td>
<td>Electromagnetic Model</td>
</tr>
<tr>
<td>6.3</td>
<td></td>
<td>Setting Up the Coupled Simulation with MpCCI GUI</td>
</tr>
<tr>
<td>6.3.1</td>
<td></td>
<td>Models Step</td>
</tr>
<tr>
<td>6.3.2</td>
<td></td>
<td>Algorithm Step</td>
</tr>
<tr>
<td>6.3.3</td>
<td></td>
<td>Regions Step</td>
</tr>
<tr>
<td>6.3.4</td>
<td></td>
<td>Settings Step</td>
</tr>
<tr>
<td>6.4</td>
<td></td>
<td>Go Step</td>
</tr>
<tr>
<td>6.5</td>
<td></td>
<td>Running the Computation</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>Three Phase Transformer</td>
</tr>
<tr>
<td>7.1</td>
<td></td>
<td>Problem Description</td>
</tr>
<tr>
<td>7.2</td>
<td></td>
<td>Model Preparation</td>
</tr>
<tr>
<td>7.2.1</td>
<td></td>
<td>Fluid Model</td>
</tr>
<tr>
<td>7.2.2</td>
<td></td>
<td>Electromagnetic Model</td>
</tr>
<tr>
<td>7.3</td>
<td></td>
<td>Setting Up the Coupled Simulation with MpCCI GUI</td>
</tr>
<tr>
<td>7.3.1</td>
<td></td>
<td>Models Step</td>
</tr>
<tr>
<td>7.3.2</td>
<td></td>
<td>Algorithm Step</td>
</tr>
<tr>
<td>7.3.3</td>
<td></td>
<td>Regions Step</td>
</tr>
<tr>
<td>7.3.4</td>
<td></td>
<td>Go Step</td>
</tr>
<tr>
<td>7.4</td>
<td></td>
<td>Running the Computation</td>
</tr>
<tr>
<td>7.5</td>
<td></td>
<td>Discussion of Results</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>Pipe Nozzle</td>
</tr>
<tr>
<td>8.1</td>
<td></td>
<td>Problem Description</td>
</tr>
<tr>
<td>8.2</td>
<td></td>
<td>Model Preparation</td>
</tr>
<tr>
<td>8.2.1</td>
<td></td>
<td>Fluid Model</td>
</tr>
</tbody>
</table>
11.5 Discussion of Results ................................................................. 132
  11.5.1 Iterative Coupling Compared to Explicit Coupling .................... 132
  11.5.2 Non-Matching Time Step Sizes ........................................... 134

12 Periodic Rotating Fan Model ..................................................... 135
  12.1 Problem Description ............................................................ 135
  12.2 Model Preparation ................................................................ 136
    12.2.1 Fluid Model .................................................................. 136
    12.2.2 Solid Model .................................................................. 137
  12.3 Setting Up the Coupled Simulation with MpCCI GUI ..................... 138
    12.3.1 Models Step .................................................................. 138
    12.3.2 Algorithm Step ............................................................... 139
    12.3.3 Regions Step .................................................................. 139
    12.3.4 Settings Step .................................................................. 141
    12.3.5 Go Step ........................................................................ 141
  12.4 Running the Computation ......................................................... 141
  12.5 Discussion of Results .............................................................. 142

VIII Programmers Guide .................................................................. 1
  1 Introduction .............................................................................. 5

  2 MpCCI API .................................................................................. 6
    2.1 Code Integration and Simulation Code Requirements .................. 7
      2.1.1 Data Exchange and Data Access ........................................ 7
      2.1.2 MpCCI Interface for Code Integration .................................. 8
    2.2 Code Integration with the MpCCI API Kit .................................. 9
      2.2.1 A Simple Example ........................................................... 9
      2.2.2 Step-by-Step Procedure for Code Integration ................. 12
      2.2.3 Code Coupling with the Example .................................... 20
      2.3 Code Configuration Directory ............................................. 22
      2.4 MpCCI GUI Configuration File gui.xcf .................................. 23
        2.4.1 Code Information: <CodeInfo> .................................... 23
        2.4.2 Codes Menu: <CodesMenuEntries> ............................... 23
        2.4.3 Models Step: <ModelsMenuEntries> .............................. 24
        2.4.4 Component Types: <ComponentTypeDimensions> ........ 25
        2.4.5 List of Quantities: <SupportedQuantities> ................... 25
        2.4.6 Algorithm Step: <CouplingMenuEntries> ..................... 26
        2.4.7 Go Step: <GoMenuEntries> ........................................ 27
        2.4.8 Environments for Scanner, Checker, Starter, Stopper and Killer 29
        2.4.9 General MpCCI GUI Elements ...................................... 31
        2.4.10 Testing gui.xcf .......................................................... 39
      2.5 Perl Scripts ......................................................................... 40
        2.5.1 Using Information from gui.xcf in Scripts .................... 40
        2.5.2 Scanner.pm .................................................................. 40
        2.5.3 Checker.pm .................................................................. 42
        2.5.4 Starter.pm ................................................................... 42
        2.5.5 Stopper.pm .................................................................. 43
        2.5.6 Killer.pm .................................................................... 43
        2.5.7 Info.pm ....................................................................... 43
        2.5.8 Subcmd.pm .................................................................. 44
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.6</td>
<td>MPCI Adapter Implementation</td>
<td>45</td>
</tr>
<tr>
<td>2.6.1</td>
<td>How to Initialize the Code?</td>
<td>45</td>
</tr>
<tr>
<td>2.6.2</td>
<td>How to Define the Mesh?</td>
<td>46</td>
</tr>
<tr>
<td>2.6.3</td>
<td>How to Deal with Angular Coordinates?</td>
<td>47</td>
</tr>
<tr>
<td>2.6.4</td>
<td>How to Transfer Data?</td>
<td>48</td>
</tr>
<tr>
<td>2.6.5</td>
<td>How to Terminate the Coupling?</td>
<td>48</td>
</tr>
<tr>
<td>2.6.6</td>
<td>How to Notify a Remeshing?</td>
<td>48</td>
</tr>
<tr>
<td>2.7</td>
<td>MPCI Coupling Manager Functions</td>
<td>49</td>
</tr>
<tr>
<td>2.7.1</td>
<td>Definition of Output Functions: umpcci_msg_funcs</td>
<td>50</td>
</tr>
<tr>
<td>2.7.2</td>
<td>Definition of Output Prefix: umpcci_msg_prefix</td>
<td>51</td>
</tr>
<tr>
<td>2.7.3</td>
<td>Get Transfer Information: ampcci_tinfo_init</td>
<td>52</td>
</tr>
<tr>
<td>2.7.4</td>
<td>Initialize the Code Information Structure</td>
<td>52</td>
</tr>
<tr>
<td>2.7.5</td>
<td>Connect and Initialize an MPCI Server: mpcci_init</td>
<td>53</td>
</tr>
<tr>
<td>2.7.6</td>
<td>Configure the Code Adapter: ampcci_config</td>
<td>54</td>
</tr>
<tr>
<td>2.7.7</td>
<td>Define Part: smcci_defp</td>
<td>55</td>
</tr>
<tr>
<td>2.7.8</td>
<td>Delete a Part: smcci_delp</td>
<td>57</td>
</tr>
<tr>
<td>2.7.9</td>
<td>Definition of Nodes: smcci_pnod</td>
<td>58</td>
</tr>
<tr>
<td>2.7.10</td>
<td>Definition of Elements: smcci_pels</td>
<td>59</td>
</tr>
<tr>
<td>2.7.11</td>
<td>Definition of the Moving Reference Frame: smcci_pmot</td>
<td>61</td>
</tr>
<tr>
<td>2.7.12</td>
<td>Definition of the Baffle Thickness: smcci_pshf</td>
<td>62</td>
</tr>
<tr>
<td>2.7.13</td>
<td>Data Exchange: ampcci_transfer</td>
<td>63</td>
</tr>
<tr>
<td>2.7.14</td>
<td>Notifying the Remeshing: ampcci_remesh</td>
<td>65</td>
</tr>
<tr>
<td>2.7.15</td>
<td>End of Coupled Simulation: mpcci_quit and mpcci_stop</td>
<td>66</td>
</tr>
<tr>
<td>2.8</td>
<td>MPCI Driver Functions</td>
<td>67</td>
</tr>
<tr>
<td>2.8.1</td>
<td>Description Values</td>
<td>70</td>
</tr>
<tr>
<td>2.8.2</td>
<td>Driver Methods Called Before/After Some Action</td>
<td>71</td>
</tr>
<tr>
<td>2.8.3</td>
<td>Driver Mesh Definition Methods</td>
<td>73</td>
</tr>
<tr>
<td>2.8.4</td>
<td>Driver Data Exchange Methods</td>
<td>75</td>
</tr>
<tr>
<td>2.9</td>
<td>Data Structures and Predefined Macros</td>
<td>76</td>
</tr>
<tr>
<td>2.9.1</td>
<td>Supported Element Types</td>
<td>76</td>
</tr>
<tr>
<td>2.9.2</td>
<td>Coordinates System Definition</td>
<td>85</td>
</tr>
<tr>
<td>2.9.3</td>
<td>Mesh Dimension Definition</td>
<td>85</td>
</tr>
<tr>
<td>2.9.4</td>
<td>Moving Reference Frame Definition</td>
<td>85</td>
</tr>
<tr>
<td>2.9.5</td>
<td>Remesh Flag Information</td>
<td>86</td>
</tr>
<tr>
<td>2.9.6</td>
<td>Transfer Information: MPCCI_TINFO</td>
<td>86</td>
</tr>
<tr>
<td>2.9.7</td>
<td>Code Specific Information: MPCCI_CINFO</td>
<td>92</td>
</tr>
<tr>
<td>2.9.8</td>
<td>Coupling Components</td>
<td>94</td>
</tr>
<tr>
<td>2.9.9</td>
<td>Quantities</td>
<td>95</td>
</tr>
<tr>
<td>2.9.10</td>
<td>Loop Functions</td>
<td>96</td>
</tr>
<tr>
<td>2.9.11</td>
<td>Memory Management</td>
<td>98</td>
</tr>
</tbody>
</table>

**IX How To**

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Automotive Thermal Management</td>
<td>4</td>
</tr>
<tr>
<td>1.1</td>
<td>Quick Information</td>
<td>4</td>
</tr>
<tr>
<td>1.2</td>
<td>Problem Description and Motivation</td>
<td>4</td>
</tr>
<tr>
<td>1.3</td>
<td>Simulation Procedure</td>
<td>5</td>
</tr>
<tr>
<td>1.3.1</td>
<td>Model Preparation</td>
<td>5</td>
</tr>
<tr>
<td>1.3.2</td>
<td>MPCI Setup</td>
<td>6</td>
</tr>
<tr>
<td>1.3.3</td>
<td>Running the Simulation</td>
<td>9</td>
</tr>
</tbody>
</table>
# Contents of All MpCCI Manuals

## I Overview

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.4</td>
<td>Results</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>Simulation of Fluid-Structure Interaction for a New Hydraulic Axial Pump</td>
<td>13</td>
</tr>
<tr>
<td>2.1</td>
<td>Quick Information</td>
<td>13</td>
</tr>
<tr>
<td>2.2</td>
<td>Problem Description and Motivation</td>
<td>13</td>
</tr>
<tr>
<td>2.3</td>
<td>Simulation Procedure</td>
<td>14</td>
</tr>
<tr>
<td>2.3.1</td>
<td>Axial Hydraulic Pumps with Compensation Chambers</td>
<td>14</td>
</tr>
<tr>
<td>2.3.2</td>
<td>Model Preparation</td>
<td>16</td>
</tr>
<tr>
<td>2.3.3</td>
<td>MpCCI Setup</td>
<td>17</td>
</tr>
<tr>
<td>2.3.4</td>
<td>Running the Simulation</td>
<td>18</td>
</tr>
<tr>
<td>2.3.5</td>
<td>Results</td>
<td>19</td>
</tr>
</tbody>
</table>

## X MpCCI FSIMapper

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MpCCI FSIMapper Overview</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>MpCCI FSIMapper Installation</td>
<td>9</td>
</tr>
<tr>
<td>2.1</td>
<td>Part of MpCCI Installation</td>
<td>9</td>
</tr>
<tr>
<td>2.2</td>
<td>Standalone Version</td>
<td>9</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Local Microsoft Windows Installation</td>
<td>9</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Linux or Multi-Platform Installation</td>
<td>9</td>
</tr>
<tr>
<td>2.2.3</td>
<td>Perl</td>
<td>10</td>
</tr>
<tr>
<td>2.2.4</td>
<td>License</td>
<td>11</td>
</tr>
<tr>
<td>2.2.5</td>
<td>Configure a License Manager as UNIX Service</td>
<td>12</td>
</tr>
<tr>
<td>2.2.6</td>
<td>Configure a License Manager as Windows Service</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>MpCCI FSIMapper Command</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>MpCCI FSIMapper GUI</td>
<td>16</td>
</tr>
<tr>
<td>4.1</td>
<td>Starting the MpCCI FSIMapper</td>
<td>16</td>
</tr>
<tr>
<td>4.2</td>
<td>The “What to map” Panel</td>
<td>17</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Selection of Cases, Parts and Quantities</td>
<td>17</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Geometry Compare</td>
<td>19</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Quantity Identification for CSM Solver Output</td>
<td>19</td>
</tr>
<tr>
<td>4.2.4</td>
<td>Execute a Mapping Process</td>
<td>20</td>
</tr>
<tr>
<td>4.3</td>
<td>The “Transformation” Panel</td>
<td>22</td>
</tr>
<tr>
<td>4.3.1</td>
<td>The “Geometry” Subpanel</td>
<td>22</td>
</tr>
<tr>
<td>4.3.2</td>
<td>The “Quantity” Subpanel</td>
<td>24</td>
</tr>
<tr>
<td>4.4</td>
<td>The “How to map” Panel</td>
<td>25</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Mapping Algorithms and Neighborhood Parameters</td>
<td>25</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Orphan Filling</td>
<td>26</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Quantity Location</td>
<td>27</td>
</tr>
<tr>
<td>4.4.4</td>
<td>Saving Mapping Configurations</td>
<td>27</td>
</tr>
<tr>
<td>4.5</td>
<td>The “Result” Panel</td>
<td>27</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Average over Rotation Axis</td>
<td>28</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Apply Fourier Transformation</td>
<td>28</td>
</tr>
<tr>
<td>4.5.3</td>
<td>MSC NASTRAN Export Options</td>
<td>29</td>
</tr>
<tr>
<td>4.6</td>
<td>The “Preferences” Panel</td>
<td>29</td>
</tr>
<tr>
<td>4.7</td>
<td>The “Harmonic Wizard” Panel</td>
<td>30</td>
</tr>
<tr>
<td>4.8</td>
<td>The “Log” Panel</td>
<td>31</td>
</tr>
<tr>
<td>5</td>
<td>Codes and Formats Information</td>
<td>32</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>5.1</td>
<td>EnSight Gold</td>
<td></td>
</tr>
<tr>
<td>5.1.1</td>
<td>Supported Quantities</td>
<td></td>
</tr>
<tr>
<td>5.2</td>
<td>FLUENT</td>
<td></td>
</tr>
<tr>
<td>5.2.1</td>
<td>Supported Quantities</td>
<td></td>
</tr>
<tr>
<td>5.2.2</td>
<td>Exporting wallfuncHTC to UDM0 in Data File</td>
<td></td>
</tr>
<tr>
<td>5.3</td>
<td>FINE/Turbo</td>
<td></td>
</tr>
<tr>
<td>5.3.1</td>
<td>Supported Quantities</td>
<td></td>
</tr>
<tr>
<td>5.4</td>
<td>CFX</td>
<td></td>
</tr>
<tr>
<td>5.4.1</td>
<td>Supported Quantities</td>
<td></td>
</tr>
<tr>
<td>5.5</td>
<td>FloTHERM</td>
<td></td>
</tr>
<tr>
<td>5.5.1</td>
<td>Supported Quantities</td>
<td></td>
</tr>
<tr>
<td>5.5.2</td>
<td>Combining Static Result Files</td>
<td></td>
</tr>
<tr>
<td>5.6</td>
<td>FloEFD</td>
<td></td>
</tr>
<tr>
<td>5.6.1</td>
<td>Supported Quantities</td>
<td></td>
</tr>
<tr>
<td>5.7</td>
<td>MagNet</td>
<td></td>
</tr>
<tr>
<td>5.7.1</td>
<td>Limitations</td>
<td></td>
</tr>
<tr>
<td>5.7.2</td>
<td>Elements</td>
<td></td>
</tr>
<tr>
<td>5.7.3</td>
<td>Supported Quantities</td>
<td></td>
</tr>
<tr>
<td>5.8</td>
<td>ANSYS Maxwell</td>
<td></td>
</tr>
<tr>
<td>5.8.1</td>
<td>Elements</td>
<td></td>
</tr>
<tr>
<td>5.8.2</td>
<td>Supported Quantities</td>
<td></td>
</tr>
<tr>
<td>5.9</td>
<td>Abaqus</td>
<td></td>
</tr>
<tr>
<td>5.9.1</td>
<td>Limitations</td>
<td></td>
</tr>
<tr>
<td>5.9.2</td>
<td>Model Preparation</td>
<td></td>
</tr>
<tr>
<td>5.9.3</td>
<td>Include Boundary Conditions</td>
<td></td>
</tr>
<tr>
<td>5.9.4</td>
<td>Elements</td>
<td></td>
</tr>
<tr>
<td>5.9.5</td>
<td>Supported Quantities</td>
<td></td>
</tr>
<tr>
<td>5.10</td>
<td>ANSYS</td>
<td></td>
</tr>
<tr>
<td>5.10.1</td>
<td>Requirements</td>
<td></td>
</tr>
<tr>
<td>5.10.2</td>
<td>Model Preparation</td>
<td></td>
</tr>
<tr>
<td>5.10.3</td>
<td>Supported Quantities</td>
<td></td>
</tr>
<tr>
<td>5.10.4</td>
<td>ANSYS Scanner and Converter</td>
<td></td>
</tr>
<tr>
<td>5.11</td>
<td>MSC NASTRAN</td>
<td></td>
</tr>
<tr>
<td>5.11.1</td>
<td>Limitations</td>
<td></td>
</tr>
<tr>
<td>5.11.2</td>
<td>Include Boundary Conditions</td>
<td></td>
</tr>
<tr>
<td>5.11.3</td>
<td>Elements</td>
<td></td>
</tr>
<tr>
<td>5.11.4</td>
<td>Supported Quantities</td>
<td></td>
</tr>
<tr>
<td>5.12</td>
<td>LS-Dyna</td>
<td></td>
</tr>
<tr>
<td>5.12.1</td>
<td>Limitations</td>
<td></td>
</tr>
<tr>
<td>5.12.2</td>
<td>Elements</td>
<td></td>
</tr>
<tr>
<td>5.12.3</td>
<td>Supported Quantities</td>
<td></td>
</tr>
<tr>
<td>5.12.4</td>
<td>Include Boundary Conditions</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Batch Usage of the MpCCI FSIMapper</td>
<td></td>
</tr>
<tr>
<td>6.1</td>
<td>File Scanners</td>
<td></td>
</tr>
<tr>
<td>6.1.1</td>
<td>FLUENT</td>
<td></td>
</tr>
<tr>
<td>6.1.2</td>
<td>MentorGraphics</td>
<td></td>
</tr>
<tr>
<td>6.1.3</td>
<td>FloEFD</td>
<td></td>
</tr>
<tr>
<td>6.1.4</td>
<td>FloTHERM</td>
<td></td>
</tr>
<tr>
<td>6.1.5</td>
<td>ANSYS</td>
<td></td>
</tr>
<tr>
<td>6.1.6</td>
<td>Abaqus</td>
<td></td>
</tr>
<tr>
<td>6.1.7</td>
<td>EnSight Gold Case</td>
<td></td>
</tr>
</tbody>
</table>
## Contents of All MpCCI Manuals

### I Overview

6.2 Comparing Geometries ............................................... 53
6.3 Mapping Quantities .................................................... 54
6.4 Files Written by the MpCCI FSIMapper .............................. 54
6.5 Configuration File ...................................................... 54
6.6 Example for a Configuration File .................................... 57

### 7 Theory Guide

7.1 Numerical Methods ..................................................... 60
7.1.1 Mapping Algorithms .................................................. 60
7.1.2 Parameters for the Mapping ......................................... 60
7.1.3 Orphan Filling ........................................................ 62
7.2 Geometry and Quantity Transformations ............................. 63
7.2.1 Fourier Transformation and Windowing ............................ 63
7.2.2 Cyclic Symmetry of Quantities ..................................... 64

### 8 Tutorial

8.1 Mapping of Electromagnetic Forces for Transient and Harmonic Analyses ......................................... 66
8.1.1 Problem Description .................................................. 66
8.1.2 Source Result File ................................................... 67
8.1.3 Target Mesh File ...................................................... 70
8.1.4 Mapping .............................................................. 70
8.1.5 Target Simulation ..................................................... 76
8.2 Mapping of Harmonic Pressure Excitations in Turbomachinery ......................................................... 77
8.2.1 Using the Harmonic Balance Method of STAR-CCM+ .............. 77
8.2.2 Using the Nonlinear Harmonic Method of FINE/Turbo .................. 89
8.3 Mapping of Temperature for Microelectronic Devices .............. 98
8.3.1 Problem Description .................................................. 98
8.3.2 Source Result File ................................................... 98
8.3.3 Target Mesh Files .................................................... 99
8.3.4 Mapping of Temperature ............................................. 99
8.3.5 Results and Target Simulation ...................................... 100

### XI Appendix

XII Quantity Reference .................................................... 4
Literature ................................................................. 38
Glossary ................................................................. 39
Keyword Index ........................................................... 42
Part II

Release Notes

Version 4.6.1
II Release Notes – Contents

1 Introduction MpCCI 4.6 5

2 Changes and New Features in MpCCI 4.6 6
   2.1 MpCCI 4.6.1-1 6
   2.1.1 MpCCI FSImapper 6
   2.1.2 Bug fixes 6
   2.1.3 Code Specific Changes 6
   2.1.4 MpCCI Tutorial 8
   2.2 MpCCI 4.6.0-2 8
   2.2.1 Bug fixes 8
   2.3 MpCCI 4.6.0-1 8
   2.3.1 MpCCI Licensing 8
   2.3.2 New Features 8
   2.3.3 Further Enhancements of Existing Features 9

3 Changes and New Features in the Earlier Releases 13
   3.1 MpCCI 4.5.2-1 13
   3.1.1 New Features 13
   3.1.2 Further Enhancements of Existing Features 13
   3.2 MpCCI 4.5.1-1 16
   3.2.1 New Features 16
   3.2.2 Further Enhancements of Existing Features 17
   3.3 MpCCI 4.5.0-1 18
   3.3.1 MpCCI Licensing 18
   3.3.2 MpCCI Platforms 18
   3.3.3 New Features 18
   3.4 MpCCI 4.4.2-1 23
   3.4.1 Further Enhancements of Existing Features 23
   3.5 MpCCI 4.4.1-1 25
   3.5.1 Further Enhancements of Existing Features 25
   3.6 MpCCI 4.4.0-1 27
   3.6.1 New Features 27
   3.6.2 Further Enhancements of Existing Features 28

4 Prerequisites for MpCCI Installation 32

5 Supported Platforms in MpCCI 4.6 33
   5.1 Platforms Supported by the MpCCI 4.6 Server 33
   5.2 Codes Supported by MpCCI 4.6 on Different Platforms 34
6 Third Party License Information

6.1 OpenJDK Java ................................................................. 35
6.2 LAPACK ........................................................................ 40
6.3 JDOM ............................................................................ 41
6.4 LOG4J ............................................................................ 41
1 Introduction MpCCI 4.6

The release notes relate to MpCCI 4.6. MpCCI 4.6 represents a new release change over MpCCI 4.5. Please read the changes and features section for more details. MpCCI 4.6 is available as an upgrade from an existing MpCCI 4.5 installation.
2 Changes and New Features in MpCCI 4.6

2.1 MpCCI 4.6.1-1

2.1.1 MpCCI FSIMapper

- Support FLUENT CFF file format from FLUENT 20.0

2.1.2 Bug fixes

- Fix synchronization issue for massive parallel simulation codes.
- Fix management of convergence decision for implicit coupling.
- Additional checks are done to validate the Type of coupling step size in case of Sent by code according to the selected coupling algorithm.
- Fix report by adding parameters with non-existent files when checking the simulation parameters in the Go step. These parameters and those with incorrect values are now also given a note in their input field (›IV-2.3.1 Parameter Notes in the MpCCI GUI ◀).
- Fix blocking issue with a project saved at the Models step. The user could not access the Regions step because of following error: There are not enough components respectively quantities for coupling!
- Adapt shortcuts in the MpCCI GUI to the general standard (Ctrl +<Key>).

2.1.3 Code Specific Changes

2.1.3.1 Abaqus

- Support of Abaqus 2021.
- Thermal co-simulation fields changed from Abaqus 2018: coupling with surface film properties is reintroduced. Following coupling fields are supported:
  - from Abaqus 2016 to Abaqus 2017: use the predefined coupling type Steady state concentrated heat transfer: exchange WallTemp with FilmTemp and HeatRate.
  - from Abaqus 2018: use the predefined coupling type Steady state surface heat transfer: exchange WallTemp with FilmTemp and WallHTCoeff.
Abaqus 2018 and 2019 releases require the installation of the latest available patch (› VI-2.7 Trouble Shooting, Open Issues and Known Bugs ◀).

2.1.3.2 ANSYS

- Support of ANSYS 2021 R1.

2.1.3.3 FINE/Open

- Support of FINE/Open 10.1. All prior releases of FINE/Open are not supported because of an issue within the morpher solver. NUMECA International will not provide any patches for the older releases.
- The MpCCI GUI co-simulation setting for FINE/Open has been reviewed to fit the current FINE/Open requirements.
2.1.3.4 FINE/Turbo

- Support of FINE/Turbo 15.1.
- It is recommended to save the provided tutorial models for FINE/Turbo in the target release you want to use. MpCCI provides the models from the minimum supported release of FINE/Turbo. You will ensure that the model is properly upgraded by FINE/Turbo prior to the usage (▷ VI-6.5 Trouble Shooting, Open Issues and Known Bugs).}

2.1.3.5 FLUENT

- Support of FLUENT 2021 R1.

2.1.3.6 J MAG

- Support of J MAG 20.0.

2.1.3.7 ProLB

- Fix issue by reading the matrices files concerning the shell component. The tools generating this information have been updated.
- Update shell component name automatically for ProLB 2.6.1 from MpCCI 4.5 to MpCCI 4.6 project.

2.1.3.8 MSC Adams

- Update the ▷ VII-11 Spring Mass System model for MSC Adams to ensure a stable solution.
- Solve adapter loading issue from MSC Adams 2018 and newer releases.

2.1.3.9 MSC NASTRAN

- Support of MSC NASTRAN 2021.0.

2.1.3.10 OpenFOAM

- Support of OpenFOAM 2012.
- Improve insertion of mpcci function object in the controlDict and dynamicMeshDict files: the motion solver keyword motionSolver is supported.
- Improve user interface for the MpCCI morpher configuration settings: only relevant information is shown when a morpher option file is selected.

2.1.3.11 STAR-CCM+


2.1.3.12 TAITherm

- Support of TAITherm 2020.2.
2.1.4 MpCCI Tutorial

• Thermal coupling tutorial for MSC Marc (see VII-5 Exhaust Manifold).

2.2 MpCCI 4.6.0-2

2.2.1 Bug fixes

• The coupling duration has been corrected if the option Set maximum number of coupling steps is used in combination with subcycling. The expected total number of steps matched with the setting in the MpCCI GUI.
• The conversion of old MpCCI project file for Abaqus, ProLB, STAR-CCM+ in MpCCI 4.6.0 properly imports the coupling configuration settings.

2.3 MpCCI 4.6.0-1

2.3.1 MpCCI Licensing

• New licensing software: for MpCCI 4.6.0 you will need the FlexNet Publisher 11.16 licensing software.

2.3.2 New Features

2.3.2.1 MpCCI GUI

• The definition of the coupling algorithm has been restructured and thus simplified by introducing an additional Algorithm step (IV-2.5 Algorithm Step – Defining the Coupling Algorithm, V-4.5 Algorithm Step).
• Additional control parameters:
  – MonitorResidual for monitoring residuals of the Quasi-Newton relaxation method (see Quasi-Newton section of IV-4.8.2 Job).
  – WriteLog for writing a logfile with Quasi-Newton relaxation information (see Quasi-Newton section of IV-4.8.2 Job).
  – ConvergenceRule to define whether all or only one quantity should be converged to consider the current time step done (see IV-4.8.2 Job).

2.3.2.2 MpCCI CouplingEnvironment

• For Quasi transient or Steady state analyses, MpCCI introduces an option to reduce the idle time if one code is solving faster than the other code: the code can use the Delay boundary updates option (see IV-3.4.4 Coupling with Delayed Boundary Updates).

2.3.2.3 MpCCI FSIMapper

• Support ANSYS Maxwell export to Universal file format .unv for mapping forces from surface.
• Extend support of reading vector quantities from EnSight Gold Case format.
• New mapping algorithm conservative is available and appropriated to map integral quantities (cf. X-7 Theory Guide).
2.3.3 Further Enhancements of Existing Features

2.3.3.1 MpCCI GUI

- Direct access to the individual steps for setting up a coupled simulation is possible. However, a selected step can get stuck on a previous step if configurations are required that have not yet been fulfilled (▷ IV-2.3.2 Navigating through the MpCCI GUI ◁).

- The selection of a model file starts the scanning process directly, without the need for an additional click for the scanning process (▷ V-4.4 Models Step ◁).

- Automatically determine the appropriate analysis type for a simulation by extracting the solution type from the model while it is being scanned (▷ V-4.5.2 Code Specific Algorithm Settings ◁).

- The tolerance value for relaxation convergence was replaced by the tolerance value of the absolute convergence check operator in order to avoid incorrect user input (▷ V-4.6.7.4 Applying Operators to Quantities of Mesh Based Components ◁).

- Clearer output from checking the simulation parameters in the Go step:
  - The overall result is listed directly in the header of the code names.
  - Summary information about the coupling configuration is displayed.
  - The check button shows the last overall result.

  (▷ IV-2.8.3 Checking the Application Configuration ◁)

- The output of the running simulation codes can be requested at any time using the Running button, and not only at the end of the simulation. Closing the output window does not terminate the code (▷ V-4.9.4 Status of the Simulation ◁).

- The Edit step has been renamed Settings step for a clearer description (▷ V-4.8 Settings Step ◁).

- The coupled simulation is limited to two codes, since the new definition of the algorithm has to be adapted for more than two codes, which is currently not possible due to the lack of use cases.

2.3.3.2 MpCCI CouplingEnvironment

- Improvement of the Quasi-Newton algorithm:
  - Improvement in terms of robustness and acceleration.
  - Better conditioning of related matrices by adding three filtering methods.
  - Quasi-Newton is now able to relax quantities using information from previous time steps (cf. ▷ V-3.3.3.4 Quasi-Newton Method ◁). This reduced the average number of iterations by almost 30% for the two-dimensional case of ▷ VII-3 Driven Cavity ◁ (5.4 iterations using four previous time steps, while it takes 7.5 iterations on average for version MpCCI 4.5.2-1).
  - The Quasi-Newton method can now also be used for parallel coupling.

- MpCCI is now capable of implicit coupling with an adaptive time step size for the supporting codes (▷ V-3.4.5.2 Coupling with Exchange of Time Step Size ◁).

- From MpCCI 4.6.0 onwards, OpenJDK will be provided. The java VM is upgraded to OpenJDK Version 11.0.6+10.

- The initial quantity scaling for Quasi-Newton relaxation, which was introduced in version MpCCI 4.5.2-1, has been removed.

- Additional codes (FLUENT, MSC NASTRAN) for the tutorial ▷ VII-3 Driven Cavity ◁.
2.3.3.3 MpCCI Client

- Changes in initialization step for inhouse code adapter: in order to properly define the used coupling scheme and algorithm defined in the MpCCI GUI, the code adapter needs to call an additional function to initialize the code information data structure. You can check the changes at VIII-2.6.1 How to Initialize the Code?.

2.3.3.4 MpCCI Batch

- Support SLURM batch system (see V-5.7.6 mpcci batch SLURM).
- The `mpcci batch` command provides an additional option (-mpmode) to modify the parallel method mode to be set for the code:
  - `none` run on one cpu,
  - `smp` shared memory model,
  - `dmp` distributed memory model.
- So, the project file needs not to be edited anymore (see V-3.6.1.1 Run a Job in Batch Mode with MpCCI Command Line).

2.3.3.5 Code Specific Changes

**Abaqus**

- Fix thermal coupling based on heat rate and film temperature quantities.
- Enhancement for implicit coupling, Abaqus supports the *negotiate time step size* option (V-3.4.5.2 Coupling with Exchange of Time Step Size).

**ANSYS**


**FINE/Open**

- Support of FINE/Open 7.2, 8.1, 8.2, 9.1, 9.2.

**FINE/Turbo**


**FLUENT**

- Fix thermal coupling based on heat rate quantity calculation.
- Fix thermal coupling with FLUENT running in parallel on Microsoft Windows. In prior MpCCI release the FLUENT computation was blocked.

**ANSYS Icemak**


**JMAG**

- Support of JMAG 17.1, 18.0, 18.1, 19.0.
- Coupling limitation with JMAG model using multiple parts for the co-simulation for all versions until JMAG 18.0.
  Need to install JMAG 18.0 service pack or use JMAG 18.1 version.

MATLAB
- Support of MATLAB R2019a on Linux platform only, MATLAB R2019b on Microsoft Windows only.

MSC Adams

MSC Marc

MSC NASTRAN
- Fix for implicit coupling when the coupling time step has converged. The converged force load is kept until the next time step begins.

OpenFOAM
- Older versions than OpenFOAM v1606+ are not maintained anymore. New features are not implemented in the older versions.
- Fix support for customer internal release number.
- Enhancement for implicit coupling. OpenFOAM supports the negotiate time step size option (\(\text{\textless V-3.4.5.2 Coupling with Exchange of Time Step Size\text{\textgreater}}\)).
- Fix thermal coupling based on heat rate quantity calculation.
- Improve management of original "controlDict" and "dynamicMeshDict" files: the file backup is done only once and are assigned with the suffix -mpcci_orig on new model. Direct modification of dictionary files by user will be taken in account for the calculation. If the user deactivates the usage of the MpCCI morpher with OpenFOAM, then the original setting will be restored.

RadTherm
- RadTherm versions have been removed from the maintenance list. Newer versions are available as TAITherm now.

TAITherm
- Support of TAITherm 12.5.2, 12.6.0, 12.7.0, 13.0.0, 13.1.0, 2020.1.0.

STAR-CCM+
- Support of STAR-CCM+ 13.06.
- For thermal simulation, the user can select the type of heat transfer coefficient to be used for the coupling.
- Update morpher setting according to the new STAR-CCM+ releases.
- Fix thermal coupling based on heat rate quantity calculation.
STAR-CD

- Co-simulation with STAR-CD is provided only on demand. It is not part of the software release.
3 Changes and New Features in the Earlier Releases

3.1 MpCCI 4.5.2-1

3.1.1 New Features

3.1.1.1 MpCCI CouplingEnvironment

- The relaxation convergence of relaxed iterative couplings can be checked in order to reduce the number of coupling iterations per time step (cf. §V-3.3.3 Quantity Relaxation). When relaxation is activated, the relative difference between the unrelaxed and the relaxed quantity value is computed. In the end of the coupling time step (max. number of iterations or quantity convergence reached) this difference is printed to the log-file. If the value is higher than the user given tolerance (or $10^{-5}$), a warning is outputted. If the value is lower than the user given tolerance, the quantity state is set to CONVERGED.

- Usually, in the first iteration of an iteratively coupled time step where Quasi-Newton relaxation is used, the relaxed quantity value is equated with the unrelaxed value. Now, it is possible to set the very first relaxed quantity value to the scaled (by $\omega$) unrelaxed value (cf. §V-3.3.3.4 Quasi-Newton Method).

- Additional log file (§V-3.7.3 Log Files) with information about the job setup, neighborhood calculation and the elapsed wall clock time per coupling step. The log filename is created by following this rule: "mpcci_<JOB_NAME_PREFIX>_server.log".

- New PreCheck mode option available in the Edit Step (§V-3.2.5 Pre-Check Mode). The user can perform a partial run until the coupled mesh definition and stop the job. This requires an additional license token.

3.1.2 Further Enhancements of Existing Features

3.1.2.1 MpCCI GUI

- The unit system SI-mm-t-s has been renamed as mm-t-s.

- The methods Jacobi and Broyden have been removed from the MpCCI GUI since the computational effort and the memory requirement of the Anderson Mixing Quasi-Newton method grows only linearly with the number of degrees of freedom.

- Support for finding equally named components which are automatically generated by the scanner (see %userInfo in §VIII-2.5.2 Scanner.pm). This feature is used when applying rules for automatic region building (see §V-4.6.5 Automatic Generation of Regions by Rules).

- Enhanced rules lists by offering options for generating only one region as a result of combining some rules. This can be used to extend or reduce the components building a region (§V-4.6.5.1 Rules Lists).

- To gain greater insight into the quantity sets in the coupling step of the MpCCI GUI they are now named automatically by default. Their name will also change immediately when quantities are added or removed. (§IV-2.6.2 Define Quantities to be Exchanged)

- Simplified look and feel by removal of unused or replaced features (Undo button in edit step; options area for synchronized scrolling and selecting components in components panel of coupling step).

- New user property for automatically saving project data before starting the coupled simulation (§V-4.3.2 Edit Menu).
• Improved start-up of the coupled application by offering only one start button which starts the server and also the codes one after the other (▷ IV-2.8.4 Starting the Coupled Simulation ◁).

• The output windows from each code are now hidden and can be opened on demand. The user now recognizes the status of the finalized applications directly and has specific access to the interesting application output (▷ V-4.9.4 Status of the Simulation ◁).

3.1.2.2 MpCCI CouplingEnvironment

• Handle the warning about the usage of a deprecated Perl glob() function. This warning was displayed from Perl 5.26.

• Fix the loading of user specific environment file for a simulation code having a directory as model file. The naming convention for the file supports lowercase and the code name case proposed by MpCCI.

• Enhancement for the syntax rule for the specific code environment file "mpcci_<CODENAME>.env". User can define if a variable should be set by using the symbol = or appended by using the symbol += (▷ V-3.6.1.2 Configure a Simulation Code for Running on a Remote Machine ◁).

3.1.2.3 MpCCI Client

• Signature of the function ampcci_config has changed. Please update your code adapter (▷ VIII-2.7.6 Configure the Code Adapter: ampcci_config ◁).

3.1.2.4 MpCCI Batch

• The -np option of the mpcci batch supports the specification of number of cpus for the MpCCI morpher.

3.1.2.5 MpCCI Visualizer and MpCCI Monitor

• Improve the handling of range settings: the range can be set automatically or manually. The automatic range setting offers the option to build the minimum and the maximum of the range from all frames or from the current frame. Moreover, it is possible to build a common range over all views or to build ranges separately for each single view.

• New selection field in Settings Panel: a matching pattern can be defined to select parts from the Table of Parts.

• New option to group automatically the parts by mesh Id.

• Pause the animation during coupling: when new data is coming into the MpCCI Monitor, the automatic jump to the last frame can be suppressed by activating the Pause button.

• Iteration norms are plotted in logarithmic scale.

3.1.2.6 Code Specific Changes

Abaqus

• Support of Abaqus 2018.
• Enable Abaqus/Standard to run parallel using MPI parallel method. The run parallel using MPI option has been introduced to enable the distributed memory parallelization method. This menu has been merged with the Abaqus/Explicit settings.

• New Abaqus subcommand restartPoints to help the user construct a restart input deck by displaying the available restart points.

• Quantity WallHTCoeff is replaced by HeatRate.

ANSYS
• Support of ANSYS 18.2, 19.0.

FINE/Open
• Support of FINE/Open 6.2.

FINE/Turbo
• Support of FINE/Turbo 11.2, 12.1, 12.2.

FLUENT
• Support of FLUENT 18.2, 19.0.

• Additional checks before using the scheme function \texttt{mpcci-solve} to ensure that the case file is initialized.

• Support FLUENT in double precision mode only.

ANSYS Icepak
• Support of ANSYS Icepak 18.2, 19.0.

• Support ANSYS Icepak in double precision mode only.

JMAG
• Support of JMAG 17.0.

MSC Adams
• Support of MSC Adams 2018.

MSC Marc
• Support of MSC Marc 2017.1.

MSC NASTRAN
• Support of MSC NASTRAN 2018.0, 2018.1.

• Fix for iterative coupling definition:
  – In the NLSTEP section: the total time is correctly read out and keep the predefined output interval definition.
  – If the iterative solution converged and the total number of iterations for MSC NASTRAN is not reached, the converged force values are reused until the next time step begins.

• Additional option to select the MPI vendor to use.

OpenFOAM
• Support of OpenFOAM v1712.
• Corrections for the implicit transient coupling case:
  – Fix iterative coupling solution approach with OpenFOAM solver.
  – The time stepping is not allowed to adjust the time step to fit the writeControl option.
• Improve support of restart simulation with mesh morphing. The initial mesh is correctly setup for the MpCCI Grid Morph.
• Support of OpenFOAM velocity motion solver.
• New OpenFOAM subcommand cleanTimeDir to mark time directories of a OpenFOAM case containing empty functionObjectProperties file with &lt;time&gt;_DELETE.

RadTherm, TAITherm
• Support of TAITherm 12.4.0, 12.4.2, 12.5.1.
• Fix the MPI bootstrap option for using rsh.
• Model with curve setting for the time step size will be kept unchanged for the co-simulation.
• Steady state simulation setup with a tolerance slope value and a coupling iteration start value will compute until the reach of the coupling iteration start without taking in account the tolerance slope setting. After the coupling begins the tolerance slope setting is re-activated.
• Transient simulation setup with a coupling start time now includes the calculated solution for the time period in the exchanged data.
• Changes related to the mpcci radtherm subcommand:
  – The importCFD command accepts the usage of a relative path to the csp file.
  – The importCFD command will not include parts with assigned temperature setting. If parts list contains only parts with boundary type assigned temperature the command will not be executed.
  – The clean command will additionally reset the convection property to none of parts with assigned temperature and activated HandTfluid property. This parts may have been selected for a co-simulation and have been modified for this purpose. This command removes the MpCCI hook function properly from the hook list having other user functions.

STAR-CCM+
• Support of STAR-CCM+ 12.04, 12.06.

3.2 MpCCI 4.5.1-1

3.2.1 New Features

3.2.1.1 MpCCI FSIMapper
• New additional variant for the Fourier Transformation: spectrogram analysis can be selected in order to have for each defined subwindow a Fourier transformation. You get as results a complex frequency dependent loads changing over time (▷ X-4.5.2 Apply Fourier Transformation ◄).
  The results are available for Abaqus, ANSYS, MSC NASTRAN codes.
• New support for model defined with mirror symmetric properties. You can select a symmetric model as source to be mapped on a full model. (▷ X-4.3 The “Transformation” Panel ◄)
3.2.2 Further Enhancements of Existing Features

3.2.2.1 MpCCI GUI

- New option to export the Region Rules to a file and share the settings. (V-4.6.5 Automatic Generation of Regions by Rules)
- New option to import the Region Rules from a file. (V-4.6.5 Automatic Generation of Regions by Rules)

3.2.2.2 MpCCI CouplingEnvironment

- Fix issue during error handling and message broadcasting.
- Improve the monitor plot for iterative coupling. The user can visualize the evolution of the value during the iterative process instead of having at the end of the process the updated plot of the completed iterative coupling step.
- Fix OpenSSH installer issue with PATH update.

3.2.2.3 MpCCI Batch

- Bug fixed in case of using the useAbsolutePath option when the MpCCI server will be started on a different file system.

3.2.2.4 Code Specific Changes

Abaqus

- Iterative coupling is not available with Abaqus/Explicit solver.
- Handle recovery restart co-simulation case: the user should provide a restart time in case of performing a recovery co-simulation in order to correctly resynchronize the Abaqus calculation.

ANSYS

- Support of ANSYS 18.1.

FLUENT

- Support of FLUENT 18.1.
- Optimization for the implicit coupling. The solver convergence check is taken into account in addition to the maximum number of iterations within each iterative loop. As soon as one of the criterion is satisfied the next iterative loop will be initiated.
- Fix shifting effect on boundary value with profile. The received values are now correctly considered in the next iteration.

ANSYS Icepak

- Support of ANSYS Icepak 18.1.

J MAG

- Support of J MAG 16.0, J MAG 16.1.
MSC Adams

MSC Marc

MSC NASTRAN
- Support of MSC NASTRAN 2017.1.

OpenFOAM
- Support of OpenFOAM v1706.

RadTherm, TAITherm
- Support of TAITherm 12.3.
- Enable Distributed Memory Parallel option for the thermal solution up to RadTherm 11.3, TAITherm 12.0.
- Parallel processing options set from MpCCI GUI are transferred to RadTherm, TAITherm GUI now.
- Additional tdf file version check against the selected RadTherm, TAITherm thermal solver version.
- New command `mpcci radtherm -importCFD` to import CFD convection data from a TCD file in a coupled TDF file. The import CFD setting is based on the selected coupled parts defined in the MpCCI co-simulation project file (VI-17.3 Code-Specific MpCCI Commands).
3.3.3.2 MpCCI GUI

- Support for automatic region building (see ☞ V-4.6.5 Automatic Generation of Regions by Rules ◄).
- Region specific definition of neighborhood search properties (☞ V-4.6.6 Applying Region Properties ◄).
- Quantities are defined in own sets which may be assigned to several coupled regions. Among other things, this simplifies the assignment and modification of quantities and their settings (see ☞ IV-2.6.2 Define Quantities to be Exchanged ◄).
- Facility to use operators for transferred quantities which allow to set e.g. limiters, convergence checkers, relaxation methods or scaling functions (☞ V-4.6.7.4 Applying Operators to Quantities of Mesh Based Components ◄).
- An overview tab is added which summarizes the setup of the coupled regions. It displays code specific information like the number of coupled and uncoupled components and gives an overview of the built regions and their assigned quantity sets (see ☞ V-4.6.9 Overview of the Coupled Regions ◄).
- Feature to cross-check the application configuration to ensure that formal conditions are fulfilled ahead of starting the coupled simulation (☞ IV-2.8.3 Checking the Application Configuration ◄).
- New preferences file with settings to be used in multiple MpCCI GUI sessions - e.g. rules which are defined for automatic region generation (see ☞ V-4.2 MpCCI GUI Properties ◄).

3.3.3.3 MpCCI CouplingEnvironment

- A new relaxation approach by Quasi-Newton (☞ V-3.3.3.4 Quasi-Newton Method ◄) is available. For example, the Quasi-Newton approach is recommended for fluid-structure interactions application with incompressible gas, models with a density aspect ratio of the materials close to 1. This method treats changes of residuals on the coupled interface starting with a predictor step and further iterations. The aim is to drive the residuals to zero to quickly reach the dynamic and kinematic conservation on the interface. The information from one time-step might be used in following time-steps. This can be applied to steady and iterative transient couplings.
- Implementation of quaternion interpolation for angular coordinates.
- Global quantities are synchronized leading to properly coupling of time step size.
- Improved log information: quit is logged in the log file.
- New operators: scale, limiter, etc. (see ☞ V-4.6.7.4 Applying Operators to Quantities of Mesh Based Components ◄).
- Allow definition of neighborhood multiplicity and normal distance parameters locally per region (☞ V-4.6.6 Applying Region Properties ◄).
- Allow definition of the relaxation method for a quantity locally per region.

3.3.3.4 MpCCI Client

- Subcycling definition with a table function (see ☞ V-4.5.2 Code Specific Algorithm Settings ◄).

3.3.3.5 MpCCI FSIMapper

- Enhanced processing of transient data which is to be transferred to the frequency domain by Fourier transformation, e.g. windowing, time range filtering and result frequency truncation. ☞ X-7.2.1 Fourier Transformation and Windowing ◄
3 Changes and New Features in the Earlier Releases

II Release Notes

- Introduction of “Harmonic Wizard” tab to map a set of harmonic turbomachinery CFD results at once to a CSM model. \(\Rightarrow\) X-4.7 The “Harmonic Wizard” Panel

- Support for transient nodal force results of JMAG via MSC NASTRAN bulk data format (keywords TLOAD1, DAREA, TABLED1). \(\Rightarrow\) X-5.11.1 Reading

- LS-Dyna added as target structural code format in transient force excitation mapping applications where a Fourier transformation is applied. \(\Rightarrow\) X-5.12 LS-Dyna

3.3.3.6 Further Enhancements of Existing Features

3.3.3.7 MpCCI GUI

- Navigation buttons now are labelled according to the name of the step they lead to. An additional label also indicates the actual step.

- For more clarity some parameters in the Models and Go Step are arranged in groups which can be collapsed or expanded.

- The Coupling Step has been reworked for a more intuitive setup of the coupled regions (see \(\Rightarrow\) IV-2.6 Regions Step - Defining Coupling Regions and Quantities).

- Regions can be renamed automatically by the name of their added components which leads to an easier distinction between them.

- The state of the coupling regions now is displayed by more different icons. They differentiate between 1:1 and n:m coupled components or between uni- and bidirectional transferred quantities depending on the environment where the regions are listed.

- The settings for orphans are collected in one central area (\(\Rightarrow\) V-4.6.7.3 Orphans Settings for Mesh Based Components).

3.3.3.8 MpCCI CouplingEnvironment

- Enhancement for coupling with parallel codes using remeshing and periodic models. Time step and data are better synchronized.

- Fixed handling of models with moving mesh motion with remeshing event. Mesh velocity is not anymore affected by the remeshing event. This enables the fluid model to run massively in parallel for large models. The enhancement focuses on fluid-structure interaction with rotating parts.

The usual approach to perform such a simulation is to model each part of the solid and the fluid in the same reference frame (also known as lab frame). But when the application takes into account the deformation due to the rotation, it may lead to significant convergence and stability problems. Due to the large rotation and pressure deformation, the solid part introduces large displacement nonlinear geometric effects into the structural stiffness matrix. In order to overcome such stability problem issue, you can mix different reference frame models without affecting the physics of the problem (\(\Rightarrow\) V-3.3.5.2 Mesh Motion Application). The improved methodology for such application would be to model the fluid in the lab frame and to model the solid part in a local rotating frame of reference. Using this approach for the solid problem will remove the nonlinear, large displacement effects in the stiffness matrix conducting to a stable and faster computation.

3.3.3.9 MpCCI Client

- Fixed coupling of global quantities. A code which received only a global quantity and sent a mesh-based quantity never got the global quantity value but only the default value.
- Fixed iterative coupling: Gauss-Seidel communication scheme was not repeated for each time step.
- Fix for parallel code coupling having load balancing: avoid blocking issues during synchronisation of parallel processes.

3.3.3.10 MpCCI FSIMapper
- Improved scheme for transient mapping applications where neighborhood now is only computed once.

3.3.3.11 MpCCI Grid Morpher
- Fix for supporting models without fixed boundary conditions.

3.3.3.12 MpCCI Batch
- Improved log output about the job management: process management events are displayed.
- New batch command options (§ V-3.6.1.1 Run a Job in Batch Mode with MpCCI Command Line):
  - np Specify the number of cores for each code without editing the ".csp" file with MpCCI GUI.
  - useAbsolutePath Define the current MpCCI installation path as reference installation on the remote machine. The user is not constrained to have a configured user ".profile" or ".bashrc" file with the current MpCCI installation. This is mostly set up by a dedicated environment module file under a batch system.
- Define the environment to set before the code runs on a remote machine. The environment defined in a file is automatically exported on each remote connection (§ V-3.6.1.2 Configure a Simulation Code for Running on a Remote Machine).
- Fixed host list definition for distributed memory code: the master node should be set as master process with rank 0 in the definition.

3.3.3.13 MpCCI Visualizer and MpCCI Monitor
- Fix for plot of steady state calculations. Plot results were shifted with one iteration.
- Enhanced zoom function for the plot. During the monitoring, the zoom of the plot is not reset automatically on new incoming data.
- The plot range can be set constant during the monitoring. The defined range will be kept during the update of the plot in order to monitor the value evolution in this selected range.
- The range values can be displayed with the current minimum and maximum values of the current step.

3.3.3.14 MpCCI Tutorial
- New tutorial for iterative coupling with Quasi-Newton (see § VII-3 Driven Cavity).
- Added Abaqus input deck model for the iterative coupling tutorial Elastic Flap in Water
3.3.3.15 Code Specific Changes

Abaqus
- Fixed iterative coupling to respect the maximum number of coupled iterations.
- Improved support for coupling with MBS code.

ANSYS
- Support of ANSYS 17.0, 17.1, 17.2, 18.0.

FINE/Hexa
- Not supported anymore. Please update to FINE/Open and contact your local NUMECA International office.

FINE/Open
- Support of FINE/Open 4.2, 4.3, 5.1, 5.2, 6.1.

FINE/Turbo
- Support of FINE/Turbo 10.2, 11.1.

FLUX
- Support for FLUX software (FLUX 10.2, 10.3) only on demand. It is not anymore part of the distribution.

FLUENT
- Support of FLUENT 17.0, 17.1, 17.2, 18.0.
- Fixed issue with coupling of multiple FLUENT wall boundaries with Dynamic Mesh option.
- Fixed iterative coupling mesh update for thin wall models.
- Automatic loading of compiler environment settings using FLUENT under Microsoft Windows as it is done by the FLUENT launcher.
- Automatic setting of thermal boundary condition type. The thermal conditions heatflux and temperature will be automatically adjusted.

ANSYS Icepak
- Support of ANSYS Icepak 17.0, 17.1, 17.2, 18.0.

JMAG
- Support of JMAG 15.0, 15.1.

MSC Adams
- Support MSC Adams Car application.
- Support static to dynamic simulation.
FMI
- New adapter for the Fraunhofer EAS Master, a simulation platform for FMI coupling (prototype on demand).

MSC Marc

MSC NASTRAN
- Support of MSC NASTRAN 2016.0, 2016.1, 2017.0.
- Code selection name change from MD NASTRAN to MSC NASTRAN.

OpenFOAM
- Support OpenFOAM 3.0.1, v1606+, v1612+.
- Fixed heat flux calculation.

RadTherm, TAITherm
- Support of TAITherm 12.1, 12.2.
- Option to define the initialization of the calculation by using:
  - Part initial temperature definition.
  - Seed a steady state solution with a loaded tdf file.
  - Transient initialization with a tdf file.
  - Transient restart with a tdf file.
- Automatic calculation of view factors file if the file is not available before the simulation starts.
- Dynamic subcycling option implemented: User can define the convergence criterion to use for the steady state calculation with subcycling. The subcycling iteration may be stopped by a constant maximum number of iterations or by the defined tolerance slope value of the TAITherm solver.
- Configure solver runtime from the MpCCI GUI: e. g. solver time step size, solver duration, total number of iterations for the simulation, etc. .
- Option to write intermediate results during the TAITherm simulation.
- Option to modify the solution frequency output.

STAR-CCM+
- Support of STAR-CCM+ 10.06, 11.02, 11.04, 11.06.
- Java template macro supports subcycling with table definition.

3.4 MpCCI 4.4.2-1

3.4.1 Further Enhancements of Existing Features

3.4.1.1 Coupling Environment
- Fixed issue for a co-simulation between a transient and steady state simulation where both codes are doing a remeshing. Data and simulation code are properly synchronized.
- MpCCI Visualizer provides a new option to open a ccvx file. You can activate the automatic loading of files series having the same stem or the opening of a single file which is the default now.
• Fix issue with cut plane definition with MpCCI Visualizer. The cut plane edit line supports the two type of decimal mark dot and comma.

3.4.1.2 MpCCI FSIMapper

• MpCCI FSIMapper offers the possibility to transform cyclic symmetric source models to the corresponding full model (mesh and quantity). This allows to map data from a periodic section to a full target mesh or to a periodic section with different shape or position, cf. Figure 8 (part X).

• MpCCI FSIMapper supports the mapping of harmonic CFD simulation results, as they are calculated by the “Nonlinear Harmonic method” by FINE/Turbo or the “Harmonic Balance method” by STAR-CCM+. MpCCI FSIMapper maps the pressure excitation (represented as complex pressure field) to the structural mesh where a vibrational analysis is performed.

• For the combination of a cyclic symmetric source mesh and harmonic simulation results (as it is the case for e.g. harmonic turbine simulations) MpCCI FSIMapper supports to define the nodal diameter/cyclic symmetry mode of the excitation, see ▶ X-4.3 The “Transformation” Panel ◄.

• The EnSight Gold format is supported also for transient and harmonic source models.

• Since meshes saved in the EnSight Gold format are always 3D, MpCCI FSIMapper offers for 2D source meshes the virtual format EnSight Gold (2D).

• Besides Abaqus and MSC NASTRAN, MpCCI FSIMapper now also supports the target code ANSYS for harmonic (frequency response) analyses.

• MpCCI FSIMapper tutorials, cf. ▶ X-8 Tutorial ◄:
  – Two tutorials concerning the mapping of harmonic pressure excitations in turbines. The harmonic CFD methods in FINE/Turbo and in STAR-CCM+ are used.
  – One tutorial concerning the mapping of electromagnetic forces in motors for vibrational analyses. The source simulation code is MagNet.

3.4.1.3 Code Specific Changes

Abaqus

• Support of Abaqus 2016 pre-release.

ANSYS

• Support of ANSYS 16.1.0, ANSYS 16.2.0.

• Enhancement for ANSYS mesh smoothing event: only nodes are redefined for the co-simulation.

• Support ANSYS parallel mode with distributed domain solver. The feature is supported for the co-simulation without the exchange of the quantity Nodal Position.

• Option to activate the GPU solver for ANSYS.

FINE/Turbo

• Support of FINE/Turbo 10.1.

• Thermal coupling issue is resolved in FINE/Turbo 10.1.

FLUENT

• Support of FLUENT 16.1.0, FLUENT 16.2.0
ANSYS Icepak
- Support of ANSYS Icepak 16.1.0, ANSYS Icepak 16.2.0

JMAG
- Support of JMAG 14.1.

MSC Adams
- Support of MSC Adams 2015.
- Support co-simulation MSC Adams/Chassis product template.
- Co-simulation enhancement to support multiple statements for MSC Adams/Car.
- Support the link of additional user subroutines for user element for the co-simulation.

MSC Marc
- Support of MSC Marc 2014.2.

MSC NASTRAN
- Support of MSC NASTRAN 2014.1.

OpenFOAM
- Support of OpenFOAM 2.4.0.
- MpCCI Grid Morpher supports now the definition of a zone for a partial domain morphing.

RadTherm
- Support of RadTherm 12.0.0, RadTherm 12.0.3.

SIMPACK

STAR-CCM+
- Support of STAR-CCM+ 10.04.009.

3.5 MpCCI 4.4.1-1

3.5.1 Further Enhancements of Existing Features

3.5.1.1 Coupling Environment
- Fix missing file for the MpCCI installation on Microsoft Windows.
- Fix number of cores allocation by the MpCCI batch management system when submitting a job with a deactivated MpCCI Grid Morpher. This issue only concerns the simulation codes OpenFOAM and STAR-CD.
- Definition of global variable as monitor set is correctly supported by the MpCCI server. This variable will be defined as X-Y plot.
- The MpCCI GUI displays the option of setting a convergence tolerance for steady state coupling simulation. For the codes which support the convergence check during the iterative process this feature of terminating the coupling process can be used.
• Optional user defined timeout value can be defined for the socket connection between the client simulation code with the MpCCI server, cf. \textit{V-3.5.1 Client-Server Structure of MpCCI}. In case of connecting an additional tool like the MpCCI Grid Morpher, the MpCCI Grid Morpher can take some time to start up and perform all checks before accepting the connection with the simulation code using it.

• Fixed bug in mapping error correction procedure for nodal based flux density / integral quantities, e.g. forces.

3.5.1.2 MpCCI FSIMapper

• New option to select parts by using a string matching definition, cf. \textit{X-4.2.1 Parts Selection}.

• Improve the support of large EnSight Gold Case file data under Microsoft Windows 64 Bit.

• EnSight Gold Case file support for mapping of transient pressure fields for vibration analysis.

• Extension of the MSC NASTRAN keyword support: the surfaces defined with the following keywords are supported: WETSURF and WETELMG. 2D surface definition is also supported.

• Fix mapping of quantities Temperature and HTC for FLUENT.cas file. This has caused the MpCCI FSIMapper to crash.

3.5.1.3 Code Specific Changes

Abaqus

• Support of Abaqus 6.14-2

• Several fixes and improvements for Abaqus 6.14 have been made:
  – Start issue under Microsoft Windows has been fixed. The start sequence is smoother now and does not hang.
  – Support for pressure quantities has been reintroduced.
  – Coupling with quantity WallHeatFlux is now correctly imported in Abaqus.

ANSYS

• Support of ANSYS 16.0.0.

FINE/Turbo

• Support of FINE/Turbo 9.1.2, FINE/Turbo 9.1.3.

• Add a known bugs list concerning thermal coupling (\textit{VI-6.5 Trouble Shooting, Open Issues and Known Bugs}).

FLUENT

• Support of FLUENT 16.0.0.

ANSYS Icepak

• Support of ANSYS Icepak 16.0.0.

JMAG

• Support of JMAG 13.1 and JMAG 14.0.
MSC Adams

- Support of MSC Adams 2014.0.1.
- Fix update of the partial derivatives module in MpCCI adapter when several points are used for the co-simulation.

MSC Marc

- Support of MSC Marc 2014.1.

OpenFOAM

- Improve environment setting to run OpenFOAM simulation. The library environment setting by MpCCI is now similar to the OpenFOAM environment source file.
- Additional support of third party intel MPI vendor, cf. VI-14.2.6.2 External 3rd party MPI.
- Improve data communication performance with OpenFOAM: the MpCCI Grid Morpher is not called during the subcycling step.
- Support for coupling a decomposed OpenFOAM case project. The list of boundaries is correctly parsed.

RadTherm

- During the coupling process the tolerance slope convergence check is disabled as long as the coupling process does not converge.

SIMPACK

- Fix handling of force element when a duplicated force element has been created in MpCCI GUI. The coupled force and torque quantities were not properly applied to the force element.
- Fix connection issue with MpCCI server for all SIMPACK versions greater than 9.6.

STAR-CCM+

- Support of STAR-CCM+ 9.04.011, STAR-CCM+ 9.06.011 and STAR-CCM+ 10.02.010.

3.6 MpCCI 4.4.0-1

3.6.1 New Features

3.6.1.1 MpCCI FSIMapper

- MpCCI FSIMapper is available as standalone installation and can be started with fsimapper (cf. X-1 MpCCI FSIMapper Overview).
- Mapping of MagNet results for stationary and transient simulations is supported (cf. X-5.7 MagNet).
- Extension for Vibration Simulations:
  - a Fourier Transformation can be applied to transient results for usage in frequency response analyses in Abaqus and MSC NASTRAN.
  - Enhanced visualization of complex excitation forces as a function of time (cf. X-4.5.2 Apply Fourier Transformation).
- Support for FINE/Turbo cgns files.
- Enhancement for Abaqus element types: support of 2d elements.
### 3.6.1.2 Coupling Environment

- Enhanced support of cyclic symmetric periodic models:
  - Coupling of full and periodic models.
  - Coupling of periodic models with different section definitions, cf. V-3.3.6 Quantity Transformation for Cyclic Symmetric Meshes.
  - MpCCI GUI supports definition of periodicity, cf. V-4.6.2.3 Periodic Components.
  - Periodic model can be shown as full model in MpCCI Visualizer and written as .ccvx, cf. V-4.8 Settings Step.
  - A new tutorial VII-12 Periodic Rotating Fan Model – demonstrating the new periodic functionalities – is available.

- Adaptive under-relaxation is available in the MpCCI GUI.
  - Calculation of optimal factor is based on the Aitken’s method, cf. V-3.3.3 Quantity Relaxation.
  - Available for steady state or iteratively transient coupled simulations.
  - The used relaxation factor can be plotted in the MpCCI Monitor, cf. V-4.8 Settings Step.

- Information about orphaned regions is displayed in the MpCCI Monitor as soon as the neighborhood computation has been performed.

- New managing of tracefiles:
  - MpCCI creates a new tracefile directory for each job based on job name and a time stamp.
  - Option to delete old tracefiles automatically, cf. V-4.9.1 Configuring the MpCCI Coupling Server.

### 3.6.2 Further Enhancements of Existing Features

#### 3.6.2.1 Coupling Environment

- Fix for rotating models created not in the SI unit system.
- Java is now part of the MpCCI installation.
- Improved mapping for fluid-structure interactions with large displacement: the deformed geometry is used for integral calculations depending on the quantity type.
- New option for applications with rotating models using different motion types (e.g. rotating and stationary reference frame): not computing new neighborhoods after the initialization permits usage of bigger time steps, cf. V-4.8.2 Job.
- Improvement for simulations using remeshing: after remeshing the complete mesh information (characteristic length, bounding box,...) is updated.
- Fix for writing orphan, slaves and domain information for some tracefiles, e.g. vtk or EnSight Gold.
- In case of a fatal error the current mesh and quantity information is saved to tracefile. Useful for checking the setup.
- Correct update of the coupling information before the code starts.
- MpCCI GUI: minor fixes for managing batch mode and saving project files in all steps.
- Microsoft Windows 32 bit platform is no longer supported.
- Support of Linux 32 bit platform will end with this release.
3.6.2.2 Code Specific Changes

3.6.2.3 Abaqus

- Support of Abaqus 6.14:
  - MpCCI coupling solution for Abaqus 6.14 is based on SIMULIA’s Co-Simulation Services (CSE).
  - Iterative coupling is available starting with Abaqus 6.14.
  - For thermal coupling with Abaqus 6.14 a new quantity configuration should be used: Abaqus receives the quantities concentrated heat flux and ambient temperature. This pair of quantity definition corresponds to HeatRate and FilmTemp MpCCI quantities.
  - For fluid-structure coupling Abaqus 6.14 supports only force quantities (WallForce and RelWallForce).
  - Limitations: Axisymmetric co-simulation surface is not supported by Abaqus 6.14 SIMULIA Co-Simulation Services.
- For Abaqus versions 6.12 and 6.13 the coupling solution is still based on the direct coupling interface. This will remain available for a certain transition period. For the future, Abaqus users are encouraged to migrate to Abaqus 6.14. For the transition period Abaqus users should contact their local support to update or extend their cosimulation license token.
- Point coupling with Abaqus/Explicit 6.12 and 6.13 can be realized with Abaqus user subroutines VUAMP combined with definition of sensors in the input deck. Contact the MpCCI support for detailed information about the procedure.

3.6.2.4 ANSYS

- ANSYS 14.5 model files are now recognized by the Scanner.
- Fix for the MpCCI ANSYS remesh command ~mpcci, remesh user request is taken into account and morphed mesh is properly updated in the MpCCI server.

3.6.2.5 FINE/Open

- FINE/Open 3.x is not supported by MpCCI.
- Support of FINE/Open 4.1.

3.6.2.6 Flowmaster

- Additional check if the quantity mass flow rate is positive.
- Proper handling of subcycling for steady state simulations.

3.6.2.7 FLUENT

- Support of FLUENT 15.0.7.
- Coupling a rigid body model with 1D codes is supported.
- Mesh update for steady state FSI simulation is performed automatically by using the MpCCI Run FSI panel from FLUENT or via the TUI command [mpcci-solve steps], cf. VI-8.2.6.2 The MpCCI Run FSI c.
- Improvements for the FLUENT co-simulation control in batch.
• Fix issue with time step size value for FLUENT single precision simulations.
• Fix for sending the calculated time step size to MpCCI in case of adaptive time stepping.
• FLUENT can access additional RP variables to check the MpCCI status.

3.6.2.8 J MAG
• Support scanning of J MAG model file with white spaces.

3.6.2.9 MSC Adams
• Support of MSC Adams 2014.

3.6.2.10 MSC Marc
• Support of MSC Marc 2014.

3.6.2.11 MD NASTRAN
• Support of MD NASTRAN 2014.

3.6.2.12 OpenFOAM
• Support of OpenFOAM 2.2, OpenFOAM 2.3.
• Additional support of third party MPI vendor, e.g. sgi with improvements for starting the application with the proper co-simulation environment, cf. ▷ VI-14.2.6.2 External 3rd party MPI ◁.

3.6.2.13 RadTherm
• Support of RadTherm 11.2, RadTherm 11.3.
• RadTherm scanner now recognizes volume parts.
• Additional checks during the preparation of the tdf file. The type of the H and Tfluid boundary is checked for type “value”.
• Proper handling of user defined part id in RadTherm model.
• Support coupling of a part with imposed wall temperature.

3.6.2.14 SIMPACK
• Support of SIMPACK 9.6, 9.7.

3.6.2.15 STAR-CCM+
• Support of STAR-CCM+ 9.04.009.
• Fix support of models with multiple region definitions.
3.6.2.16 STAR-CD

- Support of STAR-CD 4.18, STAR-CD 4.20, STAR-CD 4.22.
4 Prerequisites for MpCCI Installation

Please see ▶ 5 Supported Platforms in MpCCI 4.6 ◀ for platform-specific prerequisites.

Required Disc Space
A full MpCCI installation (all platforms and all code adapters, plus a multi-platform Java JRE and the MpCCI-RSH and the OpenSSH for Microsoft Windows) requires a free disc space of approx. 1.0 GB.

Third Party Software
Perl
Java
MpCCI-RSH for Windows is provided by MpCCI. This allows access to all Microsoft Windows operating systems (See ▶ III-2.6 MpCCI-RSH for Windows ◀).
5 Supported Platforms in MpCCI 4.6

Platform lists for supported simulation codes are given in the Codes Manual.

5.1 Platforms Supported by the MpCCI 4.6 Server

<table>
<thead>
<tr>
<th>Platform</th>
<th>Bits</th>
<th>MpCCI arch.</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux with glibc 2.3 on AMD64 or EM64T processor</td>
<td>32/64</td>
<td>lnx3_x64</td>
<td>not supported</td>
</tr>
<tr>
<td>Linux with glibc 2.4 on AMD64 or EM64T processor</td>
<td>32/64</td>
<td>lnx4_x64</td>
<td>OK</td>
</tr>
<tr>
<td>Microsoft Windows 32 bit on AMD/Intel</td>
<td>32</td>
<td>windows_x86</td>
<td>not supported</td>
</tr>
<tr>
<td>Microsoft Windows 64 bit on AMD/Intel</td>
<td>32/64</td>
<td>windows_x64</td>
<td>OK</td>
</tr>
</tbody>
</table>

The above list is valid for MpCCI alone, i.e. the MpCCI executables. This does not automatically include all code adapters. Some simulation codes can only be coupled on a subset of the above platforms. A list of platforms for the supported codes is given in the following section and for each code in the corresponding chapter of the Codes Manual.
5.2 Codes Supported by MpCCI 4.6 on Different Platforms

Codes can only be supported on platforms they support themselves.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Codes Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Linux with glibc 2.4 on AMD64 or EM64T processor (lnx4_x64)</strong></td>
<td><strong>(1)</strong></td>
</tr>
<tr>
<td>ANSYS</td>
<td>160, 161, 162, 170, 171, 172, 180, 181, 182, 190, 191, 192, 193, 194, 195, 201, 202, 211</td>
</tr>
<tr>
<td>FINE/Open</td>
<td>101</td>
</tr>
<tr>
<td>FINE/Turbo</td>
<td>111, 112, 121, 122, 131, 132, 141, 142, 151</td>
</tr>
<tr>
<td>FLUENT</td>
<td>16.0.0, 16.1.0, 16.2.0, 17.0.0, 17.1.0, 17.2.0, 18.0.0, 18.1.0, 18.2.0, 19.0.0, 19.1.0, 19.2.0, 19.3.0, 19.4.0, 19.5.0, 20.1.0, 20.2.0, 21.1.0</td>
</tr>
<tr>
<td>ANSYS Icepak</td>
<td>16.0.0, 16.1.0, 16.2.0, 17.0.0, 17.1.0, 17.2.0, 18.0.0, 18.1.0, 18.2.0, 19.0.0, 19.1.0, 19.2.0, 19.3.0, 19.4.0, 19.5.0, 20.1.0, 20.2.0, 21.1.0</td>
</tr>
<tr>
<td>JMag</td>
<td>15.0, 15.1, 16.0, 16.1, 17.0, 17.1, 18.0, 18.1, 19.0, 20.0</td>
</tr>
<tr>
<td>MATLAB</td>
<td>R2016b, R2017a, R2018a, R2019a</td>
</tr>
<tr>
<td>SIMPACK</td>
<td>9.7, 9.8.1</td>
</tr>
<tr>
<td>TAITherm</td>
<td>12.0.0, 12.0.3, 12.1.0, 12.1.1, 12.2.0, 12.3.0, 12.3.1, 12.4.0, 12.4.2, 12.5.1, 12.5.2, 12.6.0, 12.7.0, 13.0.0, 13.1.0, 2020.1.0, 2020.2.0</td>
</tr>
<tr>
<td><strong>Microsoft Windows 64 bit on AMD/Intel (windows_x64)</strong></td>
<td><strong>(2)</strong></td>
</tr>
<tr>
<td>ANSYS</td>
<td>160, 161, 162, 170, 171, 172, 180, 181, 182, 190, 191, 192, 193, 194, 195, 201, 202, 211</td>
</tr>
<tr>
<td>FINE/Open</td>
<td>101</td>
</tr>
<tr>
<td>FINE/Turbo</td>
<td>111, 112, 121, 122, 131, 132, 141, 142, 151</td>
</tr>
<tr>
<td>FLUENT</td>
<td>16.0.0, 16.1.0, 16.2.0, 17.0.0, 17.1.0, 17.2.0, 18.0.0, 18.1.0, 18.2.0, 19.0.0, 19.1.0, 19.2.0, 19.3.0, 19.4.0, 19.5.0, 20.1.0, 20.2.0, 21.1.0</td>
</tr>
<tr>
<td>FLOMASTER</td>
<td>7.6, 7.7, 7.8, 8.0, 8.1, 8.2, 9.0, 9.2</td>
</tr>
<tr>
<td>ANSYS Icepak</td>
<td>16.0.0, 16.1.0, 16.2.0, 17.0.0, 17.1.0, 17.2.0, 18.0.0, 18.1.0, 18.2.0, 19.0.0, 19.1.0, 19.2.0, 19.3.0, 19.4.0, 19.5.0, 20.1.0, 20.2.0, 21.1.0</td>
</tr>
<tr>
<td>JMag</td>
<td>15.0, 15.1, 16.0, 16.1, 17.0, 17.1, 18.0, 18.1, 19.0, 20.0</td>
</tr>
<tr>
<td>MATLAB</td>
<td>R2016b, R2017a, R2018a, R2019b</td>
</tr>
<tr>
<td>SIMPACK</td>
<td>9.7, 9.8.1</td>
</tr>
<tr>
<td>TAITherm</td>
<td>12.0.0, 12.0.3, 12.1.0, 12.1.1, 12.2.0, 12.3.0, 12.3.1, 12.4.0, 12.4.2, 12.5.1, 12.5.2, 12.6.0, 12.7.0, 13.0.0, 13.1.0, 2020.1.0, 2020.2.0</td>
</tr>
</tbody>
</table>
6 Third Party License Information

6.1 OpenJDK Java

GNU General Public License, version 2, with the Classpath Exception

The GNU General Public License (GPL)

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".
Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program’s source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program. You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:
   a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
   b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
   c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
   a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
   b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
   c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in
object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients’ exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

One line to give the program's name and a brief idea of what it does.

Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See

II MpCCI 4.6.1-1
the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'. This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show w' and 'show c'; they could even be mouse-clicks or menu items–whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program 'Gnomovision' (which makes passes at compilers) written by James Hacker.

signature of Ty Coon, 1 April 1989

Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

"CLASSPATH" EXCEPTION TO THE GPL

Certain source files distributed by Oracle America and/or its affiliates are subject to the following clarification and special exception to the GPL, but only where Oracle has expressly included in the particular source file's header the words "Oracle designates this particular file as subject to the "Classpath" exception as provided by Oracle in the LICENSE file that accompanied this code."

Linking this library statically or dynamically with other modules is making a combined work based on this library. Thus, the terms and conditions of the GNU General Public License cover the whole combination.

As a special exception, the copyright holders of this library give you permission to link this library with independent modules to produce an executable, regardless of the license terms of these independent modules, and to copy and distribute the resulting executable under terms of your choice, provided that you also meet, for each linked independent module, the terms and conditions of the license of that module. An independent module is a module which is not derived from or based on this library. If you modify this library, you may extend this exception to your version of the library, but you are not obligated to do so. If you do not wish to do so, delete this exception statement from your version.

ADDITIONAL INFORMATION ABOUT LICENSING

Certain files distributed by Oracle America, Inc. and/or its affiliates are subject to the following clarification and special exception to the GPLv2, based on the GNU Project exception for its Classpath libraries, known as the GNU Classpath Exception.

Note that Oracle includes multiple, independent programs in this software package. Some of those programs are provided under licenses deemed incompatible with the GPLv2 by the Free Software Foundation and others. For example, the package includes programs licensed under the Apache License, Version 2.0 and may include FreeType. Such programs are licensed to you under their original licenses.

Oracle facilitates your further distribution of this package by adding the Classpath Exception to the necessary parts of its GPLv2 code, which permits you to use that code in combination with other independent
modules not licensed under the GPLv2. However, note that this would not permit you to commingle code
under an incompatible license with Oracle’s GPLv2 licensed code by, for example, cutting and pasting
such code into a file also containing Oracle’s GPLv2 licensed code and then distributing the result.
Additionally, if you were to remove the Classpath Exception from any of the files to which it applies
and distribute the result, you would likely be required to license some or all of the other code in that
distribution under the GPLv2 as well, and since the GPLv2 is incompatible with the license terms of some
items included in the distribution by Oracle, removing the Classpath Exception could therefore effectively
compromise your ability to further distribute the package.

Failing to distribute notices associated with some files may also create unexpected legal consequences.
Proceed with caution and we recommend that you obtain the advice of a lawyer skilled in open source
matters before removing the Classpath Exception or making modifications to this package which may
subsequently be redistributed and/or involve the use of third party software.

6.2 LAPACK

Copyright (c) 1992-2013 The University of Tennessee and The University of Tennessee Research Founda-
tion. All rights reserved.
Copyright (c) 2000-2013 The University of California Berkeley. All rights reserved.
Copyright (c) 2006-2013 The University of Colorado Denver. All rights reserved.
$COPYRIGHT$
Additional copyrights may follow
$HEADERS$
Redistribution and use in source and binary forms, with or without modification, are permitted provided
that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the
  following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the
  following disclaimer listed in this license in the documentation and/or other materials provided with the
  distribution.

- Neither the name of the copyright holders nor the names of its contributors may be used to endorse or
  promote products derived from this software without specific prior written permission.

The copyright holders provide no reassurances that the source code provided does not infringe any patent,
copyright, or any other intellectual property rights of third parties. The copyright holders disclaim any
liability to any recipient for claims brought against recipient by any third party for infringement of that
parties intellectual property rights.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS
IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PUR-
POSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBU-
TORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUB-
STITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPT-
ION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY
WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.
6.3 JDOM

Copyright (C) 2000-2012 Jason Hunter & Brett McLaughlin.
All rights reserved.
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
1. Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the disclaimer that follows these conditions in the documentation and/or other materials provided with the distribution.
3. The name "JDOM" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact <request_AT_jdom_DOT_org>.
4. Products derived from this software may not be called "JDOM", nor may "JDOM" appear in their name, without prior written permission from the JDOM Project Management <request_AT_jdom_DOT_org>.

In addition, we request (but do not require) that you include in the end-user documentation provided with the redistribution and/or in the software itself an acknowledgement equivalent to the following:
"This product includes software developed by the JDOM Project (http://www.jdom.org/)."
Alternatively, the acknowledgment may be graphical using the logos available at http://www.jdom.org/images/logos.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE JDOM AUTHORS OR THE PROJECT CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the JDOM Project and was originally created by Jason Hunter <jhunter_AT_jdom_DOT_org> and Brett McLaughlin <brett_AT_jdom_DOT_org>. For more information on the JDOM Project, please see <http://www.jdom.org/>.

6.4 LOG4J

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION
1. Definitions.
"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.
"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.
"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled
by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and
(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NONINFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.
To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don’t include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.
Part III

Installation Guide

Version 4.6.1
III Installation Guide – Contents

1 Installation Overview

2 Before the Installation
   2.1 Downloading MpCCI
   2.2 Where to Install
   2.3 The Perl Interpreter
   2.4 The Java Runtime Environment
   2.5 OpenSSH for Windows
   2.6 MpCCI-RSH for Windows

3 Installation of the MpCCI Software
   3.1 Multi-Platform for Linux and Windows
   3.2 Local Windows Installation with the MSI

4 Immediately After the Installation - Quick Installation Tests without a License
   4.1 Your Home Directory under Windows
   4.2 Testing the MpCCI Working Environment and Perl
   4.3 Testing whether MpCCI Finds Your Simulation Codes

5 Licensing
   5.1 Request for a License File
   5.2 Installing and Activating a License
     5.2.1 Troubleshooting License Start on Linux
     5.2.2 Troubleshooting Getting License on Linux from a Windows System
     5.2.3 Configure a License Manager as Linux Service
     5.2.4 Configure a License Manager as Windows Service
   5.3 Defining the License Server
   5.4 Multiple License Servers
   5.5 Testing the License Server

6 Configuring the MpCCI Users Environment
   6.1 Accessing Remote Hosts
   6.2 Configuring MpCCI via Environment Variables

7 Testing the MpCCI Installation and Communication

8 Troubleshooting
   8.1 Secure Shell in General
   8.2 OpenSSH under Windows
   8.3 rsh, rcp and rlogin under Windows
9 Installing Perl

9.1 Linux ......................................................... 32
9.2 Windows ....................................................... 32
  9.2.1 Strawberry Perl for Windows ......................... 32
  9.2.2 ActivePerl for Windows ................................. 33
  9.2.3 File Name Associations for Perl under Windows ...... 33
1 Installation Overview

Please check the Release Notes for installation prerequisites and supported platforms first.

MpCCI installation does not require much, there are just a few steps you need to do. There are following ways to install MpCCI:

- via Linux installation
- via Windows installer
- via Multi-platform installation which is recommended for installations on a file server or for MpCCI distributors.

Download Account

Before the installation you should contact the MpCCI team at mpcci@scapos.de to obtain an account and a password for the software download.

Linux Installation

1. Download zip archive:
   - Visit the MpCCI website of Fraunhofer SCAI at www.mpcci.de and navigate to the download area or
   - use directly https://download.scai.fraunhofer.de
   and download a compressed archive "mpcci-*-lnx4_x64.zip" containing the MpCCI software.

2. Extract MpCCI software:
   Create a directory and unzip the downloaded file within this directory. The directory contains at least following files:
   - the installer script "mpcci_install.pl"
   - the MpCCI end user license agreement "MpCCI_License_Agreement.txt"
   - the MpCCI core package "mpcci-core.zip"
   - the MpCCI free tools package "mpcci-free.zip"
   Execute the installer script "mpcci_install.pl" and follow the instructions. The user is requested to provide the path for the MpCCI installation. This one will be extended by the installer with the MpCCI release number. This new sub directory is the MpCCI root directory which is referred to as <MpCCI_home>.

3. Add MpCCI to your PATH:
   Append the MpCCI binaries directory "<MpCCI_home>/bin" to your PATH environment variable such that the command [mpcci] is available.

A detailed description of this procedure is given in ▶3.1 Multi-Platform for Linux and Windows ◄.

Windows Installer

1. Download Windows installer:
   - Visit the MpCCI website of Fraunhofer SCAI at www.mpcci.de and navigate to the download area, or
   - use directly https://download.scai.fraunhofer.de
   and download the Windows installer for MpCCI software.

2. Run the installer:
   Start the installer, which also offers to install third-party software needed by MpCCI. The installer also sets up your environment.
Multi-Platform Installation

The MpCCI Multi-Platform Installation is constructed as the Linux installation. The only difference is the compressed archive "mpcci-*-allOS.zip" containing the MpCCI software for all platforms (Linux and Windows).

For the installation, please follow the detailed description of this procedure in 3.1 Multi-Platform for Linux and Windows.

Licensing

MpCCI uses a FlexNet Publisher license server. The setup is described in 5 Licensing.
2 Before the Installation

2.1 Downloading MpCCI

For the MpCCI download you need a download account!

MpCCI is only available via internet download:

- Go to the MpCCI homepage [www.mpcci.de](http://www.mpcci.de) and navigate to the download area or go directly to [https://download.scai.fraunhofer.de](https://download.scai.fraunhofer.de).
- Enter e-mail address (used as username) and password and press [Sign In](https://download.scai.fraunhofer.de).
- Select MpCCI CouplingEnvironment from the MpCCI menu.
- Expand the MpCCI version you want to download in the list appearing.
- There are three download variants:
  - The Linux download for Linux systems.
  - The Windows Installer download for local installations on Windows systems.
  - The Multi-Platform Download containing all platforms. It is recommended for installation on a file server and for distributors. The installation should be done on a Linux system.

Choose the appropriate variant and proceed.

Linux Download

- Download the zipped download file "mpcci-<MpCCI-version>-lnx4_x64.zip".
- Proceed with the installation as described in 3.1 Multi-Platform for Linux and Windows.

Windows Installer Download

- Please download the installer "mpcci-<MpCCI-version>-windows_x64.exe".
- Proceed with the installation as described in 3.2 Local Windows Installation with the MSI.

Multi-Platform Download

- Download the zipped download file "mpcci-<MpCCI-version>-allOS.zip".
- Proceed with the installation as described in 3.1 Multi-Platform for Linux and Windows.

2.2 Where to Install

MpCCI is not just a multi-code software but also a multi-platform software. That means all kinds of Linux and Windows platforms can live together within a single directory on any machine, no matter whether it is a Linux or Windows system. The command `mpcci` runs on any platform.

We should mention that Windows is used as a synonym for various Microsoft Windows systems: like Windows 7.

MpCCI uses architecture tokens to identify a platform. The architecture tokens are hereinafter referred to as `<MpCCI<arch>`. The architecture tokens are directory names and used to locate the binary executables.

A list of architecture tokens and currently supported platforms is given in II-5 Supported Platforms in MpCCI 4.6.

You may rename the MpCCI home directory or move the MpCCI home directory to a different location or even a different file server later on, since MpCCI never works with absolute pathnames nor is any file patched during the installation. Under Windows MpCCI never depends on any registry key.
As long as you keep your PATH environment variable up-to-date MpCCI will work by just typing `mpcci`. System dependent information is collected at runtime and the required system dependent settings are done automatically. There is no need to configure your MpCCI installation for a specific platform or for your local machine. Whatever operating system you are using, MpCCI behaves identical or at least similar.

The major difference between the Linux world and Microsoft Windows is that under Windows the X11 window system is not available per default and you cannot redirect the graphical display output from your local machine to a remote Linux or Windows computer.

If you want to use the remote execution facilities of MpCCI, please ensure that the X11 forwarding mechanism works. The server must allow connections from remote computers and no firewall may block the port 6000.

If you have a computer network and an NFS file server or Samba is available in your local area network, there is definitely no need to have a separate installation on each local desktop computer, whether it is a Linux or Windows system. Nevertheless, if you like you may install MpCCI locally on your desktop computer or copy the MpCCI home directory from the file server onto your local machine.

**Windows Only Version and the Microsoft Windows Installer**

For standalone Windows systems without access to file servers we offer a separate MSI (Microsoft-Installer) version of MpCCI.

⚠️ The Microsoft Windows MSI versions of MpCCI are identical with the Windows parts of the multi-platform version. The difference is the installation procedure itself, not the software.

During the MSI installation some Windows registry entries are created which allows uninstalling MpCCI later on using the Windows uninstaller. Furthermore MpCCI is added to the Windows Start-Commands list.

Although you used the MSI version you may move or rename the MpCCI home directory as with the multi-platform installation. MpCCI will still work as before. However your Windows uninstall information is then corrupted since all your links or your desktop icons are lost and the uninstaller will not be able to remove your MpCCI installation.

⚠️ In fact you do not need to have the MSI version if MpCCI can be installed on a file server. There is no advantage compared to the multi-platform installation. You may simply remove MpCCI from your Windows system by deleting the MpCCI home directory.

**Recommendations**

Each computer in a network used to run MpCCI jobs needs to have access to an MpCCI installation.

We recommend to have a multi-platform file server installation if you can run the installation on a Linux system, and the MpCCI home directory on the file server is then mounted to your local computer.

Otherwise, in case of a compute cluster with local discs only you would need to copy the MpCCI home directory on each computer separately.

If your HOME directory is shared among the computers in a network you may also have your private MpCCI installation, e.g. under HOME/bin. In this case your HOME directory acts as a file server. This is the preferred method in case you are in trouble with your IT.

⚠️ Avoid a local installation if possible. A file server installation is the preferred way using MpCCI in a heterogeneous network.
2.3 The Perl Interpreter

Since MpCCI is a multi-platform software you can never fire up any binary executable directly - these executables are wrapped by scripts. To avoid the maintenance of different script languages (Bourne shell "/bin/sh" under Linux and ".BAT" files under Windows) in fact all MpCCI commands are Perl scripts and MpCCI requires a working Perl installation.

Perl is a platform independent script language which allows running identical scripts under Linux as well as under Windows.

Perl was invented by Larry Wall. Perl is a public domain software distributed under the GNU Public License (GPL) and can be downloaded for free from the world wide web. Perl needs to be installed separately on your local computer.

Perl is not part of the MpCCI installation.

If Perl is already installed on your system - this is true for nearly any Linux system - you may check the Perl version by typing

```
perl -version
```

For MpCCI under Linux you need at least Perl 5.6.

Perl for Windows

Perl is never part of your Windows system. We recommend to have the Strawberry Perl 5.8.8 or higher installed. There are several issues with Perl versions before ActivePerl 5.8.0.

To be up to date with your Perl you may follow the link

strawberryperl.com/ resp.

www.activestate.com/Products/ActivePerl.

Strawberry Perl is the best available Windows port of Perl. A 64 bit version of Strawberry Perl for Windows 64 bit is available.

If you need to install Perl or upgrade your Perl installation, please see \9 Installing Perl \.<

For MpCCI under Windows you need at least Perl 5.8.

2.4 The Java Runtime Environment

The MpCCI GUI is a Java application based on OpenJDK. At least an OpenJDK Runtime Environment 11 (JRE) or higher is required. This corresponds to Java SE 11.

An OpenJDK Runtime Environment is part of the MpCCI installation and of the Microsoft Windows MSI installation. There is no need to have Java installed before you install MpCCI.

2.5 OpenSSH for Windows

OpenSSH is a free ssh distribution. OpenSSH is required under Windows to execute commands on remote Windows or Linux systems from a Windows or a Linux system. OpenSSH makes Windows interoperable with other Windows systems and Linux.

OpenSSH is part of the MpCCI installation and may be finally installed on the fly. You should not download
OpenSSH for Windows from the web and install it.

OpenSSH for Windows is Already Installed

You are familiar with OpenSSH and already worked with OpenSSH.

Recommendations

Do never install OpenSSH separately. Please let MpCCI do this job.

OpenSSH for Windows and cygwin

There are known issues if OpenSSH and cygwin are installed in parallel on the same machine, e.g. both come with an "ls" command accessing different and possibly incompatible versions of the same ".dll" files. In this case the OpenSSH may fail.

Best is to have either the OpenSSH installed or your existing cygwin installation includes a working ssh service. In the latter case the OpenSSH installation is not required.

If you need the OpenSSH and the cygwin installation in parallel then please make sure that the OpenSSH binaries directory is in front of your PATH environment variable.

2.6 MpCCI-RSH for Windows

MpCCI comes with its own rshd and rlogind services for Windows. This package includes both, the services and the rsh and rcp commands. Although this software is part of MpCCI, it is not necessarily bound to MpCCI. Therefore the service is - like the OpenSSH - a separate installation under Windows.

MpCCI-RSH is required under Windows to execute commands on remote Windows or Linux systems from a Windows or a Linux system. MpCCI-RSH makes Windows interoperable with other Windows systems and Linux.

Using the MpCCI Microsoft Windows installer, the MpCCI-RSH package is automatically installed and started.

Otherwise if you have a computer network and an NFS file server or Samba is available in your local area network and you need to install the MpCCI-RSH package, you can execute the installation program "<MpCCI_home>\mswin\mpccirsh\setup.exe".

After the installation of MpCCI-RSH you need to prepare the .rhosts file and rsh environment, see ▷ 8.3 Preparing the .rhosts File and the rsh Environment ◄.
3 Installation of the MpCCI Software

If you intend to install MpCCI under Linux or on a file server you should have downloaded a multi-platform distribution file. For a local Windows installation you download an "MSI" file.

3.1 Multi-Platform for Linux and Windows

For all supported platforms you will find a distribution on the MpCCI download site. No matter on which systems you are logged on, Linux or Windows, the installation procedure is identical.

You need not to be a Windows administrator or the Linux root user if you do not extract the MpCCI distribution file within protected directories. MpCCI can be installed everywhere, e.g. in your private MpCCI home directory, 

"$HOME/MpCCI" or "%USERPROFILE%\MpCCI".

After downloading your MpCCI distribution file you have to:

1. Create your temporary directory to extract the content of the downloaded file
2. Change to your temporary directory: `cd <temporary directory path>`
3. Extract the MpCCI files from the downloaded file with the command
   
   ```
   unzip mpcci-<MpCCI-version>.zip
   ```
4. Execute the installer script "mpcci_install.pl"
5. Accept the license agreement
6. Provide the installation path directory
7. Authorize the extraction of the files
8. Update your user PATH environment with the MpCCI bin directory.

The MpCCI software is now installed on your system.

Do Not Forget to Set Your PATH Environment!

To use MpCCI without having trouble the MpCCI binaries directory "<MpCCI_home>/bin" needs to be in the PATH. If the MpCCI binaries directory is not in the PATH a remote connection to this machine from a remote host via any MpCCI software is impossible.

If you already have a previous release of MpCCI in the PATH, please prepend the newest release in front of the PATH and do not append it at the end. Otherwise the mpcci command selected will always start the previous release.

According to your login shell under Linux (see "/etc/passwd") you need to set:

Bourne shell (/bin/sh) users

```
PATH=<MPCCI_HOME>/bin:$PATH
export PATH
```

Korn shell (ksh) and bash users

```
export PATH=<MPCCI_HOME>/bin:$PATH
```

c-shell (csh) and tc-shell (tcsh) users

```
setenv PATH <MPCCI_HOME>/bin:$PATH
```

and under Windows
3 Installation of the MpCCI Software

### III Installation Guide

```bash
set PATH=<MPCCI_HOME>in;%PATH%
```

1. Under **Windows** you need to set the PATH if you are using the multi-platform installation zip-file.
2. Please keep in mind that a list separator is ':' under **Linux** and ';' under **Windows**.

#### 3.2 Local Windows Installation with the MSI

You can execute the installation as an administrator or normal user account.

Please execute the downloaded self-extracting archive, e.g. "mpcci-<MpCCI-version>-windows_x64.exe" for **Windows 64-bit** on Intel x64 processor.

This installation procedure builds a proper MpCCI software environment under **Windows**.

You pick up those components, e.g. code adapters, you want to install from the components list.

You can provide the license server information, i.e. hostname and port number of your license server.
During the installation the prerequisites are tested and you have the chance to install the missing third party software (MpCCI-RSH, OpenSSH) on the fly. If you need to install Perl follow the provided link which leads you to the Strawberry Perl site and install it (64-bit) apart from the MpCCI installer hereafter (see \( \triangleright \) 9 Installing Perl \( \triangleright \)).

The MpCCI software is now installed on your system.

There is no need for any further configuration. The environment is automatically updated by the installer. If the installer has not been able to set the path to MpCCI automatically, please set it manually as described in the previous section.

⚠️ Installation as normal user may show for the first usage some request to enable the firewall settings on the machine. If this dialog is ignored, the tools can still work properly. Otherwise contact your local administrator to include the MpCCI executable from the bin/windows_x64 folder in the firewall rules: mpcci_visualizer.exe, mpcci_ccvxcat.exe, mpcci_test.exe, mpcci_server.exe, mpcci_fsmapper.exe, mpcci_morpher.exe, mpcci_modelcheck.exe, mpcci_nbhsearch.exe, mpcci_nbhcontrol.exe
4 Immediately After the Installation - Quick Installation Tests without a License

4.1 Your Home Directory under Windows

Under Linux your home directory is known to applications via the environment variable HOME, which is always defined.

Under Windows the environment variable HOME may be defined, however the standard under Windows is to use the variable USERPROFILE.

To make Windows behave like a Linux and vice versa MpCCI ignores any existing HOME variable and automatically (re)defines the variable HOME identical to either

- HOME=%USERPROFILE%
- HOME=%HOMEDRIVE%\%HOMEPATH%
- HOME=C:\

At least HOMEDRIVE and HOMEPATH should always be defined on Windows.

Your home directory is herein after referred to as <Home>.

- Under Windows MpCCI strictly ignores any environment variable HOME. MpCCI assigns the variable HOME.
- You need to have read-write-execute access rights for your <Home> directory.

4.2 Testing the MpCCI Working Environment and Perl

The first command you should execute is

```
mpcci env
```

If under Windows the OpenSSH and MpCCI-RSH are not installed, MpCCI will then do the post-installations. Please follow the installation instructions.

- If you are working under Windows, you need to have administrative rights.
- `mpcci env` should not fail and the output then shows the system information collected by MpCCI at runtime.
- If you are running under Windows and a virus scanner is installed, the `mpcci env` may be quite slow at the first execution. During the collection of system information a lot of files are visited and executed and the virus scanner has to update it’s cache for the first time. Once executed there will be no major delay any longer.

Testing whether Perl Can Really Compile All MpCCI Modules

Type in the command

```
mpcci test -modload
```

The last output line you should see is:

```
Successfully loaded and compiled all Perl modules from the above list.
```
Unfortunately, newer Perl releases do not contain some Perl modules any longer, you have to install theses modules additionally. For the normal use of MpCCI you will not need these modules - so you can skip this additional installation and this test.

Other Possible Tests

You may start further tests which do not require an MpCCI license. The command

```
mpcci test
```

will help you to test the MpCCI access and communication with remote systems. For help please just type `mpcci help test`.

4.3 Testing whether MpCCI Finds Your Simulation Codes

Depending on the code adapters you installed together with MpCCI you may now test whether MpCCI is able to find your code installations.

Please type just

```
mpcci
```

and check what codes are listed.

You may see the output

<table>
<thead>
<tr>
<th>Subcommands:</th>
<th>Tools related to ANSYS.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Tools related to Abaqus.</td>
</tr>
<tr>
<td>Abaqus</td>
<td>Tools related to Adams.</td>
</tr>
<tr>
<td>Adams</td>
<td>Tools related to FINEOpen.</td>
</tr>
<tr>
<td>FINEOpen</td>
<td>Tools related to FINETurbo.</td>
</tr>
<tr>
<td>FINETurbo</td>
<td>Tools related to FLUENT.</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Tools related to FloMASTER.</td>
</tr>
<tr>
<td>FloMASTER</td>
<td>Tools related to IcePak.</td>
</tr>
<tr>
<td>IcePak</td>
<td>Tools related to JMAG.</td>
</tr>
<tr>
<td>JMAG</td>
<td>Tools related to MATLAB.</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Tools related to MSC.Marc.</td>
</tr>
<tr>
<td>MSC.Marc</td>
<td>Tools related to MSC.Nastran.</td>
</tr>
<tr>
<td>MSC.Nastran</td>
<td>Tools related to OpenFOAM.</td>
</tr>
<tr>
<td>OpenFOAM</td>
<td>Tools related to SIMPACK.</td>
</tr>
<tr>
<td>SIMPACK</td>
<td>Tools related to STAR-CCM+.</td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td>Tools related to TAITherm.</td>
</tr>
<tr>
<td>TAITherm</td>
<td>Tools related to codename.</td>
</tr>
</tbody>
</table>

For each code listed you may type `mpcci <code name>` to get more help on the code specific commands and options.

To list the releases of a code that MpCCI needs to find on your system please type

```
mpcci <code name> -releases
```

or in abbreviated form `mpcci <code name> rel`.

Under Linux the executables for each code must be in the PATH. Under Windows MpCCI reads the Windows registry entry for a code. You need to have an ordinary installation of this code.

More detailed information may be listed using the command `mpcci <code name> -information`, or in
Non Standard Code Installation

If you are a developer of a simulation code and you would like to test your development version of the code - we assume this is not a standard installation and confuses the part of MpCCI which collects information for your code - you may define an environment variable that lists the names of the executable files.

```
MPCCI_CODEWORD_EXENAMES=executable1:executable2:...
```

Currently this feature is implemented for Abaqus and MSC Marc.
5 Licensing

After the installation of MpCCI you need to acquire a license file from Fraunhofer SCAI.

MpCCI uses the FlexNet Publisher based floating license mechanism (formerly known as FLEXlm). The FlexNet Publisher license server has to be started on the license server host for which you require a license file.

For MpCCI 4.6 you will need the current FlexNet Publisher 11.16 licensing software.

If you are working with a previous MpCCI release you will have to stop the running FlexNet Publisher license server and to start the new license server. You will not need a new license file if your license file is valid for some time.

The FlexNet Publisher license server requires the Linux Standard Base (LSB) package installation for any Linux distribution if not already installed. If this package is not installed, the license tools executable may not be recognized by the operating system. You will receive such error message: No such file or directory when you try to execute directly the license server daemon SVD for example (cf. \textsection 5.2.1 Troubleshooting License Start on Linux \textsection). If your license server is running on a Windows system and you try to get a license from a Linux system without the Linux Standard Base (LSB) package you will receive such error message: Failed to get information from license server - Executable lmutil exited with code 255. (cf. \textsection 5.2.2 Troubleshooting Getting License on Linux from a Windows System \textsection).

5.1 Request for a License File

We need two pieces of information from you to generate a license file:

1. Information about your license server host

2. Information about your desired MpCCI license features

First decide on which host you would like to run the license server. This could be any host in a network or in case of a local installation it is your local machine. On this computer please type in the command 

\texttt{mpcci license -sysid} or abbreviated \texttt{mpcci lic sys} You should see an output line like

\begin{verbatim}
SERVER hostname 00087519576d 47000
\end{verbatim}

A FlexNet Publisher license is bound to the MAC address of your network device. If you have multiple network devices installed (Ethernet card, Wireless LAN, Docking Station on a Notebook) you may see multiple hostids. For the MpCCI license server daemon the ID of the permanent integrated ethernet card address is the correct one.

Secondly, we need information about your desired MpCCI license features. MpCCI knows two basic FlexNet Publisher features,

\begin{verbatim}
mpcci_sessions
mpcci_clients
\end{verbatim}

which define how many independent coupled sessions (\texttt{mpcci_sessions}) can be run simultaneously and how many MpCCI coupling clients are allowed in the sum of all simultaneous MpCCI coupling sessions \texttt{(mpcci_clients)}.

The feature \texttt{mpcci_sessions} will be checked out and decremented by 1 during each single MpCCI session (each click on the Start button of the server in the Go panel of the MpCCI GUI or each call of \texttt{mpcci server} if you start the application manually).
The other feature `mpcci_clients` will be decremented by the number of MpCCI coupling clients of the session. The minimum is 1 for `mpcci_sessions` and 2 for `mpcci_clients`, i.e. you can start MpCCI with e.g. one Abaqus process and one FLUENT process.

With 1 `mpcci_sessions` license you cannot start another coupling job as long as your MpCCI job is running. If you’d like to run two coupled FSI simulations simultaneously you will need a license with at least 2 `mpcci_sessions` and 4 `mpcci_clients`. If you have several FLUENT processes you will need more `mpcci_clients` accordingly.

In addition, there are MpCCI license features for the supported code adapters: if you couple code `mycode` using theMpCCI adapter for this code, MpCCI will look for a feature `mpcci_adapter_mycode` on the license server. This feature must be contained in the license file, but it won’t be checked out or decremented.

Examples:

```
mpcci_adapter_abaqus
mpcci_adapter_ansys
mpcci_adapter_fluent
mpcci_adapter_starcd
```

The MpCCI Grid Morpher is also an extra licensed product and you need a license feature

`mpcci_morpher`

The MpCCI FSIMapper is also an extra licensed product and you need a license feature

`mpcci_fsimapper`

Please go to the download area at [www.mpcci.de](http://www.mpcci.de) to get a license file.

If you already have an account for the download area please use the License Request Form Sheet on the Fraunhofer SCAI download area at [www.mpcci.de](http://www.mpcci.de):

1. Go to Download Area.
2. Click the link Log into the MpCCI Download Area which redirects to the Fraunhofer SCAI download portal.
3. Enter your E-Mail and Password.
4. Go to the menu MpCCI and select the MpCCI CouplingEnvironment as product.
5. Then go to License Request

Please fill out the form with all required data and submit. You will receive the required license file soon after via e-mail.

### 5.2 Installing and Activating a License

Please copy your received license file into the file "<MCCI_home>/license/mpcci.lic".

Then please start the FlexNet Publisher license server, i.e. the server daemon and the SVD vendor daemon on the license server machine with the command

```
mpcci license -start or abbreviated: mpcci lic start
```

Under Windows you need to have the write permission in the "<MCCI_home>/license/" directory for the log file.

After the daemons were successfully started you can list the available licenses with the command

```
mpcci license -local or abbreviated: mpcci lic loc
```
If the license server is running with a valid license you will see an output e.g. like this:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Total</th>
<th>Used</th>
<th>Free</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpcci_sessions</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>mpcci_clients</td>
<td>10</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>mpcci_adapter_abaqus</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>mpcci_adapter_fluent</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>mpcci_adapter_.....</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>mpcci_adapter_.....</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

5.2.1 Troubleshooting License Start on Linux

The FlexNet Publisher based license software coming along with MpCCI makes use of the Linux Standard Base (LSB) executable format and does require a present LSB library installation. You can test if your system provides an LSB system installation with the command

```
lsb_release -a
```

If the LSB compatibility is not installed the FlexNet Publisher binaries cannot be executed resulting in the system error message

```
mpcci license:
Failed to launch executable
```

Minimal Ubuntu Linux installations do not include lsb_release as it’s not required for core system functionality.

You can upgrade your Linux system with the command

```
sudo apt-get update
sudo apt-get install lsb-release
```

5.2.2 Troubleshooting Getting License on Linux from a Windows System

The FlexNet Publisher based license software coming along with MpCCI makes use of the Linux Standard Base (LSB) executable format and does require a present LSB library installation. You can test if your system provides an LSB system installation (cf. 5.2.1 Troubleshooting License Start on Linux). If the LSB compatibility is not installed the FlexNet Publisher binaries to list or check out a license cannot be executed resulting in the system error message

```
mpcci license:
Failed to get information from license server: 47000@aquila
Executable
  /./MpCCI/4.5.2/license/lnx4_x64/lmutil
exited with code 255.
```

Some Red Hat Linux or Scientific Linux installations do not include lsb_release as it’s not required for core system functionality.

You can upgrade your Linux system with the command

```
sudo apt-get update
sudo apt-get install lsb-release
```
yum install redhat-lsb

5.2.3 Configure a License Manager as Linux Service

On Linux, edit the appropriate boot script, which may be "/etc/rc.boot", "/etc/rc.local", "/etc/rc2.d/Sxxx", "/sbin/rc2.d/Sxxx", etc. Include commands similar to the following.

```
<MpCCI_HOME>/bin/mpcci license -start
```

The `<MpCCI_home>` is the full path of the MpCCI installation. This command will create a log file under the directory "<MpCCI_home>/license" and you have to ensure that the process could write in the directory.

5.2.4 Configure a License Manager as Windows Service

Execute the "lmtools.exe" application from the MpCCI installation directory:

"<MpCCI_home>/license/<MpCCI_arch>/lmtools.exe"

`<MpCCI_arch>` corresponds to the output of the command:

```
mpcci arch
```

- In the Service/License File tab section select the option Configuration using Services.
- Click on the Config Services tab section.
- Enter a service name e.g. MpCCI license manager or MpCCI FLEXlm.
- Select the path of the program "lmgrd.exe" via the Browse button: 
  "<MpCCI_home>/license/<MpCCI_arch>/lmgrd.exe"
- Select the path to the license file "mpcci.lic" via the Browse button: 
  "<MpCCI_home>/license/mpcci.lic"
- Activate the Start Server at Power Up option.
- Activate the Use Services option.
- You can optionally add a log file by providing a file name for the Path to the debug log file option.
  Be sure to select a directory where you have the write permission for the log file otherwise you may omit this option.
- Click on the Save Service button.
- Click on the Start/Stop/Reread tab section.
Select the just configured license service e.g. MpCCI FLEXlm.

Click on the [Start Server] button.

The license server is now running and configured to start at power up.

## 5.3 Defining the License Server

We assume that an MpCCI FlexNet Publisher license server is already running.

Before you start your MpCCI session you have to know the hostname of the license server and the port where it is listening. The default port number is 47000, which can be changed by the administrator of the license server.

The location of the license server is defined via the variable `MPCCI_LICENSE_FILE`.

```
MPCCI_LICENSE_FILE=port@licenseserver
```

before starting any MpCCI session.

MpCCI has built in two mechanisms to check for the `MPCCI_LICENSE_FILE` variable:

First it checks a FlexNet Publisher resource file located in your `<Home>` directory: under Linux this is 
"HOME/.flexlmrc" and under Windows it is "%USERPROFILE%\flexlmrc". MpCCI scans the ".flexlmrc" for a line containing

```
MPCCI_LICENSE_FILE=...
```

Secondly it reads the environment variable `MPCCI_LICENSE_FILE`. Supposed your license server name is `aquila` please set `MPCCI_LICENSE_FILE` as follows:

Bourne shell ("/bin/sh") users

```
MPCCI_LICENSE_FILE=47000@aquila
export MPCCI_LICENSE_FILE
```

Korn shell (ksh) and bash users

```
export MPCCI_LICENSE_FILE=47000@aquila
```

c-shell (csh) and te-shell (tcsh) users
setenv MPCCI_LICENSE_FILE 47000@aquila

and under Windows

set MPCCI_LICENSE_FILE=47000@aquila

The values of both variables from the ".flexlmrc" file and the environment variable MPCCI_LICENSE_FILE are merged.

Recommendations

We recommend to set the MPCCI_LICENSE_FILE environment variable in your resource file ".flexlmrc" for FlexNet Publisher. Environment variables are volatile, the ".flexlmrc" file is not. Environment variables are sometimes not properly defined if a remote MpCCI command is started via rsh or ssh, the ".flexlmrc" file is always accessible.

If you still prefer to use the environment variable MPCCI_LICENSE_FILE instead of the ".flexlmrc" file:

- Under Linux please store the setting in one of your login scripts (".login", ".profile", ".cshrc", ".bashrc", etc.).
- Under Windows please make sure that MPCCI_LICENSE_FILE is a global environment variable for all users.

Sometimes the connection to the FlexNet Publisher license server is too slow and a license check gets a timeout. In this case you can set the MPCCI_LICENSE_TIMEOUT environment variable. Default value is 30 (seconds).

Under Windows you need to be the administrator to define global system wide variables.

5.4 Multiple License Servers

If you use several redundant FlexNet Publisher license servers you will have several entries of the form port@host - one list entry for each license server.

In this case the entries must be separated by a ";" under Linux respectively by a ";;" on Windows. This is an example for Windows:

set MPCCI_LICENSE_FILE=47000@aquila;47000@einstein;47000@zuse;...

5.5 Testing the License Server

After setting your MPCCI_LICENSE_FILE environment you should check whether licensing works.

mpcci license -servers

should list the value of MPCCI_LICENSE_FILE.

The command

mpcci license -mpcci

checks if the license server is running and can be reached from your local machine.
6 Configuring the MpCCI Users Environment

6.1 Accessing Remote Hosts

If all processes of an MpCCI job run on your local machine you may skip the following section or define the environment variable MPCCI_NOREMSH.

General information on running MpCCI in a network is given in V-3.5 Running MpCCI in a Network.

If you intend to distribute the MpCCI job on multiple hosts in a network - the preferred solution - MpCCI needs to execute commands on remote hosts and may copy files to remote hosts. In this case MpCCI uses either the rsh commands (rsh, rcp) or the ssh commands (ssh, scp).

rsh and ssh must be properly configured to avoid any password request when executing commands on remote hosts.

Environment Variable MPCCI_NOREMSH: Tell MpCCI Not to Check for rsh or ssh

If all processes of an MpCCI job run on your local machine you may define the environment variable MPCCI_NOREMSH. A non-empty value of MPCCI_NOREMSH will disable the check.

Environment Variable MPCCI_RSHTYPE: Tell MpCCI to Use Either rsh or ssh

To find out whether to use the rsh command set (rsh, remsh etc., and rcp) or the secure shell commands (ssh, scp) MpCCI inspects the optional environment variable MPCCI_RSHTYPE. A non-empty value of MPCCI_RSHTYPE may either be "rsh" or "ssh". Any other value is invalid and will result in an error.

If MPCCI_RSHTYPE is not defined or empty rsh is used as the default.

rsh Configuration under Linux

You should have an ".rhosts" file - used by rsh and ssh - and additionally may have an ".hosts" file - used by ssh only - located in your <Home> directory. Your "<Home>/.rhosts" just lists - one hostname per line - all remote hosts from which you may log on the local host without a password request. You may test a working rsh environment via

```
rsh hostname ls
```

There should be no password request.

Please make sure that "<Home>/.rhosts" can be accessed from all hosts relevant for MpCCI.

ssh Configuration

If you prefer to use the ssh commands you need to generate your private RSA-keys to avoid a password request. Therefore use the command

```
ssh-keygen
```

Please read the OpenSSH documentation on how to distribute private and public keys to all hosts in a network.

You may test a working ssh environment via

```
ssh hostname ls
```
There should be no password request.

You may further run some tests on your ssh configuration using

```
mpcci ssh [options]
```

For help please enter

```
mpcci help ssh
```

### Testing MpCCI Access to Remote Hosts via rsh and ssh

Even under Windows you should create a "<Home>/.rhosts" file. You may then test the remote access to all host listed with

```
mpcci test .rhosts
```

The test tries to connect to all hosts, via rsh and/or ssh, prints a protocol in case of failures and finally writes a new “hostlist” file containing all successfully tested hosts.

To get a full help on `mpcci test` please enter

```
mpcci help test
```

For each remote host/hostfile listed on the command line it performs the following tasks:

- Test whether the remote hostname can be resolved by the Domain Name Service (DNS).
- Test whether the remote host is alive and reachable and `ping` gets a reply.
- Test possibility of rsh/ssh connections to the remote host.
- Brief test on the MpCCI installation on the remote host from the local host.
- Test server-client connection on ports 47001 and so on.
- Creation of a protocol host-file "mpcci_test.hostlist" which can be used as an MpCCI hostfile (see \(\Rightarrow\) V-3.5.2 Hostlist File \(\langle\)).

### 6.2 Configuring MpCCI via Environment Variables

The behavior of MpCCI can be influenced through environment variables. If MpCCI does not run properly on your system you may be able to fix problems by setting some variables as described in the following. A complete overview of all environment variables used by MpCCI is given in \(\Rightarrow\) V-2.3 Environment and Environment Variables \(\langle\).

**Environment Variable** `MPCCI ARCH`

The variable `MPCCI ARCH` holds the architecture token of MpCCI and is usually set by MpCCI automatically since MpCCI is capable to figure out the platform it is running on. Only if this mechanism fails or you definitely want to try running a different version of MpCCI, you should set this variable.

`MPCCI ARCH` must always contain a valid architecture token as listed in \(\Rightarrow\) II-5 Supported Platforms in MpCCI 4.6 \(\langle\).

Please read \(\Rightarrow\) V-2.3.1 MPCCI_ARCH - Architecture Tokens \(\langle\) for more information.

**Environment Variable** `MPCCI_DEBUG`

If `MPCCI_DEBUG` is set and not false (any value except empty or 0), then detailed logging output is produced to find out some pitfalls in case of failures. This has the same effect as starting MpCCI with the `-debug` option.
Environment Variable **MPCCI_IBATCH**

If **MPCCI_IBATCH** is set and not false (any value except empty or 0), then the MpCCI GUI can be started during an interactive batch session.
7 Testing the MpCCI Installation and Communication

A Simple Test on the Local Machine, “Hello World”

We assume that the above installation step was successful, that you got an MpCCI license file and already started the license server.

The MpCCI distribution contains a “hello world” like example which you should run in order to test the installation. Please run the example via

```
mpcci test -simple
```

First the MpCCI server is started. Then - after some delay - code 1 (test code SINGLE) is started and connects itself to the MpCCI server. Finally after some delay code 2 (test code DOUBLE) is started and connects itself to the MpCCI server. The code 1 will disconnect from the MpCCI server earlier.

Besides the MpCCI Monitor you should see three windows: the two “hello world” application windows and the MpCCI server window.

The resulting output of code SINGLE ends as follows:

```
[...]
Test code SINGLE now sleeping.
QUIT command sent at <time stamp>
Test code SINGLE finished with success.
```

The resulting output of code DOUBLE ends as follows:

```
[...]
Test code DOUBLE now sleeping.
QUIT command sent at <time stamp>
Test code DOUBLE finished with success.
```

The resulting output of MpCCI server ends as follows:

```
[...]  
[MpCCI-SERVER]: ** WARNING ** No client is connected to the server.  
Exiting ...
[MpCCI-SERVER]: *** Finished MpCCI server shutdown.  
[MpCCI-SERVER]: *** Finished <time stamp>
```

If the output looks like the above the test was successful.
8 Troubleshooting

8.1 Secure Shell in General

Avoid any printout in your login script like ".login", ".profile", ".cshrc", ".bashrc", etc. This output may confuse the secure shell commands `scp` and `sftp` and may lead to strange and confusing error messages.

8.2 OpenSSH under Windows

Together with OpenSSH an OpenSSH service daemon is installed. The name of the service is `opensshd`.

If you cannot login from a remote host onto the local Windows PC you may have to restart the service daemon.

For a restart of the `opensshd` service please enter

```
net stop opensshd  : Stop the service
net start opensshd : Start the service
```

in a command shell.

1. You need to have administrative rights to restart a service under Windows.

2. If no remote access is required, you can define the environment variable `MPCCI_NOREMSH` (cf. 6.1 Accessing Remote Hosts).

8.3 rsh, rcp and rlogin under Windows

By default Windows does not have the UNIX rshd or rlogind services installed.

MpCCI comes with its own rshd and rlogind services for Windows. This package includes both, the services and the rsh and rcp commands. Although this software is part of MpCCI, it is not necessarily bound to MpCCI. Therefore the service is - like the OpenSSH - a separate installation under Windows.

After a successful installation of the MpCCI-RSH program (see 2.6 MpCCI-RSH for Windows) under e.g. "C:\Program Files\MpCCI-RSH" you need to

- Install and start the rshd and optional rlogind services (> 8.3 Installing the MpCCI rshd, rlogind Service Processes)
- Prepare your personal remote hosts file and rsh environment (> 8.3 Preparing the .rhosts File and the rsh Environment)

1. If no remote access is required, you can define the environment variable `MPCCI_NOREMSH` (cf. 6.1 Accessing Remote Hosts).

Installing the MpCCI rshd, rlogind Service Processes

If not already done during the installation of the MpCCI-RSH package you need to install the services on Windows.

Change into the installation directory of the MpCCI rsh installation, e.g. "C:\Program Files\MpCCI-RSH\bin". Then type
rshd install
rlogind install

Now both services should be installed and started. The services are automatically restarted after a reboot of your computer.

You may stop and remove the services later on by either using the Windows services panel or you simply type:

rshd remove
rlogind remove

You can start/stop the services by either using the Windows services panel or you simply type:

rshd stop or net stop mpcci_rshd
rshd start or net start mpcci_rshd
rlogind stop or net stop mpcci_rlogind
rlogind start or net start mpcci_rlogind

⚠️ You need to have administrative rights to start or stop a service under Windows.

Preparing the .rhosts File and the rsh Environment

To give a remote user access to your local computer at first you need to create a "%USERPROFILE%\rhosts" file in which the trusted remote hosts and users are listed. This file may be a copy of your Linux "rhosts" file.

This file must be located in your "%USERPROFILE%" directory, which on current Windows machines is at most "C:\Users\myname".

⚠️ If you are a Windows domain account user please make sure that your "%USERPROFILE%" directory is not removed after you logged off the Windows computer. Otherwise the rshd service is not capable to scan the required files before it creates a logon session for you.

The contents of the "rhosts" file is a list of hostnames and user login names, one host per line.

host1 user1 user2 user3 ...
# additional users for host1
host1 myname friend
host2 myself

Empty lines are ignored.

Unlike with Linux rshd at least one user login name per host is required. All host names must be fully qualified names (hostname.netname). IP4 names in dot notation are ignored.

For security reasons - e.g. a trojan horse may append new hosts and users to your "rhosts" file - the rshd and rlogind services do never scan the "rhosts" file directly but instead use an alternative keys file "%USERPROFILE%\rsh\keys".

This file is created - with special access right assigned - from the "rhosts" file by the command:

rsh -makekeys
or
rsh -k
The -k option will not check the hostname from the ".rhosts". If you have some trouble to connect to your local machine because of a DNS name resolution problem this option is recommended to be used.

If you add a new host in the ".rhosts" file you should create the "keys" again.

Any modification of the "keys" invalidates this file for use with rshd.

Do not modify the access rights of "%USERPROFILE%\.rsh\keys".

Now you should be able to execute a remote command, e.g.

```
rsh hostname dir
rsh hostname mpcci env
```

If you get Permission denied during a remote login procedure, please check the following items:

- The file ".rhosts" should contain fully qualified host names.
- The name of the host is not correctly resolved by the machine. You may add an entry corresponding to the IP adress from the machine with its fully qualified host name like:

  10.0.0.1 windows-pc.domain.org

  in the file

  "C:\WINDOWS\system32\drivers\etc\hosts".

- If you change the password, you need to re-run the command `rsh -k` to update the keys file "%USERPROFILE%\.rsh\keys".

- If these tips did not solve the Permission denied issue, you may stop rlogind service process with this command:

```
rlogind stop
```

Then start the rlogind service with debug mode -D by using this command:

```
rlogind start -D
```

This will show you the reason of the Permission denied.

After resolving this issue, please do not forget to stop the rlogind service and just start it without the debug mode -D for security issue.

You may get further help for the rsh command with

```
rsh -help
rsh /?
```

**Remote Shell and the Windows Firewall**

The rshd and rlogind services accept incoming connection requests on the TCP ports 513 and 514.

The rsh command may ask the rshd to open an additional TCP socket on any unused port in the range 1023, 1022, ...514 and the rshd may then connect to the rsh command on any of these port.

During the installation the rshd, rlogind, rsh and rcp programs have been added to the Windows firewall rules to accept incoming and outgoing connection. The listed ports above must not be blocked by the Windows firewall since the applications themselves have been added to the firewall rules.
Remote Shell and the Windows Environment Variable \texttt{PATH}

The \texttt{rshd} and \texttt{rlogind} create the local processes under the account of the logged on user. Therefore all environment variables - called environment block - for this user have to be set up in advance.

Most of the environment variables are simply individual user variables, but some of them, like the \texttt{PATH} environment, are merged from the global system \texttt{PATH} defined for all users and the users individual \texttt{PATH}.

With Windows Vista Microsoft introduced a bug in one of its kernel function calls responsible for the merge. In case the value of the systems \texttt{PATH} environment variable contains a list with more than 1920 characters, not just the merge process fails, but also the final \texttt{PATH} prepared for the user gets clipped at a length of 1024 characters. In fact the users \texttt{PATH} value then contains only the first 1024 characters from the systems \texttt{PATH}.

On most of the Windows machines you will not have a \texttt{PATH} environment variable with more than 1920 characters and the bug will not be recognised.

In case your system \texttt{PATH} has more than 1920 characters you will run into this issue. The \texttt{rshd} and \texttt{rlogind} will recognise this situation and the users \texttt{PATH} environment then is just a copy of the systems \texttt{PATH} environment variable.

Another workaround for the bug is the \texttt{PATH} file 

\begin{verbatim}
C:\Windows\system32 ; C:\Windows ; C:\Windows\System32\Wbem
F:\Perl\site\bin; F:\Perl\bin
F:\MpCCI\DEVELOP\bin
# C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v3.2\bin
C:\Program Files\Common Files\Roxio Shared\DLLShared
C:\Windows\System32\WindowsPowerShell\v1.0
# C:\Program Files\QuickTime\QTSystem
%userprofile%\bin
F:\Program Files\copSSH\bin
C:\Program Files\CMake 2.6\bin
\end{verbatim}

If the file 

\begin{verbatim}
%USERPROFILE%\rsh\PATH
\end{verbatim}

exists the users \texttt{PATH} environment is set up only from the contents of this file. Directories which do not exist are removed from the list. A recursive variable replacement for e.g. a variable like 

\begin{verbatim}
%userprofile%\bin
\end{verbatim}

is implemented.
9 Installing Perl

9.1 Linux

If you need to install or upgrade Perl under Linux please either download the recent Perl release from www.perl.com/download.csp and compile it on your local machine or use a binary distribution.

A binary distribution for all flavors of Linux platforms can be downloaded from www.perl.com/CPAN/ports/index.html.

The installation is straightforward for a Linux system administrator.

9.2 Windows

If you need to install or upgrade Perl under Windows we recommend to use Strawberry Perl because it’s free and it meets our requirements. You also may use the partly free ActivePerl. Please consider their terms of use.

We recommend using a 64 bit version.

The version should be at least Perl 5.8.8. Versions before 5.8.8 had some problems under Windows with signals, fork emulation, the backtick operator and whitespace in filename when executing external commands.

We also recommend using the Microsoft Windows MSI installer version since during the installation the MSI appends the Perl binaries directory to your PATH environment and properly modifies or sets the ".pl" file association in the HKEY_CLASSES_ROOT of the Windows registry.

⚠️ You need admin privileges to install Perl using the Microsoft Windows MSI installer.

9.2.1 Strawberry Perl for Windows

Strawberry Perl is a free of charge Perl for Windows. A 64 bit version is available. The latest version is Perl 5.32 (2020/08/02).

We recommend using the Microsoft Windows MSI installer version but if you don’t have admin privileges or want to install a local Perl version Strawberry Perl also provides ZIP and portable releases.

Follow the link http://strawberryperl.com/ and download your required Perl version.

After downloading the MSI installer distribution file ("strawberry-perl-5.*-64bit.msi"), please execute the Strawberry Perl installation program and follow the instructions. The installation is a typical Microsoft Windows MSI installation and is straightforward.

In case downloading a ZIP or portable release ("strawberry-perl-5.*-64bit.zip" resp. "strawberry-perl-5.*-64bit-portable.zip"), unzip the distribution into your install directory and follow the instructions from the extracted "README.txt" file.

Finally, please go on with 9.2.3 File Name Associations for Perl under Windows ⬤.

⚠️ Please execute at least the "relocation.pl.bat" file.

Otherwise you may face following error message on opening an MpCCI project:
Error opening project file "myproject.csp"!
Can't locate XML/LibXML.pm in @INC \ldots.

### 9.2.2 ActivePerl for Windows

ActivePerl is free only as community edition. Other editions are fee required. A 64 bit version is available. The latest version is ActivePerl 5.28.

You may follow the link [www.activestate.com/Products/ActivePerl](http://www.activestate.com/Products/ActivePerl) to get an overview of the offered ActivePerl distributions.


As mentioned above, we recommend using the Microsoft Windows MSI installer version of ActivePerl.

After downloading the distribution file ("ActivePerl-5.x.x.exe"), please execute the ActivePerl installation program and follow the instructions. The installation is a typical Microsoft Windows MSI installation and is straightforward.

### 9.2.3 File Name Associations for Perl under Windows

Please make sure that the ".pl" filename extension is properly associated with the newest Perl version installed. You may validate this by typing

```
assoc .pl
```

You should see an association pattern like `perl` and use this token as the argument for the `ftype` command

```
ftype perl
```

You should now see a line like

```
"D:\program files\perl\bin\perl.exe" "%1" %*
```

and type in the fully qualified pathname to get the version of Perl:

```
"D:\program files\perl\bin\perl.exe" -version
```
Part IV

Getting Started

Version 4.6.1
## IV Getting Started – Contents

### 1 Multiphysics Computation with MpCCI

1.1 Multiphysics ......................................................... 4
1.2 Solution of Coupled Problems ................................. 4
1.3 Code Coupling with MpCCI ............................... 5

### 2 Setting up a Coupled Simulation

2.1 A Simple Example ............................................. 7
2.2 Model Preparation ............................................. 7
   2.2.1 CFD Model ............................................. 8
   2.2.2 FE Model ............................................. 8
2.3 Starting the MpCCI GUI ....................................... 9
   2.3.1 Parameter Notes in the MpCCI GUI ......................... 9
   2.3.2 Navigating through the MpCCI GUI ....................... 9
2.4 Models Step – Choosing Codes and Model Files .......... 10
2.5 Algorithm Step – Defining the Coupling Algorithm ...... 11
2.6 Regions Step – Defining Coupling Regions and Quantities 12
   2.6.1 Build Coupling Regions ................................ 13
   2.6.2 Define Quantities to be Exchanged ..................... 15
   2.6.3 Assign Defined Quantities to Built Coupling Regions .. 17
2.7 Settings Step – Specifying Further Coupling Options .... 19
2.8 Go Step – Configuring the Application Startup and Running the Coupled Simulation 20
   2.8.1 Setting Runtime Parameters ............................ 20
   2.8.2 Creating a Project File ................................ 20
   2.8.3 Checking the Application Configuration ................. 21
   2.8.4 Starting the Coupled Simulation ....................... 22
   2.8.5 Interrupting the Computation .......................... 26

### 3 Checking the Results

3.1 The MpCCI Visualizer ........................................... 28
3.2 Post-Processing .................................................. 29
1 Multiphysics Computation with MpCCI

1.1 Multiphysics

The purpose of MpCCI is to perform multiphysics simulations. The systems under consideration are known as coupled systems. A coupled system consists of two or more distinct systems. Each system is governed by a characteristic set of differential equations, but both systems share some variables and cannot be solved separately (see also Zienkiewicz and Taylor [2000] for a more precise definition).

\[
\begin{align*}
\text{domain A:} & \\
f(a,b,c,d) &= 0 \\
\text{domain B:} & \\
g(c,d,e,f) &= 0
\end{align*}
\]

Figure 1: Coupled system: Two domains and a coupling region. Both systems share the variables c and d in the coupling region.

Figure 1 shows two coupled systems. In the coupling region, both systems share some variables and the governing equations of both systems must be solved. Depending on the dimension of the coupling region, surface coupling and volume coupling are distinguished. The shared variables (c and d in Figure 1) are called coupling quantities.

Typical multiphysics simulations are:

**Fluid-Structure Interaction (FSI):**
- First system: Fluid flow (Navier-Stokes equations)
- Second system: Solid mechanics (equilibrium)
- Quantities: Pressure (1 → 2), deformation (2 → 1)

**Thermomechanical coupling**
- First system: Solid mechanics (equilibrium)
- Second system: Heat conduction (Fourier’s law)
- Quantities: Temperature (2 → 1), deformation (1 → 2)

**Electrothermal coupling**
- First system: Electrical conduction (Maxwell’s equations)
- Second system: Heat conduction (Fourier’s law)
- Quantities: Temperature/electric conductivity (2 → 1), power loss/Joule heat (1 → 2)

1.2 Solution of Coupled Problems

To find a solution for a coupled problem, all governing equations, which can be combined in a large system, must be solved. The solution in this way is called strong coupling. However, solving a system with strong coupling is often difficult as different approaches are necessary to solve the sub-problems.

An alternative approach is through weak coupling. Here each problem is solved separately and some variables are exchanged and inserted into the equations of the other problem. This procedure usually
yields a less exact solution compared to strong coupling. The advantages of the weak coupling are that the sub-problems can be solved faster than the complete system and that specialized solvers can be used for each.

![Staggered algorithm for solution of a coupled problem.](image)

Figure 2: Staggered algorithm for solution of a coupled problem.

The staggered method is sketched in Figure 2, which is one of the weak coupling approaches of MpCCI: Code A computes one step, sends data to code B, which then computes one step and sends data back to code A and so on. In addition to the staggered approach, MpCCI supports parallel execution of both codes. The selection of the coupling algorithm is described in ▷2.8 Go Step – Configuring the Application Startup and Running the Coupled Simulation ▷.

An important issue is the data exchange. The quantities must be transferred from one code to the other. This process can be divided into two parts:

**Association:** Each point and/or element of the components of a coupling region is linked to a partner in the other system. The process of finding partners is also called neighborhood search, see ▷V-3.3 Data Exchange ▷.

**Interpolation:** The quantities must be transferred to the associated partner on the other mesh. In this process, the different mesh geometries, data distributions and the conservation of fluxes must be considered.

MpCCI fully supports the data exchange between non-conforming meshes, i.e. the meshes of each subsystem can be defined to optimize the solution of the subsystem.

### 1.3 Code Coupling with MpCCI

Running a co-simulation with MpCCI requires the following steps:

**Preparation of Model Files.** Before starting a co-simulation, each domain must be modeled separately, i.e. a model file must be created for each simulation code. The models must contain a definition of the coupling regions. The preparation of a model file depends on the simulation codes; a detailed description is given in the corresponding sections of the Codes Manual.

**Definition of the Coupling Process.** Simulation codes and corresponding model files must be selected. The coupled regions, quantities and a coupling algorithm must be selected, further coupling options can be given. This step is completely supported by the MpCCI GUI.
Running the Co-simulation. After starting the MpCCI server, both coupled codes are started. Each code computes its part of the problem while MpCCI controls the quantity exchange.

Post-processing. After the co-simulation, the results can be analyzed with the post-processing tools of each simulation code, with the MpCCI Visualizer or with general-purpose post-processing tools.

The application of MpCCI requires a good knowledge of the employed simulation codes. Therefore, it is recommended to use those codes for a co-simulation the user has already some experience with.

The data exchange between two simulation codes requires a “code adapter”, which constitutes a “code plug-in” for data transfer. Therefore only codes which are supported by MpCCI can be used. It is also possible to develop special code adapters, more details are given in the Programmers Guide.

An overview of the complete co-simulation process is given in Figure 3.
2 Setting up a Coupled Simulation

2.1 A Simple Example

Let us look at a simple example to clarify the approach, which was described in the preceding chapter. Figure 1 shows a model of a valve which consists of a flexible flap which can be open or closed depending on the direction of the flow. The purpose of the simulation is to investigate the behavior of the valve depending on inlet and outlet pressure.

The computation of the pressure distribution and the flow rates is achieved using Computational Fluid Dynamics (CFD), whereas the deformation of the flap can be computed applying numerical structural mechanics via a Finite Element (FE) code. Each domain, has a significant influence on the other, thus the problems cannot be solved separately. The CFD simulation requires the deformation of the flap as a boundary condition, whereas the FE simulation requires the fluid pressure as external load. Thus during a co-simulation, these quantities must be exchanged as shown in Figure 2.

2.2 Model Preparation

Before starting the coupled simulation, the models must be prepared in each code. In our example, this requires a model of the flow region and a model of the flap structure. Both models are created in the undeformed condition, i.e. the flap is straight. The computation of such a problem requires a CFD-code which can handle moving/deformable meshes.

It is recommended that you keep your FE and CFD model files in separate directories (see Figure 3) because of the following reasons:

- Clear storage and maintenance of simulation data.
- Ensuring that the codes do not overwrite files of other codes, when identical file names are used.
- Simplification of the porting of the analysis when running the simulation on different platforms.
2.2.1 CFD Model

To model the fluid domain, define the geometry in the CFD pre-processor and create a mesh. Figure 4 shows a possible mesh for the fluid domain. (Please note that the displayed mesh is rather coarse and would probably not yield good results).

All boundary conditions must be defined as well. They comprise the flap surface, walls, inlet and outlet in Figure 4. Also for the coupling region boundary conditions may be set - this depends on the CFD code applied, please check the Codes Manual for details.

It is recommended to run the fluid problem on its own before starting a coupled simulation to be sure that the model setup is correct. In the present example the coupling region could be defined as a rigid wall for this purpose. If the model is not appropriate for computations with the CFD code alone, it is most likely that a co-simulation will either not run.

2.2.2 FE Model

Compared to the CFD model which covers the fluid domain, the FE model covers the flap itself. Figure 5 shows a mesh of the flap with quadrilateral elements. The element sizes do not need to correspond to those of the fluid domain in any way as MpCCI can handle non-conforming meshes.

The flap is connected to the top wall, i.e. the top nodes of the mesh must be fixed. The surface of the flap must be defined as a boundary to which the pressure load can be applied in the co-simulation. Although the system will not show any deformation without an external load, it is recommended to check the validity of the mesh and boundary conditions (i.e. the fixed nodes here) by running the FE problem alone and/or by performing a problem check if it is available in the FE code. To obtain a deformation you can replace the fluid forces by a similar load.
2.3 Starting the MpCCI GUI

The MpCCI GUI is started by executing the command `mpcci gui` from a shell console. The MpCCI GUI guides you through the steps to interconnect the FE and CFD models and to configure and run the coupled simulation.

2.3.1 Parameter Notes in the MpCCI GUI

Parameters that need to be set or corrected are given a note in front of their input field. The note consists of an icon with a tooltip for detailed information:

- **Note** for a required value that has not yet been set.

- **⚠ Note** for an incorrect entry. Most parameters offer a fixed range from which only values can be selected that are valid. But some parameters such as filename or hostname can be entered directly. This can lead to an error, e.g. non-existent file or unreachable host.

2.3.2 Navigating through the MpCCI GUI

At the bottom of each step there is the navigation bar, which lists all available configuration steps, with the current step highlighted (see Figure 6). The individual steps can be selected one after the other or specifically. However, a directly selected step can get stuck on a previous step if configurations are required that have not yet been fulfilled. Required steps are marked by a bracketed asterisk (*) behind the step name. These steps must at least be configured and verified in order to run a coupled simulation.

The individual steps are:

1. Models – required step for choosing codes and model files.
2. Algorithm – required step for defining the coupling algorithm.
5. Settings – optional step for specifying further control parameters.
6. Go – required step for configuring the application startup and starting server and codes.
2.4 Models Step – Choosing Codes and Model Files

In the MpCCI Models step, the following steps should be accomplished:

- Select the analysis codes to couple.
- Set some code specific parameters if desired.
- Specify the model files to determine the potential coupling regions.

For each code, Code1 and Code2 (Figure 6), a field for selecting and configuring the analysis code is provided in the MpCCI GUI.

1. Configure some parameters like the CFD version or release.
2. Select the CFD model file by clicking on the Browse button in order to choose the file via the file browser. A scan of the model file is directly executed by MpCCI. The result is displayed next to the selected code whereas the solution type used for the model is also shown in a field below the model file.

For the FE-Code: Select the desired FE-Code - in this case FE - from the list provided by the pull-down menu. After your selection the corresponding parameters and settings are shown underneath (see Figure 6).

1. At first you may change the settings for the FE release or the scan method.
2. Select the FE model file via the file browser by clicking on the Browse button. A scan of the model file is directly executed by MpCCI. The result is displayed next to the selected code whereas the solution type used for the model is also shown in a field below the model file.
3. Select the unit system.
4. If the unit system is set to variable an additional parameter will be shown where you can select the grid length unit.

For each scanned model file the status is assigned to the corresponding analysis code:
if the extraction of the interface regions was successful.

if the scanner encountered problems during the scan which may depend on the parameters set for the analysis code.

By clicking on the status button, you get the information that has been extracted from the model file. If the scanning has failed the error message from the scanner is displayed in a pop-up window.

The solution type extracted by the scanner can have the following values:

- **Transient** for time dependent so called transient problems,
- **Steady state** for time independent so called stationary problems or
- **Undefined** when the problem cannot be specified out of the model file.

Depending on the solution type the code’s analysis type is set in the Algorithm step. With an undefined solution type, the user can choose between a transient and a stationary analysis and must set the appropriate value by hand as described in \(\text{\ref{V-4.5.2 Code Specific Algorithm Settings}}\).

If you changed the models, you may start the scan process again via the **Start Scanner** button at the bottom of each code field.

After performing a successful scan of all model files you may continue with the next step of the coupled simulation setup by clicking on the **Algorithm** button.

### 2.5 Algorithm Step – Defining the Coupling Algorithm

In the MpCCI Algorithm step you define the coupling algorithm. It is arranged in the two groups

- **Common Basics** with basic settings which are valid for the whole application and
- **Code Specifics** with code specific settings each shown in a titled frame (see Figure 7).

![Figure 7: Algorithm step](image)

The common basic algorithm settings define the global analysis type depending on the analysis type of each code, the coupling scheme - explicit or implicit, the type of the coupling step size, the coupling algorithm...
- parallel or serial and the coupling duration.

The code specific algorithm settings define the analysis type - transient or steady state - depending on the code’s model, solver specific settings, the coupling steps if they are not defined globally and the runtime of each code.

Some settings may be predefined by the current model and so will automatically be set and maybe not changeable.

For some settings there is brief information, which is represented by the symbol (e.g. Analysis type in Figure 7). If you click this, a dialog window opens with a short description. For a detailed description of the Algorithm step have a look at \( V-4.5 \) Algorithm Step.<

In our coupled valve problem we consider that the pressure will initiate the deformation of the flap. Therefore the CFD code will be the initiator and the FE code in our example will request the pressure solution before starting its computation. Both codes run the analysis computation in parallel until a coupled target time is reached. At this target time both analysis codes will exchange their solution quantities.

The Common Basics settings for this scenario are (see Figure 7):

**Analysis type** Automatically set to Transient because of the chosen models which are both transient.

**Coupling scheme** Set to Explicit for our analysis.

**Type of coupling step size** Select Defined by each code because each code has its own coupled target time. Another type of coupling step size would be sending the coupling step size by a global variable (Sent by code). But we haven’t defined such a variable.

**Order of interpolation** Because the coupling times of the codes won’t match, the exchanged values must be interpolated. MpCCI offers different interpolation orders. We recommend the Higher order interpolation to get the best approximated values.

**Coupling algorithm** Set the Parallel algorithm type, select Use initialization and choose CFD code as initializing code.

For the CFD and FE codes no more Code Specifics settings have to be done. The solver resp. coupling step size is defined by each model and coupling will be done after each solver step. No pre-coupling time is needed.

Now the coupling regions can be determined in the next step. Please click on the Regions button to continue.

### 2.6 Regions Step – Defining Coupling Regions and Quantities

In the MpCCI Regions step the following steps need to be done:

- Build the coupling regions.
- Define the quantities to be exchanged.
- Assign the defined quantities to the built coupling regions.

The Regions step offers configuration panels for the two basic types of components:

**Global Variables** These components are data structures that are not related to the CFD or FEM grids. They contain global quantities like time or time step size. These components can be found under the Global element type label.

**Mesh Based Element Components** These components comprise collections of mesh based elements and can be found under the Mesh label. They contain model parts and the related grid based quantities, e.g. nodal positions, heat values or forces. To build up the interconnection, elements should be
gathered that are part of the coupling region. Therefore the components of the codes are collected in lists of zero-dimensional integration point elements, zero-dimensional point elements, one-dimensional line elements, two-dimensional face elements and three-dimensional volume elements (see Figure 8).

![Figure 8: Integration Point, Point, Line, Face and Volume element labels](image)

To navigate between the two component types you have to click on the corresponding tab: Global or Mesh. Only active element types are offered (see Figure 9).

![Figure 9: Regions step](image)

In our example only the Mesh label is shown because global variables don’t exist in our model files.

### 2.6.1 Build Coupling Regions

The Build Regions tab is already selected by default. In the Setup tab for setting up regions it shows a Region, Components and Rules area (see Figure 9).

- **Regions** with a list of the created coupling regions.
- **Components** with the lists of code components which may be selected into the coupled components lists, which define the coupling region. Here in the component list only Face elements exist, so only this one list is shown.
- **Rules** with a list of user defined rules for automatic region building. Not used in our example, see V-4.6.5 Automatic Generation of Regions by Rules for more information.

The “Regions” area (Figure 10) already defines the default region Region_1. To change its name use Rename or Rename automatically from the menu which pops up when you click with the right mouse button onto the region name. With Rename you can choose a name by your own, Rename automatically is only available if components are added and renames the region by the components of the selected code.
2 Setting up a Coupled Simulation

IV Getting Started

Figure 10: Regions with context menu

The pop-up menu provides also a Delete entry for deleting coupling regions and an Unused entry for marking regions not to be used in this run. This can be useful for preparing coupling regions for further runs.

To add further coupling regions use the Add button at the top of the regions list.

Figure 11: Empty, Incomplete, Complete and CompleteMore region states and an Unused complete region

Regions may have different states (see Figure 11):

- **Empty** without components
- **Incomplete** not enough components specified
- **Complete** just enough components specified (one component from each code)
- **CompleteMore** more than enough components specified (more than one component from one code)
- **Unused** regions are greyed out.

2.6.1.1 Select Components for Each Interconnected Code

Figure 12: Coupled components selection
In the Components area (Figure 12) a list with available components and one with coupled components for each code is shown. The components used for coupling have to be added to the coupled components list. According to the valve example with the flexible flap this will imply:

For CFD code add the component flap-surface to the coupled list by double clicking on it, using drag & drop or using Add to Region Set from the pop-up menu which is shown by a right click on the desired component (see Figure 12).

For FE code add the component outside in the same way as mentioned above.

Now the region is complete. We can go on with defining and assigning quantities to be exchanged.

### 2.6.2 Define Quantities to be Exchanged

Quantities are handled in so called Quantity Sets. These sets can be generally defined for the coupling dimensions deduced from the built coupling regions. They need not to be assigned to a coupling region, so quantity sets may be prepared for different runs.

Select the Define Quantity Sets tab. It shows areas for Quantity Sets, Coupling Dimensions and Quantities (see Figure 13).

![Figure 13: Definition of quantity sets](image)

**Quantity Sets** with a list of sets with configured quantities to be transferred.

**Coupling Dimensions** offers the coupled dimensions from the previously built regions.

**Quantities** with a list of available quantities which may be selected and configured to be transferred from one application to the other.

The “Quantity Sets” (see Figure 14) area is constructed similar to the regions area before. The default set EmptyQuantitySet already exists. It can be renamed using Rename or Rename automatically from the pop-up menu where Rename automatically is the default and builds a name depending on the quantities added to the set. Thus the name of the quantity set changes every time the included quantities change. The pop-up menu also provides a Delete entry for deleting quantity sets and a Copy entry for copying the selected quantity set. Thereby a new quantity set with the same quantities and their settings will be constructed.

To add further quantity sets use the Add button at the top of the quantity sets list.
Quantity sets may also have different states indicated by the icon in front of the quantity set’s name:

⚠ Needs quantities to be transferred (incomplete).

→ Quantities are specified being transferred in only one direction: one code only sends, the other code only receives (unidirectional).

↔ At least two quantities are specified being transferred bidirectional: each code sends and also receives (bidirectional).

### 2.6.2.1 Select Quantities to be Transferred

In the quantities list, select the quantities to be transferred by clicking on the checkbox near the name of the quantity (Figure 15). In MpCCI the names of the quantities are standardized. In our example we want to transfer pressure and deformation. In MpCCI the pressure is handled by the quantity *RelWallForce* and the deformation by *NPosition* which stands for “nodal position”. For further information on the meaning of quantities see the Quantity Reference in the Appendix and ▷ V-3.1.1 Physical Domains ◁. For our example select and configure the following quantities:

![Figure 15: Pressure configuration](image-url)
RelWallForce for exchanging the pressure (see Figure 15). Sender is CFD code. Its pressure unit is not editable because the unit system for the CFD code is fixed to the “SI” unit system. On the other hand you may select the appropriate unit for the FE code because of the “variable” unit system setting in the Models step (see Figure 6).

NPosition for exchanging the deformation (see Figure 16). Sender is FE code. The unit for the deformation may be set as described above.

The resulting quantity set will be automatically named to RelWallForce<–CFD | NPosition<–FE.

2.6.3 Assign Defined Quantities to Built Coupling Regions

Select the Assign Quantity Sets tab. It shows areas for Regions, For selected regions and Quantity Sets (see Figure 17):

Regions with lists of built regions separated by their coupling dimension.

For selected regions with actions regarding the selected regions in the region area.

Quantity Sets with quantity sets which may be assigned to the regions of the selected coupling dimension.

The list of built regions only shows regions which are in used state and which are complete meaning that enough components are added.

The shown state of the regions now corresponds to the state of quantity sets (see 2.6.2 Define Quantities to be Exchanged):

⚠️ Region needs quantity sets to be assigned.

⇒ Region is complete with unidirectional quantity transfer.

⇔ Region is complete with bidirectional quantity transfer.
Assigning quantity sets is done by selecting regions and then marking the quantity sets which shall be assigned.

The arrow buttons between the regions and quantity sets areas change the view mode:

- Select regions and assign quantity sets to them. This view is also automatically switched to when a region is selected out of the region list.
- Show regions with all marked quantity sets assigned.

### 2.6.3.1 Assign Quantity Set to Coupled Region

In our example we built the coupling region `Region_1` and the quantity set `RelWallForce<–CFD | NPosition<–FE`. Now we assign the quantity set to the coupling region by selecting the region `Region_1` and marking the quantity set `RelWallForce<–CFD | NPosition<–FE` (see Figure 18).

The definition of the coupling interface is completed. For further information have a look at V-4.6 Regions Step <<.

The next step is the Monitors step where you can define some quantities for specific code components to be monitored. For the time being we won’t apply monitors. Therefor please skip the monitors and click on the Settings button to continue.
2.7 Settings Step – Specifying Further Coupling Options

In the MpCCI Settings step you may modify some MpCCI control parameters like

- online monitoring settings,
- attributes of the coupled job,
- parameters for search algorithm,
- switches controlling the output level for debugging or
- tracefile file format.

An overview of all control parameter properties is given in V-4.8 Settings Step a.

The MpCCI GUI provides default values for all properties which can be modified.

The Settings step window displays a tree-like list of parameters on the left. On the right side the corresponding parameters are shown in a table as depicted in Figure 19. This table provides information such as

- the parameter name displayed with an orange background,
- the editable parameter value displayed with a yellow background,
- and a parameter description.

To select a parameter click on its name or its category in the parameters tree-like list. After that edit the value by clicking the cell with the value and select or type in the desired value for this control parameter.

For most cases the default values are appropriate, but for our example you may activate the control parameter for writing orphan level information for the MpCCI Visualizer’s CCVX tracefile format which is Orphans in the category Output \(\rightarrow\) TraceFile \(\rightarrow\) Writers \(\rightarrow\) CCVX. Additionally you may edit the output level (parameter Level in category Output) which tells MpCCI how much output will be written during the simulation process. The default value is minimal output - also available are no output, additional output and maximal output.

Now click on the Go button to proceed to the startup window for the coupled simulation.
2.8 Go Step – Configuring the Application Startup and Running the Coupled Simulation

In the MpCCI Go step each application, MpCCI coupling server included, is shown in its own frame. The frame for the codes is titled with the code’s name and the name of the host where they’re running (see Figure 20). Below the code frames you find an area with buttons controlling the coupled application - Check, Start, Stop and Kill.

Before starting the coupled simulation you have to configure the startup of the applications. For the MpCCI coupling server you usually may retain the default values. For each analysis code you can set some runtime parameters if necessary.

2.8.1 Setting Runtime Parameters

For the CFD code select the Run parallel option. Now the parameters for running the CFD code in parallel are added to the window as shown in Figure 20. Set the number of processes to be applied e.g. to 4. If no hosts are specified the CFD code will run in parallel on the local machine.

For the FE code no more runtime parameters need to be set.

2.8.2 Creating a Project File

Before checking or starting a coupled simulation you have to establish a project by using Save As from the File menu. This is due to the fact that MpCCI needs the project file to generate its input file and to get information about the settings which were defined in the MpCCI GUI.

If you continue without creating a project, a dialog box appears reminding you to assign one.
2.8.3 Checking the Application Configuration

To cross-check the application configuration a \textbf{Check} button is provided (see Figure 20). Following check states can occur:

- \textbf{Check} - Check results for the current configuration are not yet available. This is true if no check for the current project has been done or if the user changed the configuration.
- \textbf{Check} - The project setup is verified.
- \textbf{Check} - The check has warnings but the simulation can be started taking the warnings into account.
- \textbf{Check} - The check failed, so that the simulation cannot be started.

For the general settings it checks

- Whether all required parameters for server and codes are set.
- Whether all files that are specified as parameters are available.
- Whether the selected coupling algorithm configuration is valid and fits to the configuration of the exchanged quantities.
- Whether the optional code specific checkers pass.

At the end of the check a report with the results is shown and a summary of the coupling configuration is displayed as follows:

\begin{verbatim}
========================================================================
server: SUCCESSFUL
========================================================================
Coupling configuration summary:
---------------------------------
Analysis : Transient
Coupling scheme : Explicit
Algorithm : Parallel, CFD initializes
Coupling frequency : Defined by each code

*** Warning:
The first grid information for the coupled coordinates 'NPosition' might not be received
when using a parallel algorithm with initializing code while the time steps of the codes don't match.

server CHECK SUCCESSFUL with warnings!

========================================================================
CFD: SUCCESSFUL
========================================================================
Coupling configuration summary:
---------------------------------
Analysis : Transient
Algorithm : Parallel, initializing
Coupling frequency : Every solver step
Coupling duration : Not predictable
\end{verbatim}
2 Setting up a Coupled Simulation

4.6.1-1

2.8.4 Starting the Coupled Simulation

To start the coupled simulation click on the Start button (see Figure 20). If no check results are available for the current configuration (>2.8.3 Checking the Application Configuration), this at first cross-checks the application. Depending on the check results - if the check was successful or warnings displayed were accepted - the server is started. When the server is up and waiting for the codes the application codes are automatically launched one after the other. A black running symbol at the bottom of each application frame indicates that the application is running (>4.9.4 Status of the Simulation).

After pressing the Start button all settings will be locked. It is also not possible to leave the Go step. Closing the MpCCI GUI lets the coupled application continue but it is not possible to connect the MpCCI GUI to it again.

Now the client codes are started and the coupling server starts the initialization phase as follows:

1. Initial handshaking: the CFD code and the FE code contact the server.

   Waiting for incoming connections on "47010@nemo"
   [MpCCI License] mpcci_adapter_CFD: feature test
   [MpCCI License] mpcci_clients: feature checkout
   Waiting for incoming connections on "47010@nemo"
   [MpCCI License] mpcci_adapter_FE: feature test
   [MpCCI License] mpcci_clients: feature checkout
   Loading code specific mesh and quantity settings for code "CFD":
   [...]

2. Exchange of the interface topology: The coupling server requests the interface topology from each client. The clients send their interface topology.

   Loading code specific mesh and quantity settings for code "CFD":
Code specific quantity properties:

"RelWallForce": Location(CODE), Default(0)
Send : Method(0), Index(0)
Receive : Method(0), Index(-1)

"NPosition": Location(CODE), Default(0)
Send : Method(0), Index(-1)
Receive : Method(0), Index(0)

Code specific parts properties:

"flap-surface": MeshId(1), PartId(1)
Send : "RelWallForce"
Receive: "NPosition"

Code specific mesh scale: 1

Code specific mesh transformation:

1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
Det: 1

Loading code specific mesh and quantity settings for code "FE":

Code specific quantity properties:

"RelWallForce": Location(CODE), Default(0)
Send : Method(0), Index(-1)
Receive : Method(0), Index(0)

"NPosition": Location(CODE), Default(0)
Send : Method(0), Index(0)
Receive : Method(0), Index(-1)

Code specific parts properties:

"outside": MeshId(1), PartId(1)
Send : "NPosition"
Receive: "RelWallForce"

Code specific mesh scale: 1

Code specific mesh transformation:

1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
3. MpCCI coupling server performs neighborhood searches and computes the mapping between CFD code and FE code meshes.

```plaintext
[...] Loaded REL operator library "rel_ml-32d.so". Loaded MAP operator library "map_ml-32d.so". Code/Mesh/Part/Quantities relationships:

Code: "CFD", ID(1), nice(64), clients(1), type(Finite Volume).

Mesh: "CFD/MESH-1", mid(1)
  Coord system: 3D
  Mesh type  : FACE
  Distances  : [0.001 .. 0.00312839]
  Bounding box: [-0.00717758 .. 0.00717758]
               [-0.023 .. 0.025]
               [-0.023441 .. 0.023441]
  Domain size : 0.004896

Send: "RelWallForce"
  Dimension : 1
  Direction : SEND
  Location  : cell
  Default   : 0
  Buffers   : 1

Recv: "NPosition"
  Dimension : 3
  Direction : RECV
  Location  : vertex
  Default   : 0
  Buffers   : 1
  Source    : "FE/MESH-1" -> rel_ml -> map_ml

Part: "CFD/MESH-1/flap-surface", pid(1)
  Coord system : 3D
  Mesh type     : FACE
  NVertices     : 997
  NCells        : 1948
    Type1  : TRIA3
    Ntypes: 1
  Total nodeids : 5844
  Total vertices: 5844
  Distances     : [0.001 .. 0.00312839]
  Bounding box  : [-0.00717758 .. 0.00717758]
                  [-0.023 .. 0.025]
                  [-0.023441 .. 0.023441]
  Domain size  : 0.004896
```
Code: "FE", ID(0), nice(64), clients(1), type(Finite Element).

Mesh: "FE/MESH-1", mid(1)
   Coord system: 3D
   Mesh type : FACE
   Distances : [0.0005 .. 0.0033941]
   Bounding box: [-0.007178 .. 0.00717784]
                  [-0.023 .. 0.025]
                  [-0.0234407 .. 0.023441]
   Domain size : 0.00489598

Send: "NPosition"
   Dimension : 3
   Direction : SEND
   Location : node
   Default : 0
   Buffers : 1

Recv: "RelWallForce"
   Dimension : 1
   Direction : RECV
   Location : element
   Default : 0
   Buffers : 1
   Source : "CFD/MESH-1" -> rel_ml -> map_ml

Part: "FE/MESH-1/outside", pid(1)
   Coord system : 3D
   Mesh type : FACE
   NNodes : 2805
   NEElements : 920
      Type1 : QUAD8
      Ntypes: 1
   Total nodeids : 7360
   Total vertices : 3680
   Distances : [0.0005 .. 0.0033941]
   Bounding box : [-0.007178 .. 0.00717784]
                  [-0.023 .. 0.025]
                  [-0.0234407 .. 0.023441]
   Domain size : 0.00489598

Neighborhood relationship: "FE/MESH-1" -> "CFD/MESH-1"
rel_ml::create_mapmesh(MESH="FE/MESH-1"): new mesh representation.
rel_ml::create_mapmesh(MESH="CFD/MESH-1"): new mesh representation.
rel_ml::execute(MAPLIB_REL=0x8111728): Configuration:
   nodetolerance=8.38e-05
   normaldistance=0.000419
   tangentialdistance=0.000419
   distance=0.000419
   precision=1e-06
   multiplicity=15.5513
   orphaninfo=true
Starting closePoints search loop...
rel_ml::execute(MAPLIB_REL=0x8111728)
   -> 0.24 seconds CPU.
Neighborhood relationship: "CFD/MESH-1" -> "FE/MESH-1"
rel_ml::create_mapmesh(MESH="CFD/MESH-1"): using existing mesh representation.
rel_ml::create_mapmesh(MESH="FE/MESH-1"): using existing mesh representation.
rel_ml::execute(MAPLIB_REL=0x825dd00): Configuration:
   nodetolerance=8.38e-05
   normaldistance=0.000419
   tangentialdistance=0.000419
   distance=0.000419
   precision=1e-06
   multiplicity=15.5513
   orphaninfo=true
Starting closePoints search loop...
rel_ml::execute(MAPLIB_REL=0x825dd00)
   -> 0.08 seconds CPU.

In our example the parallel coupling algorithm is used with CFD as initializing code as shown in Figure 21. Before starting the computation the CFD code sends the surface forces RelWallForce to the FE code (1). Then both codes compute in parallel their solution quantities pressure and deformation until the coupled target time is reached (2). Now the solution quantities are exchanged (3). Possible coupling algorithms are described in detail in V-3.4.1 Coupling Algorithm.

![Figure 21: Parallel coupling algorithm with initializing code](image)

When an application exits, the running button changes to a button which gives access to the information collected from the application output.

### 2.8.5 Interrupting the Computation

While the applications are running, the [Stop] and [Kill] buttons (see Figure 20) are enabled and provide following functionalities:

- **Stop** to terminate all applications by sending a stop signal or by executing the "Stopper" application module if available. The stop call may not immediately take effect, because it depends on the implementation of the stop procedure in the simulation code.
- **Kill** to terminate all applications by sending a kill signal or by executing the "Killer" application module if available.
To modify parameter settings you have to terminate the applications by pressing the \texttt{Kill} button. Otherwise the editing of parameter values is disabled.
3 Checking the Results

3.1 The MpCCI Visualizer

The MpCCI Visualizer is suitable for a quick check whether the coupling process was successful. The coupling region, orphaned nodes and exchanged quantities can be checked to ensure that the specified coupling has really occurred. Here only a short introduction is presented, a concise description of the MpCCI Visualizer is given in V-7 MpCCI Visualizer.

During the coupling process the MpCCI server writes a “tracefile”, i.e. a collection of the exchanged data (see Figure 3 and 2.8 Go Step – Configuring the Application Startup and Running the Coupled Simulation). To obtain a tracefile, the checkbox Use must be selected in the Settings step for the selected Writers. The default name of the tracefile is "mpccirun-0000.ccvx", it can be changed in the Go step of the MpCCI GUI by changing the Job name prefix for job files. This prefix is used to create a directory containing the tracefile. This directory begins with the prefix used, a timestamp and ends with the suffix of the type of writer used, e.g. mpccirun.<TIMESTAMP>.ccvx.

After a simulation the MpCCI Visualizer can be started by selecting Tools → Visualizer from the MpCCI menu or by entering the command mpcci visualize. Tracefiles can even be opened with the MpCCI Visualizer before the completion of a computation.

Example:

```
mpcci visualize mpccirun-0000.ccvx
```

The visualizer starts with the results window, which is depicted in Figure 1. On the left side the data to display can be selected including the exchanged quantities by activating the Result Panel. In Figure 1 the quantity NPosition is additionally displayed as deformation in the window. Both the quantities which are sent, i.e. before the interpolation, and the quantities which are received are displayed. As sent and received data are normally defined at the same locations, both parts can be superimposed or separated by using additional viewports.

Transient analyses consist of several steps, the step number can be selected in the control window.

The MpCCI Visualizer can not replace a post-processing tool, as the tracefile only obtains information from MpCCI, which solely covers the coupling region. Information on other regions of the analysis may be available by creating additional monitors in the Monitor step.
3.2 Post-Processing

During a coupled simulation both codes should write their results to appropriate files. The visualization of these results can be carried through with built-in tools of the simulation codes. Unfortunately the built-in post-processing tools solely allow the visualization of one part of the problem. In our example this means, that you can display the fluid properties with one tool and the structural properties with the other tool. There is general post-processing software, which can read output data from different standardized formats and combine results from different files. Describing these tools is however beyond the scope of this manual.
V User Manual – Contents

1 Introduction
  1.1 Basic Structure of MpCCI .................................................. 8

2 The MpCCI Software Package
  2.1 Introduction ................................................................. 10
  2.2 The MpCCI Home Directory .............................................. 11
  2.3 Environment and Environment Variables ............................. 12
    2.3.1 MPCCI_ARCH - Architecture Tokens .......................... 13
    2.3.2 MPCCI_DEBUG - for Debugging ............................... 14
  2.4 The MpCCI Resource Directory ....................................... 15
  2.5 Temporary Files .......................................................... 16
  2.6 Third Party Software Used by MpCCI ................................ 18
    2.6.1 Perl ................................................................. 18
    2.6.2 Java ............................................................... 18
    2.6.3 Remote Shell and Remote Copy ................................. 18

3 Code Coupling
  3.1 Multiphysics ............................................................. 19
    3.1.1 Physical Domains .................................................. 19
    3.1.2 Coupling Types .................................................... 20
  3.2 Mesh Checks .............................................................. 21
    3.2.1 Mesh Motion Checks .............................................. 22
    3.2.2 Bounding Box Checks .......................................... 22
    3.2.3 Domain Check ...................................................... 22
    3.2.4 Slave Node Mark .................................................. 24
    3.2.5 Pre-Check Mode ................................................... 25
  3.3 Data Exchange ............................................................ 25
    3.3.1 Association .......................................................... 26
    3.3.2 Interpolation ....................................................... 28
    3.3.3 Quantity Relaxation ............................................ 31
    3.3.4 Quantity Operators ............................................. 40
    3.3.5 Quantity Transformation for Mesh Motion .................... 44
    3.3.6 Quantity Transformation for Cyclic Symmetric Meshes ..... 47
    3.3.7 Operation Workflow for a Quantity ........................... 47
  3.4 Coupling Process ......................................................... 48
    3.4.1 Coupling Algorithm ............................................... 49
    3.4.2 Coupling Schemes ................................................ 49
    3.4.3 Coupling with Subcycling ..................................... 50
### 4.6.3 Coupling Components with Different Dimensions

4.6.4 Simplified Selection

4.6.5 Automatic Generation of Regions by Rules

4.6.6 Applying Region Properties

4.6.7 Quantity Specifications

4.6.8 Assigning Preconfigured Quantity Settings

4.6.9 Overview of the Coupled Regions

4.6.10 Requirements

4.7 Monitors Step

4.8 Settings Step

4.8.1 Monitor

4.8.2 Job

4.8.3 Relation Search

4.8.4 Output

4.9 Go Step

4.9.1 Configuring the MpCCI Coupling Server

4.9.2 Checking the Configuration

4.9.3 Starting the Coupled Simulation

4.9.4 Status of the Simulation

4.10 Remote File Browser

4.10.1 File Browser Handling

4.10.2 How to Mount a New File System

### 5 Command Line Interface

5.1 Using the Command Line Interface

5.2 Overview of All Subcommands

5.3 Starting MpCCI

5.3.1 `mpcci fsimapper`

5.3.2 `mpcci gui`

5.3.3 `mpcci logviewer`

5.3.4 `mpcci monitor`

5.3.5 `mpcci visualize`

5.4 MpCCI Tools

5.4.1 `mpcci ccvxcat`

5.4.2 `mpcci cosimpre`

5.4.3 `mpcci morpher`

5.4.4 `mpcci netdevice`

5.4.5 `mpcci observe`

5.4.6 `mpcci xterm`

5.5 Information and Environment
7 MpCCI Visualizer

7.1 Using the MpCCI Visualizer ........................................ 177
  7.1.1 Data Flow .................................................. 177
  7.1.2 Supported Platforms ............................................. 178
  7.1.3 Starting the Monitor ............................................. 178
  7.1.4 Starting the Visualizer ......................................... 178

7.2 MpCCI Visualizer for .ccvx and Online Monitoring ............... 178
  7.2.1 Introduction .................................................. 178
  7.2.2 Main Window .................................................. 179
  7.2.3 Menus and Toolbars .......................................... 179
  7.2.4 Panels ......................................................... 184
  7.2.5 Viewport Area ............................................... 190
  7.2.6 Preferences Viewer Dialog ................................... 193
  7.2.7 Preferences Server Dialog ................................... 195
  7.2.8 Store Animated Files Dialog ................................ 195
  7.2.9 Error Dialog .................................................. 197
  7.2.10 Command Line Parameters ................................... 198

7.3 Frequently Asked Questions ........................................ 199

8 MpCCI Grid Morpher .................................................. 200

8.1 Using the MpCCI Grid Morpher ..................................... 200
  8.1.1 Options Description .......................................... 200
1 Introduction

This user manual shall give an overview of the basic structures and features of MpCCI. All code-specific issues are discussed in the chapters of the Codes Manual.

1.1 Basic Structure of MpCCI

MpCCI is a software environment which enables the exchange of data between the meshes of two simulation codes in a multiphysics simulation. The architecture of this coupling process is shown in Figure 1.

![MpCCI architecture](image)

Figure 1: MpCCI architecture

MpCCI enables a direct communication between the coupled codes by providing adapters for each code. These code-adapters make use of already existing application programming interfaces (APIs) of the simulation tools. This technique allows an easy installation of MpCCI at the end users site without changing the standard installation of the simulation codes.

Since the meshes belonging to different simulation codes are not compatible in general, MpCCI performs an interpolation. In case of parallel codes MpCCI keeps track on the distribution of the domains onto different processes. MpCCI allows the exchange of nearly any kind of data between the coupled codes; e.g.
energy and momentum sources, material properties, mesh definitions, or global quantities. The details of
the data exchange are hidden behind the concise interface of MpCCI.

The MpCCI environment consists of several components:

- **MpCCI code-adapters** allow to adapt MpCCI to commercial codes through their standard code APIs
  without any changes in the source of the simulation code.

- **The MpCCI GUI** provides a comfortable way to define the coupling setup and to start the simulation
  independent of the codes involved in the coupled application, it is described in &gt; 4 Graphical User
  Interface &lt;

- **The MpCCI coupling server** is the “heart” of the MpCCI system. Environment handling, communi-
  cation between the codes, neighborhood computation, and interpolation are part of this kernel.
2 The MpCCI Software Package

2.1 Introduction

MpCCI is not a single executable, but a complex software package, which consists of the following parts:

Coupling server and clients. The “heart” of MpCCI, which is responsible for the code coupling process, see \( \triangleright 3 \) Code Coupling \(<\).

MpCCI Visualizer. The MpCCI visualizer is a tool for checking the coupling process. Meshes and transferred quantities can be displayed. See \( \triangleright 7 \) MpCCI Visualizer \(<\) for more information.

Collection of small tools. A variety of small tools is included in the MpCCI software package. These are useful for obtaining information on the computing environment, installation and licensing issues, and job control. A description of these tools is given in \( \triangleright 5 \) Command Line Interface \(<\).

MpCCI Grid Morpher. In some coupled analyses the coupling surface is moving during the simulation. If one of the coupled codes does not provide a morphing tool itself, the MpCCI Grid Morpher can be used, see \( \triangleright 8 \) MpCCI Grid Morpher \(<\).

These parts of MpCCI can basically be accessed in two ways:

- Through the graphical user interface, MpCCI GUI, which offers a convenient way to define a coupled simulation, starting and controlling jobs. Most of the MpCCI tools can be started directly from the MpCCI GUI. An introduction to using the MpCCI GUI can be found in Getting Started. A concise description of all MpCCI GUI features is given in \( \triangleright 4 \) Graphical User Interface \(<\).

- All parts of MpCCI can be run directly from the command line (i.e. by typing commands in a shell in UNIX/Linux or the DOS shell in Windows). Besides starting the main parts of MpCCI, a collection of small tools is offered. These tools can be used to gather important information, to deal with licensing issues and to control the coupling process. All available commands are described in \( \triangleright 5 \) Command Line Interface \(<\).
2.2 The MpCCI Home Directory

MpCCI resides completely within one directory with subdirectories. This MpCCI home directory is referred to as MPCCI_HOME.

This directory is created during the installation of MpCCI. For selection of the path, please see the Installation Guide. The only requirement, which must be met to run MpCCI properly is that the MpCCI executable, which is located in "MPCCI_HOME/bin" must be included in your PATH environment variable.

The MPCCI_HOME directory contains a number of subdirectories:

"bin" The MpCCI binaries, which contains the MpCCI main program "mpcci" or "mpcci.exe" and the MpCCI shell "mpccish" or "mpccish.exe". The MpCCI shell is executed by MpCCI on remote machines.

"codes" contains all code-specific files, in separate subdirectories for each code and one for the MpCCI server. Each subdirectory includes information for the MpCCI GUI ("gui.xcf"), and some Perl scripts ("*.pm") for scanning of input files, starting and stopping the coupled codes. These are called by MpCCI during the coupling process.

"dist" contains distribution information, which is required for patch updates.

"doc" contains the MpCCI documentation, which can be accessed with the mpcci doc command, see 5.5.2 mpcci doc ◄ on page 140

"gui" is the base directory of the MpCCI GUI, containing configuration files and the ".jar" archives of the MpCCI GUI.

"include" contains header files, which are necessary for developing code adapters, see Programmers Guide.

"jre" contains a Java distribution which is required for MpCCI.

"lib" contains libraries, which are necessary for developing code adapters, see Programmers Guide.

"license" contains the license software FlexNet Publisher, which is used by MpCCI. All license files should also be placed in this directory.

"perlmod" contains various Perl modules, mainly small helper tools.

"tutorial" contains the input files needed for trying the examples described in the Tutorial.

Do not edit any of the files in the MPCCI_HOME directory unless you know what you do. Improper changing of files may destroy your MpCCI distribution, which makes a new download necessary.
2.3 Environment and Environment Variables

MpCCI uses environment variables to transport information between subprocesses and processes created on remote hosts. Most of the environment variables are volatile, some are not. You may display the MpCCI related environment variables with the command `mpcci env`.

MpCCI distinguishes two categories of variables:

**Control variables** are named `[MpCCI...]` and may be defined before starting MpCCI to control the behavior of MpCCI. An overview of these variables is given below.

**Internal variables** begin with an underscore `_[MpCCI...]` are volatile environment variables used internally by MpCCI and should not be set by the user. Internal variables which are related to a specific code are also named accordingly, i.e. `[MpCCI_<code name>...]`.

All internal variables are intended for internal use only and are subject to change without notice. Changing these variables may yield malfunction of MpCCI, they should not be used by any external application.

**MpCCI Control Variables**

The MpCCI control variables are:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPCCI_ARCH</td>
<td>Contains the MpCCI architecture token, which is used to identify the platform</td>
<td>‡2.3.1 MPCCI_ARCH - Architecture Tokens ‡</td>
</tr>
<tr>
<td>MPCCI_DEBUG</td>
<td>If defined, MpCCI runs in debug mode</td>
<td>‡2.3.2 MPCCI_DEBUG - for Debugging ‡</td>
</tr>
<tr>
<td>MPCCI_HOSTLIST_FILE</td>
<td>Path to file which contains a list of possible hosts for parallel runs</td>
<td>‡3.5.1 Client-Server Structure of MpCCI ‡</td>
</tr>
<tr>
<td>MPCCI_TINFO</td>
<td>Passed to the code adapter to determine the coupling algorithm</td>
<td>‡VIII-2.4.8.1 MPCCI_TINFO Variables ‡</td>
</tr>
<tr>
<td>MPCCI_LICENSE_FILE</td>
<td>Location of the MpCCI license server</td>
<td>‡III-5.3 Defining the License Server ‡</td>
</tr>
<tr>
<td>MPCCI_LICENSE_TIMEOUT</td>
<td>Timeout for license check</td>
<td>‡III-5.3 Defining the License Server ‡</td>
</tr>
<tr>
<td>MPCCI_REMOTE_HOSTNAME</td>
<td>Name of the host seen by a remote system</td>
<td>‡3.5 Running MpCCI in a Network ‡</td>
</tr>
<tr>
<td>MPCCI_RSHTYPE</td>
<td>Type of remote shell used by MpCCI</td>
<td>‡3.5.3 Remote Shell and Remote Copy ‡</td>
</tr>
<tr>
<td>MPCCI_NETDEVICE</td>
<td>Define a network interface name to be used for the socket communication between the MpCCI server and the coupled simulation codes</td>
<td>‡3.5.1 Client-Server Structure of MpCCI ‡</td>
</tr>
<tr>
<td>MPCCI_NOREMSH</td>
<td>Disable errors if remote shell tools is not available on the local machine if defined</td>
<td></td>
</tr>
<tr>
<td>MPCCI_TIMEOUT</td>
<td>Client connection timeout</td>
<td>‡3.5.1 Client-Server Structure of MpCCI ‡</td>
</tr>
<tr>
<td>MPCCI_TMPDIR</td>
<td>Path to temporary directory</td>
<td>‡2.5 Temporary Files ‡</td>
</tr>
</tbody>
</table>

**System Variables**

In addition to the control variables, MpCCI evaluates some system variables:
General variables (all OS)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JAVA_BINDIR</td>
<td>These variables are evaluated by MpCCI to find the Java installation.</td>
</tr>
<tr>
<td>JAVA_HOME</td>
<td></td>
</tr>
<tr>
<td>JAVA_ROOT</td>
<td></td>
</tr>
<tr>
<td>JDK_HOME</td>
<td></td>
</tr>
<tr>
<td>JRE_HOME</td>
<td></td>
</tr>
<tr>
<td>PATH</td>
<td>The list of directories which is searched for executables. &quot;MPCCI_HOME/bin&quot; must be included in the path.</td>
</tr>
</tbody>
</table>

**Windows variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALLUSERSPROFILE</td>
<td>location of All Users Profile</td>
</tr>
<tr>
<td>COMSPEC</td>
<td>secondary command interpreter</td>
</tr>
<tr>
<td>HOMEDRIVE</td>
<td></td>
</tr>
<tr>
<td>HOMEPATH</td>
<td>These variables define the home directory, which is referred to as HOME in this manual.</td>
</tr>
<tr>
<td>PROCESSOR_ARCHITECTURE</td>
<td>The MpCCI architecture token is defined according to the values of these variables</td>
</tr>
<tr>
<td>PROCESSOR_ARCHITEM6432</td>
<td></td>
</tr>
<tr>
<td>USERPROFILE</td>
<td>Used to determine the home directory of a user on a remote machine.</td>
</tr>
</tbody>
</table>

**UNIX/Linux variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOME</td>
<td>The home directory, which is referred to as HOME in this manual.</td>
</tr>
</tbody>
</table>

2.3.1 **MPCCI_ARCH - Architecture Tokens**

MpCCI uses architecture tokens to identify a platform. A list of architecture tokens and currently supported platforms is given in ▷ II-5 Supported Platforms in MpCCI 4.6 ◁.

Before the setting of this variable is really required you should get a list of all installed MpCCI platforms. The command `mpcci list archs` shows all available MpCCI platforms installed on a file server.

The variable `MPCCI_ARCH` holds the architecture token of MpCCI and is usually set by MpCCI automatically since MpCCI is capable to figure out the platform it is running on. This variable should never be set by the user.

However there may be a situation where this mechanism fails:

- MpCCI is confused to find out the platform
- MpCCI selects a wrong architecture
- MpCCI can not find the installation for your platform

`MPCCI_ARCH` must be set to a valid architecture token.

Examples:

- You are running under AIX 5.3, but only the `aix51_power` is installed. MpCCI will complain: `MPCCI_ARCH=aix51_power` solves your problems since 5.1 software can run under 5.3.
- You have a Linux system on an Itanium processor, however `lnx3_ia64` is not installed: `MPCCI_ARCH=lnx3_x86`
solves your problem since Itanium is 32 bit x86 compatible.

### 2.3.2 `MPCCI_DEBUG` for Debugging

This variable is optional, it needs not be defined.

If `MPCCI_DEBUG` is defined and its value is not false (any value except empty or 0), then detailed logging output is produced to find out some pitfalls in case of failures.

Whenever the `-debug` option appears on the command-line of the `mpcci` command

```
mpcci arg1 arg2 ... -debug ... argn
```

`MPCCI_DEBUG` will be set to 1.
2.4 The MpCCI Resource Directory

For storing permanent files related to your personal account MpCCI uses the subdirectory ".mpcci" within your home directory.

If there is no directory "<Home>/mpcci" MpCCI will create one whenever you start MpCCI.

"<Home>/mpcci" contains the following MpCCI related files:

"mpcci.hosts" A hostfile listing possible remote MpCCI hosts
"tmp" The temporary MpCCI directory

You should avoid to delete this directory once it was created and contains files.
2.5 Temporary Files

At runtime MpCCI needs to store intermediate information and creates temporary scripts - shell scripts under UNIX or '.bat'-files under Microsoft Windows - which wraps commands or are copied from the local host to a remote system via the remote copy commands `rcp` or `scp`. Some of the temporary files are created in the current working directory (where the MpCCI application runs or where the MpCCI server was started) and are removed after successful use, others are kept for the duration of the MpCCI session.

MpCCI maintains a list of these long term temporary files created during a session and tries to remove them at the end. Long living temporary files from the last category are stored per default in the directory "<HOME>/.mpcci/tmp".

In some cases it might happen that MpCCI is killed or dies because of a signal or an exception. To eliminate all temporary files in such a case you may either delete the complete "tmp" directory or use theMpCCI command

```
mpcci clean
```

which removes all temporary files.

Never remove temporary files as long as a simulation is running.

An MpCCI temporary directory will be created whenever you start MpCCI.

Environment Variable `MPCCI_TMPDIR`: Shared HOME and Alternative Directory for Temporary Files

Instead of using the default directory for temporary files you may want to assign a different directory to MpCCI as a temporary directory, e.g.

- "/tmp" or "/var/tmp" under UNIX
- "C:\\WINDOWS\Temp" under Microsoft Windows

There are some reasons to redirect the temporary directory:

Your home directory may be physically located on a remote system. Then ";<Home>/mpcci/tmp" is also located on this remote file-server.

Under UNIX this may be an NFS based directory, which is mounted automatically by the automounter (NFS is the Network File System). Under Microsoft Windows your ";<Home>\mpcci\tmp" may be either on an Microsoft Windows file-server or a UNIX file-server mounted to your Microsoft Windows PC via the Samba tool.

In the latter case accessing temporary files in ";<Home>/mpcci/tmp" should be quite fast, but also NFS may sometime lead to out-of-sync delays for a few seconds. This delay may interrupt process communication if the timeout-limit has been defined with a too small value (of less than 10 seconds).

If your MpCCI job runs into trouble with timeouts and missing files and if you know about the fact that your home directory is on a remote host and mounted automatically, then it would be better to use a temporary directory on a locally mounted disc.

You may specify the pathname of an alternative temporary directory by setting the environment variable

```
MPCCI_TMPDIR=path_to_a_temporary_directory
```

to a valid directory.

Note that you need to have write access to that directory before starting MpCCI.
If the environment variable `MPCCI_TMPDIR` is not defined before you start MpCCI, then MpCCI uses the default "<Home>/.mpcci/tmp" and sets `MPCCI_TMPDIR` automatically.

If the environment variable `MPCCI_TMPDIR` is defined before you start MpCCI, then the value of this variable defines the temporary files directory.
2.6 Third Party Software Used by MpCCI

2.6.1 Perl

A large part of the platform-independent functionality of MpCCI is realized with Perl scripts. Perl is available for all platforms supported by MpCCI and already included in Linux distributions. The installation of Perl is described in ▶III-9 Installing Perl ◀. For more information on Perl please visit the Perl website www.perl.org.

2.6.2 Java

The MpCCI GUI is based on Java. To run the MpCCI GUI a Java virtual machine is required.

2.6.3 Remote Shell and Remote Copy

MpCCI uses the remote shell rsh or secure shell ssh to start processes on remote computers. For copying files rcp or scp are used. See ▶3.5 Running MpCCI in a Network ◀ for a detailed description of remote computing.

On Microsoft Windows systems, no remote shell is included in the operating system, therefore the either the MpCCI-RSH or the OpenSSH or both must be installed for MpCCI, see ▶III-2.6 MpCCI-RSH for Windows ◀, ▶III-2.5 OpenSSH for Windows ◀.
3 Code Coupling

MpCCI is a tool to perform multiphysics computations by coupling independent simulation codes. A general introduction to this approach is already given in IV-1 Multiphysics Computation with MpCCI. Each of the coupled codes is independent. The coupling is achieved by exchanging data in the coupling region. This procedure can be split up into two basic issues: How data is transferred and when data is transferred. The first issue is the topic of 3.3 Data Exchange, possible coupling algorithms, which determine the exchange time, are discussed in 3.4 Coupling Process.

This section only gives a short overview of the methods used in MpCCI and explains the options which can be chosen. The description mainly considers a surface coupling, point, line and volume coupling work analogously.

3.1 Multiphysics

3.1.1 Physical Domains

A physical domain is usually characterized by a set of unknown equations which describe certain properties of a material.

The simulation code can be classified according to the physical domains, i.e. the problems which they can solve. The domains known by MpCCI are:

<table>
<thead>
<tr>
<th>Domain</th>
<th>MpCCI Token</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solid mechanics</td>
<td>SolidStructure, FEM</td>
</tr>
<tr>
<td>Fluid mechanics</td>
<td>Fluid, CFD</td>
</tr>
<tr>
<td>Acoustics</td>
<td>SolidAcoustics</td>
</tr>
<tr>
<td>Fluid pipe systems</td>
<td>Network</td>
</tr>
<tr>
<td>Solid heat transfer</td>
<td>SolidThermal</td>
</tr>
<tr>
<td>Fluid heat transfer</td>
<td>FluidThermal</td>
</tr>
<tr>
<td>Heat radiation</td>
<td>Radiation</td>
</tr>
<tr>
<td>Electromagnetism</td>
<td>ElectroMagnetism</td>
</tr>
<tr>
<td>Multibody dynamics</td>
<td>mbdynamiics</td>
</tr>
</tbody>
</table>

In MpCCI physical domains are rather chosen to fit typical capabilities of simulation codes than to clearly distinguish branches of physics.

Solid Mechanics

Solid mechanics describes the behavior of solid bodies when exposed to external load. Usually deformation, stresses and strains of structures are computed using the Finite Element Method (FEM), thus structural mechanics codes are often simply known as Finite Element codes, although the method is not limited to solid mechanics. The governing equations of solid mechanics problems are the mechanical equilibrium and Newton’s laws of motion. These are completed by material equations which describe the behavior of different materials.

Fluid Mechanics

Fluid mechanics deals with the behavior of fluids, e.g. flows of liquids or gases. A common term for the numerical simulation of fluid mechanics is Computational Fluid Dynamics (CFD), thus fluid mechanics codes are known as CFD-codes. The governing equations are the Navier-Stokes equations and the continuum hypothesis.
**Acoustics**
Acoustics studies sound effects, i.e. wave propagation in solids and fluids. Usually this is coupled with vibration analyses. Physically this is a subbranch of solid or fluid mechanics, but the simulation methods are different.

(The name `SolidAcoustics` in MpCCI refers to solid mechanics codes which can also compute acoustic problems).

**Fluid pipe systems**
Fluid pipe systems provides computational fluid dynamics simulation at a system level.

**Solid Heat Transfer**
Heat transfer in solids is based on heat conduction. Usually boundary temperatures and heat sources are given and the heat distribution shall be computed. The governing law of heat conduction can usually be solved by solid mechanics codes. Heat transfer via radiation is discussed separately below.

**Fluid Heat Transfer**
Heat transfer in fluids differs from that in solids as heat is transferred not only by heat conduction but also by convection. Most fluid codes can also compute heat transfer.

**Heat Radiation**
Heat transfer by radiation is separated from solid and fluid heat transfer because specialized simulation codes exist for this issue. They compute the heat distribution caused by electromagnetic radiation (e.g. infrared light), which is emitted by bodies or fluids.

**Electromagnetism**
Electromagnetism deals with the computation of electric and magnetic fields. This includes the computation of electric currents and resulting forces. The governing equations are Maxwell’s equations.

### 3.1.2 Coupling Types

Among the many possible combinations of physical domains, there are some pairs which are often used in multiphysics simulations together with a typical set of quantities which are exchanged. These combinations of domains and quantities are called coupling types in MpCCI.

The MpCCI GUI offers some predefined coupling types to select, see 4.6.8 Assigning Preconfigured Quantity Settings. Some typical coupling types will be described in the following.

**Fluid-Structure Interaction (FSI)**
In an FSI simulation, usually a structure deforms due to forces caused by a fluid flow while the deformation changes the fluids boundary. The deformation must be transferred to the CFD code, which corresponds to the quantity “Nodal position” (NPosition), while forces are sent from CFD to the structural code, e.g. “Boundary absolute force vector” (WallForce), “Boundary relative force vector” (RelWallForce) or the “Relative pressure” (OverPressure).

CFD codes usually use a reference pressure, i.e. the atmospheric pressure, which is not considered in structural codes. For a coupled simulation it is therefore recommended to transfer the relative pressure only, i.e. RelWallForce or Overpressure instead of WallForce and AbsPressure which include the reference pressure. Please consult the appropriate section of the Codes Manual for more information on the reference pressure in a specific simulation code.

For Fluid-Structure Interaction there are two predefined coupling types, Absolute pressure based Fluid-
Structure Interaction and Gauge pressure based Fluid-Structure Interaction. In some cases a one-way coupling is sufficient, which corresponds to the types One way force mapping and One way pressure mapping.

FSI is used in the following tutorials:
▷ VII-2 Vortex-Induced Vibration of a Thin-Walled Structure ◀
▷ VII-4 Elastic Flap in a Duct ◀
▷ VII-8 Pipe Nozzle ◀

Thermal Coupling
Thermal coupling is applied if heat transfer is to be computed in a system consisting of a structure and a fluid. At the boundary, i.e. the surface of the structure, both share the same temperature and thermal energy is exchanged as heat flux.

There are two basic combinations of quantities which can be exchanged:

• Exchange of temperature (Temperature or WallTemp) and the heat flux (WallHeatFlux). The temperature is typically sent by the structural (or radiation) code, while the fluid code is sender of the heat flux. This corresponds to the predefined coupling type Transient surface heat transfer.

• Exchange of temperature, film temperature (FilmTemp) and the heat coefficient (WallHTCoeff). As above, the wall temperature is sent by the structural code. The fluid code sends the film temperature and wall heat transfer coefficient, a combination which can be directly applied as boundary condition by most structural codes. The heat transfer coefficient \( h \) is computed from the heat flux \( q \) and the difference between the wall temperature \( T_w \) and the adjacent fluid cell temperature \( T_c \) in the adapter of the fluid code:

\[
h = \frac{q}{T_w - T_c}
\]

The coupling types Steady state surface heat transfer and Steady state radiative heat transfer correspond to this procedure.

The second approach is recommended because it is more stable. Thermal coupling is used in the tutorial ▷ VII-5 Exhaust Manifold ◀.

Electrothermal Analysis
The resistive loss of electric currents corresponds to a heat source while the temperature influences the resistivity of conductors. This problem is addressed with an electrothermal analysis, which combines electromagnetism with heat transfer. The quantities which are actually should be exchanged are the electric resistivity and the temperature. However, most heat transfer codes cannot compute the resistivity directly, which is therefore done using user-defined functions. The quantity combination depends on the decision in which code these additional computations are carried through.

A special application in this area is electric arc computation. A simple example is shown in tutorial ▷ VII-6 Busbar System ◀.

3.2 Mesh Checks
After all the codes delivered the mesh information, MpCCI server checks the following information of the coupling regions:

• the mesh id: the coupled mesh must have an identical id.
• the mesh dimension: the coupling dimension type is checked.
• the coordinate system type: Code must work in the same coordinate system.
• the mesh motion type.
• the overlapping and bounding box.
the quantity must have a source mesh for the mapping.

Additional information concerning the defined mesh for the coupled simulation can be activated. These information may be useful to understand the origin values of the quantity on the source target. It helps to detect issue in mesh partition or boundaries definition. The setting options are available in the Job section at the MpCCI GUI Settings step (§4.8.2 Job).</step>

The option usage for DomainCheck and Slaves are described in the following section.

### 3.2.1 Mesh Motion Checks

If the moving mesh transformation within the MpCCI server is enabled (MeshMotion parameter in §4.8.2 Job), MpCCI verifies if the motion component definitions (translation vector, rotation) are matching.

### 3.2.2 Bounding Box Checks

The bounding box checks are started before beginning the neighborhood search. For each of the coupled meshes in a coupling region, the bounding box is computed. This box is a rectangular box with sides parallel to the coordinate axes, which contains the whole mesh. The bounding boxes thus represent the total size of the meshes.

The checks cover such issues for the coupling:

- Grid length issue. User has to check the unit system used.
- Coupling region does not overlap at all.
- No proper overlapping of the coupling region.

If the coupling region does not properly overlap as shown in Figure 1 and the user wants to apply default values on the orphaned region, the user should deactivate the overlap check by deselecting the OverlapCheck parameter in §4.8.2 Job.</step>

![Figure 1: Partial overlapping for the coupling region exceeding 2.5 times the overlapping size in one direction](image)

### 3.2.3 Domain Check

This option provides the capability to find the total number of domains of a mesh, part. This can be activated from the DomainCheck parameter in §4.8.2 Job.</step>
This option can be used to check the quality of the mesh partition. Strap elements of a mesh can be isolated and visualized by activating additionally the option for the Monitor and / or Writers. This feature can be used for understanding some mapping problem especially in the case of partial overlapping meshes. This check is executed for each neighborhood search and is an expensive operation in case of using it with remeshing.

The information output is logged in the MpCCI server log file after the Job information block.

```
[MpCCI-SERVER]: *** Job information:
   Interpolation order : Higher order
   Mesh motion transformation: Disable
   MRF correction : Disable
   Overlap check : Enable
   Domain check : Enable
   Baffle shift : Disable
   Conformal mesh mapping : Disable
   Time tolerance : 0.001000
```

Code/Mesh/Part/Quantities relationships:

Code: "RadTherm", ID(0), nice(64), clients(1), type(Radiation).

Mesh: "RadTherm/MESH-1", mid(1)
   Coord system : 3D
   Mesh type : FACE
   NPoints  : 54053
   Bounding box : [0.0254534 .. 1.84765] [-0.53021 .. 0.530219] [-0.130555 .. 0.308253]
   NFaces : 53167
   Total nodeids : 212238
   Total vertices: 212238
   Distances : [1.0443e-05 .. 0.0161794]
   Mean distance : 0.00611897
   Domain size : 1.45841
   Domain count : 2
      Domain 0 : 53165 faces
      Domain 1 : 2 faces

You can find for each coupled mesh the total number of domains (Domain count) and get the information about the number of elements found for each domain.

In order to visualize the different domains in the MpCCI Monitor you can activate the Domains from >4.8.1 Monitor in MpCCI GUI or later in the MpCCI Monitor by using the server preferences menu dialog (>7.2.7 Preferences Server Dialog). You have also the option to save this information in the tracefile by activating the Domains parameter for the corresponding writer used (>4.8.4 Output).

All of these options require that the DomainCheck in the job section (>4.8.2 Job) is activated.

### 3.2.3.1 Domain Check Handling in Visualizer

To visualize the different domains for a model you should
1. Select one viewport

2. Select the code you would like to investigate

3. Select in the **Settings Panel** a coupled mesh you would like to check. Coupled mesh having the same prefix for example **MESH-2** belong to the same set.

4. From the **Results Panel** activate the **Domains** results.

5. In the legend the range shows the total number of domains found for the full model. You can deactivate the **Automatic range settings** and decrease the maximum value until your model has some red colored parts. then you find the total number of domains on the current mesh.

In the **Figure 2** you can visualize two different domains: number 0 and 1.

![Figure 2: Example of domains visualization](image)

### 3.2.4 Slave Node Mark

For a mesh composed of several partitions it is possible to visualize the nodes relation at the partition boundary. The partitions connection can be identified. For using this feature the parameter **Slaves** in
4.8.1 Monitor of the MpCCI GUI must be activated. To visualize the slave node information for the model you should

1. Select one viewport
2. Select the code you would like to investigate
3. Select in the Settings Panel a coupled mesh you would like to check. Coupled mesh having the same prefix for example MESH-2 belong to the same set.
4. From the Results Panel activate the Slave-Node-Marks results.
5. The legend’s range varies from 0 to 1. Node marks with value zero are not slave from another node. Parent node are grayed out. The default setting will display nodes which are not slave in blue color. To improve the interpretation of the information you can deactivate the Automatic range settings and increase the minimum value to 0.9 to get a range from 0.9 to 1. Then activate the Out of range as undefined and all values lower than 0.9 are grayed out.

In that way to get you can better localize the results. On the right side you can visualize the difference through the data processing. The mesh regions which are colored in red show the connection between the nodes.

3.2.5 Pre-Check Mode

If the PreCheck parameter in 4.8.2 Job is activated, MpCCI performs only a mesh check with a neighborhood search and terminates the coupled simulation. This feature requires an additional license token. The summary of the neighborhood calculation is stored in the log file "mpcci_<job name prefix>_server.log" and the collected coupled regions are saved in ccvx format in the folder "<job name prefix>_<timestamp>ccvx_precheck" (3.7.3 Log Files). In case of having orphans in the coupled region you can use the MpCCI Visualizer to analyze the regions.

3.3 Data Exchange

Data exchange does not mean that values are shared, as would be the case in a monolithic code, but each quantity is determined by one code, which then sends it to the other. In the sending code, the data is defined on a mesh of some kind and shall be transferred to the mesh of the receiving code. These meshes describe the same geometric entity, but typically differ in element size and node location, which is referred to as “nonmatching grids”, see Figure 3.
The exchange procedure can be split up into following steps:

**Association** For each node or element of one grid, the “partners” on the other grid must be found. The data will then be exchanged between associated nodes and/or elements. This process is also called neighborhood search and is usually executed one time during the initialization phase as long as no remeshing event during the coupled computation occurs.

**Interpolation** The data which shall be transferred must be adapted to the target mesh, while e.g. conserving the sum of forces.

Each simulation code may work in its own unit system definition. In order to couple models created in different unit systems, MpCCI internally uses an SI unit system. The MpCCI GUI allows the user to input the unit system used for the model and for the quantities. The grid length unit can also be provided. Additionally different quantity transformations for mesh motions, e.g. for rotating parts, or cyclic symmetric meshes can be defined which need a special treatment of the coupled quantities. For convergence reasons quantities can also be relaxed or even scaled.

### 3.3.1 Association

#### 3.3.1.1 Association between Mesh Based Codes

For each point in the target mesh, the closest element in the source mesh is searched. It results a node-element relation (see Figure 4).

The neighborhood search is based on a Kd-tree implementation for an efficient searching algorithm.

![Figure 4: Node-Element relation for surface / volume coupling](image)

The search is based on different geometrical parameters:

- **Normal distance** and **Tangential distance** (for surface meshes): searching distance for the closest element in normal and tangential direction (Figure 5).
• **Distance** (for volume meshes): searching distance for closest element.
• **Multiplicity**: parameter to control the search distance (Figure 5).
• **Node tolerance**: to find doubly defined nodes for meshes with multiple parts.

![Figure 5: a)Normal distance $d_n$ and tangential distance $d_t$ for a point P b) Multiplicity factor parameter](image)

MpCCI adjusts the neighborhood search parameters depending on the mesh discretization by computing the geometric parameters according to this procedure:

- **Characteristic mesh length**:
  - $l_{\text{min}} = 0.5 \times (l_{\text{minSource}} + l_{\text{minTarget}})$
  - $l_{\text{max}} = \max(l_{\text{maxSource}}, l_{\text{maxTarget}})$
  - $l_{\text{mean}} = 0.5 \times (l_{\text{meanSource}} + l_{\text{meanTarget}})$

- **Node tolerance**: $ntol = \min(l_{\text{minSource}}, l_{\text{minTarget}})/5.0$

- **Precision of neighborhood search**: $prec = (0.9 \times l_{\text{minSource}} + 0.1 \times l_{\text{meanSource}})/20$

- **Normal distance**: $d_n = l_{\text{mean}}$

- **Tangential distance**: $d_t = l_{\text{min}}$

- **Distance**: $vdist = 0.9 \times l_{\text{min}} + 0.1 \times l_{\text{mean}}$

- **Multiplicity**:
  - $mult = \frac{l_{\text{max}}}{l_{\text{min}}}$ and is clipped to 10.0
  - $mult = mult \times mfac$ mfac is the user multiplicity factor.
  - $correction = \max(l_{\text{minSource}}, l_{\text{minTarget}})/l_{\text{min}}$ This correct the averaging from $l_{\text{min}}$ in case of degenerated elements and limits to a maximum value of 2.
  - Final multiplicity factor value: $mult = mult \times correction$

Parameters for association can be globally set for all coupling regions in the **Settings** step of the MpCCI GUI, see "4.8.3 Relation Search". These global parameter properties can be overwritten for selected regions in the **Regions Properties** tab of the **Build Regions in Regions** step, see "4.6.6 Applying Region Properties".

The only parameter to control the neighborhood search is **Multiplicity**. The parameter enables the iterative neighborhood search which leads to a more robust algorithm. The default value is 1.0. If a bigger value is chosen by the user, the searching radius (distance up to which a point is still regarded as a neighbor) will be enlarged (doubled) step-by-step until the given multiplicity of the starting radius is reached.

Another parameter which may be overwritten is **UserNormalDistance**. The provided parameter is expressed in SI unit system. This parameter should only be used if the Multiplicity parameter failed to reduce the number of orphaned nodes. In this case the Multiplicity parameter provided in the MpCCI GUI is used as parameter instead of using the computed edge ratio value from the meshes.
3.3.1.2 Association between a System Network Code and a Mesh Based Code

There is no specific association built for an integration point mapping with a surface mesh. MpCCI requires that only one integration point is paired with a surface mesh. Components under the following dimension icons (Figure 7) could be used together.

![Integration Point and Face element labels](image)

Figure 7: Integration Point and Face element labels

3.3.2 Interpolation

3.3.2.1 Interpolation for a Mesh Based Code

MpCCI offers currently one basic approach for association and interpolation the Shape Function algorithm. Shape function mapping simply interpolates a field using the shape functions. It can map linear functions exactly if linear elements are used; respectively quadratic functions need quadratic elements (i.e., elements with mid-nodes) to be mapped exactly.

3.3.2.2 Field Interpolation

In field interpolation (Figure 8), the values are kept to ensure a conservative transfer. This is used e.g. for pressures, densities or temperature.

In MpCCI some quantities are considered as flux density. This is a different term for field values, thus the interpolation is carried through in the same way.

The interpolation type is already defined for each quantity, see the quantities list in the Appendix.

After the data are mapped a global conservative transfer check is realized by comparing the source integral value to the target integral value. If a mapping deviation of 10% maximum occurs, the values will be scaled within this limit.
3.3.2.3 Orphaned Nodes and Elements

A node is called “orphaned” if it is not associated to an element during the neighborhood search. This also means that this node can not receive any data from the other mesh. Instead, MpCCI sends the default values, which can be defined for each quantity (see 4.6.7.3 Orphans Settings for Mesh Based Components). Besides assigning default values it is possible to extrapolate values (using a diffusion algorithm) from non-orphaned areas. The extrapolated values of the orphans will lie in the range of the lowest and highest values of the bordering non-orphaned areas.

If orphaned nodes are present, they are listed in the tracefile written by coupling server and visible through the MpCCI Monitor during the simulation.

To identify the regions where orphaned nodes appear, it is recommended to use the MpCCI Visualizer, which can display all orphaned nodes in the coupling region, see 7 MpCCI Visualizer. In the MpCCI Visualizer orphans information is associated with different values in the color range legend (Figure 9).

- Non orphaned node is displayed with the assigned value 0 or is grayed out. The gray color means that the complete coupled component does not contain any orphaned node.
- A weakly mapped node gets the value 1.
- All orphan information with a value greater than 1 have to be considered as orphaned and potentially critical for the coupled simulation. Depending on the orphan handling method chosen for each coupled region you may see different maximum range values:
  - By default the orphaned nodes are processed by setting a default constant value. In that case these orphaned nodes are marked with the value 6.
  - For the coupled region with an activated extrapolation option orphaned nodes are marked with the value 2.
MpCCI server prints additional orphans information after the neighborhood search. The total number of orphaned nodes for the mesh pairing is printed both as absolute value and in percent of the mesh size. The information provides a list of:

- fully orphaned nodes
- weakly mapped nodes
- childless elements: represents elements which can not send any data to the other mesh.

### 3.3.2.4 Interpolation for an Integration Point with a Mesh Based Code

The quantities are calculated as depicted exemplary for the field quantity temperature $T$ and the flux quantity mass flow $m$. in Figure 10
3.3 Data Exchange

### 3.3.3 Quantity Relaxation

Relaxation of quantities is a method to stabilize and improve the convergence of the coupled process for some cases. Depending on the coupling scheme, the relaxed coupling iterations refer to:

- **Coupling steps** in explicit steady-state coupling schemes. Here a converged solution is to be achieved at the end of the simulation.
- **Coupling step iterations** in implicit coupling schemes. Here a converged solution is to be achieved at the end of each time step.

Relaxation is only allowed for iterative coupling schemes (▷ 3.4.2 Coupling Schemes), not for explicit transient ones.

The general approach of relaxation is to modify the quantity’s value (in an intelligent way) before it is sent to the partner code. This modification is designed such that the actual quantity value \( Q^k \) and the relaxed quantity value \( Q_{\text{relax}}^k \) are approaching each other during the coupling iterations. In the converged state, where \( Q^k \approx Q_{\text{relax}}^k \), the coupled systems are in equilibrium and the solution is found.

To allow the relaxation to converge, the number of relaxed coupling iterations have to be high enough. These are the maximum number of coupling steps for steady state problems or the maximum number of coupling step iterations in implicit couplings.

The relaxation convergence can be checked in MpCCI by a specified tolerance value \( \epsilon \):

\[
\frac{2 \left( \| Q^k - Q_{\text{relax}}^k \|_\infty \right)}{\| Q^k \|_\infty + \| Q_{\text{relax}}^k \|_\infty} < \epsilon
\]  \hspace{1cm} (1)

**Figure 10:** Exemplary calculation of flux quantity mass flow \( m \) and field quantity temperature \( T \) between 1D and 3D codes.
Parameter Settings in the MpCCI GUI

The relaxation can be activated by applying the Relaxation operator (see 4.6.7.4 Applying Operators to Quantities of Mesh Based Components). Parameter settings for the relaxation convergence are:

<table>
<thead>
<tr>
<th>Meaning in method context</th>
<th>MpCCI GUI parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setting the relaxation convergence check</td>
<td>Check relaxation convergence of operator Relaxation</td>
<td>true</td>
</tr>
<tr>
<td>The tolerance value $\epsilon$ in Eq. (1)</td>
<td>Tolerance value of operator ConvergenceCheck</td>
<td>$[0.0, 1.0[$</td>
</tr>
</tbody>
</table>

The Check relaxation convergence option has to be used in combination with the tolerance value specified in the ConvergenceCheck operator, which must also be selected (see 4.6.7.4 Applying Operators to Quantities of Mesh Based Components).

The relaxation methods offered are:
- **Fixed relaxation** - Under- or over-relaxation with a constant relaxation factor.
- **Ramping** - Special type of relaxation with a linearly increasing or decreasing relaxation factor.
- **Aitken relaxation** - Builds an adaptively calculated relaxation factor using the Aitken method.
- **Quasi-Newton method** - Uses the Anderson Mixing Quasi-Newton method to strongly accelerate and stabilize the convergence of the mapped quantities.

These methods are described below.

### 3.3.3.1 Fixed Relaxation

Instead of sending the current quantity value $Q^{k+1}$, which was mapped to the receiver’s mesh, the following relaxed quantity is sent to the receiver:

$$Q_{\text{relax}}^{k+1} = \alpha \cdot Q^{k+1} + (1 - \alpha) \cdot Q_{\text{relax}}^{k}$$

where

- $\alpha$ is the relaxation factor provided by the user.
- $Q^{k+1}$ is the current mapped value (unrelaxed).
- $Q_{\text{relax}}^{k+1}$ is the relaxed value sent to the receiver.
- $Q_{\text{relax}}^{k}$ is the last relaxed value sent to the receiver.

Under-relaxation – with a value of $\alpha$ smaller than one – might lead to a more stable solution for some problems, e.g. with large quantity spikes (see Figure 11 and Figure 12). Some steady-state couplings might benefit from an over-relaxation ($\alpha > 1$) because this can improve the convergence of some slowly converging problems.

The physical background of the coupled solution needs to be taken into account when looking for an optimal relaxation parameter for a coupled quantity.
3.3.3.2 Ramping

A special kind of relaxation is called **Ramping** in MpCCI. In the coupling iteration $k + 1$, the ramping relaxation factor $\beta^k$ is computed according to the linear ramping function:

$$\beta^k = \alpha + (1 - \alpha) \cdot k$$

### 3.3.3.2 Ramping

**Parameter Settings in the MpCCI GUI**

After applying the Relaxation operator (see 4.6.7.4 Applying Operators to Quantities of Mesh Based Components), fixed relaxation is activated by choosing the relaxation method Fixed.

Additional parameter settings are:

<table>
<thead>
<tr>
<th>Meaning in method context</th>
<th>MpCCI GUI parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>The relaxation factor $\alpha$ in Eq. (2)</td>
<td>Relaxation factor</td>
<td>[0.0, 2.0]</td>
</tr>
</tbody>
</table>
\[
\beta^k = \min(1, \beta_0 + k \cdot \beta_d) \quad \text{for under-relaxation ramping} \tag{3}
\]
\[
\beta^k = \max(1, \beta_0 - k \cdot \beta_d) \quad \text{for over-relaxation ramping} \tag{4}
\]

where

- \( \beta^k \) is the ramping relaxation factor in iteration \( k + 1 \).
- \( \beta_0 \) is the initial ramping relaxation factor provided by the user. For under-relaxation, \( \beta_0 \in [0, 1] \), for over-relaxation, \( \beta_0 \in [1, 2] \).
- \( \beta_d \) is the ramping step value provided by the user, \( \beta_d \in [0.001, 1] \).

If ramping is activated, instead of sending the current quantity value \( Q^{k+1} \), a linear combination of \( Q^{k+1} \) and the last sent relaxed value \( Q_{\text{relax}}^k \) is sent to the receiver. The pre-factors depend on \( \beta^k \) and add up to 1:

\[
Q_{\text{relax}}^{k+1} = \beta^k \cdot Q^{k+1} + (1 - \beta^k) \cdot Q_{\text{relax}}^k = Q_{\text{relax}}^k + \beta^k \cdot (Q^{k+1} - Q_{\text{relax}}^k)
\]

where

- \( Q^{k+1} \) is the current mapped value (unrelaxed).
- \( Q_{\text{relax}}^{k+1} \) is the relaxed value sent to the receiver.
- \( Q_{\text{relax}}^k \) is the last relaxed value sent to the receiver.

Figure 13: Ramping effect in case of oscillating quantity values

Depending on the initial ramping relaxation factor \( \beta_0 \), the relaxation factor is linearly increasing or decreasing over the iteration counter until a value of 1, i.e., no relaxation, is reached. Starting with an \( \beta_0 \) smaller than 1 (which will be increased to 1) leads to under-relaxation whereas starting with an \( \beta_0 \) bigger than 1 (which will be decreased to 1) leads to over-relaxation. In fact, this is identical with a classical
under- (or over-) relaxation, except that the relaxation factor is increasing (or decreasing) with each new iteration until the final value 1, i.e. no relaxation, is reached. The value of the first iteration $Q_{\text{relax}}^0$ is set to the first unrelaxed quantity value $Q^0$. Ramping might help to avoid the exchange of any unrealistic spikes during the first few iterations (see Figure 14).

A ramping is recommended for forces or pressure to avoid initial pressure spikes.

![Figure 14: Ramping effect in case of an initial spike](image)

### Parameter Settings in the MpCCI GUI

- After applying the Relaxation operator (see 4.6.7.4 Applying Operators to Quantities of Mesh Based Components) ramping is activated by choosing the relaxation method Ramping.

  Additional parameter settings are:

<table>
<thead>
<tr>
<th>Meaning in method context</th>
<th>MpCCI GUI parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>The initial ramping relaxation factor $\beta_0$ in Eq. (3) (4)</td>
<td>Initial factor</td>
<td>[0.0, 2.0]</td>
</tr>
<tr>
<td>The ramping step value $\beta_d$ in Eq. (3) (4)</td>
<td>Ramping delta</td>
<td>[0.001, 1]</td>
</tr>
</tbody>
</table>

- There are some parameters that can be set via MpCCI in the Settings step (see 4.8 Settings Step). Following parameters are listed in the Monitor category:

<table>
<thead>
<tr>
<th>Meaning in method context</th>
<th>MpCCI GUI parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plot the used ramping relaxation factor $\beta_k$ (Eq. (3) (4)) in the MpCCI Monitor</td>
<td>Relax</td>
<td>true</td>
</tr>
</tbody>
</table>

### 3.3.3.3 Aitken Relaxation

MpCCI offers the possibility to build an adaptively calculated relaxation factor by Aitken’s method. The optimal relaxation factor for iteration $k + 1$ is built from the following quantity values:

- the last unrelaxed value $Q^k$
- the current unrelaxed value $Q^{k+1}$
• the last relaxed value sent to the receiver \( Q^k_{\text{relax}} \)
• the second last relaxed value sent to the receiver \( Q^{k-1}_{\text{relax}} \)

The computation of the current Aitken factor \( \mu^k \) for \( k > 0 \) is described with the following equations:

\[
\Delta Q^{k+1} = Q^k_{\text{relax}} - Q^{k+1} \quad \Delta Q^k = Q^{k-1}_{\text{relax}} - Q^k
\]

\[
\mu^k = \mu^{k-1} + (\mu^{k-1} - 1) \frac{(\Delta Q^k - \Delta Q^{k+1})^T \Delta Q^{k+1}}{(\Delta Q^k - \Delta Q^{k+1})^2}
\]

For the first iteration \( k = 0 \) the Aitken factor is set to 0.

The relaxation factor is then defined as

\[
\omega^k = 1 - \mu^k
\]

and is used as follows:

\[
Q^{k+1}_{\text{relax}} = \omega^k \cdot Q^{k+1} + (1 - \omega^k) \cdot Q^k_{\text{relax}}
\]

\( Q^{k+1}_{\text{relax}} \) is sent to the receiver instead of \( Q^{k+1} \).

The calculated optimal relaxation factor \( \omega \) is limited to the interval between 0.1 and 1.0 to prevent convergence difficulties. During the first iteration a relaxation with 0.1 is applied because the required values are not available.

**Parameter Settings in the MpCCI GUI**

- After applying the Relaxation operator (see ⊳ 4.6.7.4 Applying Operators to Quantities of Mesh Based Components ◄) Aitken relaxation is activated by choosing the relaxation method Aitken.
  
  No additional parameter settings are necessary.

- There are some parameters that can be set via MpCCI in the Settings step (see ⊳ 4.8 Settings Step ◄). Following parameters are listed in the Monitor category:

<table>
<thead>
<tr>
<th>Meaning in method context</th>
<th>MpCCI GUI parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plot the used relaxation factor ( \omega^k ) (Eq. (5)) in the MpCCI Monitor</td>
<td>Relax</td>
<td>true</td>
</tr>
</tbody>
</table>

### 3.3.3.4 Quasi-Newton Method

The Quasi-Newton method can strongly accelerate and stabilize the convergence of the mapped quantities, cf. Koch [2016].

In a multiphysical coupling, the goal is to achieve an equilibrium between (in general) two coupled codes \( \mathcal{F} \) and \( \mathcal{S} \). Due to the partitioned approach of MpCCI, this is achieved by a consecutive evaluation of the solvers.

Using serial coupling algorithms (described in ⊳ 3.4.1 Coupling Algorithm ◄) this procedure is formulated by the following iteration (within each relaxed coupling iteration)

\[
\begin{align*}
\text{I} : & \quad Q^k_{\mathcal{S}} = \mathcal{S}(Q^k_{\mathcal{F}}) \\
\text{II} : & \quad Q^{k+1}_{\mathcal{F}} = \mathcal{F}(Q^k_{\mathcal{S}})
\end{align*}
\]

where
• $Q^k_F$ is the quantity value sent by solver $F$ in iteration $k$.
• $Q^k_S$ is the result of solver $S$ evaluated at $Q^k_F$ (Eq. I). It is sent to solver $F$ to build the next iteration value.
• $Q^{k+1}_F$ is the result of solver $F$ where $Q^k_S$ has been inserted into the solver equations (Eq. II).

Inserting equation I into II results in the fixed point iteration

$$Q^{k+1}_F = F(S(Q^k_F))$$

In this iteration, only the quantity $Q^k_F$ occurs while $F(S(\cdot))$ corresponds to the serial coupling algorithm. For parallel coupling, all quantities will be relaxed simultaneously, with the fixed point problem as follows:

$$
\begin{bmatrix}
Q^{k+1}_F \\
Q^{k+1}_S
\end{bmatrix}
= 
\begin{bmatrix}
F(S(Q^k_F)) \\
S(F(Q^k_S))
\end{bmatrix}
$$

Generally formulated, the **fixed point iteration** reads

$$Q^{k+1} = C(Q^k)$$

where for parallel coupling $Q$ consists of $Q_F$ and $Q_S$ in a single block.

The coupling equilibrium is reached if the fixed point iteration reaches a convergence state where $Q^{k+1} = Q^k$.

In other words, a quantity $Q$ is to be found that fulfills

$$Q = C(Q)$$

which is also called **fixed point** of $C$.

This is where the Quasi-Newton method derivation starts. Instead of finding the fixed point of $C$, the equation is reformulated as a root problem

$$\mathcal{R}(Q) := C(Q) - Q = 0$$

Applying the well-known Newton iteration to $\mathcal{R}(Q) = 0$ gives the following **standard** relaxation iteration:

$$Q^{k+1}_{\text{relax}} = Q^k_{\text{relax}} - \left[ \frac{\partial \mathcal{R}}{\partial Q}(Q^k_{\text{relax}}) \right]^{-1} \cdot \mathcal{R}(Q^k_{\text{relax}})$$

(6)

$$Q^{k}_{\text{relax}} = Q^k_{\text{relax}} - \left[ \frac{\partial \mathcal{R}}{\partial Q}(Q^k_{\text{relax}}) \right]^{-1} \cdot (C(Q^k_{\text{relax}}) - Q^k_{\text{relax}})$$

(7)

$$Q^{k}_{\text{relax}} = Q^k_{\text{relax}} - \left[ \frac{\partial \mathcal{R}}{\partial Q}(Q^k_{\text{relax}}) \right]^{-1} \cdot (Q^{k+1} - Q^k_{\text{relax}})$$

(8)

where

• $Q^{k+1}$ is the current mapped value (unrelaxed).
• $Q^{k+1}_{\text{relax}}$ is the relaxed value sent to the receiver.
• $Q^k_{\text{relax}}$ is the last relaxed value sent to the receiver.

When applying the Newton iteration to $\mathcal{R}(Q) := Q - C^{-1}(Q) = 0$, the **inverse** relaxation iteration is generated as follows:

$$Q^{k+1}_{\text{relax}} = Q^{k+1} - \left[ \frac{\partial \mathcal{R}}{\partial Q}(Q^k_{\text{relax}}) \right]^{-1} \cdot (Q^{k+1} - Q^k_{\text{relax}})$$

(9)
The value of the first iteration $Q^0_{\text{relax}}$ is set to the first unrelaxed quantity value $Q^0$.

The inverse Jacobian of $R$ (standard method) or of $\bar{R}$ (inverse method) with respect to $Q$ is in general unknown and needs to be approximated (say, by $J^0$). In the Anderson Mixing method, which is the Quasi-Newton method used by MpCCI, they are constructed in the way that $\|J^k - J^0\|$ is minimized under the constraint $J^k V = W$:

$$J^k \approx \left[ \frac{\partial R}{\partial Q}(Q^k_{\text{relax}}) \right]^{-1} = J^0 + (W - J^0 V)(V^T V)^{-1}V^T$$

with the difference matrices

$$W = [Q^1_{\text{relax}} - Q^0_{\text{relax}}, \ldots, Q^k_{\text{relax}} - Q^{k-1}_{\text{relax}}]$$
$$\bar{W} = [Q^1 - Q^0, \ldots, Q^k - Q^{k-1}]$$
$$V = [R^1 - R^0, \ldots, R^k - R^{k-1}]$$

and $R^i = Q^{i+1} - Q^i_{\text{relax}}$. Having $\alpha = (V^T V)^{-1}V^T R^k$, the final equation reads:

$$Q^{k+1}_{\text{relax}} = Q^{k+1}_{\text{relax}} - J^0 (R - V \alpha) - \bar{W} \alpha$$

(10)

where $\alpha$ is calculated using least squares minimization problem. According to this derivation, MpCCI offers three variants of the Anderson Mixing Quasi-Newton method:

- **Standard** with $Q^{k+1}_{\text{relax}}$, $W$ and $J^0 = -\omega \text{ Id}$, $\omega > 0$ in Eq. (10)
- **Inverse** with $Q^{k+1}_{\text{relax}}$, $\bar{W}$ and $J^0 = -\omega \text{ Id}$, $\omega > 0$ in Eq. (10)
- **LeastSquares**, with $Q^{k+1}$, $W$ and $J^0 = 0$ in Eq. (10)

For iterations where the matrices $V$ and $W$ are not built, i.e. $k = 0$, the quantity is relaxed as follows:

$$Q^1_{\text{relax}} = \omega \cdot Q^1 + (1 - \omega) \cdot Q^0_{\text{relax}}$$

(11)

In order to accelerate the relaxation process for transient simulations, the previous time steps could be taken into account for the Quasi-Newton relaxation. Increase in the number of considered previous time steps might accelerate the convergence. However, very large values might lead to include irrelevant information and thus it might slow the convergence. But generally speaking, the choice of number of considered previous time steps is very problem dependent.

Ill-conditioned matrix $V$ might lead to inefficient results for the minimization problem. In order to overcome this issue and enhance the performance of Quasi-Newton relaxation, three filtering methods are applied on $\alpha$, $V$ and $W$, such as the detection and elimination of linear dependent, noisy and unused columns of $V$. The linear dependent and noisy columns have to be eliminated using QR decomposition of $V$:

$$V = QU$$

(12)

The columns with index $i$ in $V$ and $W$ will be ignored under one of the following circumstances:

$$|U_{ii}| < \epsilon_{\text{linear}} |U_{00}|$$

(13)
\[ |U_{ii}| < \epsilon_{\text{noise}} \sqrt{\sum_j U_{ij}^2} \] (14)

The filtering in Eq. (13) will be helpful when the oldest considered iterations have large errors or are irrelevant to the current iteration. It eliminates old linear dependent columns in Quasi-Newton relaxation method for each iteration. Eq. (14) is supposed to ignore noisy iterations in all considered iterations for relaxation. It eliminates noisy columns in Quasi-Newton relaxation method for each iteration. High values of \( \epsilon_{\text{linear}} \) and \( \epsilon_{\text{noise}} \) lead to more eliminations.

More details can be found in Bogaers et al. [2016], Haelterman et al. [2016] and Degroote et al. [2010].

Furthermore, for implicit schemes at the beginning of each time step, corresponding columns of \( V \) and \( W \) will be eliminated from the previous time step, if:

\[ |\alpha_i| < \epsilon_{\text{window}} \] (15)

This filtering is only relevant for transient simulations, when the first iterations of the previous time step did not affect the final iteration. It eliminates first columns in Quasi-Newton relaxation method at the beginning of the new time step. Big values of \( \epsilon_{\text{window}} \) result in a lower number of considered iterations from the last time step. It is irrelevant for steady-state simulations.

The divergence of the Quasi-Newton method is checked by the conditions:

\[ \epsilon_1 \|R^{k+1}\|_\infty > \|R^k\|_\infty \] (16)

\[ |\alpha_k| < \epsilon_2 \] (17)

\[ \|V - \alpha R^k\|_\infty > \epsilon_2 \] (18)

For transient simulations a restart in Quasi-Newton is only allowed when residuals’ slope is positive, where the slope is calculated by least squares fitting method.

Please note:

- The computational cost and the memory consumption of the Anderson Mixing Quasi-Newton method depend linearly on the quantity’s number of degrees of freedom.
- For most multiphysical problems, the Inverse approach converges faster than the Standard approach. For problems with high initial quantity peaks, it is recommended to use the Standard variant since the relaxation iteration is based on the last relaxed quantity value \( Q^k_{\text{relax}} \) (cf. Eq. (8)) and not (in the case of Inverse) on the current solver solution \( Q^{k+1} \) (cf. Eq. (9)).
- The LeastSquares Anderson Mixing method has the lowest computational cost but has strict convergence requirements which may not be fulfilled. If the requirements are not fulfilled, the method will not converge.
- For each iteration, \( Q \) is divided by its maximum value. In case of parallel coupling, the maximum of each quantity is applied to the corresponding part of \( Q \).
- Increase in number of regions might reduce the stability. If there are several regions, it is recommended to use the same relaxation values for all of them.
- In steady-state simulations, matrices \( V \) and \( W \) are automatically restarted when MaxIteration is reached (see section about parameter settings in the MpCCI GUI below).

An example using Quasi-Newton methods is the \( \triangleright \) VII-3 Driven Cavity \( \triangleleft \) tutorial.
Parameter Settings in the MpCCI GUI

- After applying the Relaxation operator (see \textit{\ref{sec:4.6.7.4 Applying Operators to Quantities of Mesh Based Components}}} \textit{a) Quasi-Newton relaxation is activated by choosing the relaxation method Quasi-Newton.}

Additional parameter settings are:

<table>
<thead>
<tr>
<th>Meaning in method context</th>
<th>MpCCI GUI parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variants of the Anderson Mixing Quasi-Newton method in Eq. (10)</td>
<td>Type of Anderson Mixing method</td>
<td>Standard Inverse LeastSquares</td>
</tr>
<tr>
<td>( \omega ) in Eq. (10) and (11)</td>
<td>Omega (( \omega ))</td>
<td>&gt; 0.0</td>
</tr>
<tr>
<td>Number of considered previous time steps for the relaxation (for transient problems only)</td>
<td>Number of reused levels</td>
<td>&gt;= 0</td>
</tr>
</tbody>
</table>

- There are some parameters that can be set via \textit{\ref{sec:4.6.1-1}} \textit{MpCCI in the Settings step (see \textit{\ref{sec:4.8 Settings Step}}} <). Following parameters are listed in the Job.QuasiNewton subcategory:

<table>
<thead>
<tr>
<th>Meaning in method context</th>
<th>MpCCI GUI parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>The maximum column number of ( V ) and ( W ) in Eq. (10) (for steady problems only)</td>
<td>MaxIteration</td>
<td>&gt; 0</td>
</tr>
<tr>
<td>The number of iterations that will be taken into account for calculating the residuals’ slope Zero deactivates the restart. (for transient problems only)</td>
<td>NbResiduals</td>
<td>&gt;= 0</td>
</tr>
<tr>
<td>Value of ( \epsilon_{linear} ) in Eq. (13)</td>
<td>LinearDependencyTol</td>
<td>[0.0, 1.0]</td>
</tr>
<tr>
<td>Value of ( \epsilon_{noise} ) in Eq. (14)</td>
<td>NoiseTol</td>
<td>[0.0, 1.0]</td>
</tr>
<tr>
<td>Value of ( \epsilon_{window} ) in Eq. (15) (for transient problems only)</td>
<td>WindowTol</td>
<td>[0.0, 1.0]</td>
</tr>
<tr>
<td>Divergence factor ( \epsilon_1 ) in Eq. (16)</td>
<td>RestartFactor</td>
<td>[0.01, 1.0]</td>
</tr>
<tr>
<td>Divergence limit ( \epsilon_2 ) in Eq. (17)(18)</td>
<td>RestartLimit</td>
<td>[1.0e^{-4}, 1.0]</td>
</tr>
<tr>
<td>Define whether residuals should be monitored</td>
<td>MonitorResidual</td>
<td>true or false</td>
</tr>
<tr>
<td>Define whether an additional log file with relaxation information should be written</td>
<td>WriteLog</td>
<td>true or false</td>
</tr>
</tbody>
</table>

Restrictions:

- For serial coupling algorithm, the Quasi-Newton relaxation method can be applied to only one code. Moreover, it is not allowed to apply a different relaxation method to one of the remaining quantities.
- For parallel coupling algorithm, the Quasi-Newton parameters Type of Anderson Mixing method, Omega and Number of reused levels must be the same for all coupled quantities within a region.
- Parallel coupling with Quasi-Newton is only possible without initializing code.

3.3.4 Quantity Operators

MpCCI provides some other operators besides relaxation. These concern:

- Enabling the convergence check.
- Disabling conservative mapping correction.
- Defining limits to which exceeding values will be clipped.
- Defining a scaling function.

These operators are described below.

### 3.3.4.1 Convergence Check

Since iterating is computationally expensive, the possibility to stop the iterations adaptively is essential to keep computation times as low as possible. This operator is only relevant for iterative couplings (▷ 3.4.2 Coupling Schemes ◁). When the coupled quantity values remain more or less constant between consecutive data exchanges, the coupled solution cannot be improved by continuing the iterations.

MpCCI monitors the relative difference between two successively coupled quantities. If $Q^k$ and $Q^{k+1}$ are the quantity values for the iterations $k$ and $k + 1$ respectively, it stops iterating when the relative difference between two consecutive quantity values lies beneath the given tolerance, $\epsilon$:

$$2 \frac{||Q^k - Q^{k+1}||_\infty}{||Q^k||_\infty + ||Q^{k+1}||_\infty} < \epsilon \quad (19)$$

### Parameter Settings in theMpCCI GUI

- Convergence check can be activated by applying the ConvergenceCheck operator (see ▷ 4.6.7.4 Applying Operators to Quantities of Mesh Based Components ◁).

Parameter settings are:

<table>
<thead>
<tr>
<th>Meaning in method context</th>
<th>MpCCI GUI parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value $\epsilon$ in Eq. (19)</td>
<td>Tolerance value</td>
<td>$[0.0, 1.0]$</td>
</tr>
</tbody>
</table>

- There are some parameters that can be set via MpCCI in the Settings step (see ▷ 4.8 Settings Step ◁). Following parameters are listed in the specified category:

<table>
<thead>
<tr>
<th>Meaning in method context</th>
<th>MpCCI GUI parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>If the convergence check is assigned to more than one quantity, you may specify whether the convergence of only one quantity should lead to a converged simulation or even all quantities should meet the convergence criterion.</td>
<td>Job.ConvergenceRule</td>
<td>All, One</td>
</tr>
<tr>
<td>Plot the residuum in Eq. (19) in the MpCCI Monitor</td>
<td>Monitor.Inorm</td>
<td>true</td>
</tr>
</tbody>
</table>

### 3.3.4.2 Disable Conservative Mapping Correction

MpCCI corrects by default the mapping error due to the mesh difference between codes, by scaling the mapped quantity, $Q_m$. The scale factor $\gamma$ is calculated as follows:

$$\gamma = \frac{\int_{\Omega_s} Q_s(x)dx}{\int_{\Omega_m} Q_m(x)dx}$$

where $\Omega_s$ and $\Omega_m$ denote the coupled space for sender and receiver codes, respectively and $Q_s$ is the sent quantity before mapping.
Parameter Settings in the MpCCI GUI

The conservative mapping correction can be deactivated by applying the DisableConservation operator (see \ref{4.6.7.4 Applying Operators to Quantities of Mesh Based Components}).

No parameter settings are necessary.

3.3.4.3 Define Limits

Each element $q$ of quantity $Q$ can be clipped as follows:

$$
q = \begin{cases} 
    l_{\text{low}} & \text{for } q < l_{\text{low}} \\
    q & \text{for } l_{\text{low}} \leq q \leq l_{\text{up}} \\
    l_{\text{up}} & \text{for } q > l_{\text{up}}
\end{cases}
$$

This operator exists as pre processor which will be applied before mapping on the sender side and as post processor which will be applied after mapping on the receiver side.

Parameter Settings in the MpCCI GUI

The limits to quantities can be activated by applying the PreLimiter operator (as pre processor) or the PostLimiter operator (as post processor) (see \ref{4.6.7.4 Applying Operators to Quantities of Mesh Based Components}).

Parameter settings are:

<table>
<thead>
<tr>
<th>Meaning in method context</th>
<th>MpCCI GUI parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>The lower limit $l_{\text{low}}$ in Eq. (20)</td>
<td>Lower limit</td>
<td>unlimited</td>
</tr>
<tr>
<td>The upper limit $l_{\text{up}}$ in Eq. (20)</td>
<td>Upper limit</td>
<td>unlimited</td>
</tr>
</tbody>
</table>

3.3.4.4 Scaling

The scaled quantity, $Q_s$, is calculated by ramping the mapped quantity, $Q_m$, using scale factor $\omega$ and reference value $Q_r$ as follows:

$$
Q_s = \omega \cdot Q_m + (1 - \omega) \cdot Q_r
$$

where the scale factor $\omega$ is calculated using initial factor $\omega_i$, final factor $\omega_f$, scale counter $j$, and number of scale factor changes $n_1$:

$$
\omega = \frac{(\omega_f - \omega_i)j}{n_1} + \omega_i
$$

where the scale counter $j$, which counts the scale factor changes and is limited by $n_1$, is calculated using the current step number $\text{StepCounter}$, and the number of consecutive steps with the same scale factor $n_2$ as follows:

$$
j = \min(\lfloor \frac{\text{StepCounter}}{n_2} \rfloor, n_1)
$$
For example, with the following scale parameters definition you will have the scaling factor plotted on Figure 15:

Reference value \( Q_r \): 200
Initial factor \( \omega_i \): 0.1
Final factor \( \omega_f \): 1
Number of consecutive steps with the same scale factor \( n_2 \): 2
Number of scale factor changes \( n_1 \): 3

![Figure 15: Scaling function on quantity](image)

Scaling could also be used, when the initial conditions of one code lead to huge changes in the other code. Please note that for explicit scheme in transient cases, this option might increase inaccuracy of the co-simulation.

### Parameter Settings in the MpCCI GUI

The scaling function can be activated by applying the Scale operator (see 4.6.7.4 Applying Operators to Quantities of Mesh Based Components §).

Parameter settings are:

<table>
<thead>
<tr>
<th>Meaning in method context</th>
<th>MpCCI GUI parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>The reference value ( Q_r ) in Eq. (21)</td>
<td>Reference value</td>
<td>unlimited</td>
</tr>
<tr>
<td>The initial factor ( \omega_i ) in Eq. (22)</td>
<td>Initial scale factor</td>
<td>unlimited</td>
</tr>
<tr>
<td>The final factor ( \omega_f ) in Eq. (22)</td>
<td>Final scale factor</td>
<td>unlimited</td>
</tr>
<tr>
<td>The number of consecutive steps with the same scale factor ( n_2 ) in Eq. (23)</td>
<td>Number of steps with the same scale factor</td>
<td>( &gt; 0 )</td>
</tr>
<tr>
<td>The number of scale factor changes ( n_1 ) in Eq. (22)/(23)</td>
<td>Number of scale factor changes</td>
<td>( &gt; 0 )</td>
</tr>
</tbody>
</table>
3.3.5 Quantity Transformation for Mesh Motion

When defining different mesh motions the MpCCI server should be able to provide the information (coordinates, quantities) in the target calculation frame of the target code. The incoming quantities are transformed into the SI unit system, pre-processed for any mesh motion and mapped. The outgoing quantities are transformed in a reverse way: they are post-processed (ramping, relaxation, mesh motion, etc.) into the target unit system.

3.3.5.1 Mesh Motion

A mesh motion is defined as

- A moving reference frame (MRF type):
  From the simulation code point of view, neither the grid is moving nor the quantity is moving in the inertial system.
  The MpCCI server will execute a transformation (which may be a rotational and/or translational motion) of the mesh and of vector based quantities if the simulation code is transient.

- A moving grid (GRID type):
  From the simulation code point of view, the grid and quantities are moved but the moved grid is not necessarily sent continuously to the MpCCI server. The quantities are already sent from the inertial system.
  The MpCCI server will not apply any transformation (rotation and/or translation) on the quantities.

The simulation, respectively the code adapter defines the mesh motion type used by the model. The mesh motion is defined for each part. The MpCCI server will perform a new neighborhood search at each coupled step if the defined mesh motion types are different, namely a MRF model with a GRID model. Basically data from a MRF transient model are transformed into the inertial system from the MpCCI server.

3.3.5.2 Mesh Motion Application

The principal reason for employing a moving reference frame MRF is to render a problem which is unsteady in the stationary (inertial) frame but steady with respect to the moving frame. It should also be noted that you can run an unsteady simulation in a moving reference frame with constant rotational speed. This would be appropriate if you want to simulate, for example, vortex shedding from a rotating fan blade. The unsteadiness in this case is due to a natural fluid instability (vortex generation) rather than induced from interaction with a stationary component.

When a time-accurate solution (rather than a time-averaged solution) e.g. for a rotor-stator interaction is desired, you must use the moving grid or sliding mesh model to compute the unsteady flow field. The sliding mesh model is the most accurate method for simulating flows in multiple moving reference frames, but also the most computationally demanding. Lately Nonlinear Harmonic Methods which reduce the model size for rotor-stator interactions came into operation.
• Fluid mechanics physical domain

<table>
<thead>
<tr>
<th>Motion type</th>
<th>Application area</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single/Multiple Reference Frame</td>
<td>Steady State approximations</td>
</tr>
<tr>
<td>Mixing Plane or Frozen Rotor</td>
<td>Weak rotor-stator interaction</td>
</tr>
<tr>
<td>Sliding Mesh Model</td>
<td>Initial solution for unsteady flow</td>
</tr>
<tr>
<td>Nonlinear Harmonic Method</td>
<td>Unsteady flow</td>
</tr>
<tr>
<td></td>
<td>Strong rotor-stator interaction</td>
</tr>
<tr>
<td></td>
<td>Vortex shedding</td>
</tr>
</tbody>
</table>

• Solid mechanics physical domain

<table>
<thead>
<tr>
<th>Motion type</th>
<th>Application area</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rotating reference frame</td>
<td>Stress analysis induced from rotation and flow force</td>
</tr>
<tr>
<td></td>
<td>Operational vibrations</td>
</tr>
<tr>
<td>Stationary reference frame</td>
<td>Unsteady flow</td>
</tr>
<tr>
<td></td>
<td>Strong rotor-stator interaction</td>
</tr>
<tr>
<td></td>
<td>Vortex shedding</td>
</tr>
</tbody>
</table>

3.3.5.3 Mesh Motion Combination

Definition of the abbreviations:
• MRF for moving reference frame or rotating reference frame.
• GRID for sliding moving mesh or stationary reference frame.

In the following scenarios some recommendations are provided about the usage of the Job.MeshMotion and Job.MRFCorrect features in the Settings step of MpCCI GUI (▷ 4.8.2 Job ◁).

MRFCorrect option only works if the code is transient. In that case the code using a MRF model will get its data transformed by using a prospective time step size.

Scenario 1

- CFD is steady state, MRF
- CSD is steady state, MRF

<table>
<thead>
<tr>
<th>Motion properties</th>
<th>Use MeshMotion</th>
<th>Use MRFCorrect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identical</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>Different</td>
<td>YES</td>
<td>NO</td>
</tr>
</tbody>
</table>

It is not recommended to use different angular velocities. Such modeling does not fit the targeted physical effect you want to study.

Scenario 2

- CFD is transient, GRID
- CSD is transient, MRF

<table>
<thead>
<tr>
<th>Motion properties</th>
<th>Use MeshMotion</th>
<th>Use MRFCorrect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Different</td>
<td>YES</td>
<td>NO</td>
</tr>
</tbody>
</table>

The mesh motion GRID on CFD side provides an accurate solution of the unsteady flow, rotor-stator interaction.
Scenario 3

- CFD is transient, GRID
- CSD is transient, GRID

<table>
<thead>
<tr>
<th>Motion properties</th>
<th>Use MeshMotion</th>
<th>Use MRFCorrect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identical</td>
<td>NO</td>
<td>NO</td>
</tr>
</tbody>
</table>

3.3.5.4 List of codes implementing the mesh motion definition

**Abaqus** Models defining a rotating reference frame and stationary reference frame are supported.

- **MpCCI** recognizes the keyword `*dload` with the option CENT or CENTRIF as a rotating reference frame.

  The keyword CENT needs some additional editing, because it requires the material density corresponding to the moving part:

  The user can redefine manually the definition of the CENT option into a CENTRIF in the MpCCI GUI Regions step. Before adding the components into the list of coupled components you should provide the CENT using the following syntax by editing the component properties (>4.6.2.2 Editing Component Properties):

  \[\text{ELSET,CENTRIF,omega}^2,x0,y0,z0,ax,ay,az\]

  where \(x0,y0,z0\) are the coordinate of the origin and \(ax,ay,az\) are the rotation axis definition. You should modify the information from the Aux Information field:

  keyword CENT becomes CENTRIF and you should provide \(omega^2\) instead of \(rho*omega^2\) value after the CENTRIF option. This should be repeated for all the components involved with this motion.

- **MpCCI** recognizes the keyword

  `*INITIAL CONDITIONS, TYPE=ROTATING VELOCITY,TYPE=COORDINATES`

  as a moving grid motion.

**FLUENT** Model with a Frame Motion is supported and defined as moving reference frame by MpCCI. MpCCI recognizes the properties defined under the Reference Frame panel.

Model with a Mesh Moving is supported and defined as moving grid by MpCCI. MpCCI recognizes the properties defined under the Mesh Motion Frame panel.

**FINE/Open** Model with a Rotation definition is supported and defined as moving reference frame by MpCCI. MpCCI defines the rotation axis along the z-axis.

**FINE/Turbo** Model defining a rotational speed is defined as moving reference frame by MpCCI. MpCCI defines the rotation axis along the z-axis.

**STAR-CCM+** Model with a Reference Frames setting is supported and defined as moving reference frame by MpCCI. MpCCI recognizes the properties (origin, rotational speed, rotation axis) defined under the Reference Frames panel.

Model with a Motions setting is supported and defined as moving grid by MpCCI. MpCCI recognizes the properties defined under the Motions panel.

46  V  MpCCI 4.6.1-1
3.3.6 Quantity Transformation for Cyclic Symmetric Meshes

If a model consists of a cyclic symmetric mesh, MpCCI provides the possibility to couple it to a full model or to a periodic model which does not coincide but describes the same virtual full model, cf. Figure 16.

![Figure 16: Periodicity panel](image)

The necessary information is inputted into the GUI, cf. 4.6.2.3 Periodic Components.

Internally, MpCCI server handles periodic meshes as full models. Incoming quantities are transformed into SI unit system and “copied” to the server internal sections building the virtual full model. This special copying rotates vector quantities by the periodic pitch angle; scalar quantities are simply copied to the corresponding nodes or elements. This reconstruction corresponds to cyclic symmetry mode zero, also known as nodal diameter zero.

The mapping is performed on the server internal virtual full models. If the target model is a periodic model, the mapped quantities on the full model are averaged for the target periodic section mesh, which guarantees periodic boundaries. At this stage postprocessing (ramping, relaxation, mesh motion, etc.) into the target unit system is performed. The outgoing quantities correspond to the periodic target model.

3.3.7 Operation Workflow for a Quantity

During the data exchange process, a quantity will be processed through different stages where some quantity operators can be defined by the user (cf. 4.6.7 Quantity Specifications).

The following table shows the different operators available at different stages where the sequence numbers describe the workflow for a quantity:
### 3.4 Coupling Process

Coupling can be realized in two ways:

- The equations of each domain of the coupled systems are combined in one large system of equations, which is solved to obtain the solution. This contradicts the approach of MpCCI, which couples two codes, each of which is highly specialized in its field.

- Each system is solved separately, but data is exchanged during each iteration step, i.e. both codes compute one time step (the only one in a stationary computation) at the same time. If one code has finished a step of the iteration, it sends its data to the other code, which uses it in its own iteration. Both iterations are continued until both converge, yielding a state which fulfills the equation of both physical domains. The only disadvantage in comparison to weak coupling is that convergence is slower, thus the computations are slower, but yield more precise results.

The latter approach is supported by MpCCI and can only be realized if codes support exchanges during iterations. The coupling process consists of three main phases:

**Initialization** The codes initialize their data and MpCCI is initialized. MpCCI establishes a connection between the codes and the association (neighborhood search) is carried through.

**Iteration** Each code computes its part of the problem and data is exchanged at certain moments.

**Finalization** The computation is ended by disconnecting the codes and stopping all codes and MpCCI.

Before and after the coupling process, the code can define some pre- and postcalculation steps. Such behavior can be defined under the option Runtime control in the Algorithm step (see [4.5.2 Code Specific Algorithm Settings](#)).

During coupling, the data is exchanged several times. MpCCI cannot control the simulation processes of the coupled codes. Therefore, it is important that whenever one side sends data, the other side should be ready to receive.
It is important to keep in mind, that sending of data is always possible: The data is stored by MpCCI, i.e. it can be received later by the other code. MpCCI can store several sets of data, which are then sent to the other code depending on the synchronization point set by the code. When one code receives data, it waits until data is available from the other code if its coupling configuration does not use the Delay boundary updates option.

There are multiple coupling algorithms and schemes, which will be discussed in the following sections.

### 3.4.1 Coupling Algorithm

The coupling algorithm can be set to Parallel or Serial, leading to a Jacobi-type or Gauss-Seidel-type coupling, respectively. In MpCCI the coupling algorithm can be selected via the MpCCI GUI, cf. >4.5.1 Common Basic Algorithm Settings a.

Figure 17 shows the serial coupling algorithm, where Code A leads the coupling, known as “leading code”. Knowing the fact that the coupling begins with initial conditions from the lagging code (Code B), the code with more stable and better known initial conditions should be chosen as the lagging code. In Figure 17, Code B sends its data for the first coupling step to Code A (1) to lead the simulation. After Code A finished computation (2), it sends its data from the first coupling step (3) and then waits to receive data for the second coupling step. Now, Code B got its data and starts computing the second coupling step (4), then sends data to Code A (5), which can then start its second coupling step calculation (6). At the end it sends the resulting data (7) and Code B can start the next coupling step.

![Figure 17: Serial coupling algorithm with leading Code A.](image)

Figure 18 demonstrates the parallel algorithm without initialization and with initializing Code B. In these coupling types, both codes can do the calculation simultaneously.

![Figure 18: Parallel coupling algorithm without code initialization (on the left) and with initialization by Code B (on the right).](image)

### 3.4.2 Coupling Schemes

Problems are usually time dependent or independent. In the former, which is called transient studies, the results change as time evolves, where usually the outcome for the new time step is dependent on the previous time steps. In these simulations, the behaviour of system over time is typically of interest. For time independent (steady-state) problems, only the final converged solution is desired.
In MpCCI, there are two available schemes, such as Explicit and Implicit. Explicit coupling can be applied for both steady-state and transient simulations, whereas Implicit coupling is only accessible for transient computations. Explicit steady-state and implicit schemes are referred to as iterative coupling.

### 3.4.3 Coupling with Subcycling

The idea of subcycling is the definition of extra iteration steps inside a coupling step, which is schematically depicted in Figure 19. There are possibilities to implement subcycling by configuring MpCCI and the employed solvers. The motivation for subcycling may be the implementation of optimal settings for each solver concerning computational costs, stability and accuracy.

For a transient simulation one solver could require smaller time steps for an accurate and/or stable solution process. Forcing the other code to apply the rather small time step might yield an inefficient process. Furthermore in steady-state calculations for thermal problems, heat propagation in fluids is subject to much faster transportation processes than in solid bodies, which will call for subcycling procedures in the fluid domain.

![Figure 19: Three examples for coupling algorithms with subcycling for the Code A:](image)

- a) The solver step size $\Delta t_s$ and the coupling step size $\Delta t_c$ are constant over the total simulation period,
- b) The solver step size $\Delta t_s$ is constant and the coupling step size $\Delta t_c$ is variable over the total simulation period,
- c) The solver step size $\Delta t_s$ is variable and the coupling step size $\Delta t_c$ is constant over the total simulation period.

In MpCCI, the configuration of the coupling step size defines the frequency of the data exchange (subcycling) in term of solver steps. MpCCI GUI offers the possibility to configure the subcycling for each solver. The setting is available under the option Coupling steps in the Algorithm step (see 4.5.2 Code Specific Algorithm Settings). A tutorial where subcycling is applied for a thermal simulation is VII-5 Exhaust Manifold.

### 3.4.4 Coupling with Delayed Boundary Updates

The idea of delaying the boundary update is to reduce the idle time of a waiting code. This feature is enabled for Steady state and Quasi transient analyses and aims to save idle time if one code runs much faster than the other one. This capability is only for codes which do a stationary analysis and support the delayed boundary update. The faster code may go on in its computation without updating its boundaries at the end of a coupling step. It then tries to do the update at a given substep frequency or at a given
number of update tries within the next coupling step. The update is delayed by no more than one coupling step.

![Diagram of Code A and Code B](image)

Figure 20: Example of delayed boundary updates applied to Code A

The Code A in Figure 20 uses the **Delay boundary updates** with a constant substep frequency to check for new values from Code B. The Code A always sends its quantities at the reached coupling point. If the quantities to receive are not yet available at this point, the receive operation is skipped and will be evaluated at the next delay point.

In MpCCI the **Delay boundary updates** can be selected via MpCCI GUI, cf. §4.5.1 Common Basic Algorithm Settings.<br>

### 3.4.5 Coupling with Transient Analysis Type

#### 3.4.5.1 Implicit Coupling

Transient coupling algorithms exchange data once during each time step. For some problems this explicit transient coupling results in severe stability and accuracy problems. Using implicit transient coupling schemes, it is possible to reach a stable solution with moderate time step sizes for these problems.

Implicit coupling schemes are necessary for coupled simulations in which the interaction between the codes is strong. Usually these are fluid-structure interactions with an incompressible fluid and a low-density solid. Other applications might benefit from implicit coupling schemes because of bigger possible time step sizes.

![Diagram of Implicit Coupling](image)

Figure 21: Jacobi algorithm for implicit coupling without any code initialization settings. (\( n \) is the time level, \( k \) is the coupling step iteration level)
Two different coupling algorithms are possible: a Gauss-Seidel and a Jacobi algorithm corresponding to the usual serial and parallel coupling algorithms which are described in §3.4.1 Coupling Algorithm. Figure 21 and Figure 22 show these two coupling algorithms.

![Diagram of coupling algorithms](image)

Figure 22: Gauss-Seidel algorithm for implicit coupling with leading Code A settings.

For both coupling algorithms the step from one time step to the next one is handled in a parallel, i.e. a Jacobi-type, way. The coupled step iterations $k$ for each time step $n$ are computed in a serial or in a parallel way.

- Each coupling step iteration continues until the convergence criteria are fulfilled (see §3.3.4.1 Convergence Check) or the maximum number of coupling step iterations is reached (see Coupling duration settings for transient analysis in §4.5.1 Common Basic Algorithm Settings).
- For some applications – especially fluid-structure interactions with incompressible fluids and moving walls – the Gauss-Seidel algorithm should produce a more stable solution.
- Make sure that both coupled codes reach the same time at the end of the coupling. If the end time in the input file is wrong or it is not a multiple of the given time step in case of constant time-stepping, numerical inaccuracy or co-simulation errors could be expected at the end of the coupling. Please refer to the adapter description in Codes Manual for more information.

### 3.4.5.2 Coupling with Exchange of Time Step Size

Instead of using a fixed coupling time step size, it is also possible to use adaptive time stepping.

- Determined by code:
  
  In this case the time step size is determined by one code and sent to the partner code, which changes its own time step to the received value. MpCCI regards the time step as a global quantity without any special meaning, its name is “PhysicalTime” or “DeltaTime”, depending on whether it is the absolute time or the time difference to the previous step (see XI-XI Quantity Reference in the appendix).

  The exchange of time step sizes may yield unwanted effects:

  **Time Shift:** It depends on when exactly each code determines the time step size and sends it to the partner code. It may happen that the partner code uses the time step size of the previous step. This problem cannot always be solved, try to use a different coupling algorithm.
Convergence Problems: If the time step size is determined by one code it may be simply too large for the partner code. Please choose carefully which code shall determine the size and set reasonable limits.

⚠️ It is strongly recommended to always define a default time step size in each simulation code, because the value of “PhysicalTime” or “DeltaTime” may not be defined at the beginning of a simulation! This depends on the coupling algorithm settings and how exactly each code evaluates the received value. The default value should be the same for both partner codes.

- Negotiate the time step size:
Each code exchanges the time step size at the begin of the coupled step and the minimum value is used for the time step (▷ 4.5.1 Common Basic Algorithm Settings ◁). An example for time step size negotiation is provided in ▷ VII-3 Driven Cavity ◁.

### 3.4.5.3 Coupling with Non-Matching Time Steps

**Coupling of Codes with Different Time Step Sizes** Setting a fixed coupling step size for all codes might be disadvantageous for some applications. With MpCCI every coupled code can use its own time step size – perhaps a time stepping scheme adapted to the specific problem – and still exchange data with each other. For these asynchronous coupling schemes quantity values are interpolated between different time steps, such that every code receives the values that match its own time step value.

The main principle of the interpolation can be seen in Figure 23. In this example codes A and B each use their own fixed, non-matching time step sizes.

![Figure 23](image)

Figure 23: Non-matching time step sizes. Red arrows show which values of code A are used for the interpolation (using a “higher-order” interpolation)

The time step of code B is smaller; the quantity data MpCCI sends to code B at the first step $t_B^0$ is the result of a linear interpolation using the two quantity values sent by code A at time steps $t_A^0$ and $t_A^1$. This is indicated by the two red arrows pointing from code A to code B.

For time step two of code B ($t_B^2$) the quantity values from up to three (depending on the selected interpolation method) steps by code A – $t_A^0$, $t_A^1$, and $t_A^2$ – are used for the interpolation. Obviously, the values that are sent from code B to code A also have to be interpolated. Figure 23 contains only the interpolation for one direction (from code A to code B) because of clarity reasons.

Depending on the selected interpolation method a constant, linear or higher order interpolation is used. The used interpolation method can be selected in the Algorithm step of the MpCCI GUI. The different methods are described in the next section.
Interpolation Methods

The interpolation is handled automatically by MpCCI when the time step sizes do not match. No additional settings are required. The user can select the interpolation method in the Algorithm step (▶ 4.5.1 Common Basic Algorithm Settings). A constant, linear and higher order interpolation is available. The different interpolation schemes are shown in Figure 24.

The constant interpolation function (i.e. zero order) uses the last available value; for a standard linear interpolation (i.e. first order) one value before and one after the requested value is used.

Cubic Hermite spline interpolation (also called cspline) is used to achieve a higher order interpolation. For the definition of a cubic Hermite spline at the interpolation point $x^*$ four values are needed: The two values at $x_1$ (smaller than $x^*$) and $x_2$ (bigger than $x^*$) and the tangents at these two points. Since the tangent values are not available, finite difference schemes are used to get an approximate value. If it is possible, three values and a central difference scheme are used to calculate an approximation of the tangent at the two points $x_1$ and $x_2$. For $x_1$ this are the values for $x_0$, $x_1$ and $x_2$ respectively.

Depending on the data points these calculated tangent values are scaled to ensure a monotone interpolation. In Figure 24, for example, the interpolating cubic spline between the third and fourth data point is monotonous. This ensures that the higher order interpolation curve does not oscillate or overshoot between the given data points.

The default interpolation method is the higher order interpolation using monotone cubic Hermite splines.

Limitations

Coupling of codes using different time scales

If the coupled simulations use very different time scales, the coupled job might abort with the error output:

*** ERROR *** The stored buffer time frame is out of the last grid time xx.
Either the grid time $xx$ has not been received by the code or the history size is too small.

Check either the "Coupling Algorithm" option or the "HistorySize" setting.

This happens if deformed coordinates (i.e. $\text{MpCCI}$ quantity $N\text{Position}$) are sent from a code with a big time step to a code with a small time step size.

Increasing the $\text{Job.HistorySize}$ in the $\text{Settings}$ step might help in these cases. The default value of 4 can be set to a larger number (up to 32).

If the used time steps ($\Delta t_{big}$ and $\Delta t_{small}$) are known,

$$\text{HistorySize} \geq \frac{\Delta t_{big}}{\Delta t_{small}} + 4$$

might be a good choice for the value.

Alternatively, this error message might occur, if the $\text{Coupling Algorithm}$ settings are not appropriate. When using parallel algorithms, code initialization should be disabled. Otherwise, it might result in the fatal message from above because the first grid information (if the coordinates are a coupled quantity) is not received.

**Serial coupling algorithms**

Currently, $\text{MpCCI}$ does not support extrapolation in time, which leads to some restrictions when codes with different time step sizes are coupled.

Serial coupling algorithms with non-matching time step sizes can only be used when the leading code uses a bigger time step size than the lagging code. Furthermore, the bigger time step size has to be a multiple of the smaller time step – resulting in matching coupling times at each big time step.

Generally it is suggested to avoid using serial algorithms for non-matching time step sizes.

### 3.4.6 Coupling with Mixed Analysis Types

A transient coupling is challenging because of the great disparities of the physical models between the fluid and the solid. The main difficulty is due to the significant discrepancy of characteristic times since the transient phenomena in the fluid usually take place at a much smaller time scale than those in the solid.

The $\text{Fully Transient Coupling}$ method describing the transient in both the fluid and the solid leads to a highly accurate solution but is too expensive. For some problems it is not a viable approach for practical design purposes.

Due to the discrepancy of characteristic times for the transient phenomena you can do the assumption that one of the problem having a smaller time scale than those used for modeling the other problem is considered as a steady-state problem. It leads to a computationally cheaper and simplified coupling procedure.

Such approach can be employed for the following application:

- **Transient surface heat transfer**
  In this coupling procedure you should adopt a coupling interval which is at least equal to the solid time step. The flow solution is considered as a sequence of steady-state solutions and in the solid the fields evolve in a fully transient manner.
  As in Figure 25 explained, the exchange of temperature ($\text{Temperature}$ or $\text{WallTemp}$) and the heat flux ($\text{WallHeatFlux}$) is performed after the solid time step and a fluid steady-state step. In the parallel coupling scheme example, both solvers perform an initial calculation step without coupling.
by defining a coupling start time $t_1$ or iteration $i_1$. The fluid side may optionally perform some subcycling iterations before transmitting the heat fluxes.
• **Electrothermal analysis case: Electric arc simulation**

In this application you may assume a steady-state computation of the electromagnetic phenomena (Karetta and Lindmayer [1998]) because the electromagnetic phenomena are faster than the gas dynamic. The Maxwell equations will be solved as a sequence of steady-state solutions whereas in the fluid the Navier-Stockes equations evolve in fully transient manner.

By selecting such model you will be warned by the MpCCI GUI that you are currently mixing the solution types (Figure 26). You can process to the next step and set up the coupled simulation as usual. For these mixed solution types, coupling scheme is automatically set to Explicit.

![Figure 26: Coupling with mixed solution types](image)

### 3.4.7 Restarting a Coupled Simulation

One should distinguish between an actual restart and a complex start. 

**Restart** means the simulation is interrupted by killing the simulation or reaching a target time. Then the simulation is restarted seamlessly, i.e. continued as if it had never been interrupted.

**Complex Start** means that the coupled simulation consists of different analysis steps, which differ in e.g. boundary conditions, simulation analyses (transient/steady state). The second step is then based on the results of the first one.

When restarting a coupled simulation, this consists of three parts. Each of the simulation codes must be restarted separately – the same way it is done without coupling – and MpCCI must be restarted.

A new neighborhood search is performed on the original or deformed meshes depending on the capability of the code to provide the original mesh and may yield different results in comparison to the first search, i.e. the restart may not be really seamless.

Restarting a coupled simulation is an advanced analysis procedure. It is difficult to choose the coupling algorithm such that the time in both codes is still synchronous after a restart. Mostly it should be set to **Parallel** for the restart. This is also code-dependent, there is information on restarts for some codes in the Codes Manual.
3.5 Running MpCCI in a Network

The use of MpCCI in a network is almost unlimited. MpCCI can couple simulation codes which run on different computers with different operation systems. It is also possible to run one or both of the simulation codes in parallel and to start the coupled simulation on a cluster using a queuing system.

3.5.1 Client-Server Structure of MpCCI

The code coupling of MpCCI is based on a client-server model: The MpCCI processes act as servers while the simulation codes – more precisely the code adapters of the simulation codes – act as clients.

If codes are coupled without parallel execution, one server is started for communication with each simulation code, see Figure 27 a).

If a simulation code is running in parallel, MpCCI starts one server and there are two possible ways of communicating for the simulation code:

- All data which is exchanged with the simulation code is passed via a main process. This is depicted in Figure 27 b) on the left.

![Figure 27: Client-server structure of MpCCI](attachment:image.png)
• Each subprocess of a simulation code communicates with the MpCCI server as shown in Figure 27 b) on the right.

**Distributing MpCCI Server on a Remote Machine**

The MpCCI server communication is socket-based. It can be run on a different machine which uses another endianness. (The “endianness” is the order in which bytes are stored or transmitted, which depends on the processor.)

The host for the MpCCI server can be provided in the Go step of the MpCCI GUI. Select the option Distribute server on remote host and enter a name of remote computer (with domain name if necessary) into the field Remote ‘host’ to be used. See > 4.9 Go Step ☐ for details.

**Distributing Simulation Codes on Different Machines**

The communication between the simulation code clients and MpCCI server is purely socket-based.

If a simulation code shall be started on a remote computer, the input file must be placed there and selected in the remote file browser (> 4.10 Remote File Browser ☐). If the code shall additionally be started in parallel, please select the appropriate options in the Go step, as described in the Codes Manual.

**Port Numbers for Socket Communication**

For a coupled simulation MpCCI uses the basic port number for the different socket connections, which can be selected in the MpCCI GUI (> 4.9.1 Configuring the MpCCI Coupling Server ☐) or when starting the server with mpcci server (> 5.7.11 mpcci server ☐).

In the example sketched in Figure 27 a) only one port is used for getting the incoming client connection:

- 47010 Connection to simulation code A
- 47010 Connection to simulation code B

For the parallel run of Figure 27 b) only one port is needed:

- 47010 Connection to main process of simulation code A
- 47010 Connection to process 1 of simulation code B
- 47010 Connection to process 2 of simulation code B
- 47010 Connection to process 3 of simulation code B

⚠️ Once all connections are established, the ports may be re-used, even if a simulation is still running.

Only if you want to start several coupled simulations at the same time, you should use different sets of ports.

If you need to change the ports due to firewall restrictions, please ensure that a sufficient number of subsequent ports can be accessed.

**Socket Communication Timeout**

The default socket communication timeout is set to 60 seconds. This timeout can be overwritten by defining the environment variable MPCCI_TIMEOUT. This accepts an integer value defining the timeout in seconds. This should be defined as global environment variable for the simulation. Setting the value to -1 will disable the timeout.

**Network Device**

The default network device is using the standard default ethernet interface for the socket communication. This setting can be overwritten by defining the environment variable MPCCI_NETDEVICE. This accepts a string value defining the name of the interface to be used. The network interface name mostly matches with the device name listed by the command ifconfig for Linux operating system. This environment variable should be defined as global environment variable for the simulation.

You can check the device name on the target machine by using the command mpcci netdevice (> 5.4.4
Example of testing the infiniband interface: `mpcci netdevice ib0`

### 3.5.2 Hostlist File

If you run coupled simulations frequently on different computers, it is recommended to use a hostlist file, which contains a list of computers. Each line in the list corresponds to one remote machine. There are two possible line formats, the standard format:

```
[user@]hostname [#rsh] [#ssh] [#home:path] [#arch:token]
```

or extended (".rhost" type) format:

```
hostname [user user user] [#rsh] [#ssh] [#home:path] [#arch:token]
```

Lines starting with a `#` are treated as comments. The entries `#rsh` and `#ssh` are used to select the network communication method (see 3.5.3 Remote Shell and Remote Copy below), the `#home:` option is used to specify the path to the home directory (only needed if not standard directory) and the `#arch:` option can be used to explicitly choose an architecture, see 2.3.1 MPCCI ARCH - Architecture Tokens for details.

Thus a hostlist file might look like:

```
fred@jupiter #ssh #home:/u/fred
saturn.example.com fred barney #arch:lnx4_x64
# this line is a comment
mars
```

The standard MPCCI hostlist file is "<Home>/mpcci/mpcci.hosts", which is located in the MPCCI Resource Directory (2.4 The MPCCI Resource Directory). If you want to use a different default hostlist file, you can set the environment variable MPCCI_HOSTLIST_FILE to the path of the file.

You may list the contents of the hostlist file with `mpcci list -hosts`.

### 3.5.3 Remote Shell and Remote Copy

To start processes on remote machines and copy files, MPCCI can use either the rsh or the ssh family of commands. The environment variable MPCCI_RSHTYPE defines which family of remote commands is used:

```
MPCCI_RSHTYPE remote shell file copy
rsh rsh or remsh ftp or rcp
ssh ssh scp or sftp
```

If MPCCI_RSHTYPE is not defined or not set the default is rsh under UNIX and Linux and ssh under Microsoft Windows.

It depends on your network configuration which method should be used.

Please also read 3.6.1 Accessing Remote Hosts for testing the remote access facilities.

### 3.6 Coupled Analysis in Batch Mode

#### 3.6.1 General Approach

To start a batch job with MPCCI, i.e. a simulation without graphical interface, a project file "*.csp" must already be present. The following procedure is recommended:
1. Set up a coupled simulation on a computer with graphical interface. Proceed up to the **Go** step in the **MpCCI GUI** and save the project file.

    ![Information](image)
    
    If a code has to run in parallel, you have to enable the parallel option as described in the **Codes Manual**.

2. Start the coupled analysis with `mpcci -batch <project file>` from the **MpCCI** project directory.

The `mpcci -batch` command will start the simulation and distribute the codes as configured in the project file.

### 3.6.1.1 Run a Job in Batch Mode with **MpCCI** Command Line

The `mpcci -batch` command is used to start the simulation. It distributes the codes as configured in the project file. The `mpcci -batch` command offers some additional options to configure the resources settings. Following options are available:

- **-checkonly**  Create the project file used for the batch job and exit. A new batch system compatible project file will only be created when running the **MpCCI** job under a batch queuing system. It may help you to adjust the host distribution for example.

- **-chwd <PATH>**  Replace the symbolic $(CWD) working directory used inside the project file by the absolute pathname specified in the **PATH** argument.

- **-listen <N>**  Specifies an alternative port number N for the communication between the **MpCCI** server and the simulation codes. The default port number is 47010.

- **-mpmode <codeA:mode><codeB:mode>**  Set the parallel method mode for each code with the following option:
  
  - **none**: run code on one cpu
  - **smp**: activate shared memory parallel mode
  - **dmp**: activate distributed memory parallel mode

- **-nocontrol**  Start the **MpCCI** batch job without activating the termination process management. The user is responsible for terminating the simulation process.

- **-nolic**  Do not check for a license before starting the batch job.

- **-norsa**  Do neither check for ssh nor ask for ssh assistance if an rsa key file does not exist.

- **-np <codeA:N><codeB:M>**  Bind each code (codeA, codeB) to a specified total number of cores (N,M) to use. You can modify the number of cores to use without need to reedit your project file.

- **-precheck**  Start **MpCCI** in pre-check mode only. The simulation codes will send their mesh data for a neighborhood search calculation. The results of the check are stored in the server log file (>3.7.3 Log Files).

- **-set codeA:[property=value] codeB:[property=value]**  Modify property value of a code before submitting the job in batch. The property value respect the following rules:
  
  - the property name matches the key string (<param name="property"...>) from the project csp file.
  - the value to modify should respect the defined type.
  - the list of property value pair is comma separated.
  - the list should be defined within the brackets.

- **-useAbsolutePath**  Use the current local **MpCCI** installation directory as reference for any remote computer access.

Example about running only a neighborhood calculation:
Example about changing the parallel method to use and change the number of cpus for each code:

```
mpcci batch -np abaqus:4 fluent:16 -mpmode abaqus:smp fluent:dmp test.csp
```

If you want to directly submit the calculation on a queuing system, e.g. SGE you can simply add the specific MpCCI batch name options:

```
mpcci batch SGE submit -np abaqus:4 fluent:16 -mpmode abaqus:smp fluent:dmp test.csp
```

This command calls the submission `qsub` and passes all parameters from the commandline.

### 3.6.1.2 Configure a Simulation Code for Running on a Remote Machine

Usually, when you have to run distributed computation over several compute nodes, you have to take care that the required environment to start the simulation code is correctly set. In some cases, especially with a job scheduler queuing system, the environment is usually set up in the job shell script where the PATH and license environments are defined. This compute environment is valid on the master node and is not necessarily exported to the slave node when a remote command is started from the master node. In order to address this issue, `mpcci -batch` command can be used with the option `-useAbsolutePath` which will use the current local MpCCI installation path on the compute node.

Additionally you may define the environment variables to be set up for each code before running the code. For this purpose `mpcci -batch` command looks for the special environment file "mpcci_<CODENAME>.env" under the directory containing the model file of the simulation code. To use this feature you need to create the special environment file. The `<CODENAME>` should be replaced with the name of the code in use, for example: Abaqus, FLUENT, etc. The name of the code can be written in lowercase or can match the naming case convention of MpCCI. This file "mpcci_<CODENAME>.env" should define some environment variables to be updated like PATH, LD_LIBRARY_PATH, LM_LICENSE_FILE, etc.

The following rule syntaxes are available to define a variable:

- to add a value to an existing variable, use the syntax `+=`. For example: `VARIABLE+=value`.
- to set a value to a variable, use the syntax `=`. For example: `VARIABLE=value`.

For example:

```
LM_LICENSE_FILE=30000@licserver
PATH+=/opt/mycode/bin
LD_LIBRARY_PATH+=/opt/mycode/lib64
```

Each environment variable will be updated with the defined value from the ".env" file.

### 3.6.2 Job Scheduler Environment

To run a coupled computation with a job scheduler (queuing system) on a computing cluster MpCCI provides a set of commands to submit, status, kill a job to/from a job scheduler.

MpCCI supports batch-systems like MS HPC, LSF, PBS, TORQUE, SGE, and SLURM.

When preparing the project files, you cannot already know on which nodes of a cluster your processes will be started. MpCCI automatically detects that it is started by a job scheduler and changes the entries in the project files accordingly, i.e. the jobs are spread onto the reserved nodes by MpCCI. MpCCI will create a new project file containing the new associated hosts for each code. This project file is named by the prefix "batch_" to the original project file name.
The following procedure is recommended in order to prepare the project file:

1. Start the MpCCI GUI on a computer with graphical interface.
2. Set up the coupled analysis as a local job. This means the model files are selected on the local file system.
3. Proceed up to the Go step in the MpCCI GUI and save the project file.

Please consider following items:

- To distribute the server on a remote host (see 4.9.1 Configuring the MpCCI Coupling Server), you have to define the Preferred remote shell type and let the Remote 'host' to be used field empty.
- To configure a code to run in parallel, you have to enable the parallel option as described in the Codes Manual for the code and specify the number of processes or threads to use.
- In case that the computing cluster has a different file system than the workstation where you have set up the project file, it is recommended to copy the project directory including all input files for the simulation codes as well as the MpCCI project file "*.csp" to the compute head node.

Up to step 3 you have prepared your project file.

Now there are three ways to submit the job:

- The MpCCI GUI may assist you to submit the job to the queue and configure the host allocation for each code if necessary (3.6.2.4 Submitting a Batch System Job with MpCCI GUI).
- You can submit the job by using the MpCCI command line (3.6.2.3 Submit a Batch System Job with MpCCI Command Line) which will create a small shell script and send the job to the submission system.
- You can write your own shell script which calls mpcci -batch command (3.6.1.1 Run a Job in Batch Mode with MpCCI Command Line) to run the co-simulation.

### 3.6.2.1 Host Management under a Job Scheduler Environment

At startup of a coupled job MpCCI checks if corresponding environment variables

- LSB_MCPU_HOSTS
- PBS_NODEFILE
- PE_HOSTFILE
- CCP_NODES
- SLURM_JOB_NODELIST

are defined. MpCCI then uses the listed host names to start MpCCI coupling server and both coupled codes on these hosts.

The total number of processors/cores for the coupled simulation is equal to the total amount of

- the total number of parallel processes/threads for all simulation codes plus
- the total number of threads used by the MpCCI Grid Morpher if used plus
- one process for the MpCCI server.

MpCCI distributes the available hosts by following this procedure:

1. Check for multi-threaded parallel codes and assign an SMP host:
• MpCCI searches for a machine having at least the number of required cores for running the
  multi-threaded parallel code.
• If no host has been found satisfying the minimal number of threads required, MpCCI will stop
  the batch process, otherwise the next multi-threaded parallel code is handled.

2. MpCCI checks the rest of available hosts for the parallel codes and warns if the number of processors
   are not enough or too much.

3. Check for specific code to host allocation resource option (see \[3.6.2.2 Application Specific Code to
   Host Allocation\]) and allocate the host to the corresponding code.

4. If no specific code to host allocation option is specified, MpCCI distributes the rest of the hosts by
   respecting this rule:
   • Try to find pairs of matching cores and code processes and allocate the host to the code.
   • Next try to find a host with a number of cores greater than the number of code processes and
     allocate the host to the code if found.
   • Otherwise if no host satisfies the minimal number of required processes for a code, try to perform
     a homogeneous distribution of processes between the hosts.
   • Distribute the codes on the hosts left.

5. Resort the host list with the current host (master node) at the begin of the list if this one is listed.

3.6.2.2 Application Specific Code to Host Allocation

MpCCI allows to specify in more detail the allocation of hosts for specific codes.
If environment variables

- \texttt{MPCCI_<CODENAME-1>_HOSTS = "host-a1 host-a2 host-a3..."}
- \texttt{MPCCI_<CODENAME-2>_HOSTS = "host-b1 host-b2 host-b3 ..."}
- \texttt{MPCCI_SERVER_HOSTS = "host-c1 host-c2 host-c3 ..."}

are set, the MpCCI startup procedure will allocate the listed hosts for the code represented by the name
\texttt{<CODENAME>} or for the server with \texttt{MPCCI_SERVER_HOSTS}. It is up to the user to define these environment variables if the MpCCI batch command line is used. Otherwise by submitting the job from MpCCI GUI
these environment variables are written into the batch script.
Instead of a list of real hostnames a regular Perl-pattern might be used to define the selection of code-specific hosts extracted from the defined PBS, LSF, SGE or LoadLeveler environment variables:
\texttt{MPCCI_<CODENAME-1>_HOSTS = "/pattern/".}

\texttt{<CODENAME-1>} must be a "word" character.

3.6.2.3 Submit a Batch System Job with MpCCI Command Line

The syntax of the command line is the following:
\texttt{mpcci -batch [OPTIONS] <BATCH_NAME> submit [BATCH-OPTIONS] <project file>}
with the arguments meaning:

- \texttt{[OPTIONS]} represents a list of the MpCCI options for batch mode (cf. \[3.6.1.1 Run a Job in Batch Mode
  with MpCCI Command Line\]).
- \texttt{<BATCH_NAME>} represents the name of the batch system e. g. PBS, SGE, etc.
- \texttt{[BATCH-OPTIONS]} represents a list of the original options of the submit command.
- \texttt{<project file>} is the project file or a shell script.
If the `<project file>` is a shell script this will be simply handed over to the submit command

By using the MpCCI batch command line you have to define the host allocation for each code by yourself. You can pass the host allocation definition as `[BATCH-OPTIONS]` argument. The variables

- `MPCCI_<CODENAME-1>_HOSTS="host=a1 host=a2 host=a3..."`
- `MPCCI_<CODENAME-2>_HOSTS="host=b1 host=b2 host=b3 ..."
- `MPCCI_SERVER_HOSTS="host=c1 host=c2 host=c3 ..."

should be given as arguments (see also `3.6.2.2 Application Specific Code to Host Allocation`).

After submitting the job the job id is saved by MpCCI in the directory `<HOME>/mpi/batch`. You can retrieve the list of submitted jobs with this command: `mpcci list jobs` which returns the job id with the associated batch system:

<table>
<thead>
<tr>
<th>Batch System</th>
<th>Job ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGE</td>
<td>1532</td>
</tr>
<tr>
<td>PBS</td>
<td>8470</td>
</tr>
</tbody>
</table>

This information can be used to status or kill the job.

Here is an example of job submission for SGE:

```
$> mpcci -batch SGE submit -J coupled_test -cwd -pe cre 8 MPCCI_SERVER_HOSTS=/.* MPCCI_FEM_HOSTS=/.* MPCCI_CFD_HOSTS=/.* test.csp
```

And this is the generated script "test.sh" that is given to the submit command:

```
#!/bin/sh
#$ -S /bin/sh
#$ -v MPCCI_LICENSE_FILE,PATH
#$ -cwd
#$ -N coupled_test
#$ -pe cre 8
MPCCI_CFD_HOSTS=/.*
MPCCI_SERVER_HOSTS=/.*
MPCCI_FEM_HOSTS=/.*
export MPCCI_CFD_HOSTS MPCCI_SERVER_HOSTS MPCCI_FEM_HOSTS
mpcci batch test.csp
```

**Status of a Batch System Job with MpCCI Command Line**

Each batch system provides a status command:

```
mpcci batch <BATCH_NAME> status <JOB ID>
```

The job id may be retrieved with the `mpcci list jobs`.

⚠️ The status command has to be executed on the machine where the job has been submitted.

**Kill a Batch System Job with MpCCI Command Line**

Each batch system provides a kill command:

```
mpcci batch <BATCH_NAME> kill <JOB ID>
```

The job id may be retrieved with the `mpcci list jobs`.

⚠️ The kill command has to be executed on the machine where the job has been submitted.
3.6.2.4 Submitting a Batch System Job with MpCCI GUI

To submit the MpCCI job with the MpCCI GUI you have to start the MpCCI GUI on the head node of your cluster. Usually the batch queue commands are available on the compute head node.

You have to open your project and select Batch → Submit Batch Job in the menu (which is enabled in the Go step). A dialog will pop up and you are able to configure the batch options (cf. § 4.3.3 Batch Menu). For specifying the host allocation you have the option to provide

- a list of hostnames or
- a pattern (regular expression) to modify/select some hostnames.

If a host allocation field is empty MpCCI will automatically assign the allocated hostnames to the code without any specific hostname processing. The hostname assignment is also explained in section § 3.6.2.2 Application Specific Code to Host Allocation.

By clicking on the Submit Batch Job button MpCCI prepares a script named ("<project name>. [sh|bat]") to be submitted to the batch system for the current project. This generated script contains all the options provided by the MpCCI GUI for the selected batch system and the call of mpcci -batch <project file>.

Below follows the description of the configuration dialog for each batch system.
Options Details for an MS HPC System (Figure 28)

![Figure 28: MS HPC options](image)

- **Job name**: You can provide the name of this job. This setting represents the option `/jobname` of the submit command.
- **Task name**: You can provide the name of the task. This setting represents the option `/name` of the submit command.
- **Working directory to be used during execution of the task**: This setting represents the option `/workdir` of the submit command. It is recommended to provide the directory path of the MpCCI project file. This should be either a UNC path or an accessible path from the compute nodes.
- **Name for the stdout output file**: You can provide a filename for the stdout collected by the batch system. This setting represents the option `/stdout` of the submit command.
- **Name for the stderr output file**: You can provide a filename for the stderr collected by the batch system. This setting represents the option `/stderr` of the submit command.
- **Exclusive**: If true, no other jobs can be run on a compute node at the same time as this job. This setting represents the option `/exclusive` of the submit command.
- **Number of cores required by this job**: This setting represents the option `/numcores` of the submit command.
**Number of nodes required by this job**  This setting represents the option `/numnodes` of the submit command.

**Node groups for this job (separated by comma)**  The job can only be run on nodes that are in all of these groups (e.g. `cfd,mpich2`). This setting represents the option `/nodegroup` of the submit command.

**List of environment variables to export (separated by semicolon)**  Specifies the environment variables to set in the task’s runtime environment (e.g. `TEST_NAME=coupling;TEST_MODE=parallel`). This setting represents the option `/env` of the submit command.

**Additional command options (separated by white space)**  You can provide a list of submit options that are not listed here. These options will be passed on to the submit command. (e.g. `/nodegroup cfd,mpich2`)
Options Details for an LSF System (Figure 29)

<table>
<thead>
<tr>
<th>Batch system:</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSF</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Job name:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue name:</td>
</tr>
<tr>
<td>Name for the stdout output file:</td>
</tr>
<tr>
<td>Name for the stderr output file:</td>
</tr>
<tr>
<td>Number of hosts/processors:</td>
</tr>
<tr>
<td>Host selection (option -m):</td>
</tr>
<tr>
<td>Resource requirements (option -R):</td>
</tr>
<tr>
<td>Additional command options:</td>
</tr>
</tbody>
</table>

Figure 29: LSF options

**Job name**  You can provide a job name for the coupled analysis. This setting represents the option -J of the submit command.

**Queue name**  You can provide a queue name to submit the job. This setting represents the option -q of the submit command.

**Name for the stdout output file**  You can provide a filename for the stdout collected by the batch system. This setting represents the option -o of the submit command.

**Name for the stderr output file**  You can provide a filename for the stderr collected by the batch system. This setting represents the option -e of the submit command.

**Number of hosts/processors**  You can provide the number of hosts/processors to be reserved for the job. This setting represents the option -n of the submit command.

**Host selection (option -m)**  You can provide a set of candidate hosts for running the coupled analysis. This setting represents the option -m of the submit command.

**Resource requirements (option -R)**  You can provide the resource requirement setting for the coupled analysis. This setting represents the option -R of the submit command.

**Additional command options**  You can provide a list of submit options that are not listed here. These options will be passed on to the submit command.
Options Details for a PBS System (Figure 30)

The options are identical for the PBS, PBSPro, OpenPBS and Torque batch systems.

**Job name**  You can provide a job name for the coupled analysis. This setting represents the option `-N` of the submit command.

**Queue name**  You can provide a queue name to submit the job. This setting represents the option `-q` of the submit command.

**Name for the stdout output file**  You can provide a filename for the stdout collected by the batch system. This setting represents the option `-o` of the submit command.

**Name for the stderr output file**  You can provide a filename for the stderr collected by the batch system. This setting represents the option `-e` of the submit command.

**Join both standard output and standard error**  select this option to merge the standard output and standard error. This setting represents the option `-joe` of the submit command.

**Keep both standard output and standard error**  select this option to keep both standard output and standard error. This setting represents the option `-koe` of the submit command.

**Defines the resources that are required by the job (option: -l)**  You can define a resource requirement for running the coupled analysis. This setting represents the option `-l` of the submit command.

**Additional command options**  You can provide a list of submit options that are not listed here. These options will be passed on to the submit command.
Options Details for a SGE System (Figure 31)

Figure 31: SGE options

The options are identical for the SGE, N1GE and SGE EE batch systems.

**Job name** You can provide a job name for the coupled analysis. This setting represents the option -J of the submit command.

**Queue name** You can provide a queue name to submit the job. This setting represents the option -q of the submit command.

**Name for the stdout output file** You can provide a filename for the stdout collected by the batch system. This setting represents the option -o of the submit command.

**Name for the stderr output file** You can provide a filename for the stderr collected by the batch system. This setting represents the option -e of the submit command.

**Change to the working directory of call** Select this option in order to request the batch system to change to the working directory of the submit call. This setting represents the option -cwd of the submit command.

**Resource requirements (option -l)** You can provide the resource requirements setting for the coupled analysis. This setting represents the option -l of the submit command.

**Define the parallel environment option (option: -pe)** You can provide the parameter for the parallel environment. This setting represents the option -pe of the submit command.

**Additional command options** You can provide a list of submit options that are not listed here. These options will be passed on to the submit command.
Options Details for a SLURM System (Figure 32)

**Job name**  You can provide a job name for the coupled analysis. This setting represents the option `--job-name` of the submit command.

**Partition**  You can provide a partition name to submit the job. This setting represents the option `--partition` of the submit command.

**Name for the stdout output file**  You can provide a filename for the stdout collected by the batch system. This setting represents the option `--output` of the submit command.

**Name for the stderr output file**  You can provide a filename for the stderr collected by the batch system. This setting represents the option `--error` of the submit command.

**Number of nodes**  You can provide a number of nodes requested for the job. This setting represents the option `--nodes` of the submit command.
3.6.2.5 Running the MpCCI GUI in an Interactive Batch Session

To enable the usage of the MpCCI GUI during an interactive batch session, the environment variable MPCCI_IBATCH should be defined in the current session. For example, define the environment variable under a Bourne shell:

```
export MPCCI_IBATCH=1
```
3.7 MpCCI Project and Output Files

3.7.1 MpCCI Project Files

The MpCCI project files are named "*.csp" and contain all relevant data for a coupled computation in XML format.

3.7.2 MpCCI Server Input Files

The MpCCI server input files are named "<job name prefix>.prop" and generated from the project files before the MpCCI server is started.

⚠️ The contents of the server input files are subject to change without notice. Do not write or modify "*.prop" files on your own!

3.7.3 Log Files

In addition to the log files of the coupled simulation codes, MpCCI writes its own files:

"mpcci_<job name prefix>_<project name>.log" The server logs the internal calls of the code coupling interface. If errors appear in any server or simulation codes, the error messages are written to this file.

"mpcci_<job name prefix>_server.log" This log contains a summary of your coupled simulation and you can find following information:

About the co-simulation job setting The job setting chosen in the MpCCI GUI (see 4.8.2 Job) will be displayed. For each code you will get the properties of the coupled meshes and parts that compose the mesh like: the bounding box, the number of nodes and elements, the mesh characteristics, the quantities defined on this mesh, etc.

About the neighborhood search calculation For each coupled mesh you will get a summary about the number of orphaned nodes and elements as well as of the childless nodes and elements for the corresponding multiplicity factor used.

About the wall clock time for the co-simulation You will find a time listing concerning:

- The mesh definition: the total wall clock time to send the mesh to the MpCCI server is measured.
- The time spent per coupling step such as:
  - SEND: the time needs by the code to send all the data to MpCCI.
  - RECEIVE: the time needs by the code to receive all the data from MpCCI. This includes the waiting time for the quantities.
  - SOLVER: the computation time for the simulation between the coupling step.
  - STEP: the total time for this coupling step.

"<job name prefix>_<timestamp>_<writer_type>" If you choose to activate some writers in the Edit Step of the MpCCI GUI (see 4.8 Settings Step) you will have some directories created with the job prefix name (default is mpccirun) given in the Go Step of the MpCCI GUI (see 4.9 Go Step) and the suffix of the writer (e.g. ccvx,vtk).

"<job name prefix>_<timestamp>_<writer_type>_<failure>" In case of a failure during the calculation the latest data from the MpCCI server are stored in this directory. You can use the MpCCI Visualizer to check them.

"<job name prefix>_<timestamp>_<writer_type>_<precheck>" In case of running a pre-check mode calculation the mesh data will be saved in this folder. If any orphans exist you will have the possibility to check the concerning region in the MpCCI Visualizer.
3.7.4 Tracefile

MpCCI server can write tracefile for other visualizer tools:

- The tracefile "*.ccvx" is used for the MpCCI Visualizer, see $\triangleright$ 7 MpCCI Visualizer $\triangleleft$.
- The tracefile "*.vtk" is used for the Paraview.
- The tracefile "*.vtfa" is used for the GLview Inova.
- The tracefile "*.case" is used for the EnSight Gold.
- The tracefile "*.csv" is a comma separated value file format. This tracefile contains only output data from a single point coupling system and global variables.
4 Graphical User Interface

4.1 Starting and Exiting MpCCI GUI

4.1.1 Starting MpCCI GUI

When you create a coupled simulation project, theMpCCI GUI generates a set of files containing the definition of the coupled simulation model, e.g. the MpCCI input file and the MpCCI log files. Consequently, before you start the MpCCI GUI, you should move to a directory where you have permission to create files. You execute MpCCI GUI by running the `mpcci` executable with the `gui` subcommand and the desired options.

Usage:
```
mpcci gui [OPTIONS] [project] [OPTIONS]
```

Synopsis:
```
'mpcci gui' is used to launch the MpCCI GUI.
```

Options:
- `chw`d `<PATH>`
  Replace the symbolic working directory $(CWD) used inside the project file by the absolute pathname specified in the `<PATH>` argument.

- `help`
  This screen.

- `new`
  Start the GUI with a new project.

- `nolic`
  Do not check for a license before starting the GUI. This option may be used when no license is available but you would like to prepare a job.

- `norsa`
  Do neither check for ssh nor ask for ssh assistance if an rsa key file does not exist.

- `useAbsolutePath`
  Use current local MpCCI installation path on remote access.

To start an MpCCI job in batch mode with a well prepared coupled simulation project use:
```
mpcci -batch <project name>
```

If you start the MpCCI GUI without any options like
```
mpcci gui
```
the following default checks are performed:
- Check and preparation of your environment.
• A check of your secure shell rsa/dsa key files. If no rsa/dsa key files are found you will be asked to create them.
• Check for an existing project file in the current directory in order to load that project to the MpCCI GUI. If more than one project file exists no project will be loaded.
• Check of the license environment. MpCCI GUI tries to check for a license e.g. MPCCI_LICENSE_FILE and also informs you when your license will soon expire. In order to disable it use the option -nolic.

If you do not include the <project name> to the mpcci gui the MpCCI GUI starts at the initial step.

4.1.2 Exiting MpCCI GUI

You can exit the MpCCI GUI session at any time by selecting File → Exit from the main menu bar, by using the key shortcut Ctrl + Q or by closing the main window of the MpCCI GUI. If you made any changes to the current coupled simulation project, the MpCCI GUI asks if you want to save the changes before exiting the session. MpCCI GUI then closes the current coupled simulation project file and exits the session.

4.2 MpCCI GUI Properties

Some settings in the MpCCI GUI are global and will persist till the next execution of the MpCCI GUI. These settings will be saved in the MpCCI properties file 

"<your home>/mpcci/coupling_environment/preferences.xcf".

Actually following properties are saved:
• Properties listed under the menu Edit → Properties (see 4.3.2 Edit Menu ●),
• The defined rules for region building (see 4.6.5 Automatic Generation of Regions by Rules ●) and
• The selected columns to be viewed in the region properties table (see 4.6.6 Applying Region Properties ●).

All persisting settings are mentioned at their description.

4.3 MpCCI GUI Menus

Before talking about the menus let’s have a look at Figure 1 the title of the MpCCI GUI. The title of the main window of the MpCCI GUI contains the following information:
• The name of the currently coupled simulation project or “noname” if no project is specified
• The version of MpCCI you are running

![Figure 1: MpCCI GUI title and main menu](image)

Below the title we have the MpCCI GUI menus. They are available all the time while MpCCI GUI is running. The main menus, also shown in Figure 2 are:

File for creating, opening, saving and exiting a coupled simulation project.
Edit for editing global MpCCI GUI properties.

Batch for submitting, querying status and killing a coupled simulation project from a batch queuing system like LSF, PBS, SGE, GLOBUS, etc. (see also 3.6 Coupled Analysis in Batch Mode)

License for managing the MpCCI licenses and displaying detailed license information.

Tools for launching various MpCCI commands directly from the MpCCI GUI.

Codes for adding and removing code selections and for providing code specific commands. If at most three codes are available the codename menus are directly shown in the menu bar and the entries in the Codes menu disappear.

Help for requesting help and for getting information about the MpCCI GUI.

The menu entries are:

### 4.3.1 File Menu

File→New Project creates a new project. The MpCCI GUI will be initialized with its default application values and the first step which is the Models step.

File→Open Project opens a project file. A file chooser pops up and you are able to select one of your previously saved projects. The project will be opened at the last step you were.

*File-Types: “Coupled Simulation Project” - Files (*.csp)*

File→Save Project saves the project. The current project settings are saved under the current project name. This command is available when a project configuration has been opened or if the project already has been saved via File→Save Project As.

File→Save Project As saves the project settings under a specified name. The MpCCI GUI displays the Save As dialog box. The active project can be renamed and saved to a new directory.

File→Exit ends the MpCCI GUI session. MpCCI GUI prompts to save the project with unsaved changes before terminating the application.

### 4.3.2 Edit Menu

Edit→Properties allows to set preferences for global MpCCI GUI properties (see 4.2 MpCCI GUI Properties).

Auto Save Project toggles the option for automatically saving a project with unsaved data. This property is regarded before checking the application configuration and starting the coupled simulation in the Go step. This option will also be set via the Unsaved Project Data dialog shown in Figure 35. It is not taken into account when exiting the MpCCI GUI.
4.3.3 Batch Menu

**Batch→Submit Batch Job** submits the current project to the batch queuing system. A configuration dialog pops up and shows the available batch systems in a list. By selecting one of the batch systems you will be able to configure some options for the job. The configuration dialog is mostly divided into three parts:

- **Application specific host allocation:** For each application you are able to provide either a list of hostnames or a regular Perl-pattern. This pattern will automatically extract the hosts from the batch environment.

- **Batch system selection:** You can select a queuing system to use.

- **Specific batch system options:** You find a list of options to set up for example the queue, the job name, the output filenames, etc.

**Batch→Status Batch Job** queries the status of a batch job. You are able to select one of your previously submitted jobs referenced by its job id (see Figure 3).

![Figure 3: Batch status dialog](image)

**Batch→Kill Batch Job** kills a batch job. You are able to select one of your previously submitted jobs referenced by its job id (see Figure 4).

![Figure 4: Batch kill dialog](image)

4.3.4 License Menu

- **License→Display license servers** lists all defined license servers.
- **License→Display license files** lists all local license files defined.
- **License→Check the MpCCI license status** displays MpCCI license features and available tokens.
- **License→Check all available licenses** displays all license features and available tokens.
- **License→MpCCI license expiration date** displays the expiration date of the MpCCI license.
License→Start MpCCI license server  starts the FHGSCAI vendor daemon on the local host.
License→Stop MpCCI license server stops the FHGSCAI vendor daemon running on the local host.

### 4.3.5 Tools Menu

Tools→Print environment variables  shows a dialog box with the environment used by MpCCI.
Tools→Display the current process list  shows a dialog box with a list of the current processes running on the local system.
Tools→Display the process list  shows the processes running on the local machine. On Windows it runs the `taskmgr` command and on UNIX systems it runs the `top` command.
Tools→Test  Submenu providing test commands delivered with MpCCI (see Figure 5).
  →MpCCI host connect...  Test the connection to a specific port@host. Opens a new dialog for entering the port number and host name.
  →MpCCI host listen locally...  Test for listening on a specific port@localhost. Opens a new dialog for entering the port number. This command correlates with MpCCI test host connect... First start this listen command and after that the connect command with the same port and with host localhost.
  →MpCCI rsh type connection...  Test a rsh type connection to a specific host with MpCCI. Opens a new dialog for entering the host name.
  →MpCCI simple test case  Run a simple test case delivered with MpCCI (see III-7 A Simple Test on the Local Machine, “Hello World”).
Tools→Logviewer  starts the MpCCI Logviewer.
Tools→Monitor  starts the MpCCI monitor.
Tools→Visualizer  starts the MpCCI Visualizer.

![Figure 5: Tools→Test submenu with MpCCI test commands.](image)

### 4.3.6 Codes Menu

Codes→Codename  Provides code specific commands which are defined in the respectively code configuration file (see VIII-2.4.2 Codes Menu: <CodesMenuEntries>).

### 4.3.7 Help Menu

Help→MpCCI Documentation  displays a help window with the MpCCI documentation.
Help→About  displays product information and third party products integrated in the MpCCI GUI.
4.4 Models Step

The MpCCI Models step is the first step for setting up a coupled simulation. Its aim is:

- Select the codes involved in the coupling.
- Select their corresponding model files.
- Extract the components needed to create the coupling regions which is done in the Regions step.

Selecting a model file will immediately launch the scanner to extract the components. At that time you don’t need to call the scanner manually. But if you have changed components in the selected model file later on, the changes must be passed on to MpCCI GUI. To do this, the scanner must be started manually by pressing the Start Scanner button. The components are extracted again and are then available in the Regions step.

4.4.1 Code Parameters

Additionally some codes may provide some further parameters to be set by the user: e.g. the version or release of the code to run or the unit system to be used for the grid length and quantities. For a description of the code specific parameters have a look at the Codes Manual.

4.4.2 Requirements

In order to proceed with another step the following requirements have to be fulfilled in this Models step:

- Selection of at least two codes for coupling.
- Setting of all required parameters for each code especially the model files.
- Successful run of the scanner for each code.
- Usage of the same model dimension by the coupled codes. Different dimensions lead to a warning message whereas undefined dimensions are treated to be proper.
- Usage of appropriate solution types by the coupled codes. The solution types will be found by the scanner and may be Steady state, Transient or undefined. Different types in the model files lead to a warning message whereas undefined types are treated as being proper.

All these criteria are checked when leaving the Models step and a dialog with an appropriate message will pop up if the requirements are not fulfilled. In case of a warning the user may decide to continue with the chosen step although it is not recommended by MpCCI. If an error occurs the user can’t go on before having corrected the complained items.

4.4.3 Changing the Model - Implications for the Setup

Changing the model file for a code leads to new components and normally implies a new setup of the coupled simulation with all regions and monitors being removed.

But sometimes it might be desirable to keep the setup while doing only some small changes to the model e.g. renaming, splitting or uniting components. Therefore MpCCI compares the components of the new scanned model file with the actual setup.

- If all actually coupled components are still in the new component list, the setup will not change. Old uncoupled components which do not exist anymore will be automatically removed from the code and new components will be added to the code components list.
- If at least one old coupled component doesn’t exist anymore a component conflict dialog (see Figure 6) appears where the user has the choice among
– Keeping all sets with the omitted components removed or replaced by new components if existing
– Removing sets with omitted components but keeping not affected sets
– A new setup with all regions and monitors removed

Figure 6: Component conflict after changing the model

The replacement of old components will be done in a special dialog (see Figure 7) where the user can choose

• One old component and one new component (1:1 replacement renaming a component)
• One old component and several new components (1:n replacement splitting one old component into several new components) or
• Multiple old components and one new component (n:1 replacement uniting several old components)

Figure 7: Replacing components in a component conflict

Only those new components will be offered as selectable components which are of the same type as the old components. Old components which are not replaced will be removed from the sets, new remaining components will be added to the code components list.

If an old component with copies is replaced, the copies are transferred to the replacing component. If several components for replacement are selected, only the first component gets the copies, the others
remain without copies. When a copied component is replaced it loses its copy state and becomes the new stand-alone component.

### 4.5 Algorithm Step

In the MpCCI Algorithm step you define the coupling algorithm (see IV-2.5 Algorithm Step – Defining the Coupling Algorithm). The settings are separated into Common Basics which define the global view and Codes Specifics defining the code’s local view of the algorithm. The algorithm is organized into following subject areas:

- Analysis
- Solver settings
- Coupling steps
- Coupling algorithm (in common basics only)
- Coupling duration (in common basics only)
- Runtime control (in code specifics only)

Only those settings will be visible which are relevant for the current configuration. E.g., the solver settings area in the common view is only visible for iterative coupling. The coupling algorithm and duration are global so they are only shown in the common basics view, whereas the runtime control refers to each code.

#### 4.5.1 Common Basic Algorithm Settings

![Common Basic Algorithm Settings](image)

Following common basic algorithm parameters can be configured (see also Figure 8):

**Coupling analysis**

- **Analysis type** shows the type of the coupling analysis depending on the analysis type of each code.
Transient if all coupled codes do a transient analysis (see \(\triangleright 3.4\) Coupling Process\(\langle\)).

Steady state if all coupled codes do a stationary analysis (see \(\triangleright 3.4\) Coupling Process\(\langle\)).

Quasi transient if the coupled codes use mixed analysis types - one transient and the other steady state (see \(\triangleright 3.4.6\) Coupling with Mixed Analysis Types\(\langle\)).

Define the coupling scheme for solving the coupled problem:

Explicit is available for all analysis types.

Implicit defines an iterative coupling for transient analysis (see \(\triangleright 3.4.5.1\) Implicit Coupling\(\langle\)).

It is a feature which is not supported by all codes and only offered when all coupled codes support the iterativeCoupling feature.

Solver settings is only offered for implicit coupling where all codes have to trigger a boundary update at the same time.

Type of solver step size specifies whether the time step size is constant or adaptive.

Constant defines a constant time step size in seconds. The value has to be given in the appearing field Constant solver step size (s).

Sent by code defines an adaptive time step size which will be exchanged by the codes via a global variable (see \(\triangleright 3.4.5.2\) Coupling with Exchange of Time Step Size\(\langle\)). The definition of a global variable can be looked up in \(\triangleright 4.6.1\) Global Variables\(\langle\).

Negotiation defines an adaptive time step size which will also be exchanged by codes. But in this case several codes send the step size and the smallest one will be propagated by the MpCCI server. This is a feature which is not supported by all codes and only offered when all coupled codes support the negotiation feature. It is directly implemented by the codes so no global variable has to be specified.

Initial solver step size (s) defines the step size used for the first solver step when the codes haven’t already negotiated a step size.

Minimum solver step size (s) specifies a minimum value for the solver step size to prohibit the usage of a step size which may be too low. Even if the codes negotiate a lower solver step size this minimum step size will be taken.

Coupling steps define the data exchanges of the codes (see \(\triangleright 3.3\) Data Exchange\(\langle\)) - how they are specified by the codes or if they correlate with the solver steps. The options depend on the chosen coupling analysis.

Type of coupling step size is offered for explicit analyses only.

Sent by code defines an adaptive coupling step size and can only be chosen for an explicit transient analysis. The time between two data exchanges is sent by one code via a global variable (see \(\triangleright 3.4.5.2\) Coupling with Exchange of Time Step Size\(\langle\) and \(\triangleright 3.3\) Data Exchange\(\langle\) for more information).

Defined by each code can be chosen for all explicit solutions and is the only option for Quasi transient and Steady state analyses. It says that each code defines its own time or iteration number for the boundary updates. For transient problems this leads to non-matching coupling steps where the exchanged values have to be interpolated.

Coupling step size shows how the point of data exchange is determined.

Every solver step is the fixed setting for implicit coupling. No other options are possible. The value of the solver step size is repeated for a better overview.

Global quantity sent by code is the fixed setting for type Sent by code.

Order of interpolation is offered for transient problems with non-matching coupling steps (see also \(\triangleright 3.4.5.3\) Coupling with Non-Matching Time Steps\(\langle\)). In this case the interpolation order for the exchanged values can be specified as follows:

Higher order is a cubic Hermite spline interpolation using up to four available quantity values and finite difference schemes to approximate the necessary tangential values.

First order uses one value before and one after the requested value for a standard linear interpolation.
Zero order is a constant interpolation function using the last available value.

Idle time reduction offers the possibility to save idle time if one code runs much faster than the other one but only for Quasi transient and Steady state analyses and only for codes which do a stationary analysis and support the delayed boundary update.

Delay boundary updates has to be selected for using this feature (see Figure 9). The faster code may go on in its computation without updating its boundaries at the end of a coupling step. It then tries to do the update at a given substep frequency or at a given number of update tries within the next coupling step. The update will be delayed by no more than one coupling step.

Implementing code offers codes supporting the delayed boundary update in the current setup. Select the faster code.

Delay definition defines whether the delay is specified by a number of update tries (Number of update tries) or by a substep frequency (Number of substeps).

Type of delayed updates defines a Constant or Variable delay value. A variable type allows different number of update tries resp. substep frequencies per coupling step whereas for a constant type the Delay value remains the same for all coupling steps.

Delay value specifies the constant value resp. variable values for the delayed update. Variable values are a list of couplingStepNo=delayValue pairs each defining the coupling step number couplingStepNo from where the delay value will be set to the given value delayValue. This value remains until a new coupling step number with another value is specified or the end of the coupling is reached. The couplingStepNo=delayValue pairs are separated by whitespace (e.g. 1=3 3=5 10=7).

Coupling algorithm specifies the algorithm for the data exchanges used in the coupled simulation (cf. \textit{3.4.1 Coupling Algorithm}).

Algorithm type specifies a serial (Gauss Seidel) or parallel (Jacobi) algorithm.

Serial needs the leading code to be specified in the field Leading code.

Parallel can be selected with or without initialization. If initialization is used check the appropriate field and select the Initializing code.

Coupling duration specifies the duration of the coupled simulation without runtimes of the codes before and after the coupling. The settings depend on the analysis type whether it is transient or stationary. Quasi transient is a mixed analysis and therefore includes settings for both types.

For transient analysis:

Maximum number of coupling step iterations can be defined for implicit coupling only. It specifies the maximum duration of the coupling step iterations and is used if no other criterion is met to stop the iterations earlier.

Set total coupling time allows to define a maximum time for the coupling process. If no total coupling time is defined the coupling will end when the convergence criteria are achieved.

Total coupling time will not be exceeded for coupling but if other criteria are fulfilled i.e. quantities are converged, the coupling might end earlier.

For steady state analysis:

Minimum number of coupling steps can be defined. This can be useful if the coupling starts with very small changes in the quantities leading to a converged solution already after the first coupling step. In order to prohibit this early coupling end enter a value greater than one.

Set maximum number of coupling steps allows to define a maximum number of coupling steps for the coupling process. If no maximum is defined the coupling will end when the convergence criteria or total simulation time are achieved.

Maximum number of coupling steps will not be exceeded for coupling but if other criterions are fulfilled i.e. quantities are converged, the coupling might end earlier.
4 Graphical User Interface

4.5.2 Code Specific Algorithm Settings

The code specific algorithm parameters may differ slightly from code to code. The most common entries are described below. Additional code specific settings are explained in the respective chapter of the Codes Manual. Please look there for the exact entries of each code.

Following most common entries for the code specific algorithm parameters can be configured (see also Figure 10):

**Analysis type** shows the type of the coupling analysis of this code and is either Transient for a transient analysis or Steady state for a stationary analysis. It is preset depending on the solution type of the model. If the solution type is undefined, both analysis types are offered and the user can choose one.

**Solver settings** include the solver step size, which is mostly Defined by model or for implicit coupling already specified in the common basic settings. Other solver specific entries can be looked up in the Codes Manual.

**Coupling steps** define the frequency of the data exchange. Normally it is given in the number of solver steps where more than one solver step leads to subcycling. The step size can be constant or variable. A variable step size allows different number of solver steps per coupling step whereas for a constant type the coupling step size remains the same for all coupling steps.
**Type of coupling step size** defines if the coupling step size remains constant or variable over the computation.

**Constant coupling step size (solver steps)** defines the constant number of solver steps without coupling. For transient simulation, the elapsed time between two data exchanges depends on the solver step size used. Only the solver step count will trigger the data exchange.

**Variable coupling step size (solver steps)** enables to set the number of solver steps without coupling depending on the coupling step number. For Variable coupling step size (solver steps) MpCCI accepts a list of couplingStepNo=nofSteps pairs each defining the coupling step number (couplingStepNo) from where the number of solver steps without coupling will be set to the given value nofSteps. This value remains until a new coupling step number with another value is specified or the end of the coupling is reached. The couplingStepNo=nofSteps pairs are separated by whitespace.

⚠️ For Implicit coupling the coupling step size is constant and restricted to the common solver step size. Subcycling is not possible.

**Runtime control** specifies the runtime of the codes before and after the coupling.

*For transient analysis:*

**Define coupling start** defines the time in the model when the coupling process should start.

*Take latest time* starts the coupling at the latest time specified in the model. This enables a model that has already been calculated, without specifying the current time of the model. If the precalculated time of the model changes, the time for starting the coupling in the MpCCI GUI is still valid using *Take latest time.*

*Specify time* starts the coupling at the point in time of the model which is specified in the field *Time of coupling start.* This enables some uncoupled calculations to be performed before the coupling starts.

*Time of coupling start* specifies the model specific point in time for starting the coupling.

When restarting from an interrupted coupled simulation, the start time of the coupling must be specified explicitly, since the latest time in the model is no longer the start of the coupling. The start time can be 0 s or later.

*For steady state analysis:*

**Define coupling start** defines the iteration number in the model when the coupling process should start.

*Take latest iteration* starts the coupling at the latest iteration number specified in the model. This enables a model that has already been calculated, without specifying the current iteration number of the model. If the precalculated number of iterations of the model changes, the iteration number for starting the coupling in the MpCCI GUI is still valid using *Take latest iteration.*

*Specify iteration* starts the coupling at the iteration number of the model which is specified in the field *Iteration number for coupling start.* This enables some uncoupled calculations to be performed before the coupling starts.

*Iteration number for coupling start* specifies the model specific iteration number for starting the coupling.

When restarting from an interrupted coupled simulation, the iteration number for starting the coupling must be specified explicitly, since the latest iteration number in the model is no longer the start of the coupling. The iteration number for coupling start can be 0 or higher.

**Number of iterations after the coupling** allows to define some iterations to be performed after the coupling has ended.
4.6 Regions Step

In the MpCCI Regions step you define the exchange of global variables in the Global tab and create and configure your coupling regions in the Mesh tab by

1. Editing components to be prepared for coupling.
2. Building coupling regions.
3. Defining quantities which will be exchanged.
4. Assigning the quantities to be transferred to the built coupling regions.

Additionally the MpCCI Regions step provides an overview over the coupled setup for the mesh based components.

The basic steps are described in IV-2.6 Regions Step – Defining Coupling Regions and Quantities. Additional features are described in the following:

4.6.1 Global Variables

![Figure 11: Defining global variables](image)

The definition of sets with global variables to be exchanged differs a little from that of mesh based coupling regions. Figure 11 shows the Global panel. The sets are defined directly with the coupled components and assigned quantities in the respective area.

Sets area provides a list of defined global variable sets. These sets are handled like regions for mesh based components (see 2.6.1 (part IV) for a description)

Components area with lists of global variables (called components) offered by each code. These lists are similar to those for mesh based components (see IV-2.6.1.1 Select Components for Each Interconnected Code). Their type is depicted by the label \( \gamma \). Global variables can only be coupled 1:1.

Quantities area provides a list of available quantities and their configuration. The selection of the quantities is analogous to those for mesh based quantities (see IV-2.6.2.1 Select Quantities to be
Transferred\(^a\). But for global variables only one quantity may be selected per set which is why they are selected by radio buttons instead of checkboxes.

### 4.6.2 Editing Components

In the *Edit Components* tab for mesh based components (see Figure 12) the components scanned from the model file can be prepared further for the coupling process. Following choices are offered:

- Copying components if the code allows this feature.
- Editing component properties like component name, its part-id or additional auxiliary information.
- Defining periodic component properties.

The component’s popup menu with the entries *Edit Properties*, *Copy* and *Delete* is also available in the components lists of the *Build Regions* tab.

#### 4.6.2.1 Copying Components

To use the component in another region, too, first your code has to provide the *CopyComponents* option in the *Code Information* section of its MpCCI GUI configuration file (see △ VIII-2.4 *MpCCI GUI Configuration File gui.xcf*\(^a\)).

After that you can choose *Copy* from the component’s pop-up menu (see Figure 12, also available in the code components lists of the *Build Regions* tab) to get a new component. This component then can be used in addition to the original component.

Only components determined by the scanner can be copied. Copied components cannot be copied again, but they can be deleted which is not possible for scanned components.
4.6.2.2 Editing Component Properties

The pop-up menu in the components list (Figure 12) provides an Edit Properties entry (also available in the code and coupled components lists of the Build Regions tab) which launches a dialog for editing some component properties (see Figure 13).

In the upper part of the dialog the component’s code and type are listed. If it is a copied component, also the name of the original component is mentioned, for periodic components (▷4.6.2.3 Periodic Components◁) also the assigned periodicity.

The editable fields are the component’s name, its part identification number and the additional aux information.

With Set the new values will be overtaken. In doing so the MpCCI GUI checks the new name which it would refuse if it already exists for another component. Cancel dismisses the input.

4.6.2.3 Periodic Components

Sometimes a model may only consist of a cyclic symmetric part and the whole component can be built by replicating the one part. MpCCI handles the quantities on the periodic section during the coupling process as described in▷3.3.6 Quantity Transformation for Cyclic Symmetric Meshes◁.

The periodicity of the coupled components needs a description of how to reconstruct the whole component out of the partial model. This can be done via the Periodicity tab (see Figure 14) shown in the right area of the Edit Components tab (see Figure 12).

The periodicity panel consists of

1. A global part at the top and
2. A periodic specific part underneath. This part is divided into three areas:
   2.1 A list of defined periodicities on the left,
   2.2 A specification area for the selected periodicity on the upper right and
   2.3 An action area on the lower right.

The specification of a periodicity is done in the specification area (2.2) by defining the parameters:
Figure 14: Periodicity panel

- **Number of periodicity** which numbers the replication factor for getting the whole part. Its minimum value is two.
- **Rotation-axis origin** specified by its X, Y and Z coordinates and
- **Rotation-axis direction** also specified by a three-dimensional point. It is interpreted as a vector starting at the rotation-axis origin which is why it must not be (0,0,0).

The periodicity list provides an **Add** button for getting new periodicities to be specified.

**To assign a periodicity** select the appropriate components - even from different codes - and click on **Apply periodicity to selected components** from the pop-up menu of the selected periodicity (see Figure 15) or from the action area (2.3 in Figure 14).

Only components of dimension Face or Volume can be applied a periodicity.

Selecting and deselecting single components can be done by holding down the **Ctrl** key when clicking on a component.

**To delete or rename a periodicity** choose the appropriate option from the periodicity pop-up menu (see Figure 15, Delete periodicity and Rename periodicity). Deleting a periodicity leads to removing it from all assigned components. Renaming will also affect assigned components.

**Identifying periodic components** is done in two ways. Firstly those components get a special label:

- **Marks a periodic Face element**
Marks a periodic Volume element

And secondly the periodicity panel provides options for marking components. These lead to a visual representation of the periodicity name in the components lists (see Figure 16):

- **Mark components with this periodicity only** marks components with the selected periodicity. This option can be checked in the action area 2.3.
- **Mark all periodic components** marks all components assigned any periodicity. This option can be checked in the global part (1) of the periodicity panel as shown in Figure 16.

Selecting components with a specific periodicity is done by selecting that periodicity and choosing **Select components with this periodicity** which is available in the periodicity’s pop-up menu (Figure 15) and also in the action area 2.3 of the periodicity panel (Figure 14).

Removing periodicity from components is part of the global options (1 in Figure 14) which regard all periodicities and so do not reference the currently selected one:

- **Remove periodicity from selected components** removes any periodicity from all selected components.
- **Remove periodicity from all components** removes any periodicity from all components in all code lists. After that no periodic components will exist anymore.
To remove a specific periodicity from all components

1. Select the periodicity to be removed
2. Choose Select components with this periodicity
3. Choose Remove periodicity from selected components

### 4.6.3 Coupling Components with Different Dimensions

If you want to couple components with different dimensions, there are some restrictions regarding the quantities which may be exchanged. Quantities can only be sent from the components with the higher mesh elements dimension to the components with the lower mesh elements dimension.

So after building the coupling region with e.g. 1-dimensional line elements from code `Code_1` and 3-dimensional volume elements from code `Code_3`, the Define Quantity Sets panel provides only those quantities for the selected coupling dimension (here Line — Volume) fulfilling the above mentioned condition:

- `Code_3` offers the exchanged `quantity` for dimension `volume` to be `sent`.
- `Code_1` offers the exchanged `quantity` for dimension `line` to be `received`.

This implies a unidirectional coupling for coupling regions with components of different dimensions.

An exception is coupling components of dimension point and integration point. In this case quantities can also be sent to components with higher mesh elements dimension.

### 4.6.4 Simplified Selection

#### 4.6.4.1 Simplified Component Selection

The selection of components out of a large list sometimes may be confusing. To simplify this, following areas exist:

- A selection area for selecting components by their name and
- An options area allowing synchronized scrolling and automatic component selection

![Component selection area](image)

Figure 17: Component selection area

The component selection area (see Figure 17) is located at the top of each components list. It consists of a textfield where a regular expression pattern can be given to match the name of the uncoupled components in the list below. Via the Select button all components matching the given pattern will be selected. Be sure to enter a valid regular expression as a pattern (e.g. ".*" to match any character). Syntax errors will be reported but logical errors might be harder to find.

#### 4.6.4.2 Simplified Region Selection

Dealing with a large number of regions may require to select them by their name or also the name of their contained components. For this an area for simplified region selection (see Figure 18) is located at the top
of several region lists namely in the tabs Build Regions, Assign Quantity Sets and Overview.

![Figure 18: Area for simplified region selection](image)

This area provides a textfield where a regular expression pattern can be given to match the name of the regions in the list below. If the Extend search field is marked, the search will be extended and include the names of the regions’ components, too. Otherwise the patterns will be applied to the name of the regions only. Via the Select button those regions matching the given pattern will be selected.

Be sure to enter a valid regular expression as a pattern (e.g. “.*” to match any character). Syntax errors will be reported but logical errors might be harder to find.

### 4.6.5 Automatic Generation of Regions by Rules

If you have specific component names which may be considered for building coupled regions, you may define rules for a fast automatic region building process. Figure 19 shows an example list of rules in the rules area which is located in the Build Regions tab of the mesh based components panel below the regions area (see Figure 9 (part IV)).

![Figure 19: Rules area for automatic region generation](image)

The defined region rules are saved in the MpCCI properties file (see 4.2 MpCCI GUI Properties) and remain available for later runs of the MpCCI GUI until they are deleted.

**To define a new rule** click on the New button in the rules area which leads to a dialog for setting the rule properties (see Figure 20).

Following properties define a rule:

**Rule name** The name of the rule.

**Region criterion** The criterion for finding components fitting to this rule. Every string comparison is done case-insensitive. The region criterions are:

- **Exact** Equally named components build a region. If components have automatically been generated by the scanner, the original name will be taken for comparison. See UserInfo in VIII-2.5.2 Scanner.pm how automatically generated components are specified.

  This region criterion results in building several regions mostly with exactly one component from each code. But if automatically generated components exist, there may be more than one component from one code included.
Matching pattern Components which name match the given pattern build a region. The pattern is handled as a regular expression. If the matching pattern could not be found in the component names, no region will be generated.

First no. matching characters Components which are named equally at the beginning build a region. The number of the first matching characters (n) has to be entered. This region criterion may result in several regions each containing a group of components equally named by the first n characters where one component’s name may be prepended by a prefix like for the Exact criterion.

Only components of code If selected, only components of the selected code are added to the resulting regions.
This criterion may be selected for all region criterions mentioned before except Exact which always uses components of all codes for its execution.

Region type Specifies whether the resulting regions shall be tagged as unused or not.

Region naming Specifies the naming for the resulting regions. The names will be constructed as follows:

Prefix This prefix extended by a serial number for each built region. This is a general naming which doesn’t provide insight to the components of the generated region.

Component of code Name of a randomly chosen added component of the selected code. This is useful when using the exact name matching criterion where exactly one component of each code is added. So the name of the regions clarify their contained components.

Matching pattern The matching pattern which led to the selection of the components. This naming is recommended for all region criteria because it allows greater insight into the composition of the built regions. Depending on the chosen region criterion following region names will be constructed.
For Exact criterion the regions take the name of the components’ name.
For Matching pattern criterion the resulting region will be named by the given pattern.
For First no. matching characters the regions will be named by the n first equal patterns.

Save the rule by clicking the OK button. Cancel or closing the dialog will dismiss the property settings and won’t create a new rule resp. leave the properties unchanged for an edited rule.
Editing a rule is done by a double click on a rule in the rules list or by selecting one rule and clicking the [Edit] button in the rules area (see Figure 19). After that the dialog for setting rule properties opens (see 4.6.5 To define a new rule).

Deleting rules is done by selecting the rules and clicking the [Delete] button in the rules area (see Figure 19). A dialog pops up to confirm the deletion. If only one rule is selected and the rule is part of a rules list (see 4.6.5.1 Rules Lists), the user will be informed and has to confirm the removal from that list, too. On the other hand several selected rules will be automatically removed from affected rules lists. Rules lists which emptied out after deleting rules will also be deleted.

Generate regions by selecting one rule and clicking the [Generate] button in the rules area (see Figure 19). The rule will be applied to the code components lists selected in the Components area (see Figure 9 (part IV)). Therefore be sure to select the right component dimension list if more than one element type exists. The generated regions appear in the region list and a dialog will show the results of the applied rule.

Exporting rules and rules lists is done by clicking the [Export] button in the rules area (see Figure 19). In the upcoming file chooser dialog provide a file name for saving the definitions of all defined Rules and Rules Lists.

Importing rules and rules lists is done by clicking the [Import] button in the rules area (see Figure 19). In the upcoming file chooser dialog select the file containing the rule definitions (i.e. a file with exported rules and rules lists). If some imported rule names are identical to existing ones, a dialog will appear (see Figure 21) and the user will need to choose an option on how to handle the doubly defined rule name: Replace, Copy or Skip.

Choosing one of the options Replace all, Copy all or Skip all will automatically apply the chosen option to all remaining name conflicts of this import.

Cancel will stop importing this and any further rule and rules list.

Figure 21: Import dialog for solving rule name conflicts

4.6.5.1 Rules Lists

The defined region rules can be concatenated so that one rule after another will be automatically applied by one call. These rules lists are accessed by the Rules Lists tab in the rules area (see Figure 22).
Handling rules lists is similar to that of rules. The action buttons New, Edit, Delete and Generate are the same. The defined region rules lists are also saved in the MpCCI properties file (see 4.2 MpCCI GUI Properties) and remain available for later runs of the MpCCI GUI until they are deleted. They can be exported and imported together with the rules as described in 4.6.5 Exporting rules and rules lists and 4.6.5 Importing rules and rules lists.

Editing a rules list, while creating a new one via New, editing an existing one via Edit or double clicking a rules list, is done via the dialog shown in Figure 23. A rules list is defined by its name, some region creation options and the list of rules. Therefore the dialog offers following input and editing options:

**Rules list name** The name of the rules list.

**Region creation** Options for region creation:

- **Generate separate regions** Creates regions from each applied rule. The regions are named as defined by the applied rule.
- **Generate one merged region** Creates only one region as a result of the applied rules. The resulting region will be named after this rules list name.
by reduction Each rule will be applied to the resulting components of the rule before. I.e. the first rule at the top will be applied to all uncoupled components, the second rule to the components selected by the first rule, and so on. The components selected after applying the last rule build the new region.

by join The components selected by each applied rule are collected and put together into one region. This compares to Generate separate regions but joining all regions to one.

Available rules List of rules which are available meaning listed in the Rules tab and not already used in this rules list.

Rules list List of rules building this rules list. The rule at the top will be the first one executed continued by the next one up to the last one at the bottom of the list.

Adding and removing rules is done either by double clicking the respective rule which moves it to the appropriate other list or by using the direction buttons arranged between the two rules lists.

 Moves the selected available rules to the end of the rules list.

 Moves the selected rules from the rules list. They will be sorted into the available rules and can be added again.

Using the direction buttons allows to handle more than one rule. The lists allow multiple selection by using the Shift or Ctrl keys or selecting all rules by using the Ctrl+A shortcut.

Editing the rules list is done with the buttons at the right of the rules list.

 Moves the selected rules to the top of the list.

 Moves the selected rules one position to the top.

 Moves the selected rules one position to the bottom.

 Moves the selected rules to the bottom of the list.

Save the rules list by clicking the OK button. Cancel or closing the dialog will dismiss the editing and won’t create a new rules list resp. leave the rules list unchanged.

Deleting rules lists is done by selecting the rules lists and clicking the Delete button in the rules area (see Figure 22). A dialog pops up to confirm the deletion.

Generate regions by selecting one rules list and clicking the Generate button in the rules area (see Figure 22). Each rule of the rules list will be applied as described in 4.6.5 Generate regions. A dialog will show a summary with the results of the applied rules.

4.6.6 Applying Region Properties

The attributes for the relation search are globally defined among the control parameters for MpCCI in the Settings step (see 4.8.3 Relation Search where they are described in detail). These properties can be overwritten for selected regions in the Region Properties tab of the Build Regions panel shown in Figure 24.

Special relation search parameters for several regions are set by picking the appropriate regions in the table and applying the desired values to the picked regions:

• Picking is done either by checking the pick box of a region in the table or in the Regions area by selecting the appropriate regions in the table and then using the Apply button for the pick flag. This applies the pick flag value to the selected regions.

• Applying relation search values is done by setting the desired values in the For picked regions area and using the Apply button of the desired property to apply its values to the visible picked regions (the selection of the regions is irrelevant and will not be regarded, only the pick flag is the clincher). After that the new values are assigned and will be displayed in the table.
Beyond that the Region Properties tab offers following possibilities:

- Filtering the list of regions. Only regions matching the specified filters will be shown in the table. The Apply filters area offers the filters

  **Pattern** For specifying a regular expression pattern which will be applied to the names of the visible regions.

  **Picked** For specifying whether to show only picked regions.

- Configuring table column visibility. The Columns area allows to configure which columns of the table are visible:

  **All** All table columns are shown.

  **Minimum** The at least necessary columns are shown.

  **Custom** The user selected columns are shown.

  This columns choice will be saved in the properties file of the MpCCI GUI (see 4.2 MpCCI GUI Properties) to remain this way till next run of the MpCCI GUI.

- In the Components area lists of all coupled components of the selected regions are shown.

- Selecting all visible regions is offered by the **Select all** button.

- Deselecting all visible regions in the table is offered by the **Clear selection** button.

- Resetting the pick flag for all regions is offered by the **Reset applied flags** button. Thereby all pick flags will be deselected even the ones of invisible regions.

The region column offers a popup menu with following entries:

**Copy region name into clipboard** so that the region name can be pasted anywhere else.

**Unused** marking the selected regions to be unused. This causes that the affected regions will be removed from this list. In this list only *ready* regions are listed, incomplete and unused regions aren’t.
4.6.7 Quantity Specifications

4.6.7.1 Quantity Sender

Sometimes a quantity can be sent and received by two or more codes. In that case the user has the option to select the sender of the quantity in the Sender part of the quantity configuration as shown in figure Figure 25.

![Selection of the sender/receiver for a quantity](image)

Figure 25: Selection of the sender/receiver for a quantity

4.6.7.2 Default Value for Global Variables

For the quantities of global variables default values can be specified for the sender. This value will be sent when no computed value is available. The setting is done in the Settings area at the bottom of the quantity configuration (see Figure 11).

4.6.7.3 Orphans Settings for Mesh Based Components

The assignment of values to orphans can be configured in the Orphans settings area of the quantity configuration (see Figure 26) by

![Quantity configuration areas](image)

Figure 26: Quantity configuration areas

**Use mean value** The mean value as an average of the values for this quantity will be set into the orphaned regions.

**Use fix default value** The given fix default value for the quantity receiver will be set into the orphaned regions.
Extrapolate is an additional option for handling values for orphaned regions. If selected, values will be extrapolated into the orphaned regions where this is possible. For orphans which cannot be extrapolated a mean or default value will be set depending on its selection. If extrapolate isn’t selected, all orphaned regions take the mean resp. default value.

4.6.7.4 Applying Operators to Quantities of Mesh Based Components

The MpCCI server provides some operators in the Apply operators area which can be applied to the transferred quantities (see Figure 26). Each operator incorporates a description, a type and usually parameters. By clicking on the operator name the respective description and type will be shown next to the operator list. When an operator is selected the corresponding parameters and the name of the code for which the parameters will be applied (sender code for pre processors, receiver code for post processors) are listed (see Figure 27).

![Figure 27: Operators area](image)

The type of an operator is either pre processor or post processor and the necessary parameters are operator dependent. Following operators are supported by MpCCI.

**ConvergenceCheck** Pre processor
- Enables convergence check (for iterative couplings only, see ▶ 3.3.4.1 Convergence Check ◁ for details).
- Parameters:
  - **Tolerance value** Specifies the convergence criterium. If the variance of a quantity value is below this tolerance between two iterations, the quantity is considered as converged.

**Disable Conservation** Post processor
- Disables conservative mapping correction (see ▶ 3.3.4.2 Disable Conservative Mapping Correction ◁ for details) (default: enabled).
  - <No parameters>

**PostLimiter** Post processor
- Defines limits to which exceeding values will be clipped (see ▶ 3.3.4.3 Define Limits ◁ for details).
- Parameters:
  - **Sequence number** The rank of this operator in the list of applied post processors.
  - **Lower limit** The lower limit.
  - **Upper limit** The upper limit.

**PreLimiter** Pre processor
- Defines limits to which exceeding values will be clipped (see ▶ 3.3.4.3 Define Limits ◁ for details).
- Parameters:
  - **Lower limit** The lower limit.
  - **Upper limit** The upper limit.

**Relaxation** Post processor
- Enables relaxation (for iterative couplings only, see ▶ 3.3.3 Quantity Relaxation ◁).
- Parameters:
Relaxation method

Specifies the method used for relaxation. The methods offered and their parameters are:

**Fixed**  Uses a constant relaxation factor as described in ▷3.3.3.1 Fixed Relaxation ◄.
Relaxation factor  Specifies the relaxation factor.

**Aitken**  Uses an adaptively calculated relaxation factor as described in ▷3.3.3.3 Aitken Relaxation ◄.

**Quasi-Newton**  Uses the Anderson Mixing Quasi-Newton method for relaxation as described in ▷3.3.3.4 Quasi-Newton Method ◄.

- Quasi-Newton method
- Type of Anderson Mixing method  Specifies the variant to use. MpCCI offers Inverse, Standard and LeastSquares.
- Omega  \((\omega)\)  Specifies the \(\omega\) used in the calculation of variants Inverse and Standard.
- Number of reused levels  Determines the number of previous time steps to be taken into account for the Quasi-Newton relaxation (for transient simulations only).

**Ramping**  Is a special kind of relaxation with a relaxation factor varied over the iterations (▷3.3.3.2 Ramping ◄).

- Initial factor  Specifies the initial ramping relaxation factor. A value between 0 and 1 leads to under-relaxation whereas a value between 1 and 2 leads to over-relaxation.
- Ramping delta  Specifies the ramping step value.

Check relaxation convergence  Enables a relaxation convergence check. This option has to be used in combination with the ConvergenceCheck operator (▷3.3.4.1 Convergence Check ◄) which specifies the tolerance value to be used.

**Scale**  Post processor

Defines a scaling function (see ▷3.3.4.4 Scaling ◄ for details).

Parameters:

- Sequence number
- Only first time step  The rank of this operator in the list of applied post processors. Scale only for the first time step. This option will only be applied for implicit schemes. If this option is activated, no scaling is done for the remaining time steps.
- Reference value
- Initial scale factor  The reference value for the ramping function.
- Final scale factor  The scale factor at the beginning.
- Number of steps with the same scale factor
- Number of scale factor changes  The number of consecutive steps with the same scale factor. Defines how often the scale factor changes.

### 4.6.8 Assigning Preconfigured Quantity Settings

Furthermore the MpCCI GUI provides some Predefined Sets of already preconfigured quantities at the top of the quantities list (see Figure 28). The predefined quantity settings are selectable from a list. To apply a coupling specification you have to select a Coupling Type and click on the [Set] button. The associated quantities with their sender and receiver codes will be selected automatically.
Select Coupling Type provides different coupling types according to the type of the codes you couple. Each provided Coupling Type is characterized by some properties like:

- The dimension which corresponds to the coupling dimension.
- The quantities which will be sent by special types of codes. Each code may have assigned more than one type (see VIII-2.4 MpCCI GUI Configuration File gui.xcf). The known code types are:
  - CFD.
  - ElectroMagnetism.
  - FluidPlasma.
  - FluidThermal.
  - InjectionMoulding.
  - Radiation.
  - SolidStructure.
  - SolidThermal.

For at least one code type a list of quantities to be sent is associated.

Only those coupling types are shown in the list which fit to the coupled codes and supported quantities of the coupled simulation setup. This means that each code type in a coupling type has to match at least one coupled code and that all quantities listed for the code type have to be supported as send-quantity by the matching code and as receive-quantity by the other coupled codes. Below the different coupling types are listed.

**Transient electric arc coupling: Method 1**
Dimension of coupling region: 3

<table>
<thead>
<tr>
<th>Code type</th>
<th>Quantities to send</th>
</tr>
</thead>
<tbody>
<tr>
<td>FluidPlasma</td>
<td>ElectrRes3</td>
</tr>
<tr>
<td>ElectroMagnetism</td>
<td>LorentzForce JouleHeat</td>
</tr>
</tbody>
</table>

**Transient electric arc coupling: Method 2**
Dimension of coupling region: 3

<table>
<thead>
<tr>
<th>Code type</th>
<th>Quantities to send</th>
</tr>
</thead>
<tbody>
<tr>
<td>FluidPlasma</td>
<td>ElectrRes3</td>
</tr>
<tr>
<td>ElectroMagnetism</td>
<td>MagneticField CurrentDensity</td>
</tr>
</tbody>
</table>

**Coupled magnetic field and thermal analysis**
Dimension of coupling region: 3

<table>
<thead>
<tr>
<th>Code type</th>
<th>Quantities to send</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFD</td>
<td>Temperature</td>
</tr>
<tr>
<td>ElectroMagnetism</td>
<td>JouleHeat</td>
</tr>
</tbody>
</table>
### One way force mapping
Dimension of coupling region: 2

<table>
<thead>
<tr>
<th>Code type</th>
<th>Quantities to send</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFD</td>
<td>RelWallForce</td>
</tr>
</tbody>
</table>

### One way pressure mapping
Dimension of coupling region: 2

<table>
<thead>
<tr>
<th>Code type</th>
<th>Quantities to send</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFD</td>
<td>OverPressure</td>
</tr>
</tbody>
</table>

### Steady state radiative heat transfer
Dimension of coupling region: 2

<table>
<thead>
<tr>
<th>Code type</th>
<th>Quantities to send</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radiation</td>
<td>WallTemp</td>
</tr>
<tr>
<td>FluidThermal</td>
<td>FilmTemp WallHTCoeff</td>
</tr>
</tbody>
</table>

### Steady state surface heat transfer
Dimension of coupling region: 2

<table>
<thead>
<tr>
<th>Code type</th>
<th>Quantities to send</th>
</tr>
</thead>
<tbody>
<tr>
<td>SolidThermal</td>
<td>WallTemp</td>
</tr>
<tr>
<td>FluidThermal</td>
<td>FilmTemp WallHTCoeff</td>
</tr>
</tbody>
</table>

### Steady state concentrated heat transfer
Dimension of coupling region: 2

<table>
<thead>
<tr>
<th>Code type</th>
<th>Quantities to send</th>
</tr>
</thead>
<tbody>
<tr>
<td>SolidThermal</td>
<td>WallTemp</td>
</tr>
<tr>
<td>FluidThermal</td>
<td>FilmTemp HeatRate</td>
</tr>
</tbody>
</table>

### Transient MHD Coupling
Dimension of coupling region: 3

<table>
<thead>
<tr>
<th>Code type</th>
<th>Quantities to send</th>
</tr>
</thead>
<tbody>
<tr>
<td>FluidPlasma</td>
<td>ElectrRes3</td>
</tr>
<tr>
<td>ElectroMagnetism</td>
<td>LorentzForce</td>
</tr>
</tbody>
</table>

### Transient radiative heat transfer
Dimension of coupling region: 2

<table>
<thead>
<tr>
<th>Code type</th>
<th>Quantities to send</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radiation</td>
<td>WallTemp</td>
</tr>
<tr>
<td>FluidThermal</td>
<td>WallHeatFlux</td>
</tr>
</tbody>
</table>

### Transient surface heat transfer
Dimension of coupling region: 2

<table>
<thead>
<tr>
<th>Code type</th>
<th>Quantities to send</th>
</tr>
</thead>
<tbody>
<tr>
<td>SolidThermal</td>
<td>WallTemp</td>
</tr>
<tr>
<td>Radiation</td>
<td>WallTemp</td>
</tr>
<tr>
<td>FluidThermal</td>
<td>WallHeatFlux</td>
</tr>
<tr>
<td>InjectionMoulding</td>
<td>WallHeatFlux</td>
</tr>
</tbody>
</table>
### 4.6.9 Overview of the Coupled Regions

To get an overview of the mesh based coupled regions the **Mesh** tab of the **Regions** step provides an **Overview** tab (see **Figure 29**).

The **Overview** tab is divided into a **Summary** and a **Coupled Information** area.

The **Summary** area gives an overview of the used codes, their assigned model files and their number of coupled and uncoupled components. Coupled components are those contained in ready and used regions. They will be considered in the coupled simulation run. Uncoupled components are those not added to any region and those contained in not ready and unused regions.

The **Coupled Information** area gives an overview of the built regions, their state and their assigned quantities. Ready regions are displayed with a quantities transfer direction icon showing a uni- (→) or bidirectional (↔) quantities transfer. Additionally, not ready regions get an icon and the icons for unused regions are greyed out. Via the **Hide / Show** buttons on the top of the **Regions** area the listed regions can be controlled by showing or hiding regions of a special type. As **Figure 29** shows for the incomplete region...
outlet, the tooltip of a region displays an explanation for not ready regions. For ready regions the tooltip displays the names of its contained components (three at most). By the direction button all quantity sets will be marked which are assigned overall to the selected regions. The button on the other hand marks the regions which have assigned the selected quantity sets.

4.6.10 Requirements

In the Regions step the following requirements have to be fulfilled:

At least

- One coupling region with at least
- Two components of at least
- Two different codes with at least
- One quantity applied

has to be set up.

If one or more of the above mentioned items is missing a click on one of the following step buttons will cause a dialog popping up with an appropriate message and will inhibit the next step.

4.7 Monitors Step

In the MpCCI Monitors step you create and configure a region for monitoring (see Figure 30).

![Figure 30: Monitors step](image)

Create a monitor by:

1. Selecting the component type (global variable or mesh based element component).
2. Selecting the code.
3. Selecting the components separated by their dimension for each monitoring region.
4. Specifying the quantities to monitor.

Add additional monitors by clicking the [Add] button and repeating the procedure.

4.8 Settings Step

The Settings step provides access to the control parameters for MpCCI. They are divided into the main categories Monitor, Job, Relation Search and Output (see Figure 31).

The parameters are shown in a tree-like view on the left. The parameters for the selected category are shown in a table on the right. Here you have columns for the name of the parameter, its value and a description. To change a parameter click on its value. Depending on the value type the field changes to an editable field. A category may also contain subcategories like Output. The table only shows direct parameters of a category. Subcategories must be selected explicitly.

Figure 31: Settings step

Following is a description of the parameters for each main category and its subcategories.

4.8.1 Monitor

In this category you may specify some parameters used by MpCCI to control the online monitor.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>Activate the MpCCI Monitor. (Default: on)</td>
</tr>
<tr>
<td>NodeIds</td>
<td>Send the node IDs to the MpCCI Monitor. (Default: off)</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ElemIds</td>
<td>Send the element IDs to the MpCCI Monitor. (Default: off)</td>
</tr>
<tr>
<td>Size</td>
<td>Send the element sizes information to the MpCCI Monitor. (Default: off)</td>
</tr>
<tr>
<td>Orphans</td>
<td>Send the nodal orphan quality level (0...6) to the MpCCI Monitor. (Default: on)</td>
</tr>
<tr>
<td>Globals</td>
<td>Monitor global variables. (Default: off)</td>
</tr>
<tr>
<td>Slaves</td>
<td>Send slave node flags to the MpCCI Monitor. A slave node flag of 1 identifies a node of a mesh partition as a slave node of another partition of the same coupled mesh. Partitions without any slave node (pure master partitions) are displayed in gray. (Default: off)</td>
</tr>
<tr>
<td>Domains</td>
<td>Send the node domains' IDs to the MpCCI Monitor (only active in conjunction with a domain check - see \ref{4.8.2 Job}). If the maximum domain is greater than 0, your coupled mesh is split into multiple separated domains. In case you have to deal with orphaned nodes due to partial overlapping you might consider the fact that orphans are filled with default values instead of some kind of &quot;diffusive&quot; extrapolation. (Default: off)</td>
</tr>
<tr>
<td>Normals</td>
<td>Send face normal vectors to the MpCCI Monitor. (Default: off)</td>
</tr>
<tr>
<td>Peaks</td>
<td>Send the peak and mean values of a quantity on a partition to the MpCCI Monitor. (Default: off)</td>
</tr>
<tr>
<td>AbsDiff</td>
<td>Display the absolute differences between iterations or time steps. (Default: off)</td>
</tr>
<tr>
<td>RelDiff</td>
<td>Display the relative differences between iterations or time steps. (Default: off)</td>
</tr>
<tr>
<td>Inorm</td>
<td>In case of iterative coupling (whether transient or steady state) send the peak and mean values for the convergence norm to the MpCCI Monitor. (Default: off)</td>
</tr>
<tr>
<td>Relax</td>
<td>Include quantity relaxation factor plot. (Default: off)</td>
</tr>
<tr>
<td>Periodic</td>
<td>Display the replicated periodic parts and quantities. (Default: off)</td>
</tr>
<tr>
<td>Modulo</td>
<td>Set the maximum number of steps kept by the MpCCI Monitor. (Default: 100 steps)</td>
</tr>
<tr>
<td>Memory</td>
<td>Set the maximum memory in MB used by the MpCCI Monitor. (Default: 50 MB)</td>
</tr>
<tr>
<td>AutoLaunch</td>
<td>Auto launch the MpCCI Monitor by the server. If set to off, the user can use the command mpcci monitor [optional] to launch the MpCCI Monitor &quot;manually&quot;. (Default: on)</td>
</tr>
<tr>
<td>Host</td>
<td>Specifies the name of the host machine where the MpCCI Monitor runs. (Default: localhost)</td>
</tr>
<tr>
<td>Port</td>
<td>Provide the MpCCI Monitor port address to use. (Default: 47002)</td>
</tr>
<tr>
<td>NetDevice</td>
<td>Specify a network device or an IP address for MpCCI Monitor communication. You may provide the network device for e.g. using Infiniband or Myrinet hardware. Under UNIX this is either the network address (IP address of a special socket adapter) or the name of the special device, e.g. ib0 in case of Infiniband. (Default: empty)</td>
</tr>
</tbody>
</table>

### 4.8.2 Job

In this category you specify the attributes of the coupled job.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PreCheck</td>
<td>Perform only a mesh check and neighborhood search then exit. (Default: off)</td>
</tr>
<tr>
<td></td>
<td>Requires additional license token.</td>
</tr>
</tbody>
</table>
### AngularInterpolation
Select the interpolation method for angular coordinates. (Default: NLERP)

- **Zero Order**: Constant interpolation.
- **NLERP**: Normalized Linear Interpolation. This is computationally cheap, but the angular velocity during a time interval is not constant.
- **SLERP**: Spherical Linear Interpolation on the unit sphere in 4 dimensions. This guarantees constant angular velocity in each time interval.
- **SQUAD**: Spherical QUADrangle interpolation. The spherical analogon to cubic splines. This interpolation method yields smooth rotation.

### MeshMotion
Enable moving mesh transformation within the MpCCI server. (Default: off)
If enabled, for meshes with a motion definition all incoming quantities and coordinates are transformed into a space-fixed reference system and outgoing values are transformed back into the target reference frame.

Please enable mesh motion under the following coupling scenarios:

- a code defines a motion for at least one mesh.
- for a pair of two coupled meshes motions are defined, but the motions are not identical.
- for a pair of two coupled meshes, on one mesh a motion is defined, but not on the partner mesh.

Disabling motions will result in a better mapping performance and numerical stability, since the mesh-mesh relationship is determined only once.

⚠️ Mesh motions are a powerful feature. It should only be enabled if there are incompatible motions defined in the coupled models and the code adapters support the automatic definition of mesh motion parameters.

### MRFCorrect
Enable MRF corrections within the MpCCI server. (Default: off)

⚠️ NOT YET DOCUMENTED. Currently this is an experimental feature and might not work as expected.

### UseInitialNBH
Use the initial mesh-mesh relationship during the moving mesh transformation. (Default: off)

Only valid with the mesh motion option.

### OverlapCheck
Enable a quick overlap check during the creation of mesh pairs. (Default: on)
In case you have only tiny overlapping regions between two meshes you should deactivate this check to avoid complain and an abortion by the MpCCI server.

### DomainCheck
Enable a check on the domains of parts or meshes. (Default: off)
With an activated domain check the server finds the number of domains of a mesh resp. part and also searches for isolated strap elements resp. cells of a mesh which might cause mapping problems, especially in the case of partial overlapping meshes.

⚠️ A domain check is expensive and should be deactivated in case of production runs with remeshing.

### BaffleShift
Enable the shifting of baffle front and rear side for a geometrical separation of the two sides of a baffle. (Default: off)
In case of an activated baffle shift there is no need to put the front side and rear side of a baffle into separate coupled meshes.
The code-adapter has to support the baffle shift feature and must define the thickness of a baffle to the MpCCI server.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConformalMesh</td>
<td>Enable the conformal mesh mapping algorithm (Default: off)</td>
</tr>
</tbody>
</table>
|                          | In case two partner meshes are conformal (geometry and element resp. cell type) a special mapping algorithm can be applied. The relationship is much faster, the mapping does not produce any mapping errors.  
  This is a global switch and all meshes of all partner codes must be geometrically identical. In case of remeshing all partner meshes have to be recreated in the same way.  
  Currently this is an experimental feature and might not always work as expected.                                                                                                                                                                                                                                                                                                                                                                         |
| TimeTolerance            | Define the (relative) tolerance of any time difference resp. time step size of a code below which the two different times are assumed to be identical. (Default: 0.001)  
  Valid values are in the range \([0..1/2]\). Any value less than 0 removes the tolerance check and the time difference must be identical to 0. A value \(\geq 0.5\) resets the value to its default.  
  This value is only relevant for transient cases with multiple buffers.                                                                                                                                                                                                                                                                                                                                           |
| ConvergenceRule          | Specify whether all or only one quantity should have converged if the ConvergenceCheck operator has been assigned to more than one quantity. (Default: All) (See \(\ref{section:4.6.7.4}\) Applying Operators to Quantities of Mesh Based Components)                                                                                                                                                                                                                                                                                                                                                     |
| HistorySize              | Define the minimum number of buffers for storing the quantity history. A number between 4 and 32 may be chosen. (Default: 4)                                                                                                                                                                                                                                                                                                                                                               |
| NetDevice                | Specify a network device name or an IP address for MpCCI communication between server and clients. You may provide the network device for e.g. using Infiniband or Myrinet hardware. Under UNIX this is either the network address (IP address of a special socket adapter) or the name of the special device, e.g. ib0 in case of Infiniband. (Default: empty)                                                                                                                                                                                                                     |
| QuasiNewton              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| MaxIteration             | Define the maximum number of saved iteration values to be used in the Quasi-Newton relaxation applied to steady simulations. (Default: 20) For more details see \(\ref{section:3.3.3.4}\) Quasi-Newton Method \(<\).                                                                                                                                                                                                                                                                                                                                 |
| LinearDependencyTol      | Define the tolerance to eliminate linear dependent columns of the approximated Jacobian in the Quasi-Newton relaxation for better conditioning. (Default: 1e-6) A value between 1e-6 and 1e-10 is recommended. Zero deactivates the option. For more details see \(\ref{section:3.3.3.4}\) Quasi-Newton Method \(<\).                                                                                                                                                                                                                     |
| NoiseTol                 | Define the tolerance to eliminate noisy columns of the approximated Jacobian in the Quasi-Newton relaxation for better conditioning. (Default: 1e-4) A value between 1e-2 and 1e-6 is recommended. Zero deactivates the option. For more details see \(\ref{section:3.3.3.4}\) Quasi-Newton Method \(<\).                                                                                                                                                                                                                     |
| WindowTol                | Define the tolerance to clean up the approximated Jacobian in the Quasi-Newton relaxation for new time steps. It is only relevant for transient simulations. (Default: 1e-5) A value between 1e-3 and 1e-7 is recommended. Zero deactivates the option. For more details see \(\ref{section:3.3.3.4}\) Quasi-Newton Method \(<\).                                                                                                                                                                                                                     |
| NbResiduals              | Define the number of residuals which are checked to be increasing in a Quasi-Newton relaxation. (Default: 0) If the linear fit to these residuals has a positive slope, a re-initialization of the Jacobian will be allowed (restart). Zero leads to no residual check. For more details see \(\ref{section:3.3.3.4}\) Quasi-Newton Method \(<\).                                                                                                                                                                                                                     |
### 4.8.3 Relation Search

In this category you specify the attributes of the relation search.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiplicity</td>
<td>Neighborhood search default distance factor. (Default: 1.0)</td>
</tr>
<tr>
<td></td>
<td>Multiplicity is a factor by which the default search distance is multiplied in case of neighborhood search failures (with orphans). For any orphaned object the search is then repeated with a new (larger) search distance until no orphans are left. Proper values are between 1.0 and 4.0. Values less than 1.0 are for rare cases with strange degenerated meshes.</td>
</tr>
<tr>
<td>ExpertMode</td>
<td>Overwrite the calculated normal distance based on the mesh size with the value set in UserNormalDistance. (Default: off)</td>
</tr>
<tr>
<td></td>
<td>❔ This option should only be used for surface coupling. This may create orphaned nodes if the value is not appropriate to the mesh.</td>
</tr>
<tr>
<td>UserNormalDistance</td>
<td>Specify the normal distance value in meters. (Default: 0.0)</td>
</tr>
<tr>
<td></td>
<td>This is an absolute value. The multiplicity factor provided in the MpCCI GUI will be used for the neighborhood search.</td>
</tr>
</tbody>
</table>

### 4.8.4 Output

In this category you specify the output written by MpCCI and select and configure the writers for which trace files shall be written.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level</td>
<td>Determines how much output MpCCI writes:</td>
</tr>
<tr>
<td></td>
<td>0 no output,</td>
</tr>
<tr>
<td></td>
<td>1 minimal output (default),</td>
</tr>
<tr>
<td></td>
<td>2 additional output,</td>
</tr>
<tr>
<td></td>
<td>3 maximal output - for debug only.</td>
</tr>
<tr>
<td>Tracefile</td>
<td>Save trace files for visualization. (Default: on)</td>
</tr>
<tr>
<td>WriterHome</td>
<td>Enter a path to save the trace files. (Default: $MPCCI_JOBDIR)</td>
</tr>
<tr>
<td>WriterStep</td>
<td>Use writer stepping for saving trace files. (Default: off)</td>
</tr>
<tr>
<td>Enable</td>
<td>It allows saving the results at the specified step interval.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CouplingStep</td>
<td>Coupling step size for saving trace files. (Default: 1)</td>
</tr>
<tr>
<td>IterationStep</td>
<td>Iteration step size for saving trace files. (Default: -1)</td>
</tr>
<tr>
<td>TimeStep</td>
<td>Time step size for saving trace files. (Default: -1.0)</td>
</tr>
<tr>
<td><strong>Writers</strong></td>
<td></td>
</tr>
<tr>
<td>CCVX</td>
<td>Use</td>
</tr>
<tr>
<td></td>
<td>Save an MpCCI binary CCVX file. (Default: on)</td>
</tr>
<tr>
<td>Orphans</td>
<td>Include orphan level information. (Default: off)</td>
</tr>
<tr>
<td>Size</td>
<td>Include element size information. (Default: off)</td>
</tr>
<tr>
<td>NodeIds</td>
<td>Include node IDs information. (Default: off)</td>
</tr>
<tr>
<td>ElemIds</td>
<td>Include element IDs information. (Default: off)</td>
</tr>
<tr>
<td>Globals</td>
<td>Include received global variables. (Default: off)</td>
</tr>
<tr>
<td>Slaves</td>
<td>Include slave node flags. (Default: off)</td>
</tr>
<tr>
<td>Domains</td>
<td>Include node domains (only valid with domain check - see &amp;bull; 4.8.2 Job &amp;bullet). (Default: off)</td>
</tr>
<tr>
<td>Peaks</td>
<td>Include quantity peak and mean values plot. (Default: off)</td>
</tr>
<tr>
<td>Inorm</td>
<td>Include quantity iteration norm values plot. (Default: off)</td>
</tr>
<tr>
<td>Periodic</td>
<td>Include replicated periodic parts and quantities. (Default: off)</td>
</tr>
<tr>
<td>CSV</td>
<td>Use</td>
</tr>
<tr>
<td></td>
<td>Save a comma-separated-values file (CSV) for point coupling. (Default: off)</td>
</tr>
<tr>
<td>Globals</td>
<td>Include received global variables. (Default: off)</td>
</tr>
<tr>
<td><strong>VTK</strong></td>
<td>Use</td>
</tr>
<tr>
<td></td>
<td>Save Paraview VTK files. (Default: off)</td>
</tr>
<tr>
<td>Orphans</td>
<td>Include orphan level information. (Default: off)</td>
</tr>
<tr>
<td>Size</td>
<td>Include element size information. (Default: off)</td>
</tr>
<tr>
<td>Binary</td>
<td>Save in big endian binary format. (Default: off)</td>
</tr>
<tr>
<td>NodeIds</td>
<td>Include node IDs information. (Default: off)</td>
</tr>
<tr>
<td>ElemIds</td>
<td>Include element IDs information. (Default: off)</td>
</tr>
<tr>
<td>Midedge</td>
<td>Include mid-edge nodes on quadratic elements. (Default: off)</td>
</tr>
<tr>
<td>Group</td>
<td>File content grouped by mesh or part numbers. (Default: mesh)</td>
</tr>
<tr>
<td>Slaves</td>
<td>Include slave node flags. (Default: off)</td>
</tr>
<tr>
<td>Domains</td>
<td>Include node domains (only valid with domain check - see &amp;bull; 4.8.2 Job &amp;bullet). (Default: off)</td>
</tr>
<tr>
<td>Periodic</td>
<td>Include replicated periodic parts and quantities. (Default: off)</td>
</tr>
<tr>
<td><strong>EnSight</strong></td>
<td>Use</td>
</tr>
<tr>
<td></td>
<td>Save EnSight Gold binary files. (Default: off)</td>
</tr>
<tr>
<td>Sflush</td>
<td>Save all EnSight Gold files at each coupling step. (Default: off)</td>
</tr>
<tr>
<td>Orphans</td>
<td>Include orphan level information. (Default: off)</td>
</tr>
<tr>
<td>Size</td>
<td>Include element size information. (Default: off)</td>
</tr>
<tr>
<td>NodeIds</td>
<td>Include node IDs information. (Default: off)</td>
</tr>
<tr>
<td>ElemIds</td>
<td>Include element IDs information. (Default: off)</td>
</tr>
<tr>
<td>Midedge</td>
<td>Include mid-edge nodes on quadratic elements. (Default: off)</td>
</tr>
<tr>
<td>Slaves</td>
<td>Include slave node flags. (Default: off)</td>
</tr>
<tr>
<td>Domains</td>
<td>Include node domains (only valid with domain check - see &amp;bull; 4.8.2 Job &amp;bullet). (Default: off)</td>
</tr>
<tr>
<td>Periodic</td>
<td>Include replicated periodic parts and quantities. (Default: off)</td>
</tr>
<tr>
<td><strong>VTFA</strong></td>
<td>Use</td>
</tr>
<tr>
<td></td>
<td>Save Ceetron VTF ASCII files. (Default: off)</td>
</tr>
<tr>
<td>Orphans</td>
<td>Include orphan level information. (Default: off)</td>
</tr>
<tr>
<td>Size</td>
<td>Include element size information. (Default: off)</td>
</tr>
<tr>
<td>Periodic</td>
<td>Include replicated periodic parts and quantities. (Default: off)</td>
</tr>
</tbody>
</table>
4.9 Go Step

In the MpCCI Go step you configure the startup of the applications. Following is a detailed description of the MpCCI coupling server parameters. The configuration for the analysis codes is described in the appropriate code section of the Codes Manual. How to check, start and interrupt the coupled simulation can be found in IV-2.8 Go Step – Configuring the Application Startup and Running the Coupled Simulation.

4.9.1 Configuring the MpCCI Coupling Server

For the MpCCI coupling server the following parameters can be configured (see also Figure 32):

![Figure 32: Server settings](image)

**Job name prefix for job files** is used as prefix to all files generated by MpCCI like "mpccirun_<TIMESTAMP>_ccvx" trace file directory.

**Main server port address** specifies the port address on which the MpCCI server listens for the client connection. The default port is in most cases acceptable, modifications are only required if firewalls block connections.

**Optional local host alias** sets a name which is chosen to identify the local host.

**Setting for remote server start** configures MpCCI to run on a remote host:
- **Remote 'host' to be used** provides a host on which the server process will be started. If the user-name of the remote host differs from the current one, it must also be specified perhaps associated with a domain name. If this parameter is left empty, the server will be started on the local host.
- **Preferred remote shell type** is used to select the type of the remote shell to start the MpCCI server.
  - **rsh (classic rsh)** uses the classic remote shell command rsh.
  - **ssh (secure shell)** uses the secure remote shell command ssh.
Run server inside xterm lets the MpCCI GUI open an xterm window for the MpCCI server process on UNIX and Windows platforms. If this parameter is set the panel expands and shows some more options for configuring the xterm (see Figure 33):

![Figure 33: Run server inside xterm settings](image)

Select xterm background color provides colors to be used as background for the xterm which sometimes make it easier to find the right window.

Force codes to stop on termination tells MpCCI to automatically force an exit of remaining running jobs when one code already exited by itself. Perhaps the remaining jobs are blocked when one job already ended. Usually used in batch mode where the user can’t interactively stop or kill running jobs. If this parameter is set the panel expands and shows some more options (see Figure 34):

![Figure 34: Force codes to stop on termination settings](image)

Timeout in seconds specifies the time to wait between one job is exited and the remaining jobs will be forced to exit. The default value is 60 seconds.

Clean up old tracefile folders tells MpCCI to automatically remove all old tracefile folders beginning with the current job name prefix before the job starts.

4.9.2 Checking the Configuration

As mentioned in > IV-2.8.3 Checking the Application Configuration optional configuration checkers may be provided for each single code. The existence of a code checker depends on an existing checker environment in the MpCCI GUI configuration file. See > VIII-2.4.8 Environments for Scanner, Checker, Starter, Stopper and Killer for a detailed description.

4.9.3 Starting the Coupled Simulation

As mentioned in > IV-2.8.4 Starting the Coupled Simulation the coupled simulation is started via the button. If the project has unsaved data a dialog will pop up allowing to save the project. This dialog also provides an option for automatically saving the project (see Figure 35).

If this option (Don’t ask again and save automatically in this and all future projects.) is set and the project has successfully been saved via this dialog (choosing Save or Save as... without canceling them) the global property Auto Save Project will be set and this dialog won’t appear anymore unless the property is unset. See > 4.3.2 Edit Menu for more information.
4.9.4 Status of the Simulation

The status of each application is shown in the Go step at the bottom of the code’s frame. In Figure 20 (part IV) no special symbol and no button is shown which means that the codes are not running and no results from a previous run exist for the current configuration. This status is representative for a new or just opened project which has not been started or for a finalized simulation with modified project data.

Following running states may occur after the coupled simulation has been started and is running:

- ![Running](Running.png)
  
  This black symbol indicates that the code is running.

- ![Ending](Ending.png)
  
  This red symbol indicates that the code got a stop or kill signal and will end soon.

The running states are linked to a button that enables access to the current output of the application. This information is displayed in an xterm that can be closed and opened again throughout the whole simulation.

![Xterm](Xterm.png)

The output of the server is only available if its option to run in an xterm is activated. Code output is only available when started in batch mode and not for interactive runs.

Following done states may occur after the coupled simulation has been finalized:

- ![Done](Done.png)
  
  The code finished normally.

- ![Failed](Failed.png)
  
  The code failed by itself.

- ![Stopped](Stopped.png)
  
  The code finished by getting a stop signal which is sent after clicking the stop button.

- ![Killed](Killed.png)
  
  The code finished by getting a kill signal. This could be triggered by the kill button or by a failed job.

The done states are linked to a button that enables access to the last information from the application output. This information can be accessed as long as the project data has not been changed. After changing some options the application output from the previous run is not valid anymore and will be cleared.

4.10 Remote File Browser

4.10.1 File Browser Handling

By clicking on the button ![Browse](Browse.png) the remote file browser opens. After having selected up a file the file name is displayed in the text field. If you point the text field with the mouse pointer a tip appears and indicates the location of the file. This information contains the hostname and the absolute file path. If the field is empty the tip shows the message no file specified. In the text field you may enter a file name.
If the name is relative the directory and host of the old file parameter will be taken or if no old file exists the current working directory on the local computer will be taken. Then the file will be located relative to your current working directory.

The remote file browser (Figure 36) displays the files of your current directory in the center and on the right it shows a list of different file systems mounted to this remote file browser. The current directory is represented by the working directory where the MpCCI GUI is running. After you have selected a first file for your application the path of this file represent the current directory for your selected code. Each code has its own working directory.

The default file system view mounted is your local file system represented by the hostname where the MpCCI GUI is running.

If you have more than one file system mounted the current file system is highlighted with a blue background and has the name of the remote machine.

The navigation is performed by using the mouse double click.

The file selection is performed by doing a double click or a simple click to select it, followed by a click on the Open button.

To save a file you have to select the directory by using the mouse and finally enter the file name in the
4.10.2 How to Mount a New File System

On the right side of the remote file browser there are two buttons to Connect and Disconnect a file system. To disconnect a file system from the file browser, select the corresponding file system in the list and click on the Disconnect button. The local file system cannot be disconnected and is represented by the figure Figure 37.

Figure 37: Local file system symbol

To mount a new file system:

1. You have to open the connection dialog by clicking on the Connect button.
2. You have to enter the hostname of the remote machine and some user information.

![Connection Setting](connection-setting.png)

Figure 38: File browser connection dialog

- Select your remote machine.
  At the initialization phase of the MpCCI GUI a list of hosts is searched in the files ".rhosts" and ".shosts" from the user home directory. This list is used to provide some predefined hosts to select.
  You may write the name of the remote machine on the Host text field. If you type the hostname on the input field the MpCCI GUI tries to figure out the host name you want from the host list. Otherwise you may directly pick up the machine name to connect to from the combo box.
- Configure the protocol of the remote connection.
  You may use ssh service or rsh.
- Specify the user name.
  You may modify the user to be used for the connection to the remote machine or let the predefined name.

After configuring the connection you must click on the OK button to establish the connection. If the connection to the remote host is successfully established, you will see the name of the new remote machine in the list of the file system view and the remote file system. Each remote file system is characterized by the type of the protocol used for the connection. On figure Figure 39 you have the icon for the secure and non secure connection.

3. You have a view on the remote file system and you may navigate to select a file.
Figure 39: Secure connection icon (left) and a non secure connection icon (right)
5 Command Line Interface

5.1 Using the Command Line Interface

MpCCI has an extensive command line interface, which offers a lot of functionality beyond the actual coupling process, including license information, job control etc.

To use the command line interface, MpCCI must be installed properly. Especially the PATH environment variable must be set correctly and contain the binary directory "MPCCI_HOME/bin", see the Installation Guide for details.

To obtain a quick help on the MpCCI commands, it is also possible to enter the command followed by a “help key”. If you want to get further information e.g. on starting the MpCCI GUI, which is normally done by entering `mpcci gui`, type

```
mpcci gui -help
```

or

```
mpcci gui ?
```

MpCCI is very flexible in interpreting the command line options. In general dashes (-) can be omitted and commands or options can be abbreviated as long as the abbreviation is unique. Further, command interpretation is not case sensitive. The following commands are thus all equivalent, and show the expiry date of your license:

```
mpcci license -expire
mpcci License EXPIRE
mpcci lic -exp
mpcci lic e
```

If less subcommands of options than required are entered, a list of available options is printed. Entering only the command `mpcci` without further options consequently yields basic information:

- Version information.
- Basic usage as described in this section.
- A list of all available commands.

The general description looks like:

```
*************************************************
MpCCI 4.6.1-0, Build (2010-10-01 ??:??)
(c) 2004-2021 Fraunhofer Institute SCAI
Scientific Computing and Algorithms Institute
Website: http://www.mpcci.de
Email : mpcci-support
*************************************************

Usage:
mpcci SUBCMD [OPTIONS] [ARGS] [HELPKEY] ...

Synopsis:
'mpcci' is the root for all MpCCI related commands. You need to specify at least one subcommand (SUBCMD) on the commandline.
```
SUBCMD, OPTIONS (not file name ARGS!) may be typed in lowercase or UPPERCASE letters or may even be abbreviated as long as there is no ambiguity.

Since the command and help system is dynamically configured at runtime some SUBCMDS and OPTIONS may not always appear in the list below or may suddenly become ambiguous as they are activated only under certain circumstances (existing file/installation etc.). If you use ‘mpcci’ in scripts you should never use the abbreviated form of SUBCMD or OPTION.

You get online help for most of the SUBCMDS and OPTIONS if the HELPKEY ([-]help, [-/]?) ...) appears on the command line.

Please type either

"mpcci SUBCMD HELPKEY" or "mpcci HELPKEY SUBCMD"

to get more detailed help on the subcommands.

The list of commands is omitted here, a concise list is given in the following section.

5.2 Overview of All Subcommands

In addition to subcommands which are related to the general functions of MpCCI, further code-specific commands are offered, which are described in the corresponding sections of the Codes Manual.

The MpCCI commands are discussed in more detail in following sections, which are sorted by the purpose of the commands. To find a command by its name, the following list shows all MpCCI commands in alphabetical order:

<table>
<thead>
<tr>
<th>Subcommand</th>
<th>Short Description</th>
<th>Discussed in</th>
</tr>
</thead>
<tbody>
<tr>
<td>arch [-n]</td>
<td>Print the MpCCI base architecture without a newline or with a newline [-n] at the end and exit. This is different from mpcci info arch which prints the used architecture.</td>
<td>§5.5.1 mpcci arch ◄ on page 139</td>
</tr>
<tr>
<td>backup &lt;file ...&gt;</td>
<td>Make a backup copy of a list of files.</td>
<td>§5.7.1 mpcci backup ◄ on page 155</td>
</tr>
<tr>
<td>batch &lt;project&gt;</td>
<td>Start an MpCCI batch job with project file &lt;project&gt;.</td>
<td>§5.7.2 mpcci batch ◄ on page 156</td>
</tr>
<tr>
<td>ccvxcat &lt;port file ...&gt;</td>
<td>Catenate and/or send .ccvx files to the MpCCI visualizer.</td>
<td>§5.4.1 mpcci ccvxcat ◄ on page 130</td>
</tr>
<tr>
<td>clean</td>
<td>Remove all files from the temporary MpCCI directory &lt;Home&gt;/mpcci/tmp.</td>
<td>§5.7.7 mpcci clean ◄ on page 163</td>
</tr>
<tr>
<td>cosimpre</td>
<td>Use co-simulation pre-check tools</td>
<td>§5.4.2 mpcci cosimpre ◄ on page 131</td>
</tr>
<tr>
<td>doc</td>
<td>View the MpCCI documentation.</td>
<td>§5.5.2 mpcci doc ◄ on page 140</td>
</tr>
<tr>
<td>env</td>
<td>Print out the environment used by MpCCI in various formats for further processing.</td>
<td>§5.5.4 mpcci env ◄ on page 143</td>
</tr>
<tr>
<td>fsimapper</td>
<td>Launch the MpCCI visualizer with FSI mapping plugin.</td>
<td>§5.3.1 mpcci fsimapper ◄ on page 123</td>
</tr>
<tr>
<td>Subcommand</td>
<td>Short Description</td>
<td>Discussed in</td>
</tr>
<tr>
<td>------------</td>
<td>------------------</td>
<td>--------------</td>
</tr>
<tr>
<td><strong>gui</strong></td>
<td>Launch the MpCCI GUI.</td>
<td>5.3.2 mpcci gui &lt; on page 124</td>
</tr>
<tr>
<td><strong>home [-n]</strong></td>
<td>Print the MpCCI home directory without a newline or with a newline [-n] at the end and exit. This is in fact a shortcut for mpcci info home.</td>
<td>5.5.5 mpcci home &lt; on page 144</td>
</tr>
<tr>
<td><strong>info</strong></td>
<td>Print general information about MpCCI.</td>
<td>5.5.3 mpcci info &lt; on page 141</td>
</tr>
<tr>
<td><strong>kill</strong></td>
<td>Platform independent process kill based on command line pattern matching.</td>
<td>5.7.8 mpcci kill &lt; on page 164</td>
</tr>
<tr>
<td><strong>license</strong></td>
<td>Manage the license server and print license related information.</td>
<td>5.6.1 mpcci license &lt; on page 147</td>
</tr>
<tr>
<td><strong>list</strong></td>
<td>List information about the MpCCI installation and the supported codes.</td>
<td>5.6.2 mpcci list &lt; on page 148</td>
</tr>
<tr>
<td><strong>lmutil</strong></td>
<td>Run the FLEXnet lmutil [OPTIONS] command delivered with MpCCI. Avoid running the lmutil command installed by codes other than MpCCI.</td>
<td>5.6.3 mpcci lmutil &lt; on page 149</td>
</tr>
<tr>
<td><strong>logviewer</strong></td>
<td>Launch the MpCCI logfile viewer.</td>
<td>5.3.3 mpcci logviewer &lt; on page 125</td>
</tr>
<tr>
<td><strong>monitor</strong></td>
<td>Launch the MpCCI online monitor.</td>
<td>5.3.4 mpcci monitor &lt; on page 126</td>
</tr>
<tr>
<td><strong>morpher</strong></td>
<td>Launch the MpCCI grid morpher.</td>
<td>5.4.3 mpcci morpher &lt; on page 132</td>
</tr>
<tr>
<td><strong>netdevice &lt;interface&gt;</strong></td>
<td>Return the IP adress corresponding to the specified network interface. (under Windows it only returns the IP for the ethernet device.)</td>
<td>5.4.4 mpcci netdevice &lt; on page 135</td>
</tr>
<tr>
<td><strong>observe</strong></td>
<td>Start the MpCCI file observer.</td>
<td>5.4.5 mpcci observe &lt; on page 136</td>
</tr>
<tr>
<td><strong>ps</strong></td>
<td>Unix ps -ef compatible ps for all platforms.</td>
<td>5.7.9 mpcci ps &lt; on page 166</td>
</tr>
<tr>
<td><strong>ptoj</strong></td>
<td>Convert an MpCCI project file into an MpCCI job properties file.</td>
<td>5.7.10 mpcci ptoj &lt; on page 167</td>
</tr>
<tr>
<td><strong>server</strong></td>
<td>Start the MpCCI server.</td>
<td>5.7.11 mpcci server &lt; on page 168</td>
</tr>
<tr>
<td><strong>ssh</strong></td>
<td>Check/fix your ssh installation.</td>
<td>5.6.4 mpcci ssh &lt; on page 150</td>
</tr>
<tr>
<td><strong>test</strong></td>
<td>Run some install/communication tests.</td>
<td>5.6.5 mpcci test &lt; on page 151</td>
</tr>
<tr>
<td><strong>top</strong></td>
<td>Display process top list / Launch the taskmanager.</td>
<td>5.7.12 mpcci top &lt; on page 172</td>
</tr>
<tr>
<td><strong>visualize</strong></td>
<td>Launch the MpCCI visualizer.</td>
<td>5.3.5 mpcci visualize &lt; on page 127</td>
</tr>
<tr>
<td><strong>where &lt;CMD&gt;</strong></td>
<td>Find all locations of the executable &lt;CMD&gt; in the PATH.</td>
<td>5.5.6 mpcci where &lt; on page 145</td>
</tr>
<tr>
<td><strong>xterm</strong></td>
<td>Start a process inside an xterm.</td>
<td>5.4.6 mpcci xterm &lt; on page 137</td>
</tr>
</tbody>
</table>
5.3 Starting MpCCI

The commands in this section start the different parts of MpCCI. Besides the actual coupling engine, MpCCI offers several helper applications.

<table>
<thead>
<tr>
<th>Subcommand</th>
<th>Short Description</th>
<th>Discussed in</th>
</tr>
</thead>
<tbody>
<tr>
<td>fsimapper</td>
<td>Launch the MpCCI visualizer with FSI mapping plugin.</td>
<td>§5.3.1 mpcci fsimapper ◄ on page 123</td>
</tr>
<tr>
<td>gui</td>
<td>Launch the MpCCI GUI.</td>
<td>§5.3.2 mpcci gui ◄ on page 124</td>
</tr>
<tr>
<td>logviewer</td>
<td>Launch the MpCCI logfile viewer.</td>
<td>§5.3.3 mpcci logviewer ◄ on page 125</td>
</tr>
<tr>
<td>monitor</td>
<td>Launch the MpCCI online monitor.</td>
<td>§5.3.4 mpcci monitor ◄ on page 126</td>
</tr>
<tr>
<td>visualize</td>
<td>Launch the MpCCI visualizer.</td>
<td>§5.3.5 mpcci visualize ◄ on page 127</td>
</tr>
</tbody>
</table>
5.3.1 \texttt{mpcci fsimapper}

Usage:
\begin{verbatim}
mpcci fsimapper [-]option
\end{verbatim}

Synopsis:
'\texttt{mpcci fsimapper}' is used to launch the MpCCI FSIMapper.

Options:
\begin{verbatim}
-batch <configFile> <source> [source_quant] <target>
Use this option to launch the MpCCI FSIMapper in batch mode.
Provide a configuration file, source and target model files and
an optional source quantity file.

-convert <FORMAT> <model>
Use this option to convert native ANSYS models into MapLib format.

-help
This screen.

-scan <FORMAT> <model>
Use this option to scan models with the MpCCI FSIMapper.
Supported scanner formats are ABAQUS, ANSYS, LSDYNA, FLUENT, NASTRAN,
CFXCSV, FLOHERMMAPLIB, FLOEDMAPLIB, FLOHERMXT, FINETURBO, MAXWELL and
ENSIHGHT.
The scanner output will be written to stdout.

-visualize <file1.ccvx> <...>
Use this option visualize serveral CCVX files at once.
\end{verbatim}

The \texttt{MpCCI Visualizer} is started with the FSI mapping plugin activated. The use of the \texttt{MpCCI FSIMapper}
is described in \texttt{X-4 MpCCI FSIMapper GUI}. 

### 5.3.2 mpcci gui

**Usage:**

```
mpcci gui [OPTIONS] [project] [OPTIONS]
```

**Synopsis:**

'`mpcci gui' is used to launch the MpCCI GUI.

**Options:**

- `-chwd <PATH>`
  Replace the symbolic working directory $(CWD) used inside the project file by the absolute pathname specified in the `<PATH>` argument.

- `-help`
  This screen.

- `-new`
  Start the GUI with a new project.

- `-nolic`
  Do not check for a license before starting the GUI.
  This option may be used when no license is available but you would like to prepare a job.

- `-norsa`
  Do neither check for ssh nor ask for ssh assistance if an rsa key file does not exist.

- `-useAbsolutePath`
  Use current local MpCCI installation path on remote access.

The MpCCI GUI is started with this command, i.e. the MpCCI GUI pops up, which is described in \[4 Graphical User Interface\].
### 5.3.3 mpcci logviewer

**Usage:**
```
mpcci logviewer [OPTIONS] [logfile]
```

**Synopsis:**
'MpCCI logviewer' is used to launch the MpCCI logfile viewer.

**Options:**
- `-help`  This screen.
- `-pf`  Read logfile type properties from file (json format).
The default logfile is `${MPCCI_HOME}/gui/logviewerTypeProperties.json`.
- `-q`  Be quiet and do not print any information message except warning or error messages.

The use of the MpCCI Logviewer is described in \( \textbf{\& 6 \textit{MpCCI Logviewer}} \).
5.3.4 mpcci monitor

Usage:
   mpcci monitor [OPTIONS]

Synopsis:
   'mpcci monitor' is used to launch the MpCCI visualiser for MpCCI server monitoring.

Options:
- connect <port@host>
   The monitor connects to the MpCCI server port@host as a client of the MpCCI server. This option may be repeated for multiple connections to several servers.

- help
   This screen.

- jobname <JOBNAME>
   Specify a job name displayed on the status line.

- listen [port]
   The monitor acts as a server and allows multiple client connections in parallel to monitor multiple MpCCI servers. The default port is 47002.

- maxmem <MB>
   Set the maximum memory used by the monitor.

- modulo <STEPS>
   Set the step modulus.

- netdevice <DEV|IP>
   Specify a network device or IP address.

The MpCCI Monitor is suitable for monitoring the coupled quantities during the simulation. You can connect to the coupled simulation with this command. MpCCI Monitor and MpCCI Visualizer are the same application and a description of the MpCCI Monitor is given in §7 MpCCI Visualizer <<.
5.3.5 mpcci visualize

Usage:
```plaintext
mpcci visualize [OPTIONS] [user@host:]filename[nnnn][.ccvx]
```

Synopsis:

'mpcci visualize' is used to launch the MpCCI .ccvx tracefile visualizer.

A tracefile may be located on a remote host. The file name should then follow the general notation for remote file names, which is

```plaintext
[user@]host:path/filename[nnnn][.ccvx]
```

In this scenario you need to have a working MpCCI installation on the remote host!

You may visualize a series of files (option -series) as long as the file name contains at least four digits within the name at any position, e.g.

```plaintext
job-000011.ccvx
job-000012.ccvx
......
```

Examples:

Display a single file

```plaintext
mpcci visualize name.ccvx
mpcci visualize my-tracefiles/name
```

Display a file located on a remote host

```plaintext
mpcci visualize remotehost.network.com:/home/user/jobs/tracefiles/last.ccvx
```

Display a series of files [job0000.ccvx job0001.ccvx ...]

```plaintext
mpcci visualize -series tracefiles/job
mpcci visualize -series remotehost.network.com:/home/user/jobs/tracefiles/job
```

Options:

- `-dir <PATH>` Define ccvx files directory to search.
- `-help` This screen.
- `-listen <port>` Set the communication port number (default=47003).
- `-maxmem <MB>` Set the maximum memory used by the visualizer.
- `-modulo <STEPS>` Set the step modulus for time or coupling steps.
- `-series` Display a series of files.
- `-skiptime <dt>` Define how many steps should be skipped.
- `-skiptime <dt>` Define the time interval which should be skipped.
The **MpCCI Visualizer** is suitable for quickly checking whether the coupling process was successful. The coupling region, orphaned nodes and exchanged quantities can be checked to ensure that a coupling has really occurred. A description of the **MpCCI Visualizer** is given in &gt; 7 **MpCCI Visualizer** &lt;.
## 5.4 MpCCI Tools

The commands in this section are several helper applications to be used with MpCCI.

<table>
<thead>
<tr>
<th>Subcommand</th>
<th>Short Description</th>
<th>Discussed in</th>
</tr>
</thead>
<tbody>
<tr>
<td>ccvxcat &lt;port file ...&gt;</td>
<td>Catenate and/or send .ccvx files to the MpCCI visualizer.</td>
<td>§ 5.4.1 mpcci ccvxcat ◇ on page 130</td>
</tr>
<tr>
<td>cosimpre</td>
<td>Use co-simulation pre-check tools</td>
<td>§ 5.4.2 mpcci cosimpre ◇ on page 131</td>
</tr>
<tr>
<td>morpher</td>
<td>Launch the MpCCI grid morpher.</td>
<td>§ 5.4.3 mpcci morpher ◇ on page 132</td>
</tr>
<tr>
<td>observe</td>
<td>Start the MpCCI file observer.</td>
<td>§ 5.4.5 mpcci observe ◇ on page 136</td>
</tr>
<tr>
<td>xterm</td>
<td>Start a process inside an xterm.</td>
<td>§ 5.4.6 mpcci xterm ◇ on page 137</td>
</tr>
</tbody>
</table>
### 5.4.1 mpcci ccvxcat

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-connect <a href="mailto:port@host">port@host</a></td>
<td>Define the socket of the visualizer (default=47003).</td>
</tr>
<tr>
<td>-help</td>
<td>This screen.</td>
</tr>
<tr>
<td>-jobname &lt;jobname&gt;</td>
<td>Set the job name for the visualizer communication.</td>
</tr>
<tr>
<td>-nopcheck</td>
<td>Do not check and warn about used port numbers.</td>
</tr>
<tr>
<td>-replace &lt;old&gt; &lt;new&gt;</td>
<td>Replace string &quot;old&quot; with &quot;new&quot; in part names.</td>
</tr>
<tr>
<td>-single</td>
<td>Load a single file.</td>
</tr>
<tr>
<td>-skipstep &lt;step&gt;</td>
<td>Define how many steps should be skipped.</td>
</tr>
<tr>
<td>-skiptime &lt;dt&gt;</td>
<td>Define the time intervall which should be skipped.</td>
</tr>
</tbody>
</table>

**Usage:**
```
mpcci ccvxcat [OPTIONS] [user@]host:]filename[nnnn][.ccvx]
```

**Synopsis:**
'`mpcci ccvxcat' is used to catenate one or more .CCVX file and send them to the MpCCI visualizer.

A ccvx tracefile may be located on a remote host. The file name should then follow the general notation for remote file names, which is

```
[user@]host:path/filename[nnnn][.ccvx]
```

In this scenario you need to have a working MpCCI installation on the remote host!

You may catenate a series of files (default) as long as the file name contains at least four digits within the name at any position, e.g.

```
job-000011.ccvx
job-000012.ccvx
......
```
### 5.4.2 mpcci cosimpre

**Usage:**

```bash
mpcci cosimpre [-]OPTIONS
```

**Synopsis:**

'mpcci cosimpre' is used to prepare the co-simulation coupled regions setup

**Options:**

- `-clean` Clean the working directory from the ccvx files.
- `-control <xcc>` Run the neighborhood control.
- `-help` This screen.
- `-modelcheck <xcc>` Run the model check.
- `-monitor` Activate online monitoring.
- `-nbhsearch <xcc>` Run the neighborhood search.
5.4.3 mpccimorpher

Usage:
mpcci morpher [LAUNCH-OPTIONS] <model-name> [OPTIONS]

Synopsis:
'mpcci morpher' is used to start a morpher daemon.

LAUNCH-OPTIONS:
- help This screen.
- lhost:[user@]hostname Launch the morpher on the remote host.
- lopts:filename Read additional morpher options from 'filename'.

Relaunching
"<Home>/work/mp-ce/bin/lnx4_x64/mpcci_morpher.exe"

with 'LD_LIBRARY_PATH'
"<Home>/work/mp-ce/bin/lnx4_x64"
"/scaihome/compeng/compiler/icc10.1/linux_em64t/lib"

Grid morpher daemon 3.1B, double precision
Build Mar 14 2021, 03:21:56
Copyright (c) 2004-2013, Fraunhofer SCAI.
License checkout...
Your license will expire in 1022 days.

Usage:
mpcci_morpher.exe model [OPTIONS]

Parameters [required]
model The grid morpher file name

Options: #d=decimal, #f=float, #s=string, #c=char

Optional job id
-j #s Job id string

Options to control the deformation of edges
- enochec k Skip all edge checks
- rlen[gth] #f #f Min/max allowed relative length change of an edge
 0.0 < Min length <= 0.9
 1.0 < Max length < ?
- alen[gth] #f #f Min/max allowed absolute edge length

Options to control the deformation of faces
-fnocheck
Skip all face checks

-rar[ea]  #f #f Min/max allowed relative change of face area
0.0 < Min area <= 0.9
1.1 <= Max area < ?

-aar[ea]  #f #f Min/max allowed absolute face area

-fas[pect]  #f Max. allowed face aspect ratio [1..200]

-fsk[ew]  #f Max. allowed skewness of faces [0.1..0.99]

Options to control the deformation of cells
-cnocheck
Skip all cell checks

-rvol[ume]  #f #f Min/max allowed relative change of cell volume
0.0 < Min volume < 1.0
1.0 < Max volume < ?

-avol[ume]  #f #f Min/max allowed absolute cell volume

-cas[pect]  #f Max. allowed cell aspect ratio [1..50]

-csk[ew]  #f Max. allowed skewness of cells [0.5..0.99]

Options to control the handling of boundaries
-db[layers]  #d Deformed boundary layer level: [0...512]

-fbl[ayers]  #d Fixed boundary layer level: [0...512]

-fixreg[ion]  #d Add non default fixed boundary regions
    #d is the region number

-fltreg[ion]  #d Add non default floating boundary regions
    #d is the region number

-corner  #f Angle to make node floating along boundaries
    #f [0.0..20.0] is the angle in degree
    the angle should not be larger than 5.0 deg

-project ......... Analytical surface for sliding nodes
    (contact support for options)

-restrict ......... Analytical surface for sliding nodes
    (contact support for options)

Options to control the morpher and solver
-solver gs|lgs|qgs Select a solver (qgs is default)

-nthreads  #d Sets the no. of OpenMP threads

-once  #s #s Morph only once: infile.vrt outfile.vrt

-steps  #d No. of steps for once morphing

-diag[onal] Take care for shear in quad faces
    and hexahedral cells

-miniter  #d Min. no. of iterations: [0...?]

-maxiter  #d Max. no. of iterations: [0...?]

-tol[erance]  #f Convergence tolerance: [0.0...0.5]

-initlast Init displacements from last step

-residual Display the solution residual (time consuming)

-mrelax  #f Morphing relaxation factor: [0.0..2.0]

-local Avoid a global distribution of deformations

-mina[ngle]  #f Min. angle allowed in faces and cells
    #f [5.0...30.0] is the angle in degrees

Options to control the smoother
-smooth  #d No. of smooth sweeps

-srelax  #f Smoothing relaxation factor: [0.0..2.0]
### Auxiliary options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>-listen</strong></td>
<td>Port number for socket communications</td>
</tr>
<tr>
<td><strong>-debug</strong></td>
<td>Save a -morph.vrt file after morphing</td>
</tr>
<tr>
<td><strong>-help</strong></td>
<td>This screen and exit</td>
</tr>
<tr>
<td><strong>-output</strong></td>
<td>Information output level: [f</td>
</tr>
<tr>
<td><strong>-setup</strong></td>
<td>List setup and exit</td>
</tr>
<tr>
<td><strong>-cscaler</strong></td>
<td>Scale vertices read by factor #f</td>
</tr>
<tr>
<td><strong>-dscaler</strong></td>
<td>Scale displacements by factor #f</td>
</tr>
<tr>
<td><strong>-wait</strong></td>
<td>Set waiting timeout in seconds, 0=never timeout</td>
</tr>
<tr>
<td><strong>-noparamcheck</strong></td>
<td>Do not check the limits of any parameter</td>
</tr>
<tr>
<td><strong>-trace</strong></td>
<td>Save received vertices in a trace file</td>
</tr>
<tr>
<td><strong>-visualize</strong></td>
<td>Visualize grid while morphing</td>
</tr>
</tbody>
</table>
5.4.4 `mpcci netdevice`

**Usage:**
- `mpcci netdevice [-help] <interface>`

**Synopsis:**
- 'mpcci netdevice' is used to return the IP address corresponding to the specified network interface.
- (under Windows it only returns the IP for the ethernet device.)
5.4.5 mpcci observe

Usage:
   mpcci observe [-help] file file file ....

Synopsis:
   'mpcci observe' is used to launch the file observer. The file observer
   waits until the file exists and then displays the tail of the
   file in a separate xterm. Just try the observer.
5.4.6 mpcci xterm

Usage:

mpcci xterm [[xterm]OPTIONS] -cmd <cmd ...>

Synopsis:

'mpcci xterm' is in fact a wrapper for the standard X11 xterm command which is used to run a command <cmd ...> inside a new window, the xterm.

Unlike the X11 'xterm [OPTIONS] -e <commandline>' the xterm is launched as a background process and the xterm remains opened after the command exited.

'mpcci xterm' is also available under MS Windows.

The input stream to the <cmd ...> may be redirected and be read from the file -input <FILE>.

In addition, if -log <FILE> is not empty or "-", a copy of the xterm output is logged into the file <FILE>.

The command <cmd ...> may contain several arguments, therefore the item -cmd <cmd ...> must be the last one.

Options: (all options not listed below are passed to the xterm command):

- cmd <cmd ...> Command to fire up.
- display <DISPLAY> Set DISPLAY before.
- help This screen.
- home <HOME> Set the start directory to <HOME>.
- input <FILE> Redirect stdin to <FILE>.
- log <FILE> Redirect stdout via |tee <FILE>.
- rev Use reverse colors.
- title <TITLE> Define the title of the xterm.
5.5 Information and Environment

The commands in this section can be used to obtain information about MpCCI and the environment.

<table>
<thead>
<tr>
<th>Subcommand</th>
<th>Short Description</th>
<th>Discussed in</th>
</tr>
</thead>
<tbody>
<tr>
<td>arch [-n]</td>
<td>Print the MpCCI base architecture without a newline or with a newline [-n] at the end and exit. This is different from mpcci info arch which prints the used architecture.</td>
<td>§5.5.1 mpcci arch ◁ on page 139</td>
</tr>
<tr>
<td>doc</td>
<td>View the MpCCI documentation.</td>
<td>§5.5.2 mpcci doc ◁ on page 140</td>
</tr>
<tr>
<td>env</td>
<td>Print out the environment used by MpCCI in various formats for further processing.</td>
<td>§5.5.4 mpcci env ◁ on page 143</td>
</tr>
<tr>
<td>home [-n]</td>
<td>Print the MpCCI home directory without a newline or with a newline [-n] at the end and exit. This is in fact a shortcut for mpcci info home.</td>
<td>§5.5.5 mpcci home ◁ on page 144</td>
</tr>
<tr>
<td>info</td>
<td>Print general information about MpCCI.</td>
<td>§5.5.3 mpcci info ◁ on page 141</td>
</tr>
<tr>
<td>where &lt;CMD&gt;</td>
<td>Find all locations of the executable &lt;CMD&gt; in the PATH.</td>
<td>§5.5.6 mpcci where ◁ on page 145</td>
</tr>
</tbody>
</table>

- [doc] View the MpCCI documentation.
- [env] Print out the environment used by MpCCI in various formats for further processing.
- [home] Print the MpCCI home directory without a newline or with a newline [-n] at the end and exit. This is in fact a shortcut for mpcci info home.
- [info] Print general information about MpCCI.
- [where] Find all locations of the executable <CMD> in the PATH.
5.5.1 `mpcci arch`

`mpcci arch` is simply a shortcut for `mpcci info arch`, see also §5.5.3 `mpcci info` on page 141. `mpcci arch` prints the MpCCI architecture token of the platform on which it is run. A list of architecture tokens is given in the §II-5 Supported Platforms in MpCCI 4.6.
5.5.2 mpcci doc

This command can be used to view MpCCI documentation, the available documentation can be listed with the option \texttt{-list}. The documentation is located in the "\texttt{<MpCCI\_home>/doc}" directory and can be accessed directly as well.

Currently two kinds of documentation are available:

\begin{itemize}
  \item \texttt{mpcci doc LicenseAdministration} \textbf{The FlexNet Publisher License Administration}
  \item \texttt{mpcci doc MpCCIdoc} \textbf{The MpCCI documentation}
\end{itemize}
The mpcci info command is used to obtain information on the environment that is used by MpCCI:

Usage:

    mpcci info [-]OPTIONS

Synopsis:

    'mpcci info' is used to print single tokens to 'stdout' for the further use in scripts or .bat files...

Options:

-arch    MpCCI basic architecture token.
-arch32  MpCCI 32 bit architecture token.
-arch64  MpCCI 64 bit architecture token.
-build   Build date of the installed MpCCI release.
-help    This screen.
-home    MpCCI home path.
-java    Java command used by MpCCI.
-javaver Java version of the java command.
-jobid   Jobid used by MpCCI.
-liba32  Pathname of the 32 bit libmpcci.a (if available).
-liba64  Pathname of the 64 bit libmpcci.a (if available).
-libdl32 Pathname of the 32 bit libmpcci.so (if available).
-libdl64 Pathname of the 64 bit libmpcci.so (if available).
-make    Make command used by MpCCI.
-patches Patchlevel of the installed MpCCI release.
-perl    Pathname of the perl command used by MpCCI.
-perlinc @INC list used by Perl.
-perlver Version of the running Perl.
-release Full MpCCI release token.
-remcp   Pathname of remote copy command used by MpCCI.
-remsh   Pathname of remote shell command used by MpCCI.
-rahtype MpCCI server remote shell type used.
-userid  Userid used by MpCCI.
-version X.Y version number of the installed MpCCI release.

The options [-arch, -arch32 and -arch64] list the architecture token of the machine. This token is used to distinguish between different hardware and operating systems. MpCCI has its own tokens for identification, which may differ from those used by the coupled codes. A list of the architecture tokens is given in the 

II-5 Supported Platforms in MpCCI 4.6.

Information on the releases and version number of MpCCI are obtained with -build, -release and -version.

The option -userid prints the user which is running MpCCI, i.e. your current user name. This can be useful to check user names on remote machines.

The job id can be obtained with -jobid; it is composed of the user name and a time-dependent value, thus changes with every call of MpCCI, but is kept during one run.

MpCCI uses external software installed on your system. Sometimes several versions are installed, thus it is important to know which was found by MpCCI.
Java is needed for the MpCCI GUI (see 4 Graphical User Interface), the full path to the Java executable is obtained with `-java`, the version of java which was found is obtained with `-javaver`.

Perl The path to the Perl executable is obtained with `-perl`, the version with `-perlver`, and `-perlinc` lists the `@INC` of Perl, which is a list of directories, which is searched for Perl modules. See also III-9 Installing Perl.

rsh `-rsh` gives the path to the remote shell `rsh`. There are two types of remote shells `rsh` and `ssh`. The type which is currently used by MpCCI is shown by `-rshtype`. The remote shell can be changed by setting the `MPCCI_RSHTYPE` to either type. See 2.6.3 Remote Shell and Remote Copy for more information on remote shells.

rcp Gives the path to the rcp command, see also 2.6.3 Remote Shell and Remote Copy.
5.5.4 mpcci env

`mpcci env` is a lookup-function for environment variables. Its sole purpose is to print a list of all environment variables, which are relevant for MpCCI. More information on these environment variables is given in > 2.3 Environment and Environment Variables<. This command is useful for debugging.

The list can be formatted in various formats for use in shell scripts.

**Usage:**

```
mpcci env [ [-]FORMAT | environment variable ]
```

**Synopsis:**

The MpCCI environment is set up at runtime and contains all informations about your system required by MpCCI.

-'mpcci env' is used to print out the MpCCI environment in various formats for the further processing in shell scripts or in a MS Windows batch file.

-'mpcci env' is used internally by MpCCI to fetch information about the MpCCI configurations on remote hosts.

**Examples:**

To save the MpCCI environment in a file which may be sent for support reasons type

```
"mpcci env pretty > mpcci_env.txt"
```

**Supported formats:**

- `-bash` UNIX bash format
- `-bat` MS-DOS .BAT format
- `-csh` UNIX csh format
- `-help` this screen
- `-java` Java properties format
- `-ksh` UNIX ksh format
- `-perl` Perl expression format
- `-plain` plain format
- `-pretty` human readable format (default)
- `-sh` UNIX sh format
- `-tcsh` UNIX tcsh format
- `-xml` XML similar format
5.5.5 mpcci home

Prints the full path of the MpCCI home directory. All files which belong to the MpCCI distribution are located in this directory. With

```
mpcci home
```

the path is given without a trailing newline character, whereas

```
mpcci home -n
```

yields the same path followed by a newline character.

The path to the MpCCI home directory is stored in the environment variable MPCCI_HOME during a run of MpCCI.
5.5.6 mpcci where

Usage:
   mpcci where [-help] command ...

Synopsis:
   'mpcci where' is used to list all commands found by investigating
   the "PATH" environment variable.

   This is useful to find out whether MpCCI catches the correct
   executable file from the "PATH". Sometimes the "PATH" should
   be reordered to help MpCCI find the command really needed.

   For some important commands MpCCI has build in alternative
   methods to find the correct executable. In this case the result
   of where does not show the executable selected by MpCCI.

Examples:
   To find the location of the mpcci command please type

   "mpcci where mpcci"
## 5.6 Installation and Licensing

The commands in this section are needed to check the validity of an installation, also including license information.

<table>
<thead>
<tr>
<th>Subcommand</th>
<th>Short Description</th>
<th>Discussed in</th>
</tr>
</thead>
<tbody>
<tr>
<td>license</td>
<td>Manage the license server and print license related information.</td>
<td>5.6.1 mpcci license on page 147</td>
</tr>
<tr>
<td>list</td>
<td>List information about the MpCCI installation and the supported codes.</td>
<td>5.6.2 mpcci list on page 148</td>
</tr>
<tr>
<td>lmutil</td>
<td>Run the FLEXnet lmutil [OPTIONS] command delivered with MpCCI. Avoid running the lmutil command installed by codes other than MpCCI.</td>
<td>5.6.3 mpcci lmutil on page 149</td>
</tr>
<tr>
<td>ssh</td>
<td>Check/fix your ssh installation.</td>
<td>5.6.4 mpcci ssh on page 150</td>
</tr>
<tr>
<td>test</td>
<td>Run some install/communication tests.</td>
<td>5.6.5 mpcci test on page 151</td>
</tr>
</tbody>
</table>
5.6.1 mpcci license

Usage:
mpcci license [-pN] [-tT] [-]OPTION ...

Synopsis:
'mpcci license' is used to manage the MpCCI licenses and to display detailed license information.

Options:
- all List all features of all available licenses.
- avail Brief display the no. of MpCCI sessions and processes available.
- check Check compatibility of the FLEXlm client and server license version.
- clean Remove all local MpCCI license logfiles.
- expire Display the expiration date of the MpCCI license.
- files List all local license files defined.
- help This screen.
- info Print MpCCI related summary.
- local MpCCI feature overview for licenses on the local host.
- log Display the MpCCI related license logfiles.
- mpcci MpCCI feature overview for all hosts in a network.
- pN Redefines the port number N used with the SVD license server.
  The current port for SVD is "-p47000".
  If used this option must be the first on the commandline.
- restart Restart the SVD license server on the local host.
- servers List all defined license servers "[port]@host".
- start Start the SVD license server on the local host.
- stop Stop the SVD license server running on the local host.
- sysid MpCCI system ID used for generating a license file.
- tT Set license request timeout to T seconds.
  If used this option must be the first/second on the commandline.
- vars List the relevant environment variables *_LICENSE_FILE.
- version Show FLEXlm version of MpCCI installation.
### 5.6.2 mpcci list

#### Usage:
```
mpcci list [-]OPTIONS
```

#### Synopsis:
'`mpcci list' is used to list information about the MpCCI installation.

#### Options:
- `--archs` List all installed MpCCI architectures.
- `--batchsystems` List all installed batch systems.
- `--codes` List installed simulation codes.
- `--help` This screen.
- `--hosts` List default hostlist entries.
- `--jobs` List all submitted batch jobs.
- `--writers` List all available MpCCI trace file writers.
5.6.3 mpcci lmutil

lmutil - Copyright (c) 1989-2018 Flexera. All Rights Reserved.

usage: lmutil lmborrow | lmborrowl -status
lmutil lmborrow | lmborrowl -purge
lmutil lmborrow | lmborrowl -purge -status
lmutil lmborrow | lmborrowl -clear
lmutil lmborrow | lmborrowl {all|vendor} dd-mmm-yyyy:[time]
lmutil lmdiag [-c licfile] [-n]
lmutil lmhostid [-ptype (PHY|AMZN|VM)] [-ether|-internet (v4|v6)|-user|-n]
    -display|-hostname|-hostdomain|-string|-long|-uuid
    -eip|-ami|-iid|-genid|-flexid
lmutil lminstall [-i infile] [-o outfile]
    [-overfmt {2, 3, 4, 5, 5.1, 6, 7.1, 8}]
    [-decimal] [-maxlen n]
lmutil lmnewlog [-c licfile] vendor new-file [-secondary], or
lmutil lmnewlog [-c licfile] feature new-file [-secondary]
lmutil lmpath -status
lmutil lmpath -override {all | vendor } path
lmutil lmpath -add {all | vendor } path
lmutil lmremovelfe [-c licfile] feature user host display
lmutil lmremovelfe [-c licfile] -h feature host port handle
lmutil lmremovelfe [-c licfile] [-tsborrow <client_host>] | [-tsborrowstat]
lmutil lmread [-c licfile] [-vendor name] [-all]
lmutil lmswitchr [-c licfile] vendor new-file, or
lmutil lmswitchr [-c licfile] feature new-file
lmutil lmstat [-c licfile] [lmstat-args]
lmutil lmswitch [-c licfile] vendor new-file, or
lmutil lmswitch [-c licfile] feature new-file
lmutil lmver { flexlm_binary | -fnls }
lmutil lmvminfo [-long]
lmutil lmtpminfo [-long]
lmutil -help (prints this message)
lmutil utility_name -help (display detailed usage information)
5.6.4 mpcci ssh

Usage:
mpcci ssh [-]OPTIONS

Synopsis:
'mpcci ssh' is used to check/update the secure shell (ssh) installation
and settings. Some ssh settings should be done to avoid password
requests for each MpCCI process launched on the local or remote
hosts when using the secure shell (ssh/scp) set of commands for
remote shell and remote file copy.

If you prefer to use the classic rsh/rcp set of commands please
make sure that in your remote hosts file

"<Home>/.rhosts"

the local and all the remote hosts used by MpCCI are listed.

Options:
-all Run all the checks below.
-env Update the MpCCI environment variables in the ssh login
environment file.
-help This screen.
-keygen Check/generate an ssh key for the local host to avoid
password requests.
-mode Test the ssh daemon configuration for "StrictModes".
5.6.5 mpcci test

Usage:
.mpcci test [OPTIONS] hostname ... hostlist ... hostfile ... [-simple]

Synopsis:
'mpcci test' is used to perform various tests:
- MpCCI local and remote installation
- Remote host connections
- Perl environment
- etc.

The remote hostnames may be specified in various formats.
hostname: [user@]host
hostlist: [user@]host[:[user@]host[:...]]...
hostfile: filename with hostnames (like .rhosts)

For each host ...
- test whether the hostname is resolved by the DNS.
- test whether the host is alive and reachable.
- test possibility of rsh/ssh connections to the remote host.
- brief test on MpCCI installation on the remote host from the local host
- test server-client connection on ports 47111 ++
- write a protocol hostfile "mpcci.hostlist" which can be used as
  an MpCCI hostfile.

Examples:
Run a simple testcase delivered with the MpCCI installation:

  on the local host:  mpcci test -simple
  on various hosts :  mpcci test testhost mpcci@server.com -simple

Test the MpCCI access and communication with remote systems:

  mpcci test [OPTIONS] [ [user@]host[:[user@]host[:...]] [ hostfile ... ]
  mpcci test fred@flintstone.family:wilma@flintstone.family
  mpcci test fred@trex.farm bmw@stone-cars.manufactory
  mpcci test flintstone.hostlist aquarium whale@waterworld.future:shark@zoo
  mpcci test ~/.rhosts ~/.shosts brontosaurus@jurassic-park.vision

Options:
- -connect <port@hostname>

    INTERNAL USE ONLY: for remote communication test.

- -help

    This screen.
-known

Test if each host is already known to ssh.

-listen <port>

INTERNAL USE ONLY: for remote communication test.

-modload

Load/compile all eventually MpCCI used Perl modules for a validation and exit. This test may help while you are integrating your code into MpCCI or to figure out if some common Perl modules are not installed on your system.

-port <port>

Set the client/server communication port no. (default=47111).

-remote <hostname>

For cases with VPN connections, IPSEC tunneling, and NAT translation:
<hostname> is the name or dotaddress of the local host seen from the remote hosts. Redefine the local hostname if necessary.
e.g. -rem 192.168.2.16
or -rem myvpnhost.company.com

-rsh

Test only rsh type connections.

-simple

Run a simple testcase delivered with MpCCI.
-simple must be the last option of the commandline.

-ssh

Test only ssh type connections.
-wait

Wait after each error or warning.
5.7 Job Control

During a coupled analysis it is often necessary to keep control of the different jobs which are started. The commands in this section help with starting, controlling and interrupting calculations.

<table>
<thead>
<tr>
<th>Subcommand</th>
<th>Short Description</th>
<th>Discussed in</th>
</tr>
</thead>
<tbody>
<tr>
<td>backup &lt;file ...&gt;</td>
<td>Make a backup copy of a list of files.</td>
<td>5.7.1 mpcci backup ◄ on page 155</td>
</tr>
<tr>
<td>batch &lt;project&gt;</td>
<td>Start an MpCCI batch job with project file &lt;project&gt;</td>
<td>5.7.2 mpcci batch ◄ on page 156</td>
</tr>
<tr>
<td>clean</td>
<td>Remove all files from the temporary MpCCI directory &lt;Home&gt;/mpcci/tmp</td>
<td>5.7.7 mpcci clean ◄ on page 163</td>
</tr>
<tr>
<td>kill</td>
<td>Platform independent process kill based on command line pattern matching</td>
<td>5.7.8 mpcci kill ◄ on page 164</td>
</tr>
<tr>
<td>ps</td>
<td>Unix ps -ef compatible ps for all platforms.</td>
<td>5.7.9 mpcci ps ◄ on page 166</td>
</tr>
<tr>
<td>ptoj</td>
<td>Convert an MpCCI project file into an MpCCI job properties file.</td>
<td>5.7.10 mpcci ptoj ◄ on page 167</td>
</tr>
<tr>
<td>server</td>
<td>Start the MpCCI server.</td>
<td>5.7.11 mpcci server ◄ on page 168</td>
</tr>
<tr>
<td>top</td>
<td>Display process top list / Launch the taskmanager.</td>
<td>5.7.12 mpcci top ◄ on page 172</td>
</tr>
</tbody>
</table>
5.7.1 mpcci backup

Usage:
   mpcci backup file [file file ...]

Synopsis:
   'mpcci backup' is used to copy one or more files into
   backup files adding an additional free suffix ".bakNNN".
   This is a system independent backup copy command.

Examples:
   mpcci backup path/to/file.ext => path/to/file.ext.bak000
5.7.2 mpcci batch

In the usage output you can see if a queuing system has been detected by MpCCI. To access the specific queuing system command help, execute `mpcci batch <Batch Name>`.

By specifying the queuing system as the batch name to the `mpcci batch` command, MpCCI provides a way to interact directly with the batch system using the suboptions: `submit`, `kill`, `status`. In addition to the batch system control commands, you are able to set additional options to modify the settings of the codes (see ▷ 3.6.1.1 Run a Job in Batch Mode with MpCCI Command Line).

Usage:

```
mpcci batch [OPTIONS] <projectname>
mpcci batch [OPTIONS] SGE ...
mpcci batch [OPTIONS] TORQUE ...
mpcci batch [OPTIONS] OPENPBS ...
mpcci batch [OPTIONS] SLURM ...
mpcci batch [OPTIONS] LSF ...
mpcci batch [OPTIONS] PBS ...
mpcci batch [OPTIONS] PBSPRO ...
```

Synopsis:

`'mpcci batch'` is used to start an MpCCI job in batch mode.
`'mpcci batch SGE'` is used to control a SGE batch job.
`'mpcci batch TORQUE'` is used to control a TORQUE batch job.
`'mpcci batch OPENPBS'` is used to control a OPENPBS batch job.
`'mpcci batch SLURM'` is used to control a SLURM batch job.
`'mpcci batch LSF'` is used to control a LSF batch job.
`'mpcci batch PBS'` is used to control a PBS batch job.
`'mpcci batch PBSPRO'` is used to control a PBSPRO batch job.

Options:

- `-checkonly`
  Create the project file used for the batch job and exit. A new batch system compatible project file will only be created when running under a batch queuing system.

- `-chwd <PATH>`
  Replace the symbolic $(CWD) working directory used inside the project file by the absolute pathname specified in the <PATH> argument.

- `-help`
  This screen.

- `-listen <N>`
  Specifies an alternative TCP/IP port number <N> for the communication between the MpCCI server and the clients. The default port number is 47010.

- `-mpmode <codeA:mode> <codeB:mode>`
  Set the parallel method "mode" for each code with the
following option:
- "none": run code on one cpu
- "smp": activate shared memory parallel mode
- "dmp": activate distributed memory parallel mode.

-nocontrol
Start the MpCCI batch job without termination process management.

-nolic
Do not check for a license before starting the batch job.

-norsa
Do neither check for ssh nor ask for ssh assistance if an rsa key file does not exist.

-<codeA:N> <codeB:M>
Bind each code to the specified total number of cores.

-precheck
Start MpCCI in pre-check mode only. The simulation codes will send their meshes for a neighborhood calculation and exit. The results of the check are stored in server log file.

-set
Modify property value of a code before submitting the job in batch.

The property value respects the following rules:
- the property name matches the key string (<param name="property"...>) from the project csp file.
- the value to modify should respect the defined type.
- the list of property value pair is comma separated.
- The list should be defined within the brackets.

-useAbsolutePath
Use current local MpCCI installation path on remote access.

-LSF batch system control.

OPENPBS batch system control.

PBS batch system control.

PBSPRO batch system control.

SGE batch system control.
### 5.6 Coupled Analysis in Batch Mode

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>slurm</strong></td>
<td>SLURM batch system control.</td>
</tr>
<tr>
<td><strong>torque</strong></td>
<td>TORQUE batch system control.</td>
</tr>
</tbody>
</table>

See \[ 3.6 \text{ Coupled Analysis in Batch Mode} \] for details.
5.7.3 mpCCI batch LSF

<table>
<thead>
<tr>
<th>Usage:</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpcci batch LSF submit [BATCH-OPTIONS] &lt;projectname&gt;</td>
</tr>
<tr>
<td>mpcci batch LSF status &lt;jobid&gt;</td>
</tr>
<tr>
<td>mpcci batch LSF kill &lt;jobid&gt;</td>
</tr>
</tbody>
</table>

Synopsis:
'mpcci batch LSF' is used to manage an MpCCI job running under LSF.

Batch commands:
- help This screen.
- kill <jobid> Kill a LSF batch job.
- status <jobid> Display a LSF batch job status.
- submit [BATCH-OPTIONS] <projectname> Submit a LSF MpCCI job.

See 3.6 Coupled Analysis in Batch Mode for details.
5.7.4 mpcci batch PBS

The following PBS family queuing systems are available and supported by MpCCI:

- PBS
- OpenPBS
- PBSPro
- Torque

Usage:

mpcci batch PBS submit [BATCH-OPTIONS] <projectname>
mpcci batch PBS status <jobid>
mpcci batch PBS kill <jobid>

Synopsis:

'mpcci batch PBS' is used to manage an MpCCI job running under PBS.

Batch commands:

- help          This screen.
  kill <jobid>   Kill a PBS batch job.
  status <jobid> Display a PBS batch job status.
  submit [BATCH-OPTIONS] <projectname> Submit a PBS MpCCI job.

See \[3.6 Coupled Analysis in Batch Mode\] for details.
5.7.5 mpcci batch SGE

The following Sun Grid Engine family queuing systems are available and supported by MpCCI:

- SGE
- N1GE
- SGEEE

Usage:

- mpcci batch SGE submit [BATCH-OPTIONS] <projectname>
- mpcci batch SGE status <jobid>
- mpcci batch SGE kill <jobid>

Synopsis:

'mpcci batch SGE' is used to manage an MpCCI job running under SGE.

Batch commands:

- -help
  This screen.
- kill <jobid>
  Kill a SGE batch job.
- status <jobid>
  Display a SGE batch job status.
- submit [BATCH-OPTIONS] <projectname>
  Submit a SGE MpCCI job.

See \textit{3.6 Coupled Analysis in Batch Mode} for details.
### 5.7.6 mpCCI batch SLURM

**Usage:**
- `mpcci batch SLURM submit [BATCH-OPTIONS] <projectname>`
- `mpcci batch SLURM status <jobid>`
- `mpcci batch SLURM kill <jobid>`

**Synopsis:**
'MpCCI batch SLURM' is used to manage an MpCCI job running under SLURM.

**Batch commands:**
- `-help` This screen.
- `kill <jobid>` Kill a SLURM batch job.
- `status <jobid>` Display a SLURM batch job status.
- `submit [BATCH-OPTIONS] <projectname>` Submit a SLURM MpCCI job.

See §3.6 Coupled Analysis in Batch Mode for details.
5.7.7 mpcci clean

Usage:
   mpcci clean

Synopsis:
   'mpcci clean' is used to remove ALL files from the temporary MpCCI directory which is currently
   "<Home>/mpcci/tmp".

Options:
   -dir <directory> Remove ALL files and directories under the provided directory name.
   -help          This screen.
5.7.8 mpcci kill

Usage:
   mpcci kill [OPTIONS] <pattern1> <pattern2> ...

Synopsis:
'mpcci kill' is used to kill processes whose commandline matches a character pattern. The commandline is read via the "ps -ef" command which is also available under Microsoft Windows.

The <patterns> are strings of printable characters, e.g. a piece of the name or an option of a running program. If the <pattern> is a decimal number it will be assumed that this number is a true process id (PID) and not a command line pattern.

The options -i, -v and -q (see below) are mutually exclusive. The last option on the commandline overwrites the others.

'mpcci kill' is self protecting in case of pattern matching: You may kill your whole family (childs, brother, sisters, aunts and uncles), but never yourself and your parents and grandparents ...

'mpcci kill' is also used internally by MpCCI to kill a group of MpCCI processes on the local or remote hosts.

Examples:
Kill all MpCCI related processes: "mpcci kill mpcci_"
Kill process with PID 1234: "mpcci kill 1234"

Options:
-f <prefix> Special definition of process ids.
Collects all files with name

"<prefix>.<PID>"

where <PID> is a number and is interpreted as a process id. The files "<prefix>.<PID>" are deleted afterwards. Typically, they are only indicator files of size 0 which were created by 'touch'. Their only purpose is the definition of PID's via their suffix.

-help This screen.

-i Interactive confirmation (default):
Individually confirm each process found.

-q Quiet kill, no confirmation and no printout.
-r  Recursively kill all processes specified via PID or pattern AND all their descendants (children, grandchildren, ...).

-s  <SIG>  Send signal <SIG> to the processes instead of killing them with signal 9 (KILL).

-v  Verbose printout, no confirmation:
    Print a list of process ids before killing them.
5.7.9 mpcci ps

Usage:
   mpcci ps [PATTERNS]

Synopsis:
   'mpcci ps' is a platform independent Unix style
   "ps -ef [|grep PATTERN]"
   'mpcci ps' is also available under MS Windows!

Examples:
   List all processes, run ps -ef:       "mpcci ps"
   List all MpCCI related processes:     "mpcci ps mpcci"
   List all processes under your account: "mpcci ps scai-cae"

Options:
   -help       This screen.
5.7.10 mpcci ptoj

Usage:
   mpcci ptoj [OPTIONS] <file> <file> ...

Synopsis:
   'mpcci ptoj' is used to convert an MpCCI .csp project
   file into an MpCCI .prop job properties file.

Options:
   -help    This screen.
5.7.11 mpcci server

Usage:
    mpcci server [OPTIONS] jobfile|projectfile

Synopsis:
    'mpcci server' is used to start the MpCCI broker and/or server
    processes on the local host or as an alternative distributed
    on multiple hosts in a network.

Options:
    -cleantrace
      Delete all tracefile folders based on the job name prefix of the
      current job directory.

    -help
      Prints this screen only and exits.

    -host <hostname>
      Launch the server on a remote host. The <hostname> may be given as
      [user@]host

    -hostalias <hostname>
      For cases with VPN connections, IPSEC tunneling and NAT translation.
      <hostname> is the name or dotaddress of the local host seen from the
      remote hosts.

    -jobdir <jobdir>
      Specify a path to save the trace files for visualization.

    -jobid <jobid>
      Specifies the job ID name to use.

    -jobname <jobname>
      Specify a general name prefix for all non temporary MpCCI generated
      output files. The default <jobname> is "mpccirun".
-killscript <user@host> <jobid> <pid> <jobdir>

Create a mpcci kill script.

-listen <N>

Suggest an alternative listening port number <N> for the communication between the MpCCI server and the clients.

The default port no. is 47010.

-monitor <port@host>

Suggest an alternative listening port@host for the monitor in case the monitor is automatically launched by the server. The default is 47002@localhost.

-nbuf <N>

Specifies the number of quantity buffers used by the server.

-netdevice <interface>

Set device interface name to use for the client server communication.

-noportcheck

Do not check the provided listening port number.

-out <N>

Specifies the message output level (0=quiet ... 3=debug).

-remotefs <username@host>

Indicates the name of the machine where the process as been started.

-remotejobfile <user@host> <remoteJobFile> <jobdir> <-[ssh|rsh]

>Get the MpCCI jobfile from remote.
-rsh

Use the (rsh/rcp) set of commands instead of the secure shell. The current default setting is defined by the environment variable

MPCCI_RSHTYPE

-ssh

Use the secure shell set of commands (ssh,scp) instead of the standard settings (rsh,rcp). The current default setting is defined by the environment variable

MPCCI_RSHTYPE

-tmpdir <dir>

Dependent on the platform MpCCI creates small temporary files (shell scripts or .BAT files with few lines only) and logging output.

The default -tmpdir is defined by the environment variable

MPCCI_TMPDIR=<Home>/mpcci/tmp

and should be located on your local system.

With -tmpdir you specify an alternative directory for temporary files.

-useAbsolutePath

Use current local MpCCI installation path on remote access.

-xterm

Run the server process in a separate X11 xterm.

-xtermbg <color>

Set xterm background color, e.g. "-xtermbg blue".

-xtermfg <color>

Set xterm foreground(text) color, e.g. "-xtermfg green".

-xtermopt <opt>
Add additional X11 xterm options <opt>, e.g.

"-geometry WxH -132".

Please use only X11 xterm options valid for the local xterm command.

Microsoft Windows with local MpCCI/xterm emulation:
All X11 xterm options may also be used since the MpCCI/xterm emulation simply ignores any unsupported option.
5.7.12 mpcci top

Usage:
   mpcci top

Synopsis:
"mpcci top" displays the process list:
If it is available the Unix 'top' command is launched in a separate xterm.
On Windows systems the taskmanager may be launched if 'top' is not available.

Options:
   -help   This screen.
6 MpCCI Logviewer

The MpCCI Logviewer is a tool to inspect the logfiles written by MpCCI.

![MpCCI Logviewer](image)

**Figure 1: MpCCI Logviewer**

As shown in **Figure 1** it shows MpCCI logfiles in a table view with numbered chronological events that can be sorted and filtered. Filters can be set for codes, levels and messages. The whole message for a selected entry is shown in the area below the table. The header of the logfile - which is everything from the beginning of the file up to the first code entry - can be shown over the **Header** button. At the bottom of the window a status bar displays short information messages.

6.1 Starting the MpCCI Logviewer

The **MpCCI Logviewer** can be started from the **MpCCI GUI** menu by selecting **Tools→Logviewer** as well as from the command line with

```
mpcci logviewer [<logfile>]
```

where the name of a logfile can optionally be given.

The command **mpcci logviewer -help** shows up a short help text. See >5.3.3 mpcci logviewer for a detailed command description.

6.2 Description

6.2.1 Menus

The menu bar offers the File and Help menus.

**File→Open**  Shows a file chooser for opening an MpCCI logfile which usually ends with ".log".
6.2.2 Information

**Logfile** Displays the name of the loaded logfile.

**Type** Displays the type of the loaded logfile.

6.2.3 Buttons

**Header** The same as File → Show Header but if no header data is available the button is disabled.

**Open** The same as File → Open.

**Clear** The button clears all filters so that all logfile entries are shown.

6.2.4 Filters

With the filters you can reduce the information shown in the logfile table and so focus on special entries. Choosing the empty entry removes each filter.

**Code** Select a code whose entries are to be shown as the only ones in the table. Additionally to the simulation codes used in the coupled simulation the MpCCI-SERVER code and MpCCI License process are also listed here.

**Level** The entries are marked with a level which is influenced by the output level (see 4.8.4 Output). Only the levels found in the logfile are listed here. The known levels are:

- **INFO** General information messages mostly sent from MpCCI and printed in output level 1. They regard license information, job settings, quantity mappings and the like.
- **ACTION** More detailed messages regarding the coupling process. They are sent from MpCCI as well as the code adapters typically in output level 2.
- **DEBUG** Detailed debugging messages printed in output level 3 only.
- **WARNING** Warning messages which may worth observing. They are independent of the set output level.
- **ERROR** Error messages which will be printed whatever output level is set.
- **UNKNOWN** Messages with unknown level.

**Message** This filter looks for the specified string in the messages, highlights it, and only displays the lines that match the input. The input is used as a string that contains all spaces, including those at the beginning and end. Multiple patterns are not supported.

6.2.5 Table with Message Area

The center of the logviewer is occupied by a table holding the event messages sent by the codes and a message area.
6.2.5.1 Table

The table holds all event messages read out of the logfile and displays it in several columns:

**Event No.** Shows the number of the event messages which are numbered chronological by their appearance in the logfile.

**Code** Shows the name of the code which sent the event.

**Level** Shows the level of the message event. See \[6.2.4 Filters\] for a detailed level description.

**Message** Shows the beginning of the event message. Messages which are too long for being displayed in the table cell are marked by ... at the end. The whole message is displayed in the message area below this table by selecting the event in this table.

Each column can be sorted by clicking on its header where the event number column will be sorted numerically the other columns alphabetically. Clicking several times on the header toggles the sort order between ascending and descending.

6.2.5.2 Message Area

The event messages may take several lines where only the beginning is shown in the table. Select a row in the table and the whole message is displayed in the message area. If a message filter is set, the matching strings are highlighted.

6.2.6 Status Line

The status line prints short information about the just opened file, the number of currently filtered events and so on.

6.3 Supported Logfile Types

As mentioned before the MpCCI Logviewer is a tool that was developed to inspect the logfiles written by MpCCI. Because running a coupled simulation produces several logfiles written by various codes it might be useful to inspect these files, too. Therefore the MpCCI Logviewer has been enhanced to be configurable for other logfile types. The configuration is done by a properties file in JSON (JavaScript Object Notation) format.

The properties file used as default is "$MPCCI_HOME/gui/logviewerTypeProperties.json" and supports following logfile types:

**MpCCI log file ".log"** Logfiles written by the MpCCI server while running a coupled simulation. Usually these files are stored in the project directory of the coupled simulation. This is the default format if no other logfile type is specified.

**Abaqus message file ".msg"** Message files written by Abaqus. These files are stored in the Abaqus model directory.

**Abaqus CSS logfile ".css.log"** Logfiles written by Abaqus’ CSE engine coupled with MpCCI. These files are also stored in the Abaqus model directory.

6.3.1 Definition of a Logfile Type

The logfile type is defined in JSON format. The type structure is a map with a string as key and a structure as value. The key must be unique to all listed types and correlates the **typeName**.
Here an example of a type definition related to the MpCCI logfile with default as key and type name:

```json
"default" : {
    "typeName" : "default",
    "fileSuffix" : ".log",
    "description" : "MpCCI log file",
    "regexEvent" : "^$(source): $(level)?$(message)$",
    "regexMessage" : "\[(\[^\[\]\]+)\]",
    "regexLevelError" : "\*\*\* ERROR \*\*\* ",
    "regexLevelWarning" : "\*\* WARNING \*\* ",
    "regexLevelInfo" : "\*\*\* ",
    "regexLevelAction" : "\*\* ",
    "regexLevelDebug" : "\* 
}
```

The `'` character must be escaped (`\`) in order to be read. All values are strings and must be quoted (" "). The strings before the ':' are keywords and must appear as stated (except default which is the unique key and must be chosen individually).

The logfile structure is defined by a key - here default. This key is a string unique to all listed types. The structure itself consists of keywords which are set as follows:

- **typeName** The internal type name of the defined logfile type which usually correlates the key (here default).
- **fileSuffix** The file suffix identifying this logfile type. If the suffix isn’t unique to all file types it is unpredictable which type will be taken for this suffix.
- **description** A description of the logfile type which will be displayed by the MpCCI Logviewer (see 6.2.2 Information ▷).
- **regexEvent** A regular expression describing the start of a new event in the logfile. All lines following this event-start-line belong to the message of this event until a new event-start is spotted. The placeholders $(source), $(level) and $(message) will be replaced by the expressions for regexSource, regexLevel* and regexMessage whereas the expressions for each level will be combined to level by or’ing them ($regexLevelError$|$regexLevelWarning$|$regexLevelInfo$|$regexLevelAction$|$regexLevelDebug$). Each placeholder must hold a grouping construct () via the regexEvent value (as done for the log level) or via its own values (possible for regexSource and regexMessage). The matching groups will be taken as entries for the code filter resp. as identifiers for level and message.
- **regexSource** A regular expression describing the source for the log event. This source will be listed in the code filter (see 6.2.4 Filters ▷).
- **regexMessage** A regular expression describing the message of the log event. The message will be extended by the lines following until the start of a new event is spotted. It will be displayed in the message area of the MpCCI Logviewer (see 6.2.5 Table with Message Area ▷).
- **regexLevelError**, **regexLevelWarning**, **regexLevelInfo**, **regexLevelAction**, **regexLevelDebug** Regular expressions describing the level of the log event. The level is not necessary and only one, several or even all levels may be omitted. The level may be useful (e.g. the error level) to quickly access this information by filtering it out (see 6.2.4 Filters ▷).
7 MpCCI Visualizer

The MpCCI Visualizer is suitable for quickly checking whether the coupling process was successful. The coupling region, orphaned nodes and exchanged quantities can be checked to ensure that a coupling has really occurred.

7.1 Using the MpCCI Visualizer

7.1.1 Data Flow

![Figure 1: Data flow for the MpCCI Visualizer. The MpCCI Server Process collects the exchanged data and writes them to the tracefile.](image)

During the MpCCI coupling process, the data exchange between the coupled codes can be observed. The exchanged data can be collected by the MpCCI server process, who writes the data into a tracefile, which can finally be read directly by the MpCCI Visualizer for ".ccvx" or sent directly through socket communication to the MpCCI Monitor.

The tracefile is only written if some writers have been selected in the Settings step. This is achieved by selecting the writer’s (e.g. CCVX, VTK or VTFA) option Use in the Settings step of the MpCCI GUI. See \textsection 4.8 Settings Step\textsection for a description of the Settings step and “Getting Started” for a general description of the coupling process.

The name of the tracefile is set in the Settings step of the MpCCI GUI. The default name is "mpccirun-0000.ccvx" saved in a subdirectory named "mpccirun_<TIMESTAMP>.ccvx", which is located in the same directory as the corresponding project (".csp") file.

More and more data is added to the tracefile during the coupling process. The file can already be opened before the process is finished to check data during the process. The visualizer can only read data from the tracefile, not manipulate or write data.
7.1.2 Supported Platforms

The MpCCI Visualizer for .ccvx and online monitoring is included in the MpCCI downloads and is at present only available on Microsoft Windows and Linux platforms but the tracefiles may be located on a remote host.

7.1.3 Starting the Monitor

The MpCCI Monitor can be started from the command-line with

```
mpcci monitor
```

as well as from the MpCCI GUI menu by selecting Tools→Monitor.

The command `mpcci monitor -h` or `mpcci monitor -help` or `mpcci help monitor` shows up a short help text.

See $\gtrless 5.3.4 mpcci monitor \gtrless$ for a detailed command description.

7.1.4 Starting the Visualizer

The MpCCI Visualizer can be started from the command-line with

```
mpcci visualize [tracefile]
```

where a filename can be given as well as from the MpCCI GUI menu by selecting Tools→Visualizer.

The command `mpcci visualize -h` or `mpcci visualize -help` or `mpcci help visualize` shows up a short help text.

See $\gtrless 5.3.5 mpcci visualize \gtrless$ for a detailed command description.

7.2 MpCCI Visualizer for .ccvx and Online Monitoring

7.2.1 Introduction

The MpCCI Visualizer for .ccvx and online monitoring is a new application for viewing of coupling regions. Compared to the old MpCCI Visualizer for ".ccv", it does not have distinct windows for controls and viewing, but arranges panels and toolbars around the central viewing area (Figure 2). The panels provide user interfaces for defining what to visualize and how to visualize it:

The Case Panel shows information on the available cases. In MpCCI there is one case available for every code used in the coupled simulation. A case can be displayed in one of the available views in the viewports area.

The Selection Panel shows information on the selected item in the active view. Selectable items are mesh nodes, finite elements, and mesh parts. To select an item, one holds down the control key $\texttt{Ctrl}$ and left-clicks onto an item in a view.

The Settings Panel allows modifying the draw style of mesh parts in the active view by selecting parts from the list and activating different draw style options. A slider on the top allows to control an offset for views that show several superimposed cases.

The Results Panel allows selecting quantities for visualization. Scalar quantities are mapped onto part surfaces as color fringes. Vector quantities are shown as 3D vector glyphs. To select and deselect a quantity for mapping, one just double-clicks on an entry in the list of quantities. Controls below the list of quantities allow adjusting a variety of mapping parameters.
7.2 Main Window

The main window of the MpCCI Visualizer for \texttt{.ccvx} and online monitoring divides into three main areas (see Figure 2): Menus and toolbars on the window top, views in the window center, and panels on the left or on the right side.

7.2.3 Menus and Toolbars

7.2.3.1 "File" Menu

<table>
<thead>
<tr>
<th>Icon</th>
<th>Option</th>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>→ Open...</td>
<td>Ctrl+O</td>
<td>Browse for a local ccvx file or type a remote ccvx file in the form ([\text{user}@\text{hostname}]\text{path/to/file.ccvx(.gz)}). This option is available in the visualizer mode.</td>
</tr>
<tr>
<td>File</td>
<td>→ Connect to Host...</td>
<td>Ctrl+H</td>
<td>Connect to an MpCCI server for monitoring. This menu is available in the monitor mode.</td>
</tr>
<tr>
<td>File</td>
<td>→ Disconnect from Host...</td>
<td></td>
<td>Disconnect from an MpCCI server. This menu is available in the monitor mode.</td>
</tr>
<tr>
<td>File</td>
<td>→ Close</td>
<td></td>
<td>Close the current file or monitored job.</td>
</tr>
<tr>
<td>File</td>
<td>→ Export Image...</td>
<td>Ctrl+S</td>
<td>Export either the active view or all views as image file in a specified resolution. It is recommended to override the text color to black and the background color to white for images which will be used in presentations. Most important file formats such as PNG and JPEG are supported.</td>
</tr>
</tbody>
</table>
File → Copy to Clipboard

- Export either the active view or all views as image to the clipboard. This option is only available for Windows versions.

File → Preferences...

- Show the application preferences dialog. See Figure 10.

File → Exit [Alt+F4]

- Exit the application.
### 7.2.3.2 "View" Menu

<table>
<thead>
<tr>
<th>Icon</th>
<th>Option</th>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>View→Info Box</td>
<td></td>
<td>- Show/hide the info box in the active view.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>View→Text</td>
<td></td>
<td>- Show/hide textual information in the active view.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>View→Legend</td>
<td></td>
<td>- Show/hide part and color legends in the active view.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>View→Modify→Reset View</td>
<td></td>
<td>- Reset the viewpoint of the active view to its initial value.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>View→Modify→Frame Model</td>
<td></td>
<td>- Reset the viewpoint so that the whole model fits the active view.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>View→Rubber Band Zoom</td>
<td></td>
<td>- Active rubber band zooming. Define new zoom by dragging a rectangle in one of the views using the left mouse button.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>View→Viewports...</td>
<td></td>
<td>- Select a viewport configuration. Several configurations are available as illustrated by the icons. A maximum number of four views is supported.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>View→Perspective Projection</td>
<td></td>
<td>- Switch between perspective and orthographic projection.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>View→Move as Outline</td>
<td></td>
<td>- Draw the model as outlines when it is being panned or rotated.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>View→Synchronous Navigation</td>
<td></td>
<td>- Apply navigation to all views synchronously.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>View→Automatic Rotation Point</td>
<td></td>
<td>- The nearest model surface point will be used as rotation point instead of a fixed point.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>View→Set Rotation Point</td>
<td></td>
<td>- Select a new fixed rotation point by clicking into one of the views using the left mouse button.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>View→Navigation Mode→Zoom</td>
<td></td>
<td>- The model will never be penetrated or clipped when zooming into it.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>View→Navigation Mode→Walk</td>
<td></td>
<td>- The model will be penetrated and clipped when zooming into it. Only compatible with perspective projection.</td>
</tr>
</tbody>
</table>
**View → Spin Model**
- Do not stop rotating once rotation interaction has finished. The model will keep on spinning until halted.

**View → Selection...**
- Activates the part, element, or node selection mode, or selection by element ID. Click into one of the views using the left mouse button to select an item.

**View → Measure Distance**
- Create a measurement of nodes distances and coordinates.

**View → Delete Distance Measurement**
- Delete the last distance measurement.

**View → Show peaks**
- Show the minimum and maximum values on the selected view and for the activated result.

**View → Show value labels**
- Show the result values for each node or element as a label annotation.

**View → Stereo**
- Enable/disable quad buffered active stereo rendering for all views. This option requires a suitable hardware and driver installation.

**View → Full Screen**
- Switch application to full screen mode or windowed mode.

**View → XY Plot Mode**
- Switch XY plot mode from continuous lines to sample points.

**View → Regression Curves**
- Switch XY regression plot mode on the current plot data.

**View → Auto Update Plot Range**
- Enable/disable the update of the plot range.

### 7.2.3.3 "Animation" Menu

<table>
<thead>
<tr>
<th>Icon</th>
<th>Option</th>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🎥</td>
<td>Animation → Play Backward</td>
<td></td>
<td>Play the animation of coupling steps in backward direction.</td>
</tr>
<tr>
<td>🎥</td>
<td>Animation → Step Backward</td>
<td></td>
<td>Go one animation step backward.</td>
</tr>
<tr>
<td>🎥</td>
<td>Animation → Pause</td>
<td></td>
<td>Pause the animation of coupling steps. During online monitoring, prevent</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>the automatic skip to the newly received data (last step).</td>
</tr>
<tr>
<td>⏯</td>
<td>Animation → Stop</td>
<td></td>
<td>Stop the animation of coupling steps and go back to the first step.</td>
</tr>
<tr>
<td>🎥</td>
<td>Animation → Step Forward</td>
<td></td>
<td>Go one animation step forward.</td>
</tr>
<tr>
<td>🎥</td>
<td>Animation → Play</td>
<td></td>
<td>Play the animation of coupling steps in forward direction.</td>
</tr>
<tr>
<td>🎥</td>
<td>Animation → Record</td>
<td></td>
<td>Record the animation of coupling steps as an animated GIF, AVI, WMV, MP4.</td>
</tr>
</tbody>
</table>
### 7.2.3.4 "Case" Menu

<table>
<thead>
<tr>
<th>Icon</th>
<th>Option</th>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Case→Next Case" /></td>
<td>Case→Next Case</td>
<td>-</td>
<td>Load the next available case (i.e. coupling results of a code) into the active view.</td>
</tr>
<tr>
<td><img src="image" alt="Case→Previous Case" /></td>
<td>Case→Previous Case</td>
<td>-</td>
<td>Load the previous available case into the active view.</td>
</tr>
</tbody>
</table>

### 7.2.3.5 "Panels" Menu

<table>
<thead>
<tr>
<th>Icon</th>
<th>Option</th>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Panels→Case Panel" /></td>
<td>Panels→Case Panel</td>
<td>-</td>
<td>Show/hide the panel containing the list of available cases.</td>
</tr>
<tr>
<td><img src="image" alt="Panels→Selection Panel" /></td>
<td>Panels→Selection Panel</td>
<td>-</td>
<td>Show/hide the panel containing information on the currently selected items in the active view.</td>
</tr>
<tr>
<td><img src="image" alt="Panels→Settings Panel" /></td>
<td>Panels→Settings Panel</td>
<td>-</td>
<td>Show/hide the panel for modification of part draw styles and superimposition offset.</td>
</tr>
<tr>
<td><img src="image" alt="Panels→Results Panel" /></td>
<td>Panels→Results Panel</td>
<td>-</td>
<td>Show/hide the panel for selection of available quantities for visualization.</td>
</tr>
<tr>
<td><img src="image" alt="Panels→Extraction Panel" /></td>
<td>Panels→Extraction Panel</td>
<td>-</td>
<td>Show/hide the panel for extraction features, i.e. cut planes and iso surfaces.</td>
</tr>
<tr>
<td><img src="image" alt="Panels→FSIMapper" /></td>
<td>Panels→FSIMapper</td>
<td>-</td>
<td>Show the FSIMapper panel for file based mapping; only available in FSIMapper mode.</td>
</tr>
</tbody>
</table>

### 7.2.3.6 "Help" Menu

<table>
<thead>
<tr>
<th>Icon</th>
<th>Option</th>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Help→Help Contents" /></td>
<td>Help→Help Contents</td>
<td>F1</td>
<td>Show help for MpCCI Visualizer.</td>
</tr>
<tr>
<td><img src="image" alt="Help→About" /></td>
<td>Help→About</td>
<td>-</td>
<td>Show application information and system diagnostics.</td>
</tr>
</tbody>
</table>
7.2.4 Panels

7.2.4.1 "Case" Panel

The case panel lists all available cases. In MpCCI, one case is created for every code in the coupled simulation.

The Table of Cases shows a case snapshot image along with the case name and date. The case snapshot image might not be available if the case has not been loaded yet. The primary case of the active view is shown in bold font. Superimposed cases, i.e. those which are shown together with the primary case of the active view, are shown in italic font.

The Show detailed checkbox allows switching to a more compact listing.

The description shown in the Case Overview area is HTML formatted. Use the context menu to copy it to the clipboard or save it to a file.

To modify the font size, one either presses the buttons + - or presses the keys \texttt{Ctrl +} and \texttt{Ctrl -}.

The available mouse interactions with case entries are all related to the assignment of cases to views. Every view has a primary case, and optionally many superimposed cases. The following table lists the mouse interactions with case entries.

<table>
<thead>
<tr>
<th>Mouse action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left mouse button click</td>
<td>Select case and show further case information in the Case Overview area.</td>
</tr>
<tr>
<td>Left mouse button double click</td>
<td>Load case into the active view as primary case.</td>
</tr>
<tr>
<td>Left mouse button drag</td>
<td>Drag case entry onto a view to show it as primary case.</td>
</tr>
<tr>
<td>Left mouse button drag + \texttt{ALT}</td>
<td>Drag case entry onto a view to superimpose it as additional case.</td>
</tr>
<tr>
<td>Right mouse button click</td>
<td>Use the context menu to assign or superimpose the selected case.</td>
</tr>
</tbody>
</table>

Figure 3: Case panel.
7.2.4.2 "Selection" Panel

The selection panel shows information on the currently selected item in the active view. This can either be a part, an element, or a mesh node. By changing the tabs and clicking into the active view with left mouse button and $\text{Ctrl}$ pressed, one can easily select new items and change the type of selection.

The description of the selected item is shown in the text area. The options Show details and Hide details allow to control the amount of information shown.

Depending on the selected item, one can click on several links, which are underlined and highlighted in blue. These links point to other selectable items. That way, one can easily obtain information on the nodes of a selected element, for instance.

Note that the information shown is only valid for the selected item in the associated animation step, since the selected item in one step might not have a corresponding item in other animation steps. Obviously, this is the case if the mesh topology changes, for example. So keep in mind that links are only valid on a "per animation step" basis.

The text and tables shown are HTML formatted. Use the context menu to copy the content to the clipboard or save it to a file. Note that any images shown will not be exported. To modify the font size, one either presses the buttons $+_{-}$ or presses the keys $\text{Ctrl}+$ and $\text{Ctrl}-$.

The Tabs on top checkbox allows moving the Part, Element, and Node tabs to the side.

Figure 4: Selection panel.
7.2.4.3 "Settings" Panel

The settings panel lists the parts of the primary case assigned to the active view. Also, the offset method for superimposition can be changed in the settings panel.

The **Superimposition offset** option allows controlling the offset factor. The slider sets the offset factor from a minimum (left), i.e. no offset, to a maximum (right).

Offsetting superimposed cases typically helps when one has configured a single view with all cases assigned, i.e. one primary case plus a number of superimposed cases. If the coupling regions associated with the cases are overlapping, then they can be separated using the offset.

Four offset axes are available: The **x-Axis**, **y-Axis**, **z-Axis**, and **Auto** axis, where the latter option selects a suitable axis depending on the current viewing position and viewing orientation.

The second, and more prominent, user interface area of the settings panel shows the list of parts of the active view’s primary case. One can select parts from the list and modify their draw style, using several checkboxes and radio-button in the **Draw Style** area right to the list of parts, as outlined in the table below. The part selection can also be done by regular expressions (case insensitive and greedy) entered in the text box. By pressing the **Select** button all matching parts are highlighted.

The **Table of Parts** lists the parts of the current active case in a hierarchic structure. As default all parts are grouped below the root node. New subgroups can be created by selecting one or multiple items and clicking the right mouse button followed by selecting **New Group** in pop up menu and definition of the group’s name. There is also the possibility to create automatically groups of parts per mesh id: select **New Groups from Meshes** in the pop up menu.

Created groups can be deleted by selecting the group item or one of its members and clicking the right mouse button followed by selecting **Delete Group**. All **Draw Style** modifications applied to a group are applied to each group member.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Visible (default)</td>
<td>Show or hide the selected parts.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Show as surface</td>
<td>Show surface geometry of selected parts as surface.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Show as lines</td>
<td>Show surface geometry of selected parts as lines.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Show as points</td>
<td>Show surface geometry of selected parts as points.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Show as outlines</td>
<td>Show surface geometry of selected parts as outlines.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Show as hidden lines</td>
<td>Show surface geometry of selected parts as lines, with all hidden lines removed.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Overlay mesh lines</td>
<td>Show the edges of finite elements as lines on top of the surface geometry of parts. Only applies to surface draw style.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Overlay mesh outlines</td>
<td>Show the outlines over on top of the surface geometry of parts. Only applies to surface draw style.</td>
</tr>
</tbody>
</table>
Shrink finite elements

Shrink all finite elements of the selected parts. Note that this setting can degrade performance significantly. However, it is useful when one superimposes cases without using a superimposition offset.

7.2.4.4 "Results" Panel

The results panel lists available scalar, vector, and displacement results of the primary case assigned to the active view. Scalar results are visualized as color fringes on the mesh surface, vector results are visualized as glyphs, and displacement results apply to node positions. Note that some datasets automatically supply displaced node positions. Thus even if the mesh appears displaced, it can be that there is no displacement result shown in the list.

One double-clicks on a result to choose it for visualization. Only one scalar result, one vector result, and one displacement result can be visualized at a time. Once a result has been chosen, an icon next to the result name is shown, indicating the currently visualized result. Also the result settings area below the list is activated. Its appearance changes depending on the chosen result type.

The options for scalar results allow to modify color legend settings used for color fringes. The options for vector results allow to modify the appearance of vector glyphs. The options for displacement results allow to modify the displacement multiplication factors, i.e. typically uses when the displacement is very small, but shall be overemphasized for visualization.

The option Synchronize settings to all cases is enabled by default. It allows applying result visualization settings to all cases synchronously. I.e. a scalar result "Wall Temp [C]" will be visualized for all cases if it is available for all cases. Other settings of the results panel are also synchronized. This synchronization is useful since corresponding results are typically available for all associated codes in MpCCI.

Settings for Scalar Results

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic range from</td>
<td>The range for color fringes is chosen automatically, depending on the selected options. With the option All frames the range is calculated by the minimum and maximum over all available frames and views. In contrast, the option Curr. frame only considers the data of the currently visible frame. Having selected here the option All views, the range's minimum and maximum are built over all views whereas the option Single view results in minimum and maximum values per view.</td>
</tr>
</tbody>
</table>
Choose manually the Minimum and the Maximum of the range for color fringes. One can either use the numeric input field or the slider. The maximum cannot be lower than the minimum value.

Modify legend coloring used for color fringes.
Select a pre-defined color scale for color fringes. The most common color scales are "Default", a rainbow color scale, and "Metal casting".
Use a leveled color scale, divided into the given number of color levels. This mode is suited to visualize iso contours for the scalar result.
Use a continuous color scale, with smooth transitions from color to color.
Apply logarithmic scaling to color fringes. This is only supported for leveled color scales.
Draw scalar values outside of the current minimum and maximum value in gray color. Otherwise, clamp to the colors of minimum and maximum value.

### Settings for Vector Results

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show absolute vector length</td>
<td>Vector length representation uses the absolute length value.</td>
</tr>
<tr>
<td>Draw vectors as lines</td>
<td>Change the vector arrows representation to lines.</td>
</tr>
<tr>
<td>Scaling</td>
<td>Scaling factor for vector glyphs. The scaling factor relates the maximum possible glyph size with the dataset’s bounding box diagonal.</td>
</tr>
<tr>
<td>Draw skip</td>
<td>Option to skip every n’th vector while drawing. This is useful for datasets with many vectors. The skip factor can reduce cluttering and improve rendering performance.</td>
</tr>
<tr>
<td>Coloring</td>
<td>Determines glyph coloring, using either a Constant color, as defined by the colored button, or Color by scalar in which case the glyph color corresponds to the color fringes of the active scalar result.</td>
</tr>
</tbody>
</table>

### Settings for Displacement Results

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scaling</td>
<td>Scaling factor for displacement vectors. A scaling factor greater than one visually overemphasizes the displacement.</td>
</tr>
</tbody>
</table>
7.2.4.5 "Extraction" Panel

The extraction panel allows to create cut planes and iso surfaces for post-processing of volumetric coupling meshes. These extraction features are either created for an individual case or synchronized for all cases, which is the default mode. Use the checkbox *Synchronize settings to all cases* to modify this behavior. All changes to cut plane and iso surface parameters are also synchronized, if possible.

To create and delete extraction features, one activates the corresponding tab in the extraction panel and clicks on the Create/Delete buttons below the list widget on top of the tab. The list item selection indicates the active cut plane or iso surface. Changes of parameters apply to the active item, as well as to its corresponding items, if synchronization is enabled. Note that the part draw style switches automatically to Outline when an iso surface is being created and the current part draw style is set to Surface.

The parameters for a cut plane are the mapped scalar and vector result, the plane point, plane normal, and several switches to define the appearance, such as Show Surface or Show Vectors. It is possible to modify the cut plane point and normal using mouse interaction. Hold down both Ctrl and Shift in addition to the usual mouse button navigation scheme for panning, rotating, and zooming, i.e. displacing the plane in normal direction.

By default, a cut plane clips the part geometry in normal direction. As an alternative, one can disable clipping and switch to Outline draw style in the Settings Panel.

When Ignore Part Visibility is enabled, the cut plane is being computed for all parts, disregarding part visibility settings as defined in the Settings Panel.

The main parameter for an iso surface is the scalar for the iso threshold value (*Compute From*). Also, one can map a scalar and vector result onto an iso surface. A line edit and a dial allows to modify the iso threshold value. This interface is the same as used in the Results Panel. All other iso surface parameters have the same meaning as for cut planes.

The parameters of color scales and vector glyphs for results which are mapped on extraction features can be changed in the Results Panel. Those results referenced by extraction features show a small x in front of the result name.
7.2.5 Viewport Area

The viewport area contains up to four views (see Figure 8). The active view is marked by a green border around it. If only one view is shown, then it is always the active one. Note that most options in the view menu apply to the active view only, except if synchronized interaction is activated (such as for synchronized navigation).

The mouse interaction options for a view are shown in the following section, as well as a description of options available in the context menu of a view.

Figure 8: Viewports area. The active view is highlighted by a green border.
7.2.5.1 View Mouse Interaction

<table>
<thead>
<tr>
<th>Mouse Button</th>
<th>Operation</th>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left</td>
<td>Drag</td>
<td>-</td>
<td>Rotate</td>
</tr>
<tr>
<td>Middle</td>
<td>Drag</td>
<td>-</td>
<td>Zoom</td>
</tr>
<tr>
<td>Right</td>
<td>Drag</td>
<td>-</td>
<td>Pan</td>
</tr>
<tr>
<td>Mouse wheel</td>
<td>Scroll</td>
<td>-</td>
<td>Zoom</td>
</tr>
<tr>
<td>Right</td>
<td>Click</td>
<td></td>
<td>Show context menu.</td>
</tr>
<tr>
<td>Left</td>
<td>Click</td>
<td>Ctrl</td>
<td>Select item. Item information will be shown in the selection panel. Click into empty view area to clear the item selection.</td>
</tr>
<tr>
<td>Right</td>
<td>Click</td>
<td>Ctrl</td>
<td>Show/hide part under cursor. Click into empty view area to show all parts again.</td>
</tr>
<tr>
<td>Left</td>
<td>Drag</td>
<td>Ctrl+Shift</td>
<td>Rotate active cut plane</td>
</tr>
<tr>
<td>Middle</td>
<td>Drag</td>
<td>Ctrl+Shift</td>
<td>Displace active cut plane into normal direction</td>
</tr>
<tr>
<td>Right</td>
<td>Drag</td>
<td>Ctrl+Shift</td>
<td>Pan active cut plane</td>
</tr>
<tr>
<td>Mouse wheel</td>
<td>Scroll</td>
<td>Ctrl+Shift</td>
<td>Displace active cut plane into normal direction</td>
</tr>
</tbody>
</table>

7.2.5.2 View Context Menu

The view context menu is available by right-clicking into a view. It bundles a set of important options from the View menu, as well as options to select the primary and superimposed cases for the view.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Select Node" /></td>
<td>Activates node selection mode. Click into the view using the left mouse button to select a node.</td>
</tr>
<tr>
<td><img src="image" alt="Select Part" /></td>
<td>Activates part selection mode. Click into the view using the left mouse button to select a part.</td>
</tr>
<tr>
<td><img src="image" alt="Select Element" /></td>
<td>Activates element selection mode. Click into the view using the left mouse button to select an element.</td>
</tr>
<tr>
<td><img src="image" alt="Reset View" /></td>
<td>Reset the viewpoint of the view to its initial value.</td>
</tr>
<tr>
<td><img src="image" alt="Frame Model" /></td>
<td>Reset the viewpoint so that the whole model fits the view.</td>
</tr>
<tr>
<td><img src="image" alt="Show/Hide Text" /></td>
<td>Show/hide textual information in the view.</td>
</tr>
</tbody>
</table>

Figure 9: View context menu.
Show/Hide → Color Legend

Show/hide part and color legends in the view.

Show/Hide → Info Box

Show/hide the info box in the view.

Assign Case...

Select a case from a list, and set it as primary case of the view.

Superimpose Case...

Select a case from a list, and add it as superimposed case to the view.
### 7.2.6 Preferences Viewer Dialog

The preferences dialog allows setting several fixed application preferences related to file type associations, navigation styles, graphics hardware settings, and stereo rendering options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Application Settings</strong></td>
<td></td>
</tr>
<tr>
<td>Navigation profile</td>
<td>Select a default navigation profile for the mouse interaction in views. The default is MpCCI style navigation. The only alternative is currently GLview Inova navigation style. Use either icons of size 24x24 pixels or 16x16 pixels for button in the toolbar. The latter is beneficial for low resolution screens and automatically chosen for screen widths lower equal 1024 pixels. Dis/Enable synchronized navigation on application start.</td>
</tr>
<tr>
<td>Toolbar icon size</td>
<td>Use either icons of size 24x24 pixels or 16x16 pixels for button in the toolbar. The latter is beneficial for low resolution screens and automatically chosen for screen widths lower equal 1024 pixels.</td>
</tr>
<tr>
<td>Synchronized navigation</td>
<td>Dis/Enable synchronized navigation on application start.</td>
</tr>
<tr>
<td><strong>Rendering Settings</strong></td>
<td></td>
</tr>
<tr>
<td>Fast single frame drawing</td>
<td>On/Auto: Draw single frames faster using features of recent graphics hardware, if available. Off: Disable option in case of hardware or graphics driver problems.</td>
</tr>
<tr>
<td>Use flipbook animation</td>
<td>Dis/Enable flipbook animation. The flipbook is updated while an animation runs and played back instead of rendering the actual geometry, as long as the viewing setup and contents do not change.</td>
</tr>
<tr>
<td>Maximum animation FPS</td>
<td>Define a maximum number of frames per second for a running animation. Typically the size of the dataset limits the frames per second, so this setting is more useful for small datasets.</td>
</tr>
<tr>
<td>Use software OpenGL</td>
<td>Use software OpenGL rendering, bypassing the graphics hardware. This option is only available for Windows platforms.</td>
</tr>
</tbody>
</table>

![Preferences viewer dialog](image.png)

*Figure 10: Preferences viewer dialog.*
**Background color**

Background color mode. The entry *Automatic* chooses the default color black or the color defined by a ccvx file. Otherwise, the colors *Black*, *White*, and *Sky* are available.

**Stereo Settings**

**Camera offset**

Define a camera offset factor for stereo rendering. Default is 1.0 (slider middle), allowed range is [0.0 (left) .. 2.0 (right)]. In effect this factor controls the amount of eye channel separation.

**Parallax balance**

**Focus mode**

Defines the mode to compute the reference point of zero parallax. The default value "Follow rotation point" is recommended. Otherwise a fixed focal length or focal reference point may be supplied.

**Focal length**

Focal length for "Fixed focal length" focus mode.

**Focal point**

Focal point for "Fixed focal point" focus mode.

**Connection Settings**

**Timeout**

Timeout value in seconds for the monitor connection.
7.2.7 Preferences Server Dialog

The preferences dialog allows setting several monitor preferences related to the co-simulation. Details for each setting can be found at § 4.8.1 Monitor.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Server Information</strong></td>
<td></td>
</tr>
<tr>
<td>Orphans</td>
<td>Dis/Enable the send of orphans information.</td>
</tr>
<tr>
<td>Node Ids</td>
<td>Dis/Enable the send of node ids.</td>
</tr>
<tr>
<td>Element Ids</td>
<td>Dis/Enable the send of element ids.</td>
</tr>
<tr>
<td>Element Sizes</td>
<td>Dis/Enable the send of element sizes.</td>
</tr>
<tr>
<td>Global Variables</td>
<td>Dis/Enable the send of global variables.</td>
</tr>
<tr>
<td>Slave Nodes</td>
<td>Dis/Enable the send of slaves node.</td>
</tr>
<tr>
<td>Node Domains</td>
<td>Dis/Enable the send of node domains information.</td>
</tr>
<tr>
<td><strong>Peak Values</strong></td>
<td></td>
</tr>
<tr>
<td>Iteration Norm And Mean Values</td>
<td>Dis/Enable the send of iteration norm and mean values.</td>
</tr>
<tr>
<td>Element Normals</td>
<td>Dis/Enable the send of element normals.</td>
</tr>
<tr>
<td>Absolute Quantity Difference</td>
<td>Dis/Enable the send of absolute quantity difference.</td>
</tr>
<tr>
<td>Relative Quantity Difference</td>
<td>Dis/Enable the send of relative quantity difference.</td>
</tr>
<tr>
<td>Periodic Replicates</td>
<td>Dis/Enable the send of the replicated periodic parts and quantities.</td>
</tr>
<tr>
<td>Aitken Relaxation Value</td>
<td>Dis/Enable the send of the Aitken relaxation value.</td>
</tr>
</tbody>
</table>

7.2.8 Store Animated Files Dialog

The store animated files dialog allows the export of animated GIF’s as well as some commonly used video formats. For the latter the MpCCI Visualizer makes use of the free available (L)GPL licensed third party video rendering library Libav (http://libav.org/) which is available for Windows and Linux systems and needs a separate installation on user side.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Store Animated Files</td>
<td>Select GIF export format.</td>
</tr>
<tr>
<td>AVI</td>
<td>Select AVI export format.</td>
</tr>
</tbody>
</table>
WMV
Select WMV export format.

MP4
Select MP4 export format.

Frame repeat
Specify the number of frame to repeat.

Frame rate
Specify the frame rate for the video export.

Export prefix
Specify the path and file name prefix for the exported file.


On Linux the user of Libav has to decide which free software license should be used during compilation. As end-user of the library option $--enable-nonfree$ would be suitable.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--prefix=PREFIX</td>
<td>install binaries in PREFIX</td>
</tr>
</tbody>
</table>

Licensing options:

- $--enable-gpl$ allow use of GPL code, the resulting libs and binaries will be under GPL [no]
- $--enable-version3$ upgrade (L)GPL to version 3 [no]
- $--enable-nonfree$ allow use of nonfree code, the resulting libs and binaries will be unredistributable [no]

so that a installation of binaries in target folder PREFIX would look like

```
./configure --disable-yasm --prefix=PREFIX --enable-nonfree
make
make install
```

Compiled binaries will then finally be located in folder PREFIX/bin.

To make use of Libav in MpCCI Visualizer the user has to extend system 'PATH' variable by Libav's binary executable folder containing the tool 'avconv' which is responsible for video conversion.

To check if 'avconv' is available in system path type `avconv --version` in your system command prompt.

If 'avconv' is not available the export options for AVI, WMV and MP4 will be grayed out. The export of animated GIF's is independent from Libav and does not require an installation.
7.2.9 Error Dialog

An error dialog appears in case of recoverable MpCCI Visualizer errors, showing the error message and optionally further detail information. One can copy the error message and details using the Copy button.

Linux Note: If MpCCI Visualizer dies in case of unrecoverable application errors, one can find the text file `bl_mpci_visualizer.txt` in the execution path which provides useful debugging information for the MpCCI support.

![Error dialog](image)

Figure 13: Error dialog with optional details.
7.2.10 Command Line Parameters

MpCCI Visualizer for .ccvx and online monitoring calling syntax:

```
mpcci_visualizer.exe <command line options>
```

Command line options (usually set by MpCCI GUI automatically):

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-help</td>
<td>Overview of command line options</td>
</tr>
<tr>
<td>-connect <a href="mailto:port@host">port@host</a></td>
<td>Connect to one or more servers</td>
</tr>
<tr>
<td>-monitor</td>
<td>Switch from Visualize mode (default) to Monitor mode</td>
</tr>
<tr>
<td>-netdevice &lt;string&gt;</td>
<td>Use the specified network device for monitoring</td>
</tr>
<tr>
<td>-modulo &lt;number&gt;</td>
<td>Save memory by showing only &lt;number&gt; steps</td>
</tr>
<tr>
<td>-maxmem &lt;number&gt;</td>
<td>Save memory by using at most &lt;number&gt; MB per case</td>
</tr>
<tr>
<td>-jobname &lt;name&gt;</td>
<td>Default job name</td>
</tr>
<tr>
<td>-updatedelay &lt;ms&gt;</td>
<td>Scene update after SYNC command limited to once per &lt;ms&gt; milliseconds, default is 0, i.e. no limit</td>
</tr>
</tbody>
</table>
7.3 Frequently Asked Questions

| Question: | The quantity NPosition is used for the FSI coupled simulation. Should it be equivalent to displacement in the FE post-processing? |
| Answer: | The MpCCI Visualizer is displaying the displacement for the NPosition quantity. The deformed structure is also displayed. |

| Question: | FE post-processing can display stresses result. Can the MpCCI Visualizer display the stresses as in the FE post-processing tools? |
| Answer: | No, MpCCI Visualizer only shows the exchanged quantity values selected in MpCCI GUI for the coupling or monitoring. |
8 MpCCI Grid Morpher

The MpCCI Grid Morpher will smooth a grid based on the displacements of some boundary or interior vertices. A moving or morphing grid capability is always required with fluid-structure interactions.

8.1 Using the MpCCI Grid Morpher

MpCCI offers a spring based morphing method which

- supports socket communication with parallel simulation codes.
- allows vertices to float along semi-planar boundaries, e.g. symmetry planes.
- may be monitored within an additional output terminal and a morpher log file in the model working directory ("mpcci_morpher_<model name>.log").
- can be used as a tool to morph grids in files e.g. geometry variations.

MpCCI Grid Morpher is actually implemented for the following simulations codes:

- ANSYS
- OpenFOAM

MpCCI Grid Morpher needs to read a "gmd" file which contains the mesh definition of the model. After reading the "gmd", the MpCCI Grid Morpher executable is waiting for the list of nodes to displace. Morphed nodes are then sent back to the simulation code. For each supported code, MpCCI provides a "gmd" file converter executable reading the native file mesh of the simulation code.

The MpCCI Grid Morpher executable has a lot more options (see command line options) for better fine control in case of difficult morphing scenarios. In this case the use of an options file instead of the MpCCI GUI is the better alternative.

The morpher controls the results of the morphing step, it may control the length of the edges, the shape and size of faces and also checks the quality of cells. There are options available to limit the compression and elongation of edges.

8.1.1 Options Description

The following options may be adjusted, whereas the default values are in many cases appropriate.

Optional morpher hostname  Enter an optional hostname for a remote execution of the MpCCI Grid Morpher.

Morpher port no.  Specifies the port address on which the MpCCI Grid Morpher listens for the client connection. The default port is in most cases acceptable, modifications are only required if firewalls block connections or multiple morphers are running in parallel on the same host.

No. of parallel morpher threads.  The MpCCI Grid Morpher is multi threaded parallelized. The amount of parallel threads used will never be larger than the number of cores on the host even if more threads are requested. However in case that some of the cores on the morpher host are used by other processes you should reduce the number of threads/cores reserved by the morpher to achieve the best performance.

Morpher options file  Instead of using the selectable options from the list below you may specify an options file as an alternative. The options file should have the suffix ".gmo". The options file contains all command line options valid for the morpher executable, except the port number and hostname and the number of threads and the list of cells zone names or cell type ids. The options file contains floating text. You may comment out a line with the # sign.
Please select the output level

- quiet: no output
- default: medium level output
- verbose: high level output
- full: highest level output (verbose and received node displacements)

Check edges
Activate to invoke quality checks for edges during morpher run.

Check faces
Activate to invoke quality checks for faces during morpher run.

Check cells
Activate to invoke quality checks for cells during morpher run.

List of cell type ids (Integers or ALL)
Enter the cell table number of the fluid domains where the MpCCI Grid Morpher should be applied. For OpenFOAM the names of the cell zones where the morpher should be used have to be entered as strings, separated by spaces. If you want to involve all fluid domains please enter “ALL”.

Min. relative edge length change
Enter a lower limit for the ratio of the edge length, calculated by the morpher, to the edge length of the input grid provided to the morpher. A modified value might be interesting if the morpher generates too small edge lengths.

Max. relative edge length change
Enter an upper limit for the ratio of the edge length, calculated by the morpher, to the edge length of the input grid provided to the morpher. This might be desired if the morpher generates too large edge lengths.

Min. change of face area
Enter a lower limit for the ratio of the face area, calculated by the morpher, to the face area of the input grid provided to the morpher. This might be desired if the morpher generates too small face areas.

Max. change of face area
Enter an upper limit for the ratio of the face area, calculated by the morpher, to the face area of the input grid provided to the morpher. This might be desired if the morpher generates too large face areas.

Max. face aspect ratio
Enter an upper limit for the cells aspect ratio. For triangular faces the aspect ratio is built by the height to the corresponding maximum edge length.

Min. angle allowed in faces and cells
Enter a lower limit for angles in faces and cells to prevent distortion.

Fixed nodes on fixed boundaries
Enter the amount of cell levels to be fixed seen from boundaries with fixed nodes.

Fixed nodes on deformed boundaries
Enter the amount of cell levels to be fixed seen from boundaries with displaced nodes. The specified number of node levels will be moved with the boundary in a rigid manner.

Optional list of floating boundary regions
Enter the boundary region numbers corresponding to the OpenFOAM,... model, where a sliding of vertices is permitted. By default floating is permitted only on symmetry and cyclic boundaries. Vertices on inlet, outlet and wall regions are fixed by default.

Optional list of fixed boundary regions
Enter the boundary region numbers corresponding to the OpenFOAM,... model, where a floating of vertices is explicitly not permitted (e.g. in the case where you might want to fix vertices on a symmetry plane).

Morphing relaxation factor
Enter a relaxation factor for solving the equation system. Larger values will speed up convergence. In case of instability it might be helpful to reduce the value.

Max. no. of iterations
Number of iterations for solving the equation system.
**Convergence tolerance**  The convergence tolerance is the ratio of the maximum node displacement at the beginning of the morphing iteration to the node displacement of the current morphing iteration step. The iterative process for solving the equation system will stop, when the convergence tolerance is reached.

**Smoothing steps**  Specify the amount of laplacian smoothing steps. Smoothing should be handled with care.
# VI Codes Manual – Contents

## 1 Overview
1.1 Common MpCCI Subcommands for Simulation Codes ........................................ 12
1.2 Unit Systems ......................................................................................... 14

## 2 Abaqus
2.1 Quick Information ............................................................................. 15
   2.1.1 Supported Coupling Methods ....................................................... 15
   2.1.2 Supported Platforms and Versions .............................................. 15
   2.1.3 References .................................................................................. 15
   2.1.4 Adapter Description .................................................................... 16
   2.1.5 Prerequisites for a Coupled Simulation ....................................... 16
2.2 Coupling Process .............................................................................. 16
   2.2.1 Model Preparation .................................................................... 16
   2.2.2 Restart ....................................................................................... 17
   2.2.3 Models Step ............................................................................... 17
   2.2.4 Algorithm Step .......................................................................... 18
   2.2.5 Regions Step ............................................................................. 18
   2.2.6 Go Step ..................................................................................... 19
   2.2.7 Running the Computation ........................................................... 20
   2.2.8 Post-Processing ......................................................................... 21
2.3 Code-Specific MpCCI Commands .............................................................. 22
2.4 Code Adapter Reference ...................................................................... 23
   2.4.1 Patched Input File ..................................................................... 23
   2.4.2 SIMULIA’s Co-Simulation Engine (CSE) ...................................... 23
2.5 Co-Simulation Restart ........................................................................ 23
2.6 Known Limitations ............................................................................ 24
2.7 Trouble Shooting, Open Issues and Known Bugs ................................. 25

## 3 ANSYS
3.1 Quick Information ............................................................................ 26
   3.1.1 Supported Coupling Methods ...................................................... 26
   3.1.2 Supported Platforms and Versions .............................................. 26
   3.1.3 References .................................................................................. 26
   3.1.4 Adapter Description .................................................................... 26
   3.1.5 Prerequisites for a Coupled Simulation ....................................... 27
   3.1.6 Supported ANSYS Product Variable ......................................... 27
3.2 Coupling Process .............................................................................. 29
   3.2.1 Model Preparation .................................................................... 29
Contents

3.2.2 APDL Script .................................................. 32
3.2.3 Models Step .................................................. 36
3.2.4 Algorithm Step .............................................. 36
3.2.5 Regions Step .................................................. 36
3.2.6 Go Step ....................................................... 39
3.2.7 Running the Computation ................................... 40
3.3 Code-Specific MpCCI Commands .............................. 41
3.4 Code Adapter Reference ........................................ 43
3.5 Frequently Asked Questions ................................... 43

4 ANSYS Icepak .................................................. 45
4.1 Quick Information .............................................. 45
4.1.1 Supported Coupling Methods ............................ 45
4.1.2 Supported Platforms and Versions ....................... 45
4.1.3 Supported Quantities ....................................... 45
4.2 Code-Specific MpCCI Commands .............................. 46

5 FINE/Open .................................................... 48
5.1 Quick Information .............................................. 48
5.1.1 Supported Coupling Methods ............................ 48
5.1.2 Supported Platforms and Versions ....................... 48
5.1.3 References .................................................. 48
5.1.4 Adapter Description ....................................... 48
5.1.5 Prerequisites for a Coupled Simulation ................ 48
5.2 Coupling Process .............................................. 49
5.2.1 Model Preparation ......................................... 49
5.2.2 Models Step .................................................. 49
5.2.3 Algorithm Step .............................................. 50
5.2.4 Regions Step ............................................... 50
5.2.5 Go Step ....................................................... 50
5.2.6 Running the Computation ................................ 51
5.2.7 Post-Processing ............................................. 53
5.3 Code-Specific MpCCI Commands .............................. 53
5.4 Code Adapter Reference ........................................ 54
5.5 Limitations ..................................................... 54
# FINE/Turbo

## 6.1 Quick Information

### 6.1.1 Supported Coupling Methods

### 6.1.2 Supported Platforms and Versions

### 6.1.3 References

### 6.1.4 Adapter Description

### 6.1.5 Prerequisites for a Coupled Simulation

---

## 6.2 Coupling Process

### 6.2.1 Model Preparation

### 6.2.2 Models Step

### 6.2.3 Algorithm Step

### 6.2.4 Regions Step

### 6.2.5 Go Step

### 6.2.6 Running the Computation

### 6.2.7 Post-Processing

---

## 6.3 Code-Specific MpCCI Commands

## 6.4 Code Adapter Reference

## 6.5 Trouble Shooting, Open Issues and Known Bugs

---

# FloMASTER

## 7.1 Quick Information

### 7.1.1 Supported Coupling Methods

### 7.1.2 Supported Platforms and Versions

### 7.1.3 References

### 7.1.4 Adapter Description

### 7.1.5 Prerequisites for a Coupled Simulation

---

## 7.2 Coupling Process

### 7.2.1 Model Preparation

### 7.2.2 Models Step

### 7.2.3 Algorithm Step

### 7.2.4 Regions Step

### 7.2.5 Go Step

---

## 7.3 Code-Specific MpCCI Commands

## 7.4 Code Adapter Reference

## 7.5 Trouble Shooting, Open Issues and Known Bugs
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>MSC Marc</td>
</tr>
<tr>
<td>12.1</td>
<td>Quick Information</td>
</tr>
<tr>
<td>12.1.1</td>
<td>Supported Coupling Methods</td>
</tr>
<tr>
<td>12.1.2</td>
<td>Supported Platforms and Versions</td>
</tr>
<tr>
<td>12.1.3</td>
<td>References</td>
</tr>
<tr>
<td>12.1.4</td>
<td>Adapter Description</td>
</tr>
<tr>
<td>12.1.5</td>
<td>Prerequisites for a Coupled Simulation</td>
</tr>
<tr>
<td>12.2</td>
<td>Coupling Process</td>
</tr>
<tr>
<td>12.2.1</td>
<td>Model Preparation</td>
</tr>
<tr>
<td>12.2.2</td>
<td>Models Step</td>
</tr>
<tr>
<td>12.2.3</td>
<td>Algorithm Step</td>
</tr>
<tr>
<td>12.2.4</td>
<td>Regions Step</td>
</tr>
<tr>
<td>12.2.5</td>
<td>Go Step</td>
</tr>
<tr>
<td>12.2.6</td>
<td>Running the Computation</td>
</tr>
<tr>
<td>12.2.7</td>
<td>Post-Processing</td>
</tr>
<tr>
<td>12.3</td>
<td>Code-Specific MpCCI Commands</td>
</tr>
<tr>
<td>12.4</td>
<td>Trouble Shooting, Open Issues and Known Bugs</td>
</tr>
<tr>
<td>13</td>
<td>MSC NASTRAN</td>
</tr>
<tr>
<td>13.1</td>
<td>Quick Information</td>
</tr>
<tr>
<td>13.1.1</td>
<td>Supported Coupling Methods</td>
</tr>
<tr>
<td>13.1.2</td>
<td>Supported Platforms and Versions</td>
</tr>
<tr>
<td>13.1.3</td>
<td>Adapter Description</td>
</tr>
<tr>
<td>13.1.4</td>
<td>Prerequisites for a Coupled Simulation</td>
</tr>
<tr>
<td>13.2</td>
<td>Coupling Process</td>
</tr>
<tr>
<td>13.2.1</td>
<td>Model Preparation</td>
</tr>
<tr>
<td>13.2.2</td>
<td>Models Step</td>
</tr>
<tr>
<td>13.2.3</td>
<td>Algorithm Step</td>
</tr>
<tr>
<td>13.2.4</td>
<td>Regions Step</td>
</tr>
<tr>
<td>13.2.5</td>
<td>Go Step</td>
</tr>
<tr>
<td>13.2.6</td>
<td>Running the Computation</td>
</tr>
<tr>
<td>13.2.7</td>
<td>Post-Processing</td>
</tr>
<tr>
<td>13.3</td>
<td>Code-Specific MpCCI Commands</td>
</tr>
<tr>
<td>13.4</td>
<td>Code Adapter Reference</td>
</tr>
<tr>
<td>13.5</td>
<td>Trouble Shooting, Open Issues and Known Bugs</td>
</tr>
</tbody>
</table>

8 VI MpCCI 4.6.1-1
14 **OpenFOAM** .................................................. 140
  14.1 Quick Information ........................................ 140
    14.1.1 Supported Coupling Methods .......................... 140
    14.1.2 Supported Platforms and Versions ..................... 140
    14.1.3 References ............................................ 140
    14.1.4 Adapter Description .................................... 140
    14.1.5 Prerequisites for a Coupled Simulation ................. 141
  14.2 Coupling Process ........................................ 141
    14.2.1 Model Preparation ..................................... 141
    14.2.2 Models Step ............................................ 143
    14.2.3 Algorithm Step ......................................... 144
    14.2.4 Regions Step ........................................... 144
    14.2.5 Go Step ................................................. 146
    14.2.6 Running the Computation ................................ 147
    14.2.7 Post-Processing ......................................... 149
  14.3 Grid Morphing ............................................. 149
    14.3.1 MpCCI Grid Morpher ..................................... 149
    14.3.2 OpenFOAM Grid Morpher ................................ 149
  14.4 Code-Specific MpCCI Commands ............................. 152
  14.5 Code Adapter Reference ................................... 154

15 **SIMPACK** .................................................. 156
  15.1 Quick Information ........................................ 156
    15.1.1 Supported Coupling Methods .......................... 156
    15.1.2 Supported Platforms and Versions ..................... 156
    15.1.3 References ............................................ 156
    15.1.4 Adapter Description .................................... 156
    15.1.5 Prerequisites for a Coupled Simulation ................. 156
  15.2 Coupling Process ........................................ 156
    15.2.1 Model Preparation ..................................... 157
    15.2.2 Simulation ............................................. 157
    15.2.3 Models Step ............................................. 157
    15.2.4 Algorithm Step ......................................... 158
    15.2.5 Regions Step ........................................... 159
    15.2.6 Go Step ................................................. 159
    15.2.7 Running the Computation ................................ 160
    15.2.8 Post-Processing ......................................... 160
  15.3 Code-Specific MpCCI Commands ............................. 161
1 Overview

This codes manual contains simulation code-specific information.

There is one chapter for each code, which contains basic information on the code itself and what you should know for preparing a model and running a co-simulation.

This Code Manual cannot replace the actual manuals of the simulation codes. References to further information in the code manuals are given as necessary.

The information given for each code is given in the same structure:

- **Quick Information** – Basic properties of the simulation code and requirements for installation, licensing and further software needed by the simulation code.
- **Coupling Process** – How to prepare and run a model for co-simulation, including code-specific options in the MpCCI GUI.
- **Code-Specific MpCCI Commands** – Subcommands `mpcci <code name>` which are needed to get information on the simulation code and prepare models for co-simulation. A number of subcommands is available for several codes, these are described in > 1.1 Common MpCCI Subcommands for Simulation Codes <.
- **Code Adapter Reference** – Short description of the code adapter.
- **Trouble Shooting, Open Issues and Known Bugs** – List of known issues.
- **Frequently Asked Questions** – List of questions from support and answers.
1.1 Common MpCCI Subcommands for Simulation Codes

\texttt{mpcci <codename> \ -releases}

The \texttt{releases} subcommand prints a short list of all available releases of a simulation code.

\texttt{mpcci <codename> \ -info}

The \texttt{info} subcommand prints a more detailed list of the available releases of a simulation code and on the corresponding code adapter. The output looks as follows:

```
> mpcci simulationcode info

SIMULATIONCODE release "3.1":
  HOME: "/opt/simulationcode/sim-3.1"
  EXEC: "/opt/simulationcode/sim-3.1/bin/start-code"
  ARCH: "lnx86-64"

  MpCCI adapter release "3.1": ** NOT INSTALLED **
  HOME: "/home/fritz/mpcci/codes/SIMULATIONCODE/adapters"
  PATH: "/home/fritz/mpcci/codes/SIMULATIONCODE/adapters/3.1/lnx86-64"

SIMULATIONCODE release "2.99":
  HOME: "/opt/simulationcode/sim-2.99"
  EXEC: "/opt/simulationcode/sim-2.99/bin/start-code"
  ARCH: "lnx86-64"

  MpCCI adapter release "2.99":
  HOME: "/home/fritz/mpcci/codes/SIMULATIONCODE/adapters"
  PATH: "/home/fritz/mpcci/codes/SIMULATIONCODE/adapters/2.99/lnx86-64"

Latest SIMULATIONCODE release for MpCCI found: "2.99"
```

Two versions of the code are installed, but a code adapter is only available for version 2.99. Therefore the latest simulation code release which can be used with MpCCI is version 2.99. \texttt{HOME} is the main directory of a code installation or adapter, \texttt{EXEC} the code executable and \texttt{PATH} the path to the code adapter, which does not exist if an adapter is not available. The architecture token given under \texttt{ARCH} is the code’s architecture token.
**mpcci** `<codename>` -**align** `<ARGS>`

The **align** subcommand is used to perform a coordinate transformation. This can be necessary if the coordinate system of the model differs from that of the partner code. The usage is as follows:

**Usage:**
```
mpcci SIMULATIONCODE align <plane-definition-file> <input-file> <output-file>
```

**Synopsis:**
'**mpcci SIMULATIONCODE align**' is used to align/transform the nodal coordinates in the `<input-file>` so that they match with the coordinate system used by the coupling partner.

The transformed result is written into the `<output-file>` - which is identical to the `<input-file>` except the nodal point coordinates.

The homogeneous 4x4 transformation matrix resp. the reference coordinate systems used are indirectly defined via two planes. Each plane is defined via three non co-linear points p1, p2 and p3.

The six points that define the two planes are specified in the `<plane-definition-file>`

which contains 18 floating point values in the following order:

```
# SIMULATIONCODE source plane specified via 3 point coordinates
p1-x  p1-y  p1-z
p2-x  p2-y  p2-z
p3-x  p3-y  p3-z

# Target coupling partner code plane specified via 3 point coordinates
p1-x  p1-y  p1-z
p2-x  p2-y  p2-z
p3-x  p3-y  p3-z
```

The transformation matrix determined transforms the SIMULATIONCODE plane into the "target" plane (rotation and scaling, but without shear) so that:

```
SIMULATIONCODE(p1) -> coupling partner(p1)
SIMULATIONCODE(p2) -> coupling partner(p2)
SIMULATIONCODE(p3) -> coupling partner(p3)
```

To use 'mpcci SIMULATIONCODE align' please just select three characteristic points in your SIMULATIONCODE model and the corresponding points in the partner model and copy the x-y-z values into the `<plane-definition-file>` in the above order.

**mpcci** `<codename>` -**scan** `<model file>`

The **scan** subcommand starts the scanner for a model file, which writes basic information about the model into the scan file "**mpcci <model file> .scan**". After creating the file, its content is printed on the screen.

If the scan file "**mpcci <model file> .scan**" exists already, no new scanning process is started, instead only
the content is printed.

\texttt{mpcci <codename> -diff <model file 1> <model file 2>}

The \texttt{diff} subcommand runs the scanner on the given two model files and prints the differences.

\section*{1.2 Unit Systems}

Many simulation codes do not use a given system of units, but do not consider units at all. This has the advantage that users can select any consistent set of units, define all values in this system and will also receive the values in this system.

For coupling a unit-less code with codes which are based on a specific system of units, MpCCI must know which system of units was used to create the model.

MpCCI supports the following unit systems, for each system some units are listed. Of course there are more units in each system, please see what is selected in the MpCCI GUI.

<table>
<thead>
<tr>
<th>unit system</th>
<th>mass</th>
<th>length</th>
<th>time</th>
<th>forces</th>
<th>el. current</th>
<th>temperature</th>
</tr>
</thead>
<tbody>
<tr>
<td>SI</td>
<td>kg</td>
<td>m</td>
<td>s</td>
<td>N</td>
<td>A</td>
<td>K</td>
</tr>
<tr>
<td>British*</td>
<td>lbm</td>
<td>ft</td>
<td>s</td>
<td>lbf</td>
<td>A</td>
<td>K</td>
</tr>
<tr>
<td>cgs</td>
<td>g</td>
<td>cm</td>
<td>s</td>
<td>dyn</td>
<td>Bi</td>
<td>K</td>
</tr>
<tr>
<td>mm-t-s</td>
<td>t</td>
<td>mm</td>
<td>s</td>
<td>N</td>
<td>A</td>
<td>K</td>
</tr>
<tr>
<td>US-ft-lbf-s</td>
<td>slug</td>
<td>ft</td>
<td>s</td>
<td>lbf</td>
<td>A</td>
<td>K</td>
</tr>
<tr>
<td>US-in-lbf-s</td>
<td>lbf s$^2$/in</td>
<td>in</td>
<td>s</td>
<td>lbf</td>
<td>A</td>
<td>K</td>
</tr>
</tbody>
</table>

Alternatively, you can also select the option \texttt{variable}, which allows you to select units individually for each quantity. However, for each quantity only a limited set of units is offered! If the unit system is set to \texttt{variable} an additional parameter will be shown where you can select the \texttt{grid length unit}, which corresponds to the length unit used in your model.

\footnote{*: The “British” unit system as defined in the MpCCI GUI is not a consistent unit system – some of the electromagnetic units are inconsistent! Use at own risk.}
2 Abaqus

2.1 Quick Information

<table>
<thead>
<tr>
<th>Company name</th>
<th>SIMULIA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company homepage</td>
<td><a href="http://www.3ds.com/products/simulia/overview">www.3ds.com/products/simulia/overview</a></td>
</tr>
<tr>
<td>Support</td>
<td><a href="http://www.3ds.com/products-services/simulia/support/support-overview/">www.3ds.com/products-services/simulia/support/support-overview/</a></td>
</tr>
<tr>
<td>Tutorials</td>
<td>▶ VII-2 Vortex-Induced Vibration of a Thin-Walled Structure ◄</td>
</tr>
<tr>
<td></td>
<td>▶ VII-3 Driven Cavity ◄</td>
</tr>
<tr>
<td></td>
<td>▶ VII-4 Elastic Flap in a Duct ◄</td>
</tr>
<tr>
<td></td>
<td>▶ VII-5 Exhaust Manifold ◄</td>
</tr>
<tr>
<td></td>
<td>▶ VII-8 Pipe Nozzle ◄</td>
</tr>
</tbody>
</table>

2.1.1 Supported Coupling Methods

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Dimension</th>
<th>Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implicit</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The MpCCI Coupling dimension corresponds to the Abaqus interface type.

2.1.2 Supported Platforms and Versions

The following versions of Abaqus are supported by MpCCI (only the main version is listed):

<table>
<thead>
<tr>
<th>MPCCI_ARCH: Code platform</th>
<th>Supported versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>win86-64:</td>
<td>X   X   X    X    X    X</td>
</tr>
</tbody>
</table>

Please look at the important hints in ▶ 2.1.4 Adapter Description ◄ and ▶ 2.1.5 Prerequisites for a Coupled Simulation ◄ and see also the System Information section of the Support page at www.simulia.com for details on the platforms supported by Abaqus.

2.1.3 References

Abaqus Documentation The Abaqus documentation is part of your Abaqus installation. Read especially the section “Co-simulation” of the Abaqus Analysis User’s Manual (Analysis Techniques).

Abaqus Fluid-Structure Interaction User’s Guide This guide is available via the Abaqus support homepage: Log into Abaqus SIMULIA knowledge base and search for “FSI guide”.

Besides general information, the FSI guide contains several examples of coupled simulations with Abaqus and FLUENT.
2.1.4 Adapter Description

The code adapter for Abaqus is developed by SIMULIA in cooperation with Fraunhofer SCAI. The adapter is distributed as part of the Abaqus software.

For transient simulations with implicit coupling scheme, if constant time-stepping is chosen but the end time is not dividable by the time step size, Abaqus will reduce the time step size for the last step automatically so that the end time is reached. This might lead to numerical inaccuracies for the last coupled time step.

2.1.5 Prerequisites for a Coupled Simulation

To run a coupled simulation you need the following:

- Ordinary Abaqus installation.
- Enough license tokens.
- Co-simulation based on the SIMULIA’s Co-Simulation Engine (CSE) requires the token “multiphysics”.

2.2 Coupling Process

Please read also ▷ IV-2 Setting up a Coupled Simulation ◁.

2.2.1 Model Preparation

The Abaqus model can be prepared with Abaqus/CAE or as an input file. Please consider the following advice:

- The model can be defined in any of the consistent systems of units supported by MpCCI (see ▷ 1.2 Unit Systems ◁). The basic unit system must be given in the Models step of the MpCCI GUI. Units of single quantities can be set in the Regions step.

- Most Abaqus features can be used in co-simulations. Please see the Abaqus documentation or contact the Abaqus support for further information.

- The Abaqus model must contain a definition of coupling components.
  - This can be either an element-based surface for surface coupling:
    
    Abaqus/CAE: Create a surface using the Surfaces tool. See also “13.7.6 Using sets and surfaces in the Assembly module” in the Abaqus/CAE User’s Manual. You can also use surfaces defined in the Part module.
    
    Input file: A surface is created with *SURFACE, NAME=<surface name>, TYPE=ELEMENT, see section “2.3.3 Defining element-based surfaces” of the Abaqus Analysis User’s Manual.
    
    The surface which serves as coupling component can be selected in the Models step of theMpCCI GUI.
  
  - This can be either a discrete point for point coupling:
    
    Abaqus/CAE: Create a node set using the Set tool. Select only one node for this set. You can also use set defined in the Part module.
    
    Input file: A node set is created with *NSET, NSET=<point name>, see section “2.1.1 Node definition” of the Abaqus Analysis User’s Manual.
• Code coupling is only run in one step of a simulation (for coupling in several steps, please use a restart). The co-simulation step is selected in the Go step of the MpCCI GUI.

It is recommended to create a complete Abaqus model first and test it separately without co-simulation. The quantities which will be transferred by the partner code can be simulated by appropriate loads or boundary conditions.

In steady thermal couplings which use relaxation, define the heat transfer step as *STEP, AMPLITUDE=STEP. Otherwise, Abaqus will ramp the thermal load over time, which is not necessarily compatible with the coupling relaxation.

2.2.2 Restart

For preparing a restart computation the output for restart must be activated for example:

*Restart, write, frequency=5

To restart a co-simulation see 2.5 Co-Simulation Restart.

2.2.3 Models Step

In the Models step, the following options must be chosen:

**Abaqus release** Select the Abaqus release you wish to use. Only supported releases installed on your system are listed. The selection latest always refers to the latest supported version (default). The release should match the input file. To select a release on a remote machine, please select the remote input file first.

**Scan method** This can be set to:

- **Scan for all regions** (default) – The input file is scanned for possible coupling regions.
- **Scan *CO-SIMULATION option only** – This setting is only needed for a restart. In this case the input file only contains the definition of the coupling regions in the *CO-SIMULATION section and no ordinary region definitions.

**Abaqus input file** Select the Abaqus input file "*.inp".

**Unit system** Select the unit system which was used in Abaqus. Abaqus has no units built into it, a self-consistent set of units should be used (see chapter “1.2.2 Conventions” of the Abaqus Analysis User’s Manual for more information). In the Models step you can select from any unit system listed in 1.2 Unit Systems.
2.2.4 Algorithm Step

For general options please refer to V-4.5.2 Code Specific Algorithm Settings. Following options are available in the code specific section in Solver settings.

**Use subcycling** Check this button if you want allow Abaqus to subcycle, i.e. use a smaller time step size than the coupling time step size. The basic principle of subcycling is described in V-3.4.3 Coupling with Subcycling.

**Enforce exact target times** If Abaqus performs some subcycling, you can choose whether Abaqus has to meet the coupling target times in an exact or loose manner by toggling this Enforce exact target times option.

2.2.5 Regions Step

Abaqus supports the following quantities for coupling:

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Dim.</th>
<th>Default Value</th>
<th>Integration Type</th>
<th>Coupling Dimension</th>
<th>Location</th>
<th>Send Option</th>
<th>Receive Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbsPressure</td>
<td>Scalar</td>
<td>0.0 N/m²</td>
<td>flux dens.</td>
<td>Face</td>
<td>Code</td>
<td></td>
<td>Buffer</td>
</tr>
<tr>
<td>Acceleration</td>
<td>Vector</td>
<td>0.0 m/s²</td>
<td>field</td>
<td>Point</td>
<td>Code</td>
<td></td>
<td>Buffer</td>
</tr>
<tr>
<td>AngularAccel</td>
<td>Vector</td>
<td>0.0 rad/s²</td>
<td>field</td>
<td>Point</td>
<td>Code</td>
<td></td>
<td>Buffer</td>
</tr>
<tr>
<td>AngularCoord</td>
<td>Quaternion</td>
<td>0.0</td>
<td>mesh coordinate</td>
<td>Point</td>
<td>Code</td>
<td></td>
<td>Buffer</td>
</tr>
<tr>
<td>AngularVel</td>
<td>Vector</td>
<td>0.0 rad/s</td>
<td>field</td>
<td>Point</td>
<td>Code</td>
<td></td>
<td>Buffer</td>
</tr>
<tr>
<td>BodyForce</td>
<td>Vector</td>
<td>0.0 N/m³</td>
<td>flux dens.</td>
<td>Volume</td>
<td>Code</td>
<td></td>
<td>Buffer</td>
</tr>
<tr>
<td>DeltaTime</td>
<td>Scalar</td>
<td>1.0 s</td>
<td>g-min</td>
<td>Global</td>
<td>global</td>
<td></td>
<td>Buffer</td>
</tr>
<tr>
<td>FilmTemp</td>
<td>Scalar</td>
<td>300.0 K</td>
<td>field</td>
<td>Face</td>
<td>Code</td>
<td></td>
<td>Buffer</td>
</tr>
<tr>
<td>Force</td>
<td>Vector</td>
<td>0.0 N</td>
<td>flux integral</td>
<td>Point, Face</td>
<td>Code</td>
<td></td>
<td>Direct Buffer</td>
</tr>
<tr>
<td>HeatRate</td>
<td>Scalar</td>
<td>0.0 W</td>
<td>flux dens.</td>
<td>Face</td>
<td>Code</td>
<td></td>
<td>Buffer</td>
</tr>
<tr>
<td>NPosition</td>
<td>Vector</td>
<td>0.0 m</td>
<td>mesh coordinate</td>
<td>Face, Volume</td>
<td>Code</td>
<td></td>
<td>Buffer</td>
</tr>
<tr>
<td>OverPressure</td>
<td>Scalar</td>
<td>0.0 N/m²</td>
<td>flux dens.</td>
<td>Face</td>
<td>Code</td>
<td></td>
<td>Buffer</td>
</tr>
<tr>
<td>PointPosition</td>
<td>Vector</td>
<td>0.0 m</td>
<td>mesh coordinate</td>
<td>Point</td>
<td>Code</td>
<td></td>
<td>Buffer</td>
</tr>
<tr>
<td>PorePressure</td>
<td>Scalar</td>
<td>0.0 N/m²</td>
<td>flux dens.</td>
<td>Face, Volume</td>
<td>Code</td>
<td></td>
<td>Buffer</td>
</tr>
<tr>
<td>PorousFlow</td>
<td>Scalar</td>
<td>0.0 m/s</td>
<td>flux dens.</td>
<td>Face, Volume</td>
<td>Code</td>
<td></td>
<td>Buffer</td>
</tr>
<tr>
<td>RelWallForce</td>
<td>Vector</td>
<td>0.0 N</td>
<td>flux integral</td>
<td>Face</td>
<td>Code</td>
<td></td>
<td>Buffer</td>
</tr>
<tr>
<td>Temperature</td>
<td>Scalar</td>
<td>300.0 K</td>
<td>field</td>
<td>Volume</td>
<td>Code</td>
<td></td>
<td>Buffer</td>
</tr>
<tr>
<td>Torque</td>
<td>Vector</td>
<td>0.0 N</td>
<td>flux integral</td>
<td>Point</td>
<td>Code</td>
<td></td>
<td>Buffer</td>
</tr>
<tr>
<td>Velocity</td>
<td>Vector</td>
<td>0.0 m/s</td>
<td>field</td>
<td>Point, Face, Volume</td>
<td>Code</td>
<td></td>
<td>Buffer</td>
</tr>
<tr>
<td>WallForce</td>
<td>Vector</td>
<td>0.0 N</td>
<td>flux integral</td>
<td>Face</td>
<td>Code</td>
<td></td>
<td>Buffer</td>
</tr>
<tr>
<td>WallHeatFlux</td>
<td>Scalar</td>
<td>0.0 W/m²</td>
<td>flux dens.</td>
<td>Face</td>
<td>Code</td>
<td></td>
<td>Buffer</td>
</tr>
<tr>
<td>WallHTCoeff</td>
<td>Scalar</td>
<td>0.0 W/m² K</td>
<td>field</td>
<td>Face</td>
<td>Code</td>
<td></td>
<td>Buffer</td>
</tr>
<tr>
<td>WallTemp</td>
<td>Scalar</td>
<td>300.0 K</td>
<td>field</td>
<td>Face</td>
<td>Code</td>
<td></td>
<td>Buffer</td>
</tr>
</tbody>
</table>
Following changes for thermal coupling to consider:
The Steady state radiative heat transfer coupling type cannot be applied with Abaqus 6.14 based on
the SIMULIA’s Co-Simulation Engine (CSE). The previous coupling type based on the exchange of
temperature (WallTemp), film temperature (FilmTemp) and the heat transfer coefficient (WallHTCoeff)
should be replaced by the exchange of temperature (WallTemp), film temperature (FilmTemp) and the
heat rate (HeatRate). Abaqus applies the heat rate and film temperature as concentrated heat flux and
ambient temperature. The heat transfer coefficient is computed on the structural side.

2.2.6 Go Step

In the Go step, the following options can be selected:

Enter a job name This is the name of the Abaqus job, which is also used as base for the Abaqus output
files.

Enter the co-simulation step number The co-simulation step is the step of the Abaqus simulation which
is coupled. You can e.g. run some initial computations first, followed by a coupled step based on the
previous results.

Additional command line options Additional command line options for Abaqus can be given here, they
will directly be used when Abaqus is started.

Optional old job name for a restart This option is required for a restart computation. It is handed over
to Abaqus as oldjob=<name> option.

Recovery restart time (Option is available if aodb file file has been selected)
This option is required for a recovery restart computation. The user should provide the time where
the restart point will be initiated by the Abaqus computation. The user may analyze the ".msg"
files from the old job to figure out at which time the restart data has been written. You can search
for the following key to find this information: RESTART INFORMATION WRITTEN IN STEP....

User Subroutine is the name of a FORTRAN file (source or object file), which is handed to Abaqus as

Figure 2: Abaqus solver specific options for solvers Standard and Explicit

Abaqus solver used shows the Abaqus solver used for this computation. The solver type is retrieved from
the scanner run.

Double precision Select this option to use the double precision. Only Explicit solver type provides this
option (cf. Figure 2 Explicit).

No. of processors Enter the number of threads for an Abaqus run. This starts Abaqus with the option
-mp_mode=threads.
Run parallel using MPI  Select this to start a parallel run distributed on several hosts. This starts Abaqus with the option `-mp_mode=mpi`. A panel with additional options appears, which are further described in 2.2.7.1 Parallel Execution.

Figure 3: Abaqus CSE parameters in the Go step

Use optional CSE parameters  Activate this option in order to access some additional parameters. It is recommended to provide different initial port numbers for an Abaqus-Abaqus co-simulation.

CSE Director Port number  Enter the initial port number for the CSE director process. 48000 is the default value if the Optional CSE parameters is not activated.

Optional configuration file  Select a configuration file when doing a co-simulation using CSE. It is recommended to use a configuration file generated by MpCCI and rename it for your purpose. This step is highly recommended in order to keep the conventional name for the co-simulation region interface. You may be able to add or activate some features from CSE that are not already automated in MpCCI GUI.

Optional CSE library path  Define the PATH to the CSE library installation. This is only recommended if you get the information UNDEFINED CSE LIBRARY PATH from the command `mpcci abaqus info`.

2.2.7 Running the Computation

When the Start button is pressed, Abaqus is started with the options given in the Go step. The setting for the coupling time step will be checked:

One of the following option is required in order to start Abaqus:

- Exchange the global quantity DeltaTime, or
- Use a constant coupling time step.
- If none of these information has been activated Abaqus will use its own time stepping to perform a co-simulation step.

If the Stop button is pressed, a stop-file is created and Abaqus will stop the next time it checks the presence of a stop file.

You can check during the Abaqus run for the Abaqus log files:

- "*.dat" log file from the Abaqus Pre process which analyzes the input deck.
- "*.msg" log message file from the Abaqus standard or explicit solver.

In case of any issue with Abaqus, please first check the Abaqus log file for further information.

2.2.7.1 Parallel Execution

Parallel runs of Abaqus are supported by MpCCI.

In the Go step, a set of additional options can be chosen for a parallel run as shown in Figure 4.
2.2 Coupling Process

No. of processors Enter the number of processors to use during the analysis. The corresponding Abaqus command line option is -cpus.

No. of parallel domains Enter the number of parallel domains. The corresponding Abaqus command line option is -domains. As parallel Abaqus is always run with the command line option -parallel=domain (-parallel=loop is not available for coupled runs), the value for No. of processors must be a divisor of the value of No. of parallel domains. (Only for Explicit solver).

Optional 'host host...' to be used Enter host names for parallel execution of Abaqus.

Optional hostlist file Specify a hostfile, from which host names are extracted ➔ V-3.5.2 Hostlist File ◄.

Use default hostfile A default hostfile can be configured by setting MPCCI_HOSTLIST_FILE ➔ V-3.5.2 Hostlist File ◄.

Please read the Abaqus documentation for a detailed explanation of the possible options.

2.2.7.2 Batch Execution

Abaqus is always run as a batch process, therefore no special settings are necessary for batch execution.

2.2.8 Post-Processing

Post-processing for the Abaqus part of the results can be performed as in ordinary computations, e.g. with Abaqus/CAE. The "<job name>.odb" file can be found in the same directory as the input file.
2.3 Code-Specific MpCCI Commands

The MpCCI subcommands available for Abaqus are:

Usage:
```
mpcci Abaqus [-]option
```

Synopsis:
'mpcci Abaqus' is used to get information about Abaqus.

Options:
- **align <ARGS>**
  Do a coordinate transformation on all nodes of an input file based on a plane definition file and align the nodal coordinates for the coupling partner.

- **diff <inp1> <inp2>**
  Run the scanner on two .inp files and print the differences.

- **help**
  This screen.

- **info**
  List verbose information about all Abaqus releases.

- **releases**
  List all Abaqus releases which MpCCI can find.

- **restartPoints <oldjob> [RELEASE]**
  Check which restart points are written to an old job by running an Abaqus datacheck using a dummy project.

- **scan <input-file>**
  Run the scanner and create a scanner output file.

- **solver <input-file>**
  Get the solver 'standard' or 'explicit' from an input deck.

The subcommands **align, diff, info, releases** and **scan** are described in §1.1 Common MpCCI Subcommands for Simulation Codes <.

**mpcci abaqus restartPoints <oldjob> [RELEASE]**
The **restartPoints** subcommand launches an Abaqus datacheck for a dummy restart from an old job. Restart points are printed e.g. STEP 2 INCREMENT 44 HAS BEEN FOUND ON THE RESTART FILE.

**mpcci abaqus -solver <input-file>**
The **solver** subcommand gets the used solver (standard or explicit) from the specified input deck and prints it to stdout.
2.4 Code Adapter Reference

2.4.1 Patched Input File

Before a coupled simulation is started, MpCCI patches the Abaqus input file. For a file "*.inp" selected in the Models step, a new file "mpcci_*.inp" is created containing a new *CO-SIMULATION block. If the original file already contains *CO-SIMULATION keywords, they are removed.

The lines inserted by MpCCI may e.g. look as follows:

```plaintext
**
** MpCCI automatically inserted this *CO-SIMULATION keyword.
** Therefore any existing *CO-SIMULATION keyword was removed.
**
*CO-SIMULATION, NAME=MPCCI, PROGRAM=MULTIPHYSICS
*CO-SIMULATION REGION, TYPE=SURFACE, EXPORT ASSEMBLY_BLOCK-1_WALL, COORD
*CO-SIMULATION REGION, TYPE=SURFACE, IMPORT ASSEMBLY_BLOCK-1_WALL, CF
```

The settings in this block reflect the user’s choices in the MpCCI GUI:

- The selected coupling components are listed below the *CO-SIMULATION REGION] keywords. Here it is ASSEMBLY_BLOCK-1_WALL and the exchanged quantities are COORD = NPosition, sender is Abaqus
  CF = RelWallForce, sender is partner code

The *CO-SIMULATION] keywords are described in detail in “Preparing an Abaqus analysis for co-simulation” of the Abaqus Analysis User’s Manual.

2.4.2 SIMULIA’s Co-Simulation Engine (CSE)

Before a coupled simulation with CSE is started, MpCCI generates automatically the proper SIMULIA CSE configuration file for the calculation.

2.5 Co-Simulation Restart

To prepare the input file for restarting the co-simulation follow these steps:

- Create a new input file named for instance "abaqus_run_restart.inp".
- Introduce the Abaqus key *RESTART with the respective option, specifying the time of increment to use.
  The defined time for the restart should also match the time of the partner code. Otherwise the grid position may not match.
- Define your analysis *Step.
- Before closing the Step definition, copy the lines introduced by MpCCI in the input file for the co-simulation, for example:

```plaintext
**
** MpCCI automatically inserted this *CO-SIMULATION keyword.
** Therefore any existing *CO-SIMULATION keyword was removed.
**
```
• Close the Step definition and save the input file.
• In the MpCCI GUI you should use the option Scan *CO-SIMULATION option only for scanning the model you just created.
• In the Go step you should select your odb file for the restart and set the co-simulation step number to 2, because the restart is considered as the step 1.
• If the restart is a recovery calculation, you should provide the time where the restart point of the calculation will begin.

2.6 Known Limitations

• Prescribing a point displacement with Abaqus/Explicit 6.12 and 6.13 does not deliver the correct reaction force. The point coupling can be realized with Abaqus user subroutines VUAMP combined with definition of sensors in the input deck. This allows to obtain the proper reaction forces. Contact the MpCCI support for detailed information about the procedure.
• Axisymmetric co-simulation surface is not supported by SIMULIA Co-Simulation Engine. The workaround consists of formulating your problem in a 3D cyclic symmetric model.
2.7 Trouble Shooting, Open Issues and Known Bugs

Version:
Abaqus 2018, 2019 with MpCCI 4.6

Problem:
For a thermal co-simulation with filmTemp, WallHTCoeff you receive during the initialization with the MpCCI the message:

MpCCI-CSE: Create field quantity definition "filmtemp" at element.
MpCCI-CSE: Registering outgoing quantity "filmtemp".
MpCCI-CSE: Create field quantity definition "wallhtcoeff" at element.
MpCCI-CSE: Registering outgoing quantity "wallhtcoeff".
MpCCI-CSE: Create field quantity definition "walltemp" at node.
MpCCI-CSE: Registering incoming quantity "walltemp".
MpCCI-CSE: Director terminated the analysis due to early termination by a client. Please check for errors for each of the analysis involved in co-simulation and also verify that the co-simulation duration time specified in the configuration file is correct.

From the Abaqus ".msg" file you can find this error message:

Variable surface_filmprop referenced in the co-simulation configuration file is not declared as incoming variable in this model.

Solution:
Update the Abaqus 2018 or 2019 with the latest hotfix available from SIMULIA: https://software.3ds.com/
3 ANSYS

3.1 Quick Information

<table>
<thead>
<tr>
<th>Company name</th>
<th>ANSYS, Inc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company homepage</td>
<td><a href="http://www.ansys.com">www.ansys.com</a></td>
</tr>
<tr>
<td>Support</td>
<td>ANSYS Customer portal at support.ansys.com</td>
</tr>
</tbody>
</table>
| Tutorials       | ◐ VII-2 Vortex-Induced Vibration of a Thin-Walled Structure ◐
|                 | ◐ VII-4 Elastic Flap in a Duct ◐
|                 | ◐ VII-5 Exhaust Manifold ◐
|                 | ◐ VII-6 Busbar System ◐
|                 | ◐ VII-8 Pipe Nozzle ◐ |

3.1.1 Supported Coupling Methods

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Dimension</th>
<th>Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit</td>
<td>Implicit</td>
<td>FEM, SolidStructure, SolidThermal, SolidAcoustics, ElectroMagnetism</td>
</tr>
</tbody>
</table>

3.1.2 Supported Platforms and Versions

The following versions of ANSYS are supported by MpCCI:

<table>
<thead>
<tr>
<th>Code platform</th>
<th>Supported versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>lnx4_x64</td>
<td>180 161 162 170 171 172 180 181 182 180 191 192 193 194 195 201 202 211</td>
</tr>
<tr>
<td>lnx64</td>
<td>X X X X X X X X X X X X X X X X X X</td>
</tr>
<tr>
<td>windows_x64</td>
<td>X X X X X X X X X X X X X X X X X X</td>
</tr>
<tr>
<td>winx64</td>
<td>X X X X X X X X X X X X X X X X X X</td>
</tr>
</tbody>
</table>

3.1.3 References

The ANSYS installation contains a detailed documentation. We especially refer to:

ANSYS APDL Programmer’s Guide. The ANSYS Parametric Design Language (APDL) is needed to run a co-simulation.

3.1.4 Adapter Description

The ANSYS adapter is a shared library which is loaded by ANSYS. The adapter functions must be called from an APDL script.
3.1.5 Prerequisites for a Coupled Simulation

To run a coupled simulation you need the following:

- Ordinary ANSYS installation.
- Know the list of available ANSYS products you may use (see section 3.1.6). This is required to start the ANSYS code.
- Set the ANSYS license environment variables properly for a ANSYS standalone computation. In order to let MpCCI figuring out the available ANSYS product you may use MpCCI is inspecting the following environment variable: ANSYSLMD_LICENSE_FILE. This environment variable should be set to the FLEXlm license server with a similar information value: portnumber@server. This may be checked by executing the command `mpcci ansys product` which lists the available product installed at your site. If the command fails because of some license connection issue, you can set the following environment variable MPCCI_ANSYS_PRODUCT_LIST containing the list of products separated by colon, semicolon or space. For example:
  ```bash
  export MPCCI_ANSYS_PRODUCT_LIST=ansys:ane3fl:strucds
  ```
  It is required that ANSYS can run in standalone correctly and can get a license without any issue.

3.1.6 Supported ANSYS Product Variable

The following table shows all ANSYS products and their associated feature name as used in the ANSYS license file INCREMENT lines supported by MpCCI.
<table>
<thead>
<tr>
<th>Product</th>
<th>Feature Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS Multiphysics/LS-DYNA</td>
<td>ane3flds</td>
</tr>
<tr>
<td>ANSYS Multiphysics 1</td>
<td>ane3f11</td>
</tr>
<tr>
<td>ANSYS Multiphysics 2</td>
<td>ane3f12</td>
</tr>
<tr>
<td>ANSYS Multiphysics 3</td>
<td>ane3f13</td>
</tr>
<tr>
<td>ANSYS Multiphysics/LS-DYNA PrepPost</td>
<td>ane3fldp</td>
</tr>
<tr>
<td>ANSYS Mechanical</td>
<td>ansys, ansysul, ansysuh, ansysrf</td>
</tr>
<tr>
<td>ANSYS Mechanical/Emag</td>
<td>ane3</td>
</tr>
<tr>
<td>ANSYS Mechanical/FLOTRAN</td>
<td>anfl</td>
</tr>
<tr>
<td>ANSYS Mechanical/LS-DYNA</td>
<td>ansysds</td>
</tr>
<tr>
<td>ANSYS Mechanical/Emag/LS-DYNA</td>
<td>ane3ds</td>
</tr>
<tr>
<td>ANSYS Mechanical/CFD-Flo/LS-DYNA</td>
<td>anflds</td>
</tr>
<tr>
<td>ANSYS Mechanical/LS-DYNA PrepPost</td>
<td>ansysdp</td>
</tr>
<tr>
<td>ANSYS Mechanical/Emag/LS-DYNA PrepPost</td>
<td>ane3dp</td>
</tr>
<tr>
<td>ANSYS Mechanical/CFD-Flo/LS-DYNA PrepPost</td>
<td>anfldp</td>
</tr>
<tr>
<td>ANSYS Structural</td>
<td>struct, struct1, struct2, struct3</td>
</tr>
<tr>
<td>ANSYS Structural/Emag</td>
<td>ste3</td>
</tr>
<tr>
<td>ANSYS Structural/FLOTRAN</td>
<td>stfl</td>
</tr>
<tr>
<td>ANSYS Structural/Emag/CFD-Flo</td>
<td>ste3f1</td>
</tr>
<tr>
<td>ANSYS Structural/LS-DYNA</td>
<td>structds</td>
</tr>
<tr>
<td>ANSYS Structural/Emag/LS-DYNA</td>
<td>ste3ds</td>
</tr>
<tr>
<td>ANSYS Structural/CFD-Flo/LS-DYNA</td>
<td>stflds</td>
</tr>
<tr>
<td>ANSYS Structural/LS-DYNA PrepPost</td>
<td>structdp</td>
</tr>
<tr>
<td>ANSYS Structural/Emag/LS-DYNA PrepPost</td>
<td>ste3dp</td>
</tr>
<tr>
<td>ANSYS Structural/CFD-Flo/LS-DYNA PrepPost</td>
<td>stfldp</td>
</tr>
<tr>
<td>ANSYS Structural/Emag/CFD-Flo/LS-DYNA PrepPost</td>
<td>ste3fldp</td>
</tr>
<tr>
<td>ANSYS Professional NLT</td>
<td>prf</td>
</tr>
<tr>
<td>ANSYS Professional/Emag</td>
<td>prfe3</td>
</tr>
<tr>
<td>ANSYS Professional/FLOTRAN</td>
<td>prffl</td>
</tr>
<tr>
<td>ANSYS Professional/Emag/FLOTRAN</td>
<td>prfe3f1</td>
</tr>
<tr>
<td>ANSYS Emag</td>
<td>emag</td>
</tr>
<tr>
<td>ANSYS Emag/FLOTRAN</td>
<td>emagfl</td>
</tr>
<tr>
<td>ANSYS Mechanical PrepPost</td>
<td>prepost</td>
</tr>
<tr>
<td>ANSYS PrepPost/LS-DYNA PrepPost</td>
<td>prpostdy</td>
</tr>
<tr>
<td>ANSYS FLOTRAN</td>
<td>flowtran</td>
</tr>
<tr>
<td>ANSYS Academic Teaching Advanced</td>
<td>aa_t_a, aa_mcad, aa_ds</td>
</tr>
<tr>
<td>ANSYS Academic Research</td>
<td>aa_r, aa_mcad</td>
</tr>
</tbody>
</table>
3.2 Coupling Process

Please read also ▶ IV-2 Setting up a Coupled Simulation ◄.

3.2.1 Model Preparation

There are some issues which you should consider while creating an ANSYS model for a co-simulation:

**Supported element types.** Not all ANSYS element types are supported, the supported types are given in Table 1.

**Dummy surface elements for surface coupling.** For surface coupling, additional surface elements must be used for quantity exchange. For instance SHELL63 elements for a fluid structure interaction or SHELL57 elements for thermal coupling can be used as dummy elements to receive forces from the partner code. The dummy elements must have the corresponding quantities you want to receive or send. Only these elements take part in the coupling process and must be either be deselected when the solution is performed or defined so weak that they do not influence the solution in case of fluid structure interaction. They can be deselected before solution is executed if only nodal quantities are sent or received, because the data transfer will then put the values to the nodes and the nodes of dummy elements are shared by the “real” solid model elements. In case of thermal coupling the dummy elements can not be deselected because the quantities wall heat transfer coefficient, wall temperature and wall heat flux are only supported as element quantities. Therefore the additional layer of shell elements (SHELL57) have to be a part of the solution. To reduce the influence on the solution you should give the shell elements the same material properties as the adjacent solid elements and a small thickness. In case of line coupling on area elements in fluid structure interaction BEAM3 elements can be used. The attached example cases for 2D and 3D fluid structure interaction contains such dummy elements. Such dummy elements are shown in Figure 1.

**Put elements for coupling in a component.** The elements of the coupling region must be grouped into one or more element components using the `cm` command.

**Predefined loads on dummy surface elements.** If element based quantities WallHTCoeff, FilmTemp, WallHeatFlux, AbsPressure or OverPressure are received by ANSYS using dummy surface elements, you have to predefine surface loads on these elements in order to enable the MpCCI ANSYS adapter to select the correct element sides to store the received values. Therefore select the coupled element set and the attached nodes and define a dummy load, depending on the received quantities:

- WallHTCoeff and FilmTemp: `~SF, ALL, CONV, 1, 300`
- WallHeatFlux: `~SF, ALL, HFLUX, 1`
- AbsPressure or OverPressure: `~SF, ALL, PRES, 1`

**Exchange of global quantities.** For each global quantity a corresponding scalar parameter must be defined. E.g. for the time step size a parameter named “DeltaTime” is needed. Please also assign a dummy value to the parameter. This parameter must be set to the correct value in the APDL script if it is received, or the parameter must be evaluated in the APDL script. Global quantities can thus be regarded as simple variables which can be filled with values by `~mpcci, receive` or whose values are read by `~mpcci, send`.

**Assign unique material property numbers.** Every element of your model must be assigned a unique material property number if you want to send material property values to ANSYS, e.g. the electrical resistivity. In ANSYS you define such numbers with the `mp` command.

**Possible quantities depend on degrees of freedom.** Normally the degree of freedom of the elements involved in the coupling process determines which quantities can be transferred. It is laborious to find out if all degrees of freedom are actually supported by the ANSYS API. As this API is used for the MpCCI ANSYS–adapter, it is not guaranteed that all theoretically supported degrees of freedom are valid.
Not carefully tested. Only few of the element type mappings are already validated with certain quantities. The compatibility index in Table 2 shows the validated element-quantity pairs. It is constantly added. Please contact us if you have problems with other combinations or need additional capabilities.

Binary database file required. You need to set up your model and store it in a binary database file (db file) because the coupled components will be extracted from this during the MpCCI scanning process.

APDL script required. For managing the co-simulation, an APDL script is required. This is described in detail in §3.2.2 APDL Script.

<table>
<thead>
<tr>
<th>ANSYS Element Types</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEAM3, BEAM4, BEAM23, BEAM24, BEAM44, BEAM54, BEAM188, BEAM189, SHELL51, SHELL61, SHELL208, SURF151, SURF153, SURF251 PIPE288, PIPE289, ELBOW290, MPC184,</td>
<td>MPCCI_ETYP_LINE2, MPCCI_ETYP_LINE3</td>
</tr>
<tr>
<td>PLANE13, PLANE25, PLANE42, PLANE55, PLANE67, PLANE181, PLANE182, SHELL28, SHELL41, SHELL43, SHELL57, SHELL63, SHELL131 SHELL143, SHELL157, HYPER56, VISC0106, CPT212, SURF252,</td>
<td>MPCCI_ETYP_TRIA3, MPCCI_ETYP_QUAD4</td>
</tr>
<tr>
<td>PLANE2, PLANE38</td>
<td>MPCCI_ETYP_TRIA3, MPCCI_ETYP_TRIA6</td>
</tr>
<tr>
<td>SHELL91, SHELL93, SHELL99, SHELL132, SHELL150, SHELL281 PLANE53, PLANE145, PLANE223, PLANE77, PLANE78, PLANE82, PLANE83, PLANE121, PLANE183, PLANE230 SURF152, SURF154 CPT213, HYPER74, VISC088, VISC0108,</td>
<td>MPCCI_ETYP_TRIA6, MPCCI_ETYP_QUAD4, MPCCI_ETYP_QUAD8</td>
</tr>
<tr>
<td>SOLID5, SOLID45, SOLID46, SOLID62, SOLID64, SOLID65, SOLID69, SOLID70, SOLID96, SOLID97, SOLID164, SOLID285, SOLID185, HYPER58, HYPER86, VISC0107, SOLSH190, CPT215</td>
<td>MPCCI_ETYP_TET4, MPCCI_ETYP_WEDGE6, MPCCI_ETYP_HEX8, MPCCI_ETYP_PYRAM5</td>
</tr>
<tr>
<td>SOLID90, SOLID95, SOLID122, SOLID117, SOLID122, SOLID128, SOLID147, SOLID186, SOLID191, SOLID226, SOLID231, SOLID236, VISC089, CPT216</td>
<td>MPCCI_ETYP_TET4, MPCCI_ETYP_TET10, MPCCI_ETYP_WEDGE6, MPCCI_ETYP_WEDGE15, MPCCI_ETYP_HEX8, MPCCI_ETYP_HEX20, MPCCI_ETYP_PYRAM5, MPCCI_ETYP_PYRAM13</td>
</tr>
<tr>
<td>MESH200</td>
<td>MPCCI_ETYP_LINE2, MPCCI_ETYP_TRIA3, MPCCI_ETYP_QUAD4, MPCCI_ETYP_QUAD8</td>
</tr>
</tbody>
</table>

Table 1: Supported ANSYS element types
### Table 2: Quantities supported for different ANSYS elements.

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Joule Heat Density</th>
<th>Lorentz Force Density</th>
<th>Electric Resistivity</th>
<th>Nodal Positions</th>
<th>Wall Forces</th>
<th>Relative Wall Forces</th>
<th>Film Temperature</th>
<th>Wall Heat-transfer Coefficient</th>
<th>Wall Temperature</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEAM3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOLID5</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLANE13</td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOLID45</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SHELL57</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHELL63</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SOLID69</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SHELL93</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SOLID97</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
<td>+</td>
<td>+</td>
<td></td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SOLID117</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SURF151</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+</td>
<td></td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>MESH200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLANE230</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOLID231</td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOLID236</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 1: ANSYS Flap and Additional SHELL63 elements for coupling**
3.2.2 APDL Script

In ANSYS, the coupling process is controlled via an APDL script, which calls functions provided by MpCCI.

3.2.2.1 Sample APDL Script

```
/batch
resume, ansys, db ! resume ANSYS database "ansys.db"
/com > > >
/com > > > set initial transfer tag (before MpCCI initialization!)
/com > > >
! time, iter
~mpcci, settag, -1, 0

~mpcci, init, 2D ! initialize MpCCI for 2D analysis
fini

/solu
antype, transient, new ! new transient analysis
trnopt, full ! full Newton-Rhapson

/com > > >
/com > > > initial data transfer controlled via 'Initial quantities transfer' setting:
/com > > > send / receive / exchange
\todo{update the above mentioned commands based on the new gui}

~mpcci, exchange ! exchange data

*do, i, 1, steps+1
  *if, i, ne, 1, then ! first run is dummy
    esel, all $ nsle ! make sure to have all of the model
    cmsel, u, TOP ! deselect the beam elements,
    cmsel, u, BOTTOM ! only used for transfer of values
    nsle
time, PhysicalTime ! time at the end of this loadstep
    solve ! solve this time step
    ~mpcci, settag, PhysicalTime, i
    ~mpcci, exchange ! exchange data via MpCCI
  *endif
*endo
d
fini ! finish solution routine
~mpcci, stop ! finalize the co-simulation
/exit, nosa ! quit ANSYS
```

Before initializing MpCCI the transfer tag should be set. This transfer tag assigns to the quantity value a time and/or iteration information.

MpCCI is initialized by `~mpcci, init, 2D`. This means the model is two dimensional.
First we do an initial quantities transfer call by executing the command `~mpcci, exchange`. In MpCCI GUI depending on the value set the script will automatically execute one of the actions:

- **receive**: ANSYS receives data.
- **send**: ANSYS sends data.
- **exchange**: ANSYS exchanges data.

Next the loop for the coupled simulation starts with one dummy run first. ANSYS sometimes has problems without this dummy run. Afterwards the solution `~mpcci, settag, PhysicalTime, i` sets the current time and iteration for the solution quantity and `~mpcci, exchange` sends the nodal positions to the partner code. ANSYS will wait until the partner code has received this data. Following it waits for the partner code to finish the solution and receives the results. The loop continues until the final step is reached.

The command `~mpcci, stop` is finishing the MpCCI process regularly and ANSYS could be finished.

⚠️ If element table items should be transferred, generate them before the send or exchange command is executed.

### 3.2.2.2 Available Commands

The command for MpCCI calls within ANSYS is `~mpcci` followed by command line options. The following command line options are valid, where *marked values are default values and options in square brackets [] are optional.

```
~mpcci, FSIMAP, filename.ml
~mpcci, WRCPL, filename.cpl
~mpcci, WRGMD, filename.gmd, element-component,
     fixed-nodes-component, floating-nodes-component
~mpcci, MORPH [, node-component | EXIT ]
~mpcci, SETTAG [, time [, iter ]]
~mpcci, STATUS
~mpcci, *HELP
~mpcci, INIT [ , 2D | 3D | AX | *AUTO ]
~mpcci, SEND [ , ALWAYS | ONEWAIT | ALLWAIT ]
~mpcci, RECEIVE [ , ALL | ANY | AVAILABLE | COMPLETE ]
~mpcci, EXCHANGE [ , ALWAYS | ONEWAIT | ALLWAIT [, ALL | ANY | AVAILABLE | COMPLETE ]]
~mpcci, REMESH [ , MOVE | *TOTAL ]
~mpcci, QSTAT
~mpcci, STOP
~mpcci, QUIT
```

The available commands have the following functionality:

#### ~mpcci, FSIMAP, <filename>.ml

Writes a component list file with all defined element components with the mesh information in "<filename>.ml". This file is normally generated and used by the MpCCI FSIMapper, but can also be generated with the `fsimap` command option. Then the MpCCI FSIMapper will use this file for the file based mapping.  
⚠️ Only for internal use.

#### ~mpcci, WRCPL, <filename>.cpl

Writes a component list file with all defined components and variables into "<filename>.cpl". This file is normally generated and used by theMpCCI GUI, but can also be generated with the `wrcpl`
command option. Then the GUI will use this file for the 
"<filename>.db" file.

Only for internal use.

~mpcci, WRGND, <filename>, grid, element-component, fixed-nodes-component, floating-nodes-component

Writes an MpCCI Grid Morpher data file.

~mpcci, MORPH [, node-component | EXIT ]

Executes the MpCCI Grid Morpher. If [EXIT] is given the MpCCI Grid Morpher exits.

~mpcci, SETTAG [, time [, iter]]

Defines the current time, iteration for the solution quantities to request.

~mpcci, STATUS

Print the actual MpCCI status.

~mpcci, HELP

Print list of available commands.

~mpcci, EXIT [, 2D | 3D | AX | *AUTO ]

Initializes the MpCCI process.

~mpcci, SEND [, ALWAYS | ONEWAIT | ALLWAIT ]

Performs a SEND transfer action, sending data from ANSYS to the partner code.

• If ALWAYS is given, ANSYS always sends all requested quantities.
• If ONEWAIT is given, ANSYS is waiting for at least one partner code (which is using the quantities) before sending all quantities for an uni-directional coupling scheme. For a bi-directional coupling scheme, ANSYS sends the data only if at least one partner code is ready to receive the data, otherwise this operation is skipped.
• If ALLWAIT is given, ANSYS is waiting for all partner codes (which is using the quantities) before sending all quantities for an uni-directional coupling scheme. For a bi-directional coupling scheme, ANSYS sends the data when all partner codes are ready to receive the data, otherwise this operation is skipped.

~mpcci, RECEIVE [, *ALL | ANY | AVAILABLE | COMPLETE ]

Performs a RECEIVE transfer action, receive data from the partner code.

• If ALL is given, ANSYS always waits for all requested quantities.
• If ANY is given, ANSYS receives all quantities if at least one quantity is available from the partner code, if not, ANSYS will continue and no data will be transferred.
• If AVAILABLE is given, ANSYS does not wait and get the available new data from the partner code.
• If COMPLETE is given ANSYS receives only if all quantities are available, if not ANSYS will continue and no data will be transferred.

~mpcci, EXCHANGE [, ALWAYS | ONEWAIT | ALLWAIT [, ALL | ANY | AVAILABLE | COMPLETE ]

Performs EXCHANGE transfer operation, first SEND followed by RECEIVE.

• The first optional argument corresponds to the send operation mode.
• The second optional argument corresponds to the receive operation mode.

~mpcci, REMESH [, MOVE | *TOTAL ]

NotifiesMpCCI that a remesh has been requested by user. If MOVE is given, this represents a mesh deformation, in opposite to TOTAL which is a remeshing of the domain.

~mpcci, QSTAT

Queries the MpCCI server about the capability to receive quantities.

• MPCCI_QSTAT > 0 ANSYS can receive.
• MPCCI_QSTAT < 0 there is no need to receive.
• MPCCI_QSTAT = 0 ANSYS cannot receive.
"mpcci, STOP
Performs an MpCCI stop and stops the whole MpCCI process.

"mpcci, QUIT
Performs an MpCCI quit and quits the MpCCI process within ANSYS.

### 3.2.2.3 Data Access

In the GUI-option receive/send method there are two methods:

- **Direct**: direct read or store of data using “UPF” (user programmable feature) subroutines
- **ETAB**: only send possible (no receive of values into “ETAB”)

Here the ETAB option means that a quantity is read out of an element-table (“ETAB”). It is only valid when sending an element based quantity.

If the quantity is a scalar quantity with dim=1 (e.g. Joule heat density), choose a storage index \( sindex \). The user has to generate an element table fulfilling the naming convention:

\[
\text{etab, MPCCI}_{<sindex>}, <item>, <component>
\]

For example: To get joule heat density from storage index 0 the APDL command should be:

\[
\text{etab, MPCCI}_{00}, \text{jheat}
\]

(see ANSYS documentation of \texttt{ETABLE} command for details)

The user has to fill the “ETAB” with sensible values before the quantities are sent to the partner application.

If the quantity has a dimension >1 (vector quantity) then you have to generate element tables following this naming convention:

\[
\begin{align*}
\text{etab, MPCCI}_{<sindex>}, & <item>, <component> \\
\text{etab, MPCCI}_{<sindex+1>}, & <item>, <component> \\
\text{etab, MPCCI}_{<sindex+2>}, & <item>, <component>
\end{align*}
\]

For example: to get define Lorentz force density vector into element tables starting with storage index 5 the APDL command could be:

\[
\begin{align*}
\text{etab, volu, volu} & \quad ! \text{element volume} \\
\text{sexp, MPCCI}_05, \text{lfx}, \text{volu},,-1 & \quad ! \text{lorentz force density} \\
\text{sexp, MPCCI}_06, \text{lfy}, \text{volu},,-1 \\
\text{sexp, MPCCI}_07, \text{lfz}, \text{volu},,-1
\end{align*}
\]
3.2.3 Models Step

In the Models step, the following options must be chosen:

**ANSYS Release**  Select the ANSYS release you wish to use. Only supported releases are listed. latest always refers to the latest supported version (default). The release should match the input file.

**ANSYS product to run**  Choose one of the ANSYS products you have licensed, see also section 3.1.6.

**ANSYS database file (*)**  Select the ANSYS database which contains your model definitions.

**Unit system**  Select the unit system which was used in ANSYS (see ▶ 1.2 Unit Systems ◀).

3.2.4 Algorithm Step

For general options please refer to ▶ V-4.5.2 Code Specific Algorithm Settings ◀. There is no special specified setting for ANSYS.

3.2.5 Regions Step

ANSYS supports the following quantities for coupling:

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Dim.</th>
<th>Default Value</th>
<th>Integration Type</th>
<th>Coupling Dimension</th>
<th>Location Option</th>
<th>Send Option</th>
<th>Receive Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbsPressure</td>
<td>Scalar</td>
<td>0.0 N/m²</td>
<td>flux dens.</td>
<td>Line, Face,</td>
<td>Element</td>
<td>Direct, Direct</td>
<td></td>
</tr>
<tr>
<td>BodyForce</td>
<td>Vector</td>
<td>0.0 N/m³</td>
<td>flux dens.</td>
<td>Volume</td>
<td>Node, Element</td>
<td>ETAB</td>
<td></td>
</tr>
<tr>
<td>CGAngle</td>
<td>Vector</td>
<td>0.0 rad</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td></td>
</tr>
<tr>
<td>CGOmega</td>
<td>Vector</td>
<td>0.0 rad/s</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td></td>
</tr>
<tr>
<td>CGPosition</td>
<td>Vector</td>
<td>0.0 m</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td></td>
</tr>
<tr>
<td>CGVelocity</td>
<td>Vector</td>
<td>0.0 m/s</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td></td>
</tr>
<tr>
<td>ChargeDensity</td>
<td>Scalar</td>
<td>0.0 C/m³</td>
<td>field</td>
<td>Line,</td>
<td>Element</td>
<td>ETAB</td>
<td></td>
</tr>
<tr>
<td>Current1</td>
<td>Scalar</td>
<td>0.0 A</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td></td>
</tr>
<tr>
<td>Current2</td>
<td>Scalar</td>
<td>0.0 A</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td></td>
</tr>
<tr>
<td>Current3</td>
<td>Scalar</td>
<td>0.0 A</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td></td>
</tr>
<tr>
<td>Current4</td>
<td>Scalar</td>
<td>0.0 A</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2: ANSYS options in the Models step
<table>
<thead>
<tr>
<th>Quantity</th>
<th>Dim.</th>
<th>Default Value</th>
<th>Integration Type</th>
<th>Coupling Dimension</th>
<th>Location</th>
<th>Send Option</th>
<th>Receive Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrentDensity</td>
<td>Vector</td>
<td>0.0 A/m²</td>
<td>flux dens.</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>DeltaTime</td>
<td>Scalar</td>
<td>1.0 s</td>
<td>g-min</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>ElectCond1</td>
<td>Scalar</td>
<td>0.0 S/m</td>
<td>field</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ElectCond3</td>
<td>Vector</td>
<td>0.0 S/m</td>
<td>field</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ElectCondX</td>
<td>Scalar</td>
<td>0.0 S/m</td>
<td>field</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ElectCondY</td>
<td>Scalar</td>
<td>0.0 S/m</td>
<td>field</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ElectCondZ</td>
<td>Scalar</td>
<td>0.0 S/m</td>
<td>field</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ElectricField</td>
<td>Vector</td>
<td>0.0 V/m</td>
<td>field</td>
<td>Line, Volume</td>
<td>Element</td>
<td>ETAB</td>
<td></td>
</tr>
<tr>
<td>ElectricFlux</td>
<td>Vector</td>
<td>0.0 C/m²</td>
<td>flux dens.</td>
<td>Line, Volume</td>
<td>Element</td>
<td>ETAB</td>
<td></td>
</tr>
<tr>
<td>ElectricPot</td>
<td>Scalar</td>
<td>0.0 V</td>
<td>field</td>
<td>Line, Face, Volume</td>
<td>Node</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ElectRes1</td>
<td>Scalar</td>
<td>0.0 ohm m</td>
<td>field</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ElectRes3</td>
<td>Vector</td>
<td>0.0 ohm m</td>
<td>field</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ElectResX</td>
<td>Scalar</td>
<td>0.0 ohm m</td>
<td>field</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ElectResY</td>
<td>Scalar</td>
<td>0.0 ohm m</td>
<td>field</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ElectResZ</td>
<td>Scalar</td>
<td>0.0 ohm m</td>
<td>field</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>Enthalpy</td>
<td>Scalar</td>
<td>0.0 W/m³</td>
<td>flux dens.</td>
<td>Volume</td>
<td>Node</td>
<td>ETAB</td>
<td></td>
</tr>
<tr>
<td>FilmTemp</td>
<td>Scalar</td>
<td>300.0 K</td>
<td>field</td>
<td>Face</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>Force</td>
<td>Vector</td>
<td>0.0 N</td>
<td>flux integral</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>HeatFlux</td>
<td>Vector</td>
<td>0.0 W/m³</td>
<td>flux dens.</td>
<td>Volume</td>
<td>Node, Element</td>
<td>ETAB</td>
<td></td>
</tr>
<tr>
<td>HeatSource</td>
<td>Scalar</td>
<td>0.0 W/m³</td>
<td>flux dens.</td>
<td>Volume</td>
<td>Node, Element</td>
<td>ETAB</td>
<td></td>
</tr>
<tr>
<td>IntFlag</td>
<td>Scalar</td>
<td>0 g-max</td>
<td></td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>IterationNo</td>
<td>Scalar</td>
<td>0 g-max</td>
<td></td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>JouleHeat</td>
<td>Scalar</td>
<td>0.0 W/m³</td>
<td>flux dens.</td>
<td>Volume</td>
<td>Element</td>
<td>Direct, ETAB</td>
<td>Direct</td>
</tr>
<tr>
<td>LorentzForce</td>
<td>Vector</td>
<td>0.0 N/m³</td>
<td>flux dens.</td>
<td>Volume</td>
<td>Node, Element</td>
<td>ETAB</td>
<td></td>
</tr>
<tr>
<td>MagneticField</td>
<td>Vector</td>
<td>0.0 A/m</td>
<td>field</td>
<td>Line, Volume</td>
<td>Element</td>
<td>ETAB</td>
<td></td>
</tr>
<tr>
<td>MagneticFlux</td>
<td>Vector</td>
<td>0.0 T</td>
<td>flux dens.</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>NPosition</td>
<td>Vector</td>
<td>0.0 m</td>
<td>mesh coordinate</td>
<td>Line, Face, Volume</td>
<td>Node</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>Quantity</td>
<td>Dim.</td>
<td>Default Value</td>
<td>Integration Type</td>
<td>Coupling Dimension</td>
<td>Location Option</td>
<td>Send Option</td>
<td>Receive Option</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------</td>
<td>---------------</td>
<td>------------------</td>
<td>--------------------</td>
<td>----------------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>OverPressure</td>
<td>Scalar</td>
<td>0.0 N/m²</td>
<td>flux dens.</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>PhysicalTime</td>
<td>Scalar</td>
<td>0.0 s</td>
<td>g-min</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>PointPosition</td>
<td>Vector</td>
<td>0.0 m</td>
<td>mesh coordinate</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>RealFlag</td>
<td>Scalar</td>
<td>0.0</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>RefPressure</td>
<td>Scalar</td>
<td>1.12e5 N/m²</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>RelWallForce</td>
<td>Vector</td>
<td>0.0 N</td>
<td>flux integral</td>
<td>Face</td>
<td>Node</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>Residual</td>
<td>Scalar</td>
<td>0.0</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>Temperature</td>
<td>Scalar</td>
<td>300.0 K</td>
<td>field</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ThermCond1</td>
<td>Scalar</td>
<td>0.0 W/mK</td>
<td>field</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>ThermCond3</td>
<td>Vector</td>
<td>0.0 W/mK</td>
<td>field</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>ThermCondX</td>
<td>Scalar</td>
<td>0.0 W/mK</td>
<td>field</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>ThermCondY</td>
<td>Scalar</td>
<td>0.0 W/mK</td>
<td>field</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>ThermCondZ</td>
<td>Scalar</td>
<td>0.0 W/mK</td>
<td>field</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>TimeStepNo</td>
<td>Scalar</td>
<td>0</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>Torque</td>
<td>Vector</td>
<td>0.0 N.m</td>
<td>flux integral</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>Velocity</td>
<td>Vector</td>
<td>0.0 m/s</td>
<td>field</td>
<td>Volume</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>Voltage1</td>
<td>Scalar</td>
<td>0.0 V</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>Voltage2</td>
<td>Scalar</td>
<td>0.0 V</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>Voltage3</td>
<td>Scalar</td>
<td>0.0 V</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>Voltage4</td>
<td>Scalar</td>
<td>0.0 V</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>WallForce</td>
<td>Vector</td>
<td>0.0 N</td>
<td>flux integral</td>
<td>Face</td>
<td>Node</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>WallHeatFlux</td>
<td>Scalar</td>
<td>0.0 W/m²</td>
<td>flux dens.</td>
<td>Face</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>WallHTCoeff</td>
<td>Scalar</td>
<td>0.0 W/m² K</td>
<td>field</td>
<td>Face</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>WallTemp</td>
<td>Scalar</td>
<td>300.0 K</td>
<td>field</td>
<td>Face</td>
<td>Node</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>
3.2.6 Go Step

In the Go step, the following options can be selected:

**Select a gui option**  It is recommended to select -b to start ANSYS in the batch mode.

With -g ANSYS is started with a graphical user interface with one of the graphics devices: -d X11 is the standard UNIX device, use -d X11C to activate light-shading on devices with more than 16 colors. -d WIN32 and -d WIN32C are the corresponding devices for Windows. The option -d 3D should be used if you have a 3-D graphics device.

**Optional GPU device name**  Define the GPU device name (intel, nvidia) to be used by ANSYS.

ANSYS command line option is `-acc`. For example: `-acc intel -na 1`.

**No. solver threads**  Set the number of solver threads used by ANSYS with SMP (Shared Memory Parallel).

ANSYS command line option is `-np`.

**Run parallel using the DDS**  Setting this option activates the Distributed Domain Solver.

ANSYS command line option is `-dis`.

Feature is not supported if the quantity NPosition is exchanged.

**No. of processors**  Enter the number of processors to use during the analysis.

ANSYS command line option is `-np`.

**Parallel communicator**  Select the parallel communicator as given in the ANSYS manual, this is passed as option `-mpi=` to ANSYS.

**Optional 'host host ...' to be used**  Enter host names for parallel execution of ANSYS.

**Optional hostlist file**  A default hostfile can be configured by setting MPCCI_HOSTLIST_FILE, see V-3.5.2 Hostlist File.<br>

**Additional command line options**  You can specify additional ANSYS command line options here, which are described in “3.1 Starting an ANSYS Session from the Command Level” of the ANSYS documentation.

**Select APDL input script**  Select the file containing the APDL script you created for the analysis (see 3.2.2 APDL Script).
3.2.7 Running the Computation

When the [Start] button is pressed, ANSYS is started with the options given in the Go step. If the [Stop] button is pressed, a stop-file "*.ABT" is created (see also “Basic Analysis Guide, 3.8 Terminating a Running Job” in the ANSYS documentation) and ANSYS will stop the next time it checks the presence of a stop file.
3.3 Code-Specific MpCCI Commands

The MpCCI commands which are available for ANSYS are:

**Usage:**
```
mpcci ANSYS [-]option
```

**Synopsis:**
'mpcci ANSYS' is used to get information about ANSYS.

**Options:**
- **-align** <ARGS>
  Do a coordinate transformation on all nodes of a .cdb file based on a plane definition file and align the nodal coordinates for the coupling partner.

- **-convert** <db file> [product] [release]
  Run the converter and create an FSIMapper input file.

- **-dbcheck** <db file> [release]
  Check a .db or .cdb file.

- **-dbsave** <jobname> <folder_name>
  Copy all esav, osav, emat, rst files with the job name to the archive folder.

- **-help**
  This screen.

- **-info**
  List verbose information about all ANSYS releases.

- **-products**
  List all available licensed ANSYS products.

- **-releases**
  List all ANSYS releases which MpCCI can find.

- **-scan** <db file> [product] [release]
  Run the scanner and create a scanner output file.

The subcommands **align**, **info**, **releases** and **scan** are described in §§1.1 Common MpCCI Subcommands for Simulation Codes. The **convert** subcommand runs the converter for the given ANSYS db file and creates an MpCCI FSIMapper input file. Optionally the ANSYS product and release can be specified. The defaults are ansys product and latest release.

**Usage:**
```
mpcci ANSYS convert <db file> [product] [release]
```
Examples:
- mpcci ANSYS convert manifold.db
- mpcci ANSYS convert busbar.db emag
- mpcci ANSYS convert elastic_flap.db ansys 11
- mpcci ANSYS convert nozzle.cdb 11

mpcci ansys -dbcheck <db file> [release]
The `dbcheck` subcommand checks a " .db" or " .cdb" file for an optional given release. The latest release will be taken as default.

Usage:
- mpcci ANSYS dbcheck <db file> [release]

Examples:
- mpcci ANSYS dbcheck manifold.db
- mpcci ANSYS dbcheck busbar.db latest
- mpcci ANSYS dbcheck elastic_flap.cdb 11

mpcci ansys -dbsave <jobname> <folder name>
The `dbsave` subcommand copies all esav, osav, emat and rst files with the given job name to the specified archive folder.

Usage:
- mpcci ANSYS dbsave <jobname> <folder_name>

Examples:
- mpcci ANSYS dbsave mpccirun mpccirun.db

mpcci ansys -products
The `products` subcommand lists the available ANSYS products, e. g. those for which licenses are available. The list looks like:

> mpcci ansys -products
ane3fl
ansys
emag
structds

See also “Installation and Licensing Documentation, 3.4 Product Variable Table” in the ANSYS documentation.
3.4 Code Adapter Reference

Within the MpCCI distribution the "adapters" directory contains the necessary software to connect the simulation programs to MpCCI depending on your license. The files are located within the subdirectory ":<MpCCI_home>/codes/ANSYS/adapters:"

This subdirectory is further divided by several release subdirectories, e.g. "110" and "120". The version directories are further divided by several architectures (e.g. "linia32"). There you find the library files of the ANSYS adapter (e.g. "libansysmpcci.so"). The connection to MpCCI is established using these shared libraries. The binding to the APDL command mpcci is set in the "ans_ext.tbl" file.

To enable coupling capabilities the mpcci command is added to the standard APDL commands. By the command line options you decide which coupling function is called. The basic tasks of the command are to initialize the coupling and to manage the data transfer to MpCCI commands.

Ensure that the directory path of your MpCCI-APDL scripts on Windows do not contain any whitespace!

3.5 Frequently Asked Questions

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Could ANSYS use automatic time stepping scheme for the coupling?</td>
<td>Currently ANSYS adapter does not support it. Instead a user defined time stepping has to be implemented in the APDL script as done for example in the &quot;VII-4 Elastic Flap in a Duct&quot;.</td>
</tr>
<tr>
<td>In MpCCI Regions step I can’t find any component for coupling the time step size in the Global Panel. How can I activate the Global Panel?</td>
<td>In ANSYS model you need to create a parameter name DELTATIME for instance with the command: <em>SET,DELTATIME,1</em> where DELTATIME is the name of the parameter and 1 the default value. That parameter name will be also used for setting the quantity tag information (see APDL example in &quot;VII-4 Elastic Flap in a Duct&quot;). You will be to save your ANSYS model and rescan it in the MpCCI GUI.</td>
</tr>
<tr>
<td>I would like to use shell elements rather than solid ones to model a flexible wall. I wonder whether I should use two layers of plate elements to model my flexible wall: one real and one dummy.</td>
<td>By using shell elements without solid for the modeling you do not need dummy elements as recommended in &quot;3.2.1 Model Preparation&quot; and do not need two layers. You can directly use shell elements. It is required to orient the shell element normal for the surface definition in the same direction and to apply a predefined load.</td>
</tr>
<tr>
<td>I noticed that the result of &quot;OverPressure&quot; transferred is quite different than with &quot;RelWallForce&quot;. I used shell elements for modeling my wall and have defined a default load on the surface. What happens?</td>
<td>In that case the surface contains a mixed orientation of the shell elements. The default load on one side of the shell element is used to know where to apply the load coming from MpCCI. By having mixed orientation of shell elements provide for force values different results than using pressure load. The model has to provide a homogeneous orientation of the shell face.</td>
</tr>
</tbody>
</table>
**Question:**
In MpCCI GUI by selecting the ANSYS product, I get the message error:

"No valid license files or servers found."

How can I select a product for scanning?

**Answer:**
Set the environment variable `ANSYSLMD_LICENSE_FILE` to the FLEXlm license server for the ANSYS product.

Execute the command `mpcci ansys product` to check ANSYS product license availability.

---

**Question:**
In MpCCI GUI by selecting the ANSYS product or by executing the command `mpcci ansys -products`, I get the message error:

```
mpcci ansys products:
Executable
   "/home/software/MpCCI/license/linux_em64t/lmutil"
 exited with code 244.
*** lmutil - Copyright (c) 1989-2006 Macrovision Europe Ltd. and/or
 Macrovision Corporation. All Rights Reserved.
*** Flexible License Manager status on Tue 7/13/2010 15:43
*** Error getting status: Invalid returned data from license server
 system. (-12,16)
```

```
mpcci ansys products:
   No valid license files or servers found.
```

```
mpcci ansys products:
   No ANSYS product found or the license is temporarily unavailable.
```

The variable `ANYSYSLIC_ADMIN` is set in my user setting environments to `1056@license.com` and MpCCI did not found any products.

**Answer:**
Set the environment variable `ANSYSLMD_LICENSE_FILE` to the FLEXlm license server for the ANSYS product.

The FLEXlm message error:

`Error getting status: Invalid returned data from license server system. (-12,16)` indicates that the port 1056 is not associate with a FLEXlm server but with the ANSYS Licensing Interconnect server.

Ask the IT for more information about the license file used for ANSYS especially for the following lines:

```
SERVER license.com 00469f763f3f 1055
VENDOR ansyslmd port=1056
```

You need to set the `ANSYSLMD_LICENSE_FILE` variable to `1055@license.com` in that example.

Execute the command `mpcci ANSYS -products` to check ANSYS product license availability.
4 ANSYS Icepak

4.1 Quick Information

| Company name | Fluent Inc. is a wholly owned subsidiary of ANSYS, Inc. |
| Company homepage | www.ansys.com |
| Support | ANSYS Customer portal at support.ansys.com |
| Tutorials | No dedicated tutorial available |

ANSYS Icepak is based on FLUENT solver technology and is dedicated for 3D CFD thermal analysis applications.

For setting up the coupling with MpCCI, please refer to the FLUENT >8.2 Coupling Process <.

### 4.1.1 Supported Coupling Methods

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Dimension</th>
<th>Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit</td>
<td>Implicit</td>
<td>(V-3.1.1 Physical Domains)</td>
</tr>
<tr>
<td>ANSYS Icepak</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

### 4.1.2 Supported Platforms and Versions

#### MpCCIARCH:

<table>
<thead>
<tr>
<th>Code platform</th>
<th>Supported versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARCH</td>
<td>16.0.0, 16.1.0, 16.2.0, 17.0.0, 17.1.0, 18.0.0, 18.1.0, 18.2.0, 19.0.0, 19.1.0, 19.2.0, 19.3.0, 19.4.0, 19.5.0, 20.1.0, 20.2.0, 21.1.0</td>
</tr>
</tbody>
</table>

#### lnx4_x64:

| lnamd64 | X X X X X X X X X X X X X X X X X X |

#### windows_x64:

| win64 | X X X X X X X X X X X X X X X X X X |

### 4.1.3 Supported Quantities

ANSYS Icepak supports the following quantities for coupling:

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Dim.</th>
<th>Default Value</th>
<th>Integration Type</th>
<th>Coupling Location</th>
<th>Sent Option</th>
<th>Receive Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeltaTime</td>
<td>Scalar</td>
<td>1.0 s</td>
<td>g-min</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
</tr>
<tr>
<td>Enthalpy</td>
<td>Scalar</td>
<td>0.0 W/m³</td>
<td>flux dens.</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
</tr>
<tr>
<td>FilmTemp</td>
<td>Scalar</td>
<td>300.0 K</td>
<td>field</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
</tr>
<tr>
<td>HeatFlux</td>
<td>Vector</td>
<td>0.0 W/m²</td>
<td>flux dens.</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
</tr>
<tr>
<td>HeatSource</td>
<td>Scalar</td>
<td>0.0 W/m³</td>
<td>flux dens.</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
</tr>
<tr>
<td>IntFlag</td>
<td>Scalar</td>
<td>0</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
</tr>
<tr>
<td>IterationNo</td>
<td>Scalar</td>
<td>0</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
</tr>
<tr>
<td>JouleHeat</td>
<td>Scalar</td>
<td>0.0 W/m³</td>
<td>flux dens.</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
</tr>
<tr>
<td>JouleHeatLin</td>
<td>Scalar</td>
<td>0.0 W/m³ K</td>
<td>flux dens.</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
</tr>
<tr>
<td>PhysicalTime</td>
<td>Scalar</td>
<td>0.0 s</td>
<td>g-min</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
</tr>
<tr>
<td>RealFlag</td>
<td>Scalar</td>
<td>0.0</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
</tr>
<tr>
<td>Quantity</td>
<td>Dim.</td>
<td>Default Value</td>
<td>Integration Type</td>
<td>Coupling Dimension</td>
<td>Location</td>
<td>Send Option</td>
</tr>
<tr>
<td>---------------</td>
<td>--------</td>
<td>---------------</td>
<td>------------------</td>
<td>--------------------</td>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>Residual</td>
<td>Scalar</td>
<td>0.0</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
</tr>
<tr>
<td>SpecificHeat</td>
<td>Scalar</td>
<td>1.0 J/kg K</td>
<td>field</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
</tr>
<tr>
<td>Temperature</td>
<td>Scalar</td>
<td>300.0 K</td>
<td>field</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
</tr>
<tr>
<td>ThermCond1</td>
<td>Scalar</td>
<td>0.0 W/m K</td>
<td>field</td>
<td>Face,</td>
<td>Code</td>
<td>Dir</td>
</tr>
<tr>
<td>ThermCond3</td>
<td>Vector</td>
<td>0.0 W/m K</td>
<td>field</td>
<td>Face,</td>
<td>Code</td>
<td>Dir</td>
</tr>
<tr>
<td>ThermCondX</td>
<td>Scalar</td>
<td>0.0 W/m K</td>
<td>field</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
</tr>
<tr>
<td>ThermCondY</td>
<td>Scalar</td>
<td>0.0 W/m K</td>
<td>field</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
</tr>
<tr>
<td>ThermCondZ</td>
<td>Scalar</td>
<td>0.0 W/m K</td>
<td>field</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
</tr>
<tr>
<td>TimeStepNo</td>
<td>Scalar</td>
<td>0</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
</tr>
<tr>
<td>WallHeatFlux</td>
<td>Scalar</td>
<td>0.0 W/m² K</td>
<td>flux dens.</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
</tr>
<tr>
<td>WallHTCoeff</td>
<td>Scalar</td>
<td>0.0 W/m² K</td>
<td>field</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
</tr>
<tr>
<td>WallTemp</td>
<td>Scalar</td>
<td>300.0 K</td>
<td>field</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
</tr>
</tbody>
</table>

### 4.2 Code-Specific MpCCI Commands

The MpCCI subcommands available for ANSYS Icepak are:

**Usage:**

```bash
mpcci IcePak [-]option
```

**Synopsis:**

Use 'mpcci IcePak' to get information about IcePak and to build/install your private adapter ...

**Options:**

- **-diff** `<cas1>` `<cas2>`
  
  Run the scanner on two case files and print the differences.

- **-help**
  
  This screen.

- **-info**
  
  List verbose information about all IcePak releases.

- **-libmpcci** `<RELEASE>` [-64] `<VERSION>`
  
  Install the IcePak

    "libmpcci/ARCH/VERSION/libudf.so"

  in your current working directory - where the .cas and .dat files are located - by either just copying the MpCCI libudf's or remaking the libudf from MpCCI and your own sources located in "libmpcci/src".

  You do NOT need to have a "libmpcci/Makefile" and/or "libmpcci/src/makefile" prepared since the makefiles are...
generated automatically from your IcePak installation.

RELEASE: The IcePak release.
ARCH : The IcePak architecture token
    (automatically determined by MpCCI)
VERSION: The version 2d, 3d_node etc.

Please specify the IcePak release (e.g. 13.0.0) you
would like to use and a list of versions (2d, 3d_node etc.).

For more information type "mpcci IcePak libmpcci".

`-libudf` `<RELEASE>` [-64] `<VERSION>` [MSVC_VERSION]
Compile the IcePak

"libudf/ARCH/VERSION/libudf.so"

from your current working directory - where the .cas and
.dat files are located - by making the libudf with your own
sources located in "libudf/src".

RELEASE : The IcePak release.
ARCH : The IcePak architecture token
    (automatically determined by MpCCI)
VERSION : The version 2d, 3d_node etc.
MSVC_VERSION: The version of MSVC compiler to use
    (MSVC_100, MSVC_110, MSVC_120, MSVC_140, MSVC).

Please specify the IcePak release (e.g. 13.0.0) you
would like to use and a list of versions (2d, 3d_node etc.).

For more information type "mpcci IcePak libudf".

`-releases`
List all IcePak releases which MpCCI can find.

`-scan` `<casfile>`
Run the scanner and create a scanner output file.

The subcommands `diff`, `info`, `releases` and `scan` are described in > 1.1 Common MpCCI Subcommands for Simulation Codes.<

The subcommands `libmpcci` and `libudf` are described in detail above.
5 FINE/Open

5.1 Quick Information

<table>
<thead>
<tr>
<th>Company name</th>
<th>NUMECA International</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company homepage</td>
<td><a href="http://www.numeca.com">www.numeca.com</a></td>
</tr>
<tr>
<td>Support</td>
<td><a href="mailto:support@numeca.com">support@numeca.com</a></td>
</tr>
<tr>
<td>Tutorials</td>
<td>▶ VII-2 Vortex-Induced Vibration of a Thin-Walled Structure ◀</td>
</tr>
<tr>
<td></td>
<td>▶ VII-4 Elastic Flap in a Duct ◀</td>
</tr>
</tbody>
</table>

5.1.1 Supported Coupling Methods

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Dimension</th>
<th>Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit</td>
<td>Implicit</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>X - - - X -</td>
</tr>
<tr>
<td>CFD, Fluid, FluidThermal</td>
</tr>
</tbody>
</table>

5.1.2 Supported Platforms and Versions

<table>
<thead>
<tr>
<th>MPCCI ARCH:</th>
<th>Supported versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code platform</td>
<td>101</td>
</tr>
<tr>
<td>lxnx64: x86_64</td>
<td>X</td>
</tr>
<tr>
<td>windows_x64: win64</td>
<td>X</td>
</tr>
</tbody>
</table>

5.1.3 References

FINE/Open Documentation is part of the FINE/Open distribution. The section “Co-simulation using MpCCI” describes some expert parameter options.

5.1.4 Adapter Description

The code adapter for FINE/Open is developed by NUMECA International in cooperation with Fraunhofer SCAI. The code adapter for FINE/Open is based on a dynamic library "libmpcci_cadapt_64.[so|dll]", which is loaded by FINE/Open. It includes the necessary interface functions.

5.1.5 Prerequisites for a Coupled Simulation

To run a coupled simulation you need the following:
- Ordinary FINE/Open installation.
- License feature (Solver_MpCCI) for coupled simulation using MpCCI and for mesh deformation module.
5.2 Coupling Process

5.2.1 Model Preparation

There are two options to prepare the FINE/Open model for a MpCCI coupled simulation:

1. If you are not familiar with FINE/Open, MpCCI can prepare the model ".run" file for the coupling. In this step the boundaries will be automatically activated by MpCCI before the code starts. Therefore the FINE/Open model must be set up for a standalone simulation run.

2. If you are familiar with FINE/Open, you can set up your model as usual and activate the boundaries by yourself for the coupling. In this scenario you will have a scan method option in the Models Step of MpCCI GUI to select only the boundaries you have prepared for the MpCCI coupling.

The FINE/Open model (".run" file) has to be prepared with FINE/Open GUI. Please consider the following approach for model preparation:

- The FINE/Open solver internally operates only in SI units. The geometrical dimensions should best defined in meters. However, it is possible to define a scaling factor in FINE/Open GUI.
- During the model preparation you must assign appropriate names to the boundaries. These ones may be used for the coupling.
- When receiving the wall temperature in FINE/Open set the boundary condition of coupled walls to “Temperature Imposed”, otherwise you will get an error message.
- For a 2D co-simulation analysis, the parameter iMpCCIzSize should be set to the length of the cell in z-direction (default value is 1 m). This parameter can be set from the Local parameters in the Advanced menu of the Control Variables in the Computation Control section.
- The FINE/Open computation must run in standalone.

5.2.2 Models Step

In the Models step, the following options must be chosen:

**FINE/Open release** Select the FINE/Open release you wish to use. Only supported releases installed on your system are listed. The selection latest always refers to the latest supported version (default).

**Scan method** This can be set to:
- **Scan for all regions** (default) – The model file is scanned for possible coupling regions.
- **Scan predefined COUPLING regions only** – The model file is scanned for the already activated coupling regions only (prepared in the FINE/Open GUI).

![Select code to couple](image-url)
Run file  Select the FINE/Open input file "*.run".

5.2.3 Algorithm Step

For general options please refer to V-4.5.2 Code Specific Algorithm Settings.<br>
In the Algorithm step, following additional option is available:

Solver settings

Compute steady solution  This is an advanced parameter setting the FINE/Open parameter <i>impcCSteadyMode</i>. This option forces the unsteady solver to compute a steady solution. Please consult the NUMECA International support team to get more information about the application case.

5.2.4 Regions Step

FINE/Open supports the following quantities for coupling:

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Dim.</th>
<th>Default Value</th>
<th>Integration Type</th>
<th>Coupling Dimension</th>
<th>Location Option</th>
<th>Send Option</th>
<th>Receive Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbsPressure</td>
<td>Scalar</td>
<td>0.0 N/m²</td>
<td>flux dens.</td>
<td>Face</td>
<td>Node,</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Element</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeltaTime</td>
<td>Scalar</td>
<td>1.0 s</td>
<td>g-min</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>Density</td>
<td>Scalar</td>
<td>1.0 kg/m³</td>
<td>field</td>
<td>Volume</td>
<td>Node,</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Element</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IterationNo</td>
<td>Scalar</td>
<td>0</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>NPosition</td>
<td>Vector</td>
<td>0.0 m</td>
<td>mesh coordinate</td>
<td>Face</td>
<td>Node</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>OverPressure</td>
<td>Scalar</td>
<td>0.0 N/m²</td>
<td>flux dens.</td>
<td>Face</td>
<td>Node,</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Element</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PhysicalTime</td>
<td>Scalar</td>
<td>0.0 s</td>
<td>g-min</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>RefPressure</td>
<td>Scalar</td>
<td>1.12e5 N/m²</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>RelWallForce</td>
<td>Vector</td>
<td>0.0 N</td>
<td>flux integral</td>
<td>Face</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>TimeStepNo</td>
<td>Scalar</td>
<td>0</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>Velocity</td>
<td>Vector</td>
<td>0.0 m/s</td>
<td>field</td>
<td>Face</td>
<td>Node,</td>
<td>Direct</td>
<td>Element</td>
</tr>
<tr>
<td>WallHeatFlux</td>
<td>Scalar</td>
<td>0.0 W/m²</td>
<td>flux dens.</td>
<td>Face</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>WallTemp</td>
<td>Scalar</td>
<td>300.0 K</td>
<td>field</td>
<td>Face</td>
<td>Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>

It is not possible for FINE/Open solver to receive DeltaTime quantity. The time step can only be sent to the counterpart code.

5.2.5 Go Step

Solver precision  Solver precision is double.

Define a reference pressure [N/m²]  If you exchange relative pressure (e.g. RelWallForce or Overpressure), you must set the reference pressure to an appropriate value, which usually corresponds to the atmospheric pressure. The unit of this value is assumed as m².

Expert mode  For FSI problems with mesh deformation involved some expert parameters have to be set correctly to choose the most appropriate deformer. Remeshing is not possible during a computation. MpCCI offers the basic FINE/Open mesh deformation settings.
**Mesh deformation:** Select the mesh deformation to use. This option refers to the FINE/Open parameter `mov_iMovingGridMeth_`.

**Inverse distance weighting (if available)** Use morphing method based on inverse distance weighting interpolation.

- The option is available since FINE/Open 6.1. If this method is selected with an older version, the default method Radial basis functions will be applied automatically.

**Quaternions method** The solving of Laplace equation computing the displacement.

- Radial basis functions The radial basis function interpolating the displacement on boundary nodes. (default)
  - If this method is selected an additional setting `RBF mesh deformation function` appears with the following options:
    - Thin plate spline
    - Wendland function By selecting this function you will be able to define a radius for the Wendland function under the `Radius of the support of the Wendland function` corresponding to the FINE/Open parameter `mov_rad_` (default is 1).

**Control if deformation is performed from initial mesh** This option refers to the FINE/Open parameter `mov_iFromInitMesh_`.

You can access further expert mode parameters for the mesh deformation method by using the FINE/Open GUI like:

- Coarsening level of the mesh boundary: `mov_iLevelCoarse_` parameter.
- etc.

Please refer to FINE/Open documentation describing further options for the mesh deformation module.

**Run parallel** Select this to start a parallel run. A panel with additional options appears, which are further described in \(\triangleright 5.2.6.1\) Parallel Execution \(\triangleright\).

### 5.2.6 Running the Computation

When the **Start** button is pressed, MpCCI will prepare the ".run" file corresponding to the FINE/Open project.

All the options for activating the coupling in FINE/Open are automatically done by MpCCI like:

- the coupling definition is set up for the corresponding boundary with the corresponding coupling definition (Thermal coupling, Mechanic coupling, Thermo-mechanic coupling) depending on the selected quantity to couple. Figure 3.
• the definition of the reference pressure and the activation of the Force and Torque Figure 4.

MpCCI creates a new ".run" file having a prefix "mpcci-" of the original ".run".

If the Stop button is pressed, a stop-file is created in the directory of the computation and FINE/Open will stop the next time it checks the presence of a stop file.
5.2.6.1 Parallel Execution

Parallel runs of FINE/Open are supported by MpCCI.

In the Go step, a set of additional options can be chosen for a parallel run as shown in Figure 5.

No. of parallel processes Enter the number of processes to use during the analysis.

Parallel settings The Automatic load balancing is used as default method for the mesh decomposition.

Optional 'host host...' to be used Enter host names for parallel execution of FINE/Open.

Optional hostlist file Specify a hostfile, from which host names are extracted. A default hostfile can be configured by setting MPCCI_HOSTLIST_FILE. A default hostfile can be configured by setting MPCCI_HOSTLIST_FILE.

Use default hostfile

Figure 5: Parallel options

5.2.6.2 Batch Execution

FINE/Open always runs as a batch process.

5.2.7 Post-Processing

Post-processing for the FINE/Open part of the results can be performed as in ordinary computations, e.g. with CFVIEW, which is part of FINE/Open package.

5.3 Code-Specific MpCCI Commands

The MpCCI subcommands available for FINE/Open are:

Usage:
mpcci FINEOpen [-]option

Synopsis:
'mpcci FINEOpen' is used to get information about FINEOpen.

Options:
-diff <run1> <run2>
Run the scanner on two .run files and print the differences.
### 5.4 Code Adapter Reference

The code adapter is distributed as a dynamic library, which is located in
```
"<MpCCI_home>/codes/FINEOpen/adapters/"
```
A link to the corresponding dynamic library has to be created in FINE/Open installation directory as explained in documentation. This dynamic library is used to establish connection between MpCCI and FINE/Open.

FINE/Open supports data exchange after solution.

### 5.5 Limitations

- Solver is always in double precision.
- FINE/Open does not run in parallel under Microsoft Windows platform if no pvm daemon has been started.
  You can not launch a remote batch job on windows if no pvm daemon is running. Additionally you should take care of correctly configuring the MPI daemon.
6 FINE/Turbo

6.1 Quick Information

<table>
<thead>
<tr>
<th>Company name</th>
<th>NUMECA International</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company homepage</td>
<td><a href="http://www.numeca.com">www.numeca.com</a></td>
</tr>
<tr>
<td>Support</td>
<td><a href="mailto:support@numeca.com">support@numeca.com</a></td>
</tr>
<tr>
<td>Tutorials</td>
<td>◄ VII-2 Vortex-Induced Vibration of a Thin-Walled Structure ◄</td>
</tr>
<tr>
<td></td>
<td>◄ VII-4 Elastic Flap in a Duct ◄</td>
</tr>
</tbody>
</table>

6.1.1 Supported Coupling Methods

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Dimension</th>
<th>Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit</td>
<td>▼</td>
<td>(► V-3.1.1 Physical Domains ◄)</td>
</tr>
<tr>
<td>Implicit</td>
<td>▲</td>
<td>CFD, Fluid, FluidThermal</td>
</tr>
</tbody>
</table>

6.1.2 Supported Platforms and Versions

<table>
<thead>
<tr>
<th>Code platform</th>
<th>Supported versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>lnx4_x64:</td>
<td>111 12 121 122 131 132 141 142 151</td>
</tr>
<tr>
<td>x86_64:</td>
<td>X X X X X X X X X X X X</td>
</tr>
<tr>
<td>windows_x64:</td>
<td>X X X X X X X X X X X X</td>
</tr>
<tr>
<td>win64:</td>
<td>X X X X X X X X X X X X</td>
</tr>
</tbody>
</table>

6.1.3 References

FINE/Turbo Documentation is part of the FINE/Turbo distribution. The section “Co-simulation using MpCCI” describes some expert parameter options.

6.1.4 Adapter Description

The code adapter for FINE/Turbo is developed by NUMECA International in cooperation with Fraunhofer SCAI. The adapter is distributed as part of MpCCI software.

6.1.5 Prerequisites for a Coupled Simulation

To run a coupled simulation you need the following:

- Ordinary FINE/Turbo installation.
- License feature (Solver, MpCCI) for coupled simulation using MpCCI and for mesh deformation module.
6.2 Coupling Process

6.2.1 Model Preparation

The FINE/Turbo model (".run" file) may be prepared with the FINE/Turbo GUI. Please consider the following approach for model preparation in FINE/Turbo GUI:

- During the model preparation in FINE/Turbo you should assign appropriate names to the boundaries you intend to couple. It might be clearer for the coupling process to group coupled boundaries. The default boundary names correspond to the FINE/Turbo block zones respectively "rows".

- When sending relative wall force (RelWallForce) or relative pressure (e.g. Overpressure) a reference pressure must be defined in the Reference Values Panel in Flow Model, which usually corresponds to the atmospheric pressure (see Figure 1).

- When receiving the wall temperature in FINE/Turbo set the boundary condition of coupled walls to "Temperature Imposed", otherwise you will get an error message.

- The FINE/Turbo computation should in any case run in standalone.

When starting the simulation MpCCI will make a patched copy of the " .run" file and activate certain parts which FINE/Turbo requires for a coupled simulation run. One of these operation is that the coupled surfaces are flagged. There are two options to prepare the FINE/Turbo model for a MpCCI coupled simulation:

1. If you are not so familiar with FINE/Turbo GUI, MpCCI can prepare the model ".run" file for the coupling. In this step the boundaries will be automatically flagged by MpCCI before the code starts.

2. If you are familiar with FINE/Turbo, you can set up your model as usual and activate the boundaries by yourself for the coupling (▷ 6.2.1 Model Preparation) in FINE/Turbo GUI:

   Optional Models - Fluid Structure - Thermal or Mechanical (see Figure 2, Figure 3).

   In this scenario you will have a scan method option in the Models Step of MpCCI GUI to select only the boundaries you have prepared for the MpCCI coupling (see ▷ 6.2.2 Models Step).

![Figure 1: Setting the reference pressure in FINE/Turbo GUI](image)
6.2 Coupling Process

### 6.2.2 Models Step

In the Models step, the following options must be chosen:

**FINE/Turbo release**  Select the FINE/Turbo release you wish to use. Only supported releases installed on your system are listed. The selection latest always refers to the latest supported version (default).

**Scan method**  This can be set to:

- **Scan for all regions** (default) – The model file is scanned for possible coupling regions.

---

Figure 2: Setting coupled thermal boundaries in FINE/Turbo GUI

Figure 3: Setting coupled mechanical boundaries in FINE/Turbo GUI

Figure 4: FINE/Turbo options in the Models step
• Scan predefined COUPLING regions only – The model file is scanned for the already activated coupling regions only (prepared in the FINE/Turbo GUI).

Run file Select the FINE/Turbo input file "*.run".

Unit system As FINE/Turbo works in SI units, no choice is given for the unit system to the user.

6.2.3 Algorithm Step

For general options please refer to V-4.5.2 Code Specific Algorithm Settings. There is no special specified setting for FINE/Turbo.

6.2.4 Regions Step

FINE/Turbo supports the following quantities for coupling:

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Dim.</th>
<th>Default Value</th>
<th>Integration Type</th>
<th>Coupling Dimension</th>
<th>Location</th>
<th>Send Option</th>
<th>Receive Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbsPressure</td>
<td>Scalar</td>
<td>0.0 N/m²</td>
<td>flux dens.</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>DeltaTime</td>
<td>Scalar</td>
<td>1.0 s</td>
<td>g-min</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>Density</td>
<td>Scalar</td>
<td>1.0 kg/m³</td>
<td>field</td>
<td>Volume</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>NPosition</td>
<td>Vector</td>
<td>0.0 m</td>
<td>mesh coordinate</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>OverPressure</td>
<td>Scalar</td>
<td>0.0 N/m²</td>
<td>flux dens.</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>PhysicalTime</td>
<td>Scalar</td>
<td>0.0 s</td>
<td>g-min</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>RelPressure</td>
<td>Scalar</td>
<td>1.12e5 N/m²</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>RelWallForce</td>
<td>Vector</td>
<td>0.0 N</td>
<td>flux integral</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>WallForce</td>
<td>Vector</td>
<td>0.0 N</td>
<td>flux integral</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>WallHeatFlux</td>
<td>Scalar</td>
<td>0.0 W/m²</td>
<td>flux dens.</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>WallTemp</td>
<td>Scalar</td>
<td>300.0 K</td>
<td>field</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>

It is not possible for FINE/Turbo solver to receive DeltaTime or PhysicalTime quantity. The time step can only be sent to the counterpart code.

6.2.5 Go Step

Solver precision Select the solver precision to use.

Set the requested memory Define the memory size to use for the computation. The first two fields show the current setting from the FINE/Turbo run file.

Expert mode For FSI problems with mesh deformation involved some expert parameters have to be set correctly to choose the most appropriate deformer. Remeshing is not possible during a computation. MpCCI offers the basic FINE/Turbo mesh deformation settings.

Mesh deformation Select the mesh deformation to use.

• Laplacian smoothing: the solving of Laplace equation computing the displacement.
• Radial basis functions (default): the radial basis function interpolating the displacement on boundary nodes.

Save inverse matrix of radial basis functions Define the solving method for system of equations.

• Do not use inverse matrix
• Compute and use inverse matrix (default)
• Use existing inverse matrix
You can access further expert mode parameters for the mesh deformation method by using the FINE/Turbo GUI like:

- Smoothing method: IMVWEI parameter.
- Choice of RBF function: MOVRBF parameter.
- etc.

Please refer to FINE/Turbo Physical Models chapter documentation describing further options for the mesh deformation module.

Run parallel  Select this to start a parallel run. A panel with additional options appears, which are further described in §6.2.6.1 Parallel Execution.

6.2.6 Running the Computation

When the [Start] button is pressed, MpCCI will prepare the ".run" file corresponding to the FINE/Turbo project. All the options for activating the coupling in FINE/Turbo are automatically done by MpCCI if needed. MpCCI creates a new ".run" file having a prefix "mpcci_" of the original ".run".

If the [Stop] button is pressed, a stop-file is created in the directory of the computation and FINE/Turbo will stop the next time it checks the presence of a stop file.

6.2.6.1 Parallel Execution

Parallel runs of FINE/Turbo are supported by MpCCI.

In the Go step, a set of additional options can be chosen for a parallel run as shown in Figure 6.

No. of processors  Enter the number of processors to use during the analysis.

Parallel settings  The automatic load balancing is used as default method for the mesh decomposition.

Optional 'host host...' to be used  Enter host names for parallel execution of FINE/Turbo.
Optional hostlist file  Specify a hostfile, from which host names are extracted &gt; V-3.5.2 Hostlist File &lt;.

Use default hostfile  A default hostfile can be configured by setting MPCCI_HOSTLIST_FILE &gt; V-3.5.2 Hostlist File &lt;.

Running FINE/Turbo in parallel the number of processors may be modified by FINE/Turbo itself if the number of processors wished is higher than the number of blocks available in the model.

6.2.6.2 Batch Execution
FINE/Turbo always runs in batch mode.

6.2.7 Post-Processing
Post-processing for the FINE/Turbo part of the results can be performed as in ordinary computations, e.g. with CFVIEW, which is part of FINE/Turbo package.

6.3 Code-Specific MpCCI Commands
The MpCCI subcommands available for FINE/Turbo are:

Usage:
mpcci FINETurbo [-]option

Synopsis:
'mpcci FINETurbo' is used to get information about FINETurbo.

Options:
-diff <run1> <run2>
Run the scanner on two .run files and print the differences.

-help
This screen.

-info
List verbose information about all FINETurbo releases.
\begin{itemize}
  \item \texttt{-intmem <RUN-file>}
    Show the number of ints used by this computation.
  \item \texttt{-realmem <RUN-file>}
    Show the number of reals used by this computation.
  \item \texttt{-releases}
    List all FINETurbo releases which MpCCI can find on the local system.
  \item \texttt{-scan <RUN-file>}
    Run the scanner and create an output file.
\end{itemize}

The subcommands \texttt{diff}, \texttt{info}, \texttt{releases} and \texttt{scan} are described in \texttt{1.1 Common MpCCI Subcommands for Simulation Codes}.

\texttt{mpcci fineturbo -intmem <RUN-file>}

The \texttt{-intmem} subcommand shows the number of ints used by the computation for the given run file.

\texttt{mpcci fineturbo -realmem <RUN-file>}

The \texttt{-realmem} subcommand shows the number of reals used by the computation for the given run file.

### 6.4 Code Adapter Reference

The code adapter is distributed as a dynamic library, which is located in

"\texttt{<MCCl\_home>/codes/FINETurbo/adapters/}".

FINE/Turbo supports exchange before solution.
### 6.5 Trouble Shooting, Open Issues and Known Bugs

**Problem:**
The MpCCI tutorial example with FINE/Turbo crashes after the MpCCI adapter library has been loaded. The message just points to an error which may be a parameter issue.

**Workaround:**
Please open and resave the project file under the FINE/Turbo release you want to run the computation. Since the project file has been created from the oldest supported release by MpCCI, it may require an upgrade in the project file format.
7 FloMASTER

7.1 Quick Information

<table>
<thead>
<tr>
<th>Company name</th>
<th>Mentor Graphics Corporation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company homepage</td>
<td><a href="http://www.mentor.com/products/mechanical/flomaster/flomaster/">www.mentor.com/products/mechanical/flomaster/flomaster/</a></td>
</tr>
<tr>
<td>Support</td>
<td><a href="http://supportnet.mentor.com">http://supportnet.mentor.com</a></td>
</tr>
<tr>
<td>Tutorials</td>
<td>⊳ VII-10 Y-Junction ⌠ ⌠</td>
</tr>
</tbody>
</table>

7.1.1 Supported Coupling Methods

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Dimension</th>
<th>Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit</td>
<td>Implicit</td>
<td>⊳ V-3.1.1 Physical Domains ⌠ ⌠</td>
</tr>
</tbody>
</table>

| FloMASTER | X | - | X | - | - | - | network |

7.1.2 Supported Platforms and Versions

MpCCI supports coupling with FloMASTER 8.0 and higher.

7.1.3 References

FloMASTER documentation is part of the FloMASTER distribution.

7.1.4 Adapter Description

The FloMASTER adapter is an executable based on the FloMASTER COM interface.

7.1.5 Prerequisites for a Coupled Simulation

To run a coupled simulation you need the following:

- Ordinary FloMASTER installation.

7.2 Coupling Process

You have to create and validate a FloMASTER model.

7.2.1 Model Preparation

Both prescribed flow (inlet/outlet) and pressure boundaries in a CFD model can be incorporated into an MpCCI co-simulation with FloMASTER. The connection to boundaries in a CFD model is realized within a FloMASTER network by a combination of flow and/or pressure sources. Figure 1 and Figure 2 show the components used to represent a prescribed flow boundary and a pressure boundary respectively.
Note that the CFD prescribed flow boundary incorporates in FloMASTER a pressure source whereas the CFD pressure boundary requires a FloMASTER flow source as coupling partner. Components within the FloMASTER network, which are attached to the coupled boundaries, are connected to the nodes of the attached network in the normal manner.

- For each boundary component you have to activate the external boundary property of the component.
- For each flow source component you need to specify a flow value to initialize the flow source.

You have to create the fmlink file in order to export the information of your network to MpCCI code scanner (see Figure 3). The necessary steps in FloMASTER are:

1. Enter the report dialog and choose the MpCCI ASCII file.
2. Select a boundary component.
3. Click on Add in order to fill the list.
4. Mark the selected component and give a name for this component.
5. Repeat step 1 to 4 for all your boundary components.
6. Select the directory to save your file.
7. Enter the name of your network model with the file suffix ".fmlink".
8. Click on Create ASCII file.
Figure 3: Export FloMASTER boundaries dialog
7 FloMASTER

7.2.2 Models Step

In the Models step, the following options must be chosen:

**FloMASTER release** Select the release of FloMASTER you want to use. latest (default) will select the latest version which is installed on your system.

**Data file** Select the Data file of your FloMASTER project.

The MpCCI scanner uses the FloMASTER data file to extract model information. This file has the suffix ".fmlink".

### Algorithms Step

For general options please refer to ⊳ V-4.5.2 Code Specific Algorithm Settings ⊲. There is no special specified setting for FloMASTER.

### Regions Step

The FloMASTER adapter only stores quantities directly ("Direct"). FloMASTER supports the following quantities for coupling:

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Dim.</th>
<th>Default Value</th>
<th>Integration Type</th>
<th>Coupling Dimension</th>
<th>Location</th>
<th>Send Option</th>
<th>Receive Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeltaTime</td>
<td>Scalar</td>
<td>1.0 s</td>
<td>g-min</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>MassFlowRate</td>
<td>Scalar</td>
<td>0.0 kg/s</td>
<td>flux integral</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>MassFluxRate</td>
<td>Scalar</td>
<td>0.0 kg/m²/s</td>
<td>flux dens.</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>PhysicalTime</td>
<td>Scalar</td>
<td>0.0 s</td>
<td>g-min</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>Temperature</td>
<td>Scalar</td>
<td>300.0 K</td>
<td>field</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>TotalPressure</td>
<td>Scalar</td>
<td>0.0 N/m²</td>
<td>flux dens.</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>VelocityMagnitude</td>
<td>Scalar</td>
<td>0.0 m/s</td>
<td>field</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>

The quantities are calculated as depicted exemplary for the field quantity temperature $T$ and the flux quantity mass flow $m$. in Figure 5.
7.2 Coupling Process

### 7.2.5 Go Step

FloMASTER offers a number of options in the Go panel, see Figure 6.

- **User name**: You have to provide the user name to log in the database.
- **User password**: The password to authenticate the database server.
- **Working project name**: The full project tree to the current project containing the network.
  - For example, the default root project name is “Flowmaster” and you have two sub-projects “coupling” and “testcases” with the following hierarchy: “Flowmaster\coupling\testcases”.
  - The project “testcases” contains some network models which one of them has been chosen for the simulation.
  - Then you will have to provide the following working project name: “Flowmaster\coupling”.
- **Analysis type**: The analysis type to use for the simulation:

Figure 5: Exemplary calculation of flux quantity mass flow \( m \) and field quantity temperature \( T \) between 1D and 3D codes.

\[
\begin{align*}
m_{1D}^3 &= m_{1D}^1 \cdot \frac{a_{3D}^1}{A} \\
T_{3D}^i &= T_{1D}^i \\
\sum_{i=0}^{n} m_i^3 &= \sum_{i=0}^{n} m_i^1 \\
\sum_{i=0}^{n} T_{3D}^i &= \sum_{i=0}^{n} T_{1D}^i \\
a_{3D}^i &= \text{element area} \\
T_{3D}^i &= \text{element temperature} \\
m_{3D}^i &= \text{element mass flow} \\
a_{1D} &= \text{total area of 3D surface} \\
i &= \text{a element of surface} \\
n &= \text{number of elements of surface}
\end{align*}
\]
7.2.5.1 Running the Computation

By pressing the Start button in the Go step of the MpCCI GUI, MpCCI starts the FloMASTER application. The output of the FloMASTER simulation is logged in a window and a file "mpcci_<model name>.log".

7.2.5.2 Post-Processing

You may use the FloMASTER interface and analyze the results by querying the database or creating a report.

7.3 Code-Specific MpCCI Commands

The MpCCI subcommands available for FloMASTER are:

```
Usage:
  mpcci FloMASTER [-]option

Synopsis:
  'mpcci FloMASTER' is used to get information about FloMASTER.

Options:
  -diff <file1> <file2> Run the scanner on two .fmlink files and print the differences.
  -help This screen.
  -info List verbose information about all FloMASTER releases.
  -releases List all FloMASTER releases which MpCCI can find.
  -scan <input-file> Run the scanner and create a scanner output file.
```

The subcommands `diff`, `info`, `releases` and `scan` are described in > 1.1 Common MpCCI Subcommands for Simulation Codes <.

7.4 Code Adapter Reference

Within the MpCCI distribution the "adapters" directory contains the necessary software to connect the simulation programs to MpCCI. The files are located within the subdirectory
"<MpCCI_home>/FloMASTER/adapters".
This subdirectory is further divided by several release subdirectories, e.g. "7.6". The version directories contain one architecture folder "mswin". There you find the executable of the FloMASTER adapter, e.g. "MpCCILink.exe". The connection to MpCCI is established using this executable and the implementation is based on the FloMASTER COM API.

For a prescribed flow boundary, the boundary component (component 3 in Figure 2) sets the pressure of the source provided by MpCCI via FloMASTER adapter. After the network analysis is complete the mass flow is read by the boundary component and returned to FloMASTER adapter.

Depending on the FloMASTER analysis type the following data exchange strategy is applied:

- Steady-state analysis
  1. FloMASTER receives the data.
  2. FloMASTER analysis is requested.
  3. FloMASTER sends the data.

- Transient analysis
  1. FloMASTER exchanges the data (send and receive).
  2. FloMASTER analysis is requested.
### 7.5 Trouble Shooting, Open Issues and Known Bugs

#### Problem:
The FloMASTER adapter failed just after starting the application with the message: *Failed to create the Flowmaster Application interface*

#### Version:
7.7

#### Fix:
You have to register three FloMASTER dynamic library files in the assembly cache (GAC) of the Microsoft Windows machine:

- "Flowmaster.Automation.Gui.dll"
- "Flowmaster.Automation.Analysis.dll"
- "Flowmaster.Interfaces.dll"

These files are located under the FloMASTER installation.

1. Open two file explorers, one pointing to "C:/WINDOWS/assembly" and the other to the FloMASTER installation directory e.g. "C:/Program files/Flowmaster/FlowmasterV7".
2. Per drag and drop, move the dll files to the assembly directory.

Then use the program "regasm.exe" to register the three previous dlls. You can find the "regasm.exe" program under "%SystemRoot%/Microsoft.NET/Framework/v2.0.50727".

Execute the commands from the FloMASTER installation directory:

- `regasm Flowmaster.Automation.Gui.dll`
- `regasm Flowmaster.Automation.Analysis.dll`
- `regasm Flowmaster.Interfaces.dll`

You should get the following message: *Types registered successfully*.

You need to have administrator privileges to execute these commands.

#### References:
Contact FloMASTER support about how to register dlls to the GAC or assembly registry.
8 FLUENT

8.1 Quick Information

<table>
<thead>
<tr>
<th>Company name</th>
<th>Fluent Inc. is a wholly owned subsidiary of ANSYS, Inc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company homepage</td>
<td><a href="http://www.ansys.com">www.ansys.com</a></td>
</tr>
<tr>
<td>Support</td>
<td>ANSYS Customer portal at support.ansys.com</td>
</tr>
<tr>
<td>Tutorials</td>
<td>VII-2 Vortex-Induced Vibration of a Thin-Walled Structure</td>
</tr>
<tr>
<td></td>
<td>VII-3 Driven Cavity</td>
</tr>
<tr>
<td></td>
<td>VII-4 Elastic Flap in a Duct</td>
</tr>
<tr>
<td></td>
<td>VII-5 Exhaust Manifold</td>
</tr>
<tr>
<td></td>
<td>VII-6 Busbar System</td>
</tr>
<tr>
<td></td>
<td>VII-7 Three Phase Transformer</td>
</tr>
<tr>
<td></td>
<td>VII-8 Pipe Nozzle</td>
</tr>
<tr>
<td></td>
<td>VII-9 Cube in a Duct Heater</td>
</tr>
<tr>
<td></td>
<td>VII-10 Y-Junction</td>
</tr>
</tbody>
</table>

8.1.1 Supported Coupling Methods

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Dimension</th>
<th>Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit</td>
<td>Implicit</td>
<td>X X X X X X X X X X X X</td>
</tr>
<tr>
<td>FLUENT</td>
<td></td>
<td>CFD, Fluid, FluidThermal, FluidPlasma</td>
</tr>
</tbody>
</table>

8.1.2 Supported Platforms and Versions

<table>
<thead>
<tr>
<th>MPCCI_ARCH:</th>
<th>Supported versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code platform</td>
<td></td>
</tr>
<tr>
<td>lnx4_x64:</td>
<td>16.0.0 16.1.0 16.2.0 17.0.0 17.1.0 17.2.0 18.0.0 18.1.0 18.2.0 19.0.0 19.1.0 19.2.0 19.3.0 19.4.0 19.5.0 20.1.0 20.2.0 21.1.0</td>
</tr>
<tr>
<td>lnamd64:</td>
<td>X X X X X X X X X X X X X X X X X X</td>
</tr>
<tr>
<td>windows_x64:</td>
<td>X X X X X X X X X X X X X X X X X</td>
</tr>
<tr>
<td>win64:</td>
<td>X X X X X X X X X X X X X X X X X</td>
</tr>
</tbody>
</table>

8.1.3 References

FLUENT Documentation is part of the FLUENT distribution.

Abaqus Fluid-Structure Interaction User’s Guide This guide is available for registered Abaqus users via the Abaqus support homepage abaqus.custhelp.com. Log into Abaqus Answers and search for “FSI guide”. The FSI guide contains several examples of coupled simulations with Abaqus and FLUENT.

8.1.4 Adapter Description

MpCCI uses the user-defined function (“UDF”) interface, which is provided by FLUENT. A set of such
functions is included in the MpCCI distribution as a shared library, which must be loaded by FLUENT. The user-defined functions must be called at appropriate stages of the simulation. For this, FLUENT offers a number of so-called “function hooks”, where these functions must be “hooked”.

8.1.5 Prerequisites for a Coupled Simulation

To run a coupled simulation you need the following:

- Ordinary FLUENT installation.

8.2 Coupling Process

8.2.1 Model Preparation

Models can be prepared as usually. The coupled surface or cell zones must be defined as separate surfaces or volumes. Figure 1 shows the list of surfaces defined in a FLUENT model, which then appear in the Coupling Step of the MpCCI GUI.

If you exchange a relative pressure (e.g. RelWallForce or Overpressure), you must set the reference pressure to an appropriate value, which usually corresponds to the atmospheric pressure. For more information on relative and absolute pressures see the FSI section in V-3.1.2 Coupling Types. In FLUENT the reference pressure is set in the Reference Values panel, which can be found under Report→Reference Values.

All further options required for coupling can be set in the MpCCI GUI.

Figure 1: List of surfaces in FLUENT (left) and in the Regions step of the MpCCI GUI (right).

8.2.1.1 Setting UDF-Hooks

The user-defined functions for the MpCCI interface must be hooked to perform initialization of the coupled simulation and data transfer. It is recommended to let MpCCI handle loading and hooking of these functions automatically. Please choose the corresponding options in the Go Step of the MpCCI GUI.
8.2.1.2 Using Own UDFs and MpCCI

The MpCCI functions are stored in the separate directory "libmpcci". Therefore the MpCCI interface functions do not directly interfere with other interface functions - you can define and hook functions as if you were not using MpCCI.

⚠️ It is not possible to hook your own user-defined functions at places where MpCCI functions need to be hooked for data transfer. It does not make sense to receive data and set the same values by a user-defined function!

8.2.1.3 Deforming Meshes

In typical FSI problems, deformations are computed by the solid mechanics code and sent to FLUENT, i.e. FLUENT has to move boundaries and deform the mesh.

The FLUENT settings to achieve this are made automatically if you select Auto hook functions as well as Auto set MDM zones (“MDM” stands for Moving and Deforming Meshes) from the Go Step of the MpCCI GUI. This enables mesh motion, however the exact parameters cannot be chosen by MpCCI and should be set when preparing the model or before starting the iteration.

To set up the mesh deformation without using the MpCCI GUI, the dynamic mesh option must be enabled by selecting Define → Dynamic Mesh → Parameters... and selecting Dynamic Mesh in FLUENT. It is recommended to use Smoothing and Remeshing as Mesh Methods.

The coupled zones must be defined as dynamic mesh zones which is described in ➤8.4.2 UDF-Hooks.<

- For a transient FSI problem, FLUENT will automatically update the mesh and the deformations are then imported on the boundaries.

- For steady state FSI problems, MpCCI activates the mesh update by calling the (steady-update-dynamic-mesh) during the calculation. Otherwise the deformations are not imported in FLUENT. MpCCI provides for this purpose a new menu entry in FLUENT user interface at MpCCI→MpCCI Run FSI (➤8.2.6.2 The MpCCI Run FSI.).

The MpCCI default setting defines a cell height value of 0 m for the adjacent zone. In the latest FLUENT 15.0 release the option Deform Adjacent Boundary Layer with Zone is not activated.

If you have a symmetry plane zone adjacent to the coupled wall zone boundary, you will need to define the zone as deforming in the dynamic mesh zones. Otherwise you will encounter issues during the mesh morphing step which will not properly deform the symmetry plane with the coupled zones.

⚠️ Current limitation: remeshing is not supported during an iterative coupling with FLUENT.
8.2.1.4 Rigid Body Motion

In some FSI problems a rigid body motion can be computed by the solid mechanics code or the multi body code and sent to FLUENT.

The FLUENT settings to achieve this are made automatically if you select Auto hook functions as well as Auto set MDM zones (“MDM” stands for Moving and Deforming Meshes) from the Go Step of the MpCCI GUI. This enables mesh motion, however the exact parameters cannot be chosen by MpCCI and should be set when preparing the model or before starting the iteration.

To set up the mesh deformation without using the MpCCI GUI, the dynamic mesh option must be enabled by selecting Define→Dynamic Mesh→Parameters... and selecting Dynamic Mesh in FLUENT. It is recommended to use Smoothing and Remeshing as Mesh Methods.

The MpCCI default setting defines the center of gravity at (0,0,0) with a center of gravity orientation theta-z of 0 deg. The layering type is constant with a cell height value of 0 m.

8.2.2 Models Step

![FLUENT options in the Models step](image)

In the Models step, the following options must be chosen:

**FLUENT version** Select the FLUENT version from 2d, 2ddp, 3d, 3ddp. The version must match your case file.

**FLUENT release** Select the release of FLUENT you want to use. The version must match your case file. latest (default) will select the latest version which is installed on your system.

**Create rigid body CG for each wall BC** A new component with the prefix mpcci_cg will be created in association with the wall boundary condition.

**FLUENT case file** Select the case file of your FLUENT model.

8.2.3 Algorithm Step

For general options please refer to ▷ V-4.5.2 Code Specific Algorithm Settings ◁.

In the Algorithm step, following additional option is available:

**Coupling steps**
### Number of inner iterations

*(Only for Implicit coupling scheme.)* Provide the number of inner iterations to perform within a coupling step iteration. The value will be automatically imported into FLUENT, for internal iterations.

### 8.2.4 Regions Step

The FLUENT adapter only stores some quantities directly ("Dir"), most quantities are first written to user-defined memory ("UDM").

FLUENT supports the following quantities for coupling:

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Dim.</th>
<th>Default Value</th>
<th>Integration Type</th>
<th>Coupling Dimension</th>
<th>Location</th>
<th>Send Option</th>
<th>Receive Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbsPressure</td>
<td>Scalar</td>
<td>0.0 N/m²</td>
<td>flux dens.</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>AcstPressure</td>
<td>Scalar</td>
<td>0.0 N/m²</td>
<td>flux dens.</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>AngularVelocity</td>
<td>Vector</td>
<td>0.0 rad/s</td>
<td>field</td>
<td>Point</td>
<td>Code</td>
<td>Buf</td>
<td></td>
</tr>
<tr>
<td>BodyForce</td>
<td>Vector</td>
<td>0.0 N/m³</td>
<td>flux dens.</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>CGAngle</td>
<td>Vector</td>
<td>0.0 rad</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>CGOmega</td>
<td>Vector</td>
<td>0.0 rad/s</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>CPosition</td>
<td>Vector</td>
<td>0.0 m</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>CGVelocity</td>
<td>Vector</td>
<td>0.0 m/s</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>ChargeDensity</td>
<td>Scalar</td>
<td>0.0 C/m³</td>
<td>field</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>Current1</td>
<td>Scalar</td>
<td>0.0 A</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>Current2</td>
<td>Scalar</td>
<td>0.0 A</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>Current3</td>
<td>Scalar</td>
<td>0.0 A</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>Current4</td>
<td>Scalar</td>
<td>0.0 A</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>CurrentDensity</td>
<td>Vector</td>
<td>0.0 A/m²</td>
<td>flux dens.</td>
<td>Face</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>DeltaTime</td>
<td>Scalar</td>
<td>1.0 s</td>
<td>g-min</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td></td>
</tr>
<tr>
<td>Density</td>
<td>Scalar</td>
<td>1.0 kg/m³</td>
<td>field</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>DynPressure</td>
<td>Scalar</td>
<td>0.0 N/m²</td>
<td>flux dens.</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td></td>
</tr>
<tr>
<td>ElectrCond1</td>
<td>Scalar</td>
<td>0.0 S/m</td>
<td>field</td>
<td>Face</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>ElectrCond3</td>
<td>Vector</td>
<td>0.0 S/m</td>
<td>field</td>
<td>Face</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>ElectrCondX</td>
<td>Scalar</td>
<td>0.0 S/m</td>
<td>field</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>ElectrCondY</td>
<td>Scalar</td>
<td>0.0 S/m</td>
<td>field</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>ElectrCondZ</td>
<td>Scalar</td>
<td>0.0 S/m</td>
<td>field</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>ElectricField</td>
<td>Vector</td>
<td>0.0 V/m</td>
<td>field</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>ElectricFlux</td>
<td>Vector</td>
<td>0.0 C/m²</td>
<td>flux dens.</td>
<td>Face</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>ElectrRes1</td>
<td>Scalar</td>
<td>0.0 ohmm</td>
<td>field</td>
<td>Face</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>ElectrRes3</td>
<td>Vector</td>
<td>0.0 ohmm</td>
<td>field</td>
<td>Face</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>ElectrResX</td>
<td>Scalar</td>
<td>0.0 ohmm</td>
<td>field</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>ElectrResY</td>
<td>Scalar</td>
<td>0.0 ohmm</td>
<td>field</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>ElectrResZ</td>
<td>Scalar</td>
<td>0.0 ohmm</td>
<td>field</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>Enthalpy</td>
<td>Scalar</td>
<td>0.0 W/m³</td>
<td>flux dens.</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>FilmTemp</td>
<td>Scalar</td>
<td>300.0 K</td>
<td>field</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>Quantity</td>
<td>Dim.</td>
<td>Default Value</td>
<td>Integration Type</td>
<td>Coupling Dimension</td>
<td>Location Option</td>
<td>Send Option</td>
<td>Receive Option</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------</td>
<td>---------------</td>
<td>------------------</td>
<td>--------------------</td>
<td>-----------------</td>
<td>-------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Force</td>
<td>Vector</td>
<td>0.0 N</td>
<td>flux integral</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td></td>
</tr>
<tr>
<td>Heat Flux</td>
<td>Vector</td>
<td>0.0 W/m²</td>
<td>flux dens.</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>Heat Rate</td>
<td>Scalar</td>
<td>0.0 W</td>
<td>flux dens.</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td></td>
</tr>
<tr>
<td>Heat Source</td>
<td>Scalar</td>
<td>0.0 W/m³</td>
<td>flux dens.</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>Int Flag</td>
<td>Scalar</td>
<td>0 g-max</td>
<td>Global</td>
<td>global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>Iteration No</td>
<td>Scalar</td>
<td>0 g-max</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
<td></td>
</tr>
<tr>
<td>Joule Heat</td>
<td>Scalar</td>
<td>0.0 W/m³</td>
<td>flux dens.</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>Joule Heat Lin</td>
<td>Scalar</td>
<td>0.0 W/m³ K</td>
<td>flux dens.</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>Lorentz Force</td>
<td>Vector</td>
<td>0.0 N/m³</td>
<td>flux dens.</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>Magnetic Field</td>
<td>Vector</td>
<td>0.0 A/m</td>
<td>field</td>
<td>Volume</td>
<td>UDM UDS</td>
<td>UDS</td>
<td></td>
</tr>
<tr>
<td>Magnetic Flux</td>
<td>Vector</td>
<td>0.0 T</td>
<td>flux dens.</td>
<td>Face</td>
<td>Code</td>
<td>UDM</td>
<td></td>
</tr>
<tr>
<td>Mass Flow Rate</td>
<td>Scalar</td>
<td>0.0 kg/s</td>
<td>flux integral</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>Mass Flux Rate</td>
<td>Scalar</td>
<td>0.0 kg/m³ s</td>
<td>flux dens.</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>N Position</td>
<td>Vector</td>
<td>0.0 m</td>
<td>mesh coordinate</td>
<td>Face,</td>
<td>Code</td>
<td>Dir Buf</td>
<td></td>
</tr>
<tr>
<td>Over Pressure</td>
<td>Scalar</td>
<td>0.0 N/m²</td>
<td>flux dens.</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>Physical Time</td>
<td>Scalar</td>
<td>0.0 s</td>
<td>g-min</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>Pore Pressure</td>
<td>Scalar</td>
<td>0.0 N/m²</td>
<td>flux dens.</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>Real Flag</td>
<td>Scalar</td>
<td>0.0 g-max</td>
<td>Global</td>
<td>global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>Ref Pressure</td>
<td>Scalar</td>
<td>1.12e5 N/m²</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>Rel Wall Force</td>
<td>Vector</td>
<td>0.0 N</td>
<td>flux integral</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>Residual</td>
<td>Scalar</td>
<td>0.0 g-max</td>
<td>Global</td>
<td>global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>Specific Heat</td>
<td>Scalar</td>
<td>1.0 J/kg K</td>
<td>field</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>Temperature</td>
<td>Scalar</td>
<td>300.0 K</td>
<td>field</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>Therm Cond 1</td>
<td>Scalar</td>
<td>0.0 W/m K</td>
<td>field</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>Therm Cond 3</td>
<td>Vector</td>
<td>0.0 W/m K</td>
<td>field</td>
<td>Face,</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>Therm Cond X</td>
<td>Scalar</td>
<td>0.0 W/m K</td>
<td>field</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>Therm Cond Y</td>
<td>Scalar</td>
<td>0.0 W/m K</td>
<td>field</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>Therm Cond Z</td>
<td>Scalar</td>
<td>0.0 W/m K</td>
<td>field</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>Time Step No</td>
<td>Scalar</td>
<td>0 g-max</td>
<td>Global</td>
<td>global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>Total Pressure</td>
<td>Scalar</td>
<td>0.0 N/m²</td>
<td>flux dens.</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>Velocity</td>
<td>Vector</td>
<td>0.0 m/s</td>
<td>field</td>
<td>Point,</td>
<td>Code</td>
<td>Dir Buf</td>
<td>UDM</td>
</tr>
<tr>
<td>Velocity Magnitude</td>
<td>Scalar</td>
<td>0.0 m/s</td>
<td>field</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>Voltage 1</td>
<td>Scalar</td>
<td>0.0 V</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>Voltage 2</td>
<td>Scalar</td>
<td>0.0 V</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>Voltage 3</td>
<td>Scalar</td>
<td>0.0 V</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>Voltage 4</td>
<td>Scalar</td>
<td>0.0 V</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>Wall Force</td>
<td>Vector</td>
<td>0.0 N</td>
<td>flux integral</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>Wall Heat Flux</td>
<td>Scalar</td>
<td>0.0 W/m²</td>
<td>flux dens.</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>Wall HT Coeff</td>
<td>Scalar</td>
<td>0.0 W/m³ K</td>
<td>field</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>Wall Temp</td>
<td>Scalar</td>
<td>300.0 K</td>
<td>field</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
</tbody>
</table>
8.2.5 Go Step

FLUENT offers a number of options in the Go panel, see Figure 3.

- **Run in batch mode**  If this option is selected, FLUENT will not start the graphical user interface, i.e. no user interaction is possible. For batch mode a journal file, which contains the FLUENT commands for analysis execution, is required. See 8.2.6.4 Batch Execution for details.

- **Data file**  For the analysis a data file can be read. Same as loading a data file in the FLUENT GUI.

- **Optional journal files**  A semicolon-separated list of journal files can be given, which will be read by FLUENT when it is started. A journal file must be given for batch execution.

- **Additional command line options**  Additional command line options for FLUENT can be given here, they will directly be used when FLUENT is started.

- **Auto install/make libmpcci**  Automatically copy the user-defined library which contains the MpCCI interface to the working directory. If this button is not selected, the library must be copied by hand. Therefore it should always be used in a standard analysis.

- **Auto read case data**  Read the case file which was selected in the Models step when FLUENT is started. If a dat file with the same filename is available, it will be automatically selected at start.

- **Auto load libmpcci**  Load the code adapter library. This options should only be unselected for special purposes.

- **Auto hook functions**  Hooks the user-defined functions of the MpCCI code adapter library automatically. If not set, the functions must be hooked manually as described in 8.2.1.1 Setting UDF-Hooks.

- **Auto set MDM zones**  Sets the interface zones for the Moving Deforming Mesh automatically. If nodal displacements are transferred from a structural code, this option should be selected to allow deformations of the FLUENT mesh.

- **Auto set BC’s**  Automatically set the boundary conditions for the coupling regions, which is necessary to enable data transfer on these boundaries.

- **Run parallel**  Select this, if you want to run FLUENT in parallel mode. See 8.2.6.3 Parallel Execution.
8.2.6 Running the Computation

Coupled simulations can be run using the FLUENT window. For running a simulation without a graphical interface, see 8.2.6.4 Batch Execution.

Before FLUENT is started by pressing the [Start] button in the Go Step of the MpCCI GUI, MpCCI creates a journal file "mpcci_<problem name>.jou", which performs the tasks selected in the MpCCI GUI Go Step. These steps are also reflected in FLUENT’s output which partially depend on the settings (e.g. quantities, steady state/transient):

- Load "cosimtools.scm" which contains helper functions needed for a coupled simulation:
  ```
  Loading "/home/user/mpcci/codes/FLUENT/cosimtools.scm"
  Done.
  ```

- Enable coupling with MpCCI,
  ```
  > (rpsetvar 'mpcci/on? #t)mpcci/on?
  ```

- Read the case file if Auto read case data is selected,
  ```
  > /file/read-case "example.cas"
  Reading "example.cas"
  ```

- Load the MpCCI adapter library "libmpcci" if Auto load libmpcci is checked,
  ```
  > /define/user-defined/compiled-functions/load libmpcci
  "/home/user/computations/fluent"
  Opening library "libmpcci"
  Library "libmpcci/lnx86/3d/libudf.so" opened
  UDF_List_globals
  UDF_Write_globals
  ```

- Hook the MpCCI functions if Auto hook functions is set, e.g.
  ```
  > (co-set-udf-hook 'init "UDF_Initialize::libmpcci")
  > (co-set-udf-hook 'adjust "UDF_Adjust::libmpcci")
  > (register-dynamic-function "exchange-mpcci-quantities" #t #f \%udf-on-demand
  "UDF_Dynamic_exchange::libmpcci")
  ```

- Set dynamic mesh zones if Auto set MDM zones is selected,
  ```
  > /define/models/dynamic-mesh? yes
  > /define/dynamic-zones/create 6 motion type: (stationary rigid-body deforming
  user-defined)
  user-defined UDF_Grid_position::libmpcci
  adjacent cell zone name/id [2]
  cell height (air-remesh) (m) [0]
  ```

- Set boundary conditions if Auto set BC’s is enabled,
  ```
  (co-set-zone-profile "wall" "wall-temperature" 
  "UDM00_Profile::libmpcci")
  Fluent MpCCI_UDMprofile: Set default zero before MpCCI init.
  ```

- Load the MpCCI Control Panel.
  ```
  load "/home/user/mpcci/codes/FLUENT/mpccipanel.scm"
  Loading "/home/user/mpcci/codes/FLUENT/mpccipanel.scm"
  Done.
  ```
If all coupling functions are hooked - either automatically or as described in 8.2.1.1 Setting UDF-Hooks - no special steps need to be carried through to start the simulation:

- Initialize the solution, this will also trigger MpCCI initialization.
- Start the iteration, MpCCI data transfer routines will be called automatically.

8.2.6.1 The MpCCI Control Panel

The MpCCI Control Panel (Figure 4) is an extra panel in the FLUENT GUI, which is available if FLUENT is started during a coupled simulation. It allows to perform some coupling actions manually. Select MpCCI → MpCCI Control... from the FLUENT menu to open the panel.

Select Use MpCCI to enable coupling with MpCCI.

The second box of options hooks or unhooks MpCCI functions. If Initialize MpCCI is checked, the MpCCI initialization will be carried through automatically with FLUENT initialization. If you select the box next to Automated Exchange of Quantities you can decide at which points in the simulation data transfers should take place. Choose one of Before each iteration (recommended for steady-state problems), Before each time step (i.e. exchange before iteration) or After each time step (exchange after iteration). For each selection a specific MpCCI function is hooked.

The further buttons allow to perform coupling tasks on demand. Initialize initialized coupling with MpCCI, Finalize ends the coupling process gracefully, and Abort directly aborts the coupling process.

The transfer buttons trigger a transfer of quantities. Send will yield sending of all quantities which were selected in the Regions step of the MpCCI GUI. Receive yields receiving all quantities selected to receive and Exchange will yield a complete transfer of all selected quantities.

Steady Update Mesh yields an update of the mesh, which makes sense after receiving node positions.
8.2.6.2 The MpCCI Run FSI

The MpCCI Run FSI panel is an extra panel in the FLUENT GUI, which is available from the FLUENT menu \texttt{MpCCI→MpCCI Run FSI...} if FLUENT is starting for

- an implicit coupled simulation (Figure 5):
  
  It allows to perform some time steps with the time step size defined from the MpCCI GUI, to set the Reporting Interval, Profile Update Interval.

  If the adaptive convergence control is used, the FLUENT monitor convergence settings will be used to terminate the time step once the coupled inner iterations have reached the defined convergence. This will lead to an earlier progression to the next time step when the original FLUENT .cas file contained a convergence check for the residual monitors. When the coupled quantities are converged, the settings of the FLUENT convergence check are activated. If the residuals are converged, the next time step is started.

- a steady state FSI simulation (Figure 6):
  
  The MpCCI Run FSI panel is activated if the model has been started in interactive mode and with the Auto hook functions. It allows to perform some iterations and to set the Reporting Interval, Profile Update Interval.

  If the subcyling has been activated from MpCCI GUI, the MpCCI Run FSI panel will automatically take it in consideration.

In the text interface or journal file the simulation can be started with the command:

\[ > \text{(mpcci-solve } n) \]

where \( n \) represents the number of time steps or iterations to be calculated.

![Figure 5: MpCCI Run FSI panel in FLUENT for transient implicit solution](image)

8.2.6.3 Parallel Execution

For a parallel run of FLUENT (i.e. FLUENT itself uses several parallel processes) additional options can be selected in the Go Step as shown in Figure 7.

- Please read also chapter “31. Parallel Processing” of the FLUENT 6.3 User’s Guide.

**Number of parallel processes**  Select how many FLUENT processes you wish to start.

**Interconnector**  Select the interconnector type as given in the FLUENT manual, this is passed as option \(-p\) to FLUENT.
8.2 Coupling Process

Figure 6: MpCCI Run FSI panel in FLUENT for steady state solution

Figure 7: FLUENT options for a parallel run

**Parallel communicator** Select the parallel communicator as given in the FLUENT manual, this is passed as option `-mpi=` to FLUENT.

**Optional ‘host host …’ to be used** Enter host names for parallel execution of FLUENT.

**Optional hostlist file** Specify a hostfile, from which host names are extracted.

**Use default hostfile** A default hostfile can be configured by setting `MPCCI_HOSTLIST_FILE`, see > V-3.5.2 Hostlist File.

### 8.2.6.4 Batch Execution

FLUENT can be started without the graphical user interface. This is important if you run a simulation on a remote computer with text access only. For batch mode in a coupled simulation you must check the Run in batch mode button in the Go Step and provide a journal file which must be given under Optional journal files.

As no graphical interface is available in batch mode you should perform all steps needed for the simulation in the journal file. A simple journal file for a transient analysis would contain:

```plaintext
;; Initialize flow and connect to MpCCI
/solve initialize initialize-flow
;; Perform unsteady iterations for a specified number of time steps
```
If you want to do subcycling (iterations without exchange), you should provide the number of steps without coupling as coupling step size in the Coupling steps area of the Algorithm step. The journal file might contain the usual commands to start the iterations:

```
;s; Define autosave of case and data files
/file/autosave/case-frequency 10
/file/autosave/data-frequency 10
/file/autosave/overwrite-existing-files yes
;s; Initialize flow
/solveinitialize initialize
;s; Iterate
/solve/iterate 1000
;s; Write case and data file
wcd file
;s; Exit simulation
/exit yes
```

It is required that the MpCCI libmpcci is loaded and the necessary functions are hooked. This can be achieved by selecting the required Auto-options in the Go Step (select all if in doubt) or including the corresponding commands in the journal file. The commands for loading and functions hooking can be found in the journal file which is written by MpCCI at start-up of a FLUENT simulation, they are also described in > 8.2.1.1 Setting UDF-Hooks <.

### 8.2.6.5 MpCCI Scheme Commands and Variables

List of available variables to control the calculation flow:

- **mpcci/mpcci_used** provides the MpCCI connection status. If the value is negativ, FLUENT is not anymore connected to MpCCI.
- **mpcci/mpcci_init** provides the MpCCI initialisation status. If the value is equal 0, FLUENT has not been initialized with MpCCI. A value equals to 1 indicated that the initialisation with MpCCI has been done.
- **mpcci/convergence** contains the status of the coupled job after the data transfer occurs. The convergence status may have the following value defined as constant:
  - **mpcci/conv_state_error** indicates an error.
  - **mpcci/conv_state_invalid** indicates an invalid status.
  - **mpcci/conv_state_diverged** indicates a divergence status.
  - **mpcci/conv_state_stop** indicates a stop request.
  - **mpcci/conv_state_converged** indicates a convergence status.
  - **mpcci/conv_state_continue** indicates a continue request.
- **mpcci/coupling-inittact** contains the value of the quantities initial transfer action selected in the MpCCI GUI.
  It may have the following values:
  - **mpcci/init_send** indicates the send action.
  - **mpcci/init_recv** indicates the receive action.
  - **mpcci/init_xchg** indicates the exchange action.
  - **mpcci/init_skip** indicates the skip action.
• **mpcci/coupling-substep** contains the number of substeps defined in MpCCI GUI.

• **mpcci/coupling-type** provides the coupling mode selected from the MpCCI GUI. It may have the following value:
  - **mpcci/explicit-coupling**
  - **mpcci/implicit-coupling**

You can use the ON_DEMAND functions listed in 8.4.2 UDF-Hooks as function from the journal file. You need to use the following declaration in your journal file to name the functions:

```
(define udf-eod-init (co-get-udf-lib-name "UDF_Init_MpCCI") )
(define udf-eod-exit (co-get-udf-lib-name "UDF_Exit_MpCCI") )
(define udf-eod-abort (co-get-udf-lib-name "UDF_Abort_MpCCI") )
(define udf-eod-send (co-get-udf-lib-name "UDF_Send_on_demand") )
(define udf-eod-recv (co-get-udf-lib-name "UDF_Recv_on_demand") )
(define udf-eod-xchg (co-get-udf-lib-name "UDF_Xchg_on_demand") )
```

Then the defined function can be used by using this syntax: `(%udf-on-demand udf-eod-xchg)`

### 8.3 Code-Specific MpCCI Commands

The MpCCI subcommands available for FLUENT are:

**Usage:**

```
mpcci FLUENT [-]option
```

**Synopsis:**

Use 'mpcci FLUENT' to get information about FLUENT and to build/install your private adapter ...

**Options:**

- **-diff** `<cas1>` `<cas2>`
  - Run the scanner on two case files and print the differences.

- **-help**
  - This screen.

- **-info**
  - List verbose information about all FLUENT releases.

- **-libmpcci** `<RELEASE>` [-64] `<VERSION>`
  - Install the FLUENT

```
"libmpcci/ARCH/VERSION/libudf.so"
```

in your current working directory - where the .cas and .dat files are located - by either just copying the MpCCI libudf’s or remaking the libudf from MpCCI and your own sources located in "libmpcci/src". You do NOT need to have a "libmpcci/Makefile" and/or "libmpcci/src/makefile" prepared since the makefiles are...
generated automatically from your FLUENT installation.

RELEASE: The FLUENT release.
ARCH : The FLUENT architecture token
   (automatically determined by MpCCI)
VERSION: The version 2d, 3d_node etc.

Please specify the FLUENT release (e.g. 13.0.0) you would like to use and a list of versions (2d, 3d_node etc.).

For more information type "mpcci FLUENT libmpcci".

-libudf <RELEASE> [-64] <VERSION> [MSVC_VERSION]
Compile the FLUENT

   "libudf/ARCH/VERSION/libudf.so"

from your current working directory - where the .cas and .dat files are located - by making the libudf with your own sources located in "libudf/src".

RELEASE : The FLUENT release.
ARCH : The FLUENT architecture token
   (automatically determined by MpCCI)
VERSION : The version 2d, 3d_node etc.
MSVC_VERSION: The version of MSVC compiler to use
   (MSVC_100, MSVC_110, MSVC_120, MSVC_140, MSVC).

Please specify the FLUENT release (e.g. 13.0.0) you would like to use and a list of versions (2d, 3d_node etc.).

For more information type "mpcci FLUENT libudf".

-releases
List all FLUENT releases which MpCCI can find.

-scan <casfile>
Run the scanner and create a scanner output file.

The subcommands [diff], [info], [releases] and [scan] are described in 1.1 Common MpCCI Subcommands for Simulation Codes <.

mpcci fluent -libmpcci <release#|latest> [-64] version version
The [libmpcci] subcommand installs and compiled if needed a version of the MpCCI udf library as described above.

Usage:
   mpcci FLUENT libmpcci <release#|latest> [-64] version version version ...

Examples:
   mpcci FLUENT libmpcci 12.0.16 3d 3ddp 2d 2ddp
   mpcci FLUENT libmpcci latest 3ddp 3ddp_host
mpcci fluent -libudf <release#|latest> [-64] version version [MSVC_VERSION]

The `libudf` subcommand compiles a version of the user FLUENT udf library as described above.

**Usage:**

```
mpcci FLUENT libudf <release#|latest> [-64] version version version ...
```

**Examples:**

```
mpcci FLUENT libudf 6.2.18 3d_host 3d_node
mpcci FLUENT libudf 6.3.26 3d 3ddp 2d 2ddp
mpcci FLUENT libudf latest 3ddp 3ddp_host
```

8.4 Code Adapter Reference

8.4.1 The MpCCI UDF Library

The MpCCI code adapter for FLUENT uses user-defined functions ("UDF") and user-defined memory ("UDM") to manage the data transfer. The code adapter is distributed as a dynamic library, which is located in 
"<MpCCI_home>/codes/FLUENT/adapters/<FLUENT release>/<platform>/<FLUENT code>". 

Before running the analysis, the code adapter must be copied to a subdirectory "libmpcci" of the working directory. This task is performed automatically by MpCCI if the option Auto install/make libmpcci is selected in the Go Step. Otherwise the command `mpcci FLUENT libmpcci` can be used. The working directory is the directory where the case file "*.cas" is located and where FLUENT is executed.

The adapter functions must be hooked at different places in FLUENT, see ▶ 8.2.1.1 Setting UDF-Hooks ◀. A list of all adapter functions is given in Table 2 on page 88.

More information on user-defined functions (UDFs) can be found in the “FLUENT UDF Manual” which is part of the FLUENT documentation.

8.4.2 UDF-Hooks

The data transfer with MpCCI is realized with user-defined functions (“UDF”), which are included in the MpCCI distribution. A list of all UDFs of the MpCCI adapter is given in Table 2 on page 88.

In some cases it is preferable to set or check the necessary functions hooks manually.

Most UDF functions can be hooked using the User-Defined Function Hooks panel, which opens when selecting Define → User-Defined → Function Hooks... from the FLUENT menu.

For a coupled simulation the following steps must be carried through by user-defined functions:

**MpCCI Initialization** is realized in `UDF_Initialize`, which can be hooked in the UDF-hooks panel or by pressing the Initialize button of the MpCCI Control panel.

**Data exchange** for transient problems with dynamic mesh is carried before the iteration.

For explicit coupling this is carried through by one of these functions `UDF_Dynamic_exchange`, `UDF_Grid_Motion_All` or `UDF_Adjust_Once` depending of the configuration (the time step is sent, not exchanged, received).

For iterative coupling this is carried through by one of these functions `UDF_Dynamic_exchange_implicit`, `UDF_Grid_Motion_All_implicit` or `UDF_Adjust_implicit` depending of the configuration (the time step is sent, not exchanged, received).

For steady state problems `UDF_Adjust` should be used to achieve an exchange for each iteration.
UDF_Adjust and UDF_At_end can be hooked over the FLUENT function hook panel, whereas UDF_Dynamic_exchange can be hooked over the MpCCI Control panel or by the Auto hook functions in MpCCI Go Step.

Data storage For some quantities the data is stored in user defined memory (UDM). To distinguish the quantities, a storage index is set for each quantity in the Regions step, as shown in Figure 8. If you do not change it, the index is increased automatically. In addition the corresponding [UDM...] functions must be hooked at appropriate places.

There are three types of such quantities:

- Profiles for various boundary conditions like inlet or outlet boundaries use the boundary functions UDM00_Profile to UDM10_Profile to set the received quantities on the boundaries. Hook the function in the Boundary Conditions panel of FLUENT (Define→Boundary Conditions... in the menu).
- Source terms are also set in the Boundary Conditions panel but not for inlet or outlet surfaces but for fluid type boundaries: Check Source Terms in the Fluid window and select the [UDM...Source] function for the appropriate source.
- Material property quantities can be set in the Materials panel (Define→Materials). Select user-defined and the appropriate [UDM...Property] function in the User-Defined Functions panel.

![Figure 8: UDM storage index for quantities received by FLUENT.](image)

In addition to this minimal subset of user-function calls, certain situation require additional functions:

Analysis with Deforming Mesh: If the quantity NPosition or rigid body motions are received by FLUENT, the mesh must be updated by a user-defined function. The appropriate functions can be hooked automatically by activating the option Auto set MDM zones. Otherwise it must be manually hooked by selecting Define→Dynamic Mesh→Zones... from the FLUENT window. Select the coupled zones from the list of Zone Names, choose User-Defined as type and select on the selected zone:

- For explicit coupling:
  - UDF_Grid_position::libmpcci if time step is received or for steady state model.
  - UDF_Grid_Motion_All::libmpcci if time step is sent or not exchanged.
- For iterative coupling:
  - UDF_Grid_position Implicit::libmpcci if time step is received.
  - UDF_Grid_Motion_All Implicit::libmpcci if time step is sent or not exchanged.

For the rigid body quantities CGAngle, CGOmega, CGPosition or CGVelocity the type Rigid Body must be selected together with UDF_Grid_position::libmpcci.
An alternative for the rigid body quantities AngularVelocity, Velocity the type Rigid Body must be selected together with UDF.CG.motion::libmpcci.

**Restart Analysis:** To store global values in the "*.dat" file after the first analysis and reread it from there in a restart you should hook UDF_Read_globals and UDF_Write_globals in the UDF-hooks panel as Read Data and Write Data functions.

**Exchange of time step size:** If the time step size is received by FLUENT it is required to hook a function in the Run Calculation panel of FLUENT ((Solve|Solution|Solving)→Run Calculation...). MpCCI will set the Time Stepping Method to Adaptive and select UDF_Deltat::libmpcci as user-defined time step.
<table>
<thead>
<tr>
<th>UDF Hook</th>
<th>MpCCI Function</th>
<th>Purpose / Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>UDF_Initialize</td>
<td>Initialize the connection to MpCCI.</td>
</tr>
<tr>
<td>EXECUTE_AT_EXIT</td>
<td>UDF_At_exit</td>
<td>Shutdown the MpCCI connection.</td>
</tr>
<tr>
<td>ADJUST</td>
<td>UDF_Adjust</td>
<td>MpCCI data transfer.</td>
</tr>
<tr>
<td></td>
<td>UDF_Adjust.Once</td>
<td>MpCCI data transfer.</td>
</tr>
<tr>
<td></td>
<td>UDF_Adjust.implicit</td>
<td>MpCCI data transfer.</td>
</tr>
<tr>
<td></td>
<td>UDF_Adjust.Empty</td>
<td>Empty call for the FLUENT FSI panel.</td>
</tr>
<tr>
<td>DELTAT</td>
<td>UDF_Deltat</td>
<td>Set the time step size if it is received by FLUENT.</td>
</tr>
<tr>
<td>ON_DEMAND</td>
<td>UDF_ListGlobals</td>
<td>List global variables.</td>
</tr>
<tr>
<td></td>
<td>UDF_Dynamic.exchange</td>
<td>MpCCI data transfer in explicit coupling.</td>
</tr>
<tr>
<td></td>
<td>UDF_Dynamic.Set_Adjust.Once</td>
<td>MpCCI updates control flags in explicit coupling.</td>
</tr>
<tr>
<td></td>
<td>UDF_Dynamic.Explicit_Step_begin</td>
<td>MpCCI updates control flags in explicit coupling.</td>
</tr>
<tr>
<td></td>
<td>UDF_Dynamic.Implicit_Step_begin</td>
<td>MpCCI updates control flags in iterative coupling.</td>
</tr>
<tr>
<td></td>
<td>UDF_Dynamic.Implicit_Exchange_Step_begin</td>
<td>MpCCI data transfer in iterative coupling.</td>
</tr>
<tr>
<td></td>
<td>UDF_Exit_MpCCI</td>
<td>Shutdown the MpCCI connection.</td>
</tr>
<tr>
<td></td>
<td>UDF_Abort_MpCCI</td>
<td>Abort the MpCCI connection.</td>
</tr>
<tr>
<td></td>
<td>UDF_Init_MpCCI</td>
<td>Initialize the connection to MpCCI.</td>
</tr>
<tr>
<td></td>
<td>UDF_Send_on_demand</td>
<td>MpCCI data transfer: Send only.</td>
</tr>
<tr>
<td></td>
<td>UDF_Recv_on_demand</td>
<td>MpCCI data transfer: Receive only.</td>
</tr>
<tr>
<td></td>
<td>UDF_Xchg_on_demand</td>
<td>MpCCI data transfer.</td>
</tr>
<tr>
<td></td>
<td>UDF_Remesh_total.on_demand</td>
<td>MpCCI receives a full remeshing information.</td>
</tr>
<tr>
<td></td>
<td>UDF_Remesh_move_on_demand</td>
<td>MpCCI receives a deformation information.</td>
</tr>
<tr>
<td>RW_FILE</td>
<td>UDF_ReadGlobals</td>
<td>Read global data from &quot;*.dat&quot; file.</td>
</tr>
<tr>
<td></td>
<td>UDF_WriteGlobals</td>
<td>Write global data into &quot;*.dat&quot; file.</td>
</tr>
<tr>
<td>CG_MOTION</td>
<td>UDF_CG_motion</td>
<td>Set the rigid body motion.</td>
</tr>
<tr>
<td></td>
<td>UDF_CG_motion.All</td>
<td>MpCCI data transfer and set the rigid body motion.</td>
</tr>
<tr>
<td>GRID_MOTION</td>
<td>UDF_Grid.position</td>
<td>Set nodal coordinates to the received position in iterative coupling.</td>
</tr>
<tr>
<td></td>
<td>UDF_Grid.Motion.All.implicit</td>
<td>MpCCI data transfer and set nodal coordinates to the received position in explicit coupling.</td>
</tr>
<tr>
<td></td>
<td>UDF_Grid.position.exchange</td>
<td>MpCCI data transfer and set nodal coordinates to the received position in explicit coupling.</td>
</tr>
<tr>
<td></td>
<td>UDF_Grid.steadyMDM</td>
<td>MpCCI data transfer and set nodal coordinates to the received position in iterative coupling.</td>
</tr>
<tr>
<td></td>
<td>UDF_Grid.Motion.implicit</td>
<td>MpCCI data transfer and set nodal coordinates to the received position in iterative coupling.</td>
</tr>
<tr>
<td>PROFILE</td>
<td>UDM00_Profile</td>
<td>Set boundary profile values stored in UDM 0 - 10.</td>
</tr>
<tr>
<td></td>
<td>UDM10_Profile</td>
<td></td>
</tr>
<tr>
<td>SOURCE</td>
<td>UDM00_Source</td>
<td>Set a cell source value stored in UDM 0 - 10.</td>
</tr>
<tr>
<td></td>
<td>UDM10_Source</td>
<td></td>
</tr>
<tr>
<td>PROPERTY</td>
<td>UDM00_Property</td>
<td>Set a cell property value stored in UDM 0 - 10.</td>
</tr>
<tr>
<td></td>
<td>UDM10_Property</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: User-defined functions (UDFs) of the MpCCI FLUENT adapter
8.5 Trouble Shooting, Open Issues and Known Bugs

Feature:
FLUENT parallel under Microsoft Windows
Version:
12.x, 13

Problem:
FLUENT calculation hangs during the initialization stage in parallel on a Microsoft Windows workstation if FLUENT graphical interface is used. The Interconnector and Parallel communicator are set to default.

Workaround:
First delete the directory "libmpcci" located under the FLUENT *.cas* model directory. Then in the MpCCI GUI, select the option net from Parallel communicator list.
Using FLUENT interactively requires the net parallel communicator contrary to a batch calculation which may use the default parallel communicator.

By switching the Parallel communicator requires to delete the "libmpcci" each time before the start.
### 8.6 Frequently Asked Questions

#### Question:
FEM model's structure is using shell element. In FLUENT since the structure has no thickness hence no volume, the FSI interface is defined as wall type. Can MpCCI get correct pressure loading from two sides of the shell?

**Answer:**
For such thin wall, FLUENT model will create a shadow wall. For the coupling if one side is selected MpCCI will additionally get the forces on the shadow wall and provide the forces on two sides of the shell. This is not applied to pressure quantity.

#### Question:
I do a steady state FSI simulation and FLUENT could not get the nodal position from the FE code.

**Answer:**
After FLUENT has done one data exchange you should call the "steady update mesh" function manually from the MpCCI Control panel. If you are using a journal file you need to call that command:

```
(steady-update-dynamic-mesh)
```

after the data exchange occurs in order to get the new nodal position.

**Update:**
This setting is now automatically set by MpCCI from release MpCCI 4.4

#### Question:
I have installed FLUENT on Microsoft Windows and can run it. But when I command “mpcci FLUENT” or select the FLUENT version in the MpCCI GUI I receive the message:

“mpcci FLUENT: Can’t find any FLUENT installation for this architecture”.

**Answer:**
Please add the path of your FLUENT installation to your PATH environment variable.

#### Question:
I want to receive WallTemp in FLUENT but apparently FLUENT does not account for the received temperature.

**Answer:**
Check if for the coupled boundaries in FLUENT the thermal conditions are set to “temperature” and theMpCCI lib is hooked for temperature value (\ref{sec:ui-udf-hooks}).

#### Question:
How to autosave FLUENT data in compressed format?

**Answer:**
You should add the file suffix ".gz" to the file name you want to save.

- In the journal file you can provide the root name of the file to save with the following command:
  ```plaintext
  /file/autosave/root-name mpcci_job_%t.gz
  ```
- From the FLUENT UI you can set this from the menu **Solve → Calculation Activities → Edit** from the Autosave Every option. Add the suffix ".gz" to the file name.
9 J MAG

9.1 Quick Information

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Dimension</th>
<th>Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit</td>
<td>Implicit</td>
<td>X X X X X X X X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MPCCI_Arch:</th>
<th>Supported versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code platform</td>
<td>15.0</td>
</tr>
<tr>
<td>lnx4_x64:</td>
<td>X</td>
</tr>
<tr>
<td>windows_x64:</td>
<td>X</td>
</tr>
</tbody>
</table>

9.1.2 Supported Platforms and Versions

The following versions of J MAG are supported by MpCCI:

9.1.3 References

J MAG Documentation is part of the J MAG distribution.

9.1.4 Adapter Description

The J MAG-MpCCI multiphysics coupling is enabled via the JCoupled module. The J MAG-MpCCI code adapter is developed by Fraunhofer SCAI in cooperation with JSOL Corporation. It is delivered as shared library implementing the J MAG Co-Simulation API.

9.1.5 Prerequisites for a Coupled Simulation

To run a coupled simulation you need the following:

- Ordinary J MAG installation.
- JW_MpCCI token license for J MAG-MpCCI multiphysics capability.
- A J MAG code adapter license token for MpCCI.
9.1.6 Supported JMAG Modules

The following modules are supported for a co-simulation:

- FQ - Time Harmonic Magnetic (3D)
- TR - Transient Magnetic (3D)

9.2 Coupling Process

9.2.1 Model Preparation

There are some issues which you should consider while creating a JMAG model for a co-simulation study with JMAG-Designer:

**Enabling a coupled analysis** You have to activate the Two way coupled analysis option from the Coupling study properties window (Figure 1).

![JMAG coupled analysis](image)

**Figure 1: JMAG coupled analysis**

**Enabling the output of quantities** For the current list of supported quantities (> 9.2.4 Regions Step ◾️) sent by JMAG you should activate the following output from the Output control in the Show Advanced Settings (Figure 2):

- Current Density
- Joule Loss Density
- Lorentz Force Density
Export the JCF  For performing a JMAG calculation, the JCF file has to be exported. To run the solver in parallel you have to select the solver option (Figure 3) by activating the Show Advanced Settings option and activate the appropriate parallel method: shared memory or distributed memory.

Modeling temperature dependent material  In order to use the temperature or electrical conductivity quantities provided by MpCCI, the electric properties should be modified to Temperature Dependent (Fig-
Figure 4) and graph of type Conductivity vs Temperature should be created.

Figure 4: JMAG material properties
9.2.2 Models Step

In the Models step, the following options must be chosen:

**JMAG release**  Select the release of JMAG you want to use. *latest* (default) will select the latest version which is installed on your system.

**JCF file**  Select the JCF file of your JMAG project.

The MpCCI scanner uses the JMAG JCF file to extract model information.

9.2.3 Algorithm Step

For general options please refer to [V-4.5.2 Code Specific Algorithm Settings](#).

In the Algorithm step, following additional option is available:

**Solver settings**

**JMAG analysis type**  This shows the current solver used for the model. You will see the following possible solver types:

- 3D\_transient\_ (TR)
- 3D\_frequency\_ (FQ)

In case of a 3D\_transient\_ (TR) application, JMAG will use the time step and number of steps provided in the Step Control properties of the JCF file.

In case of a 3D\_frequency\_ (FQ) you should provide the number of steps you want JMAG to perform in the *Coupling duration* panel in the Common Basics settings. For each step a co-simulation will happen.

9.2.4 Regions Step

The JMAG adapter stores the quantities using a direct memory access to the JMAG solver (“Direct”). JMAG supports the following quantities for coupling:

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Dim.</th>
<th>Default Value</th>
<th>Integration Type</th>
<th>Coupling Dimension</th>
<th>Location Option</th>
<th>Send Option</th>
<th>Receive Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrentDensity</td>
<td>Vector</td>
<td>0.0 A/m²</td>
<td>flux dens.</td>
<td>Volume</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>DeltaTime</td>
<td>Scalar</td>
<td>1.0 s</td>
<td>g-min</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>ElectrCond1</td>
<td>Scalar</td>
<td>0.0 S/m</td>
<td>field</td>
<td>Volume</td>
<td>Node</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>ElectrCond3</td>
<td>Vector</td>
<td>0.0 S/m</td>
<td>field</td>
<td>Volume</td>
<td>Node</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>
ElectrCond1 provides a direction independent scalar electric conductivity distribution for the material. This is appropriate for an isotropic material.

ElectrCond3 provides the electric conductivity for an orthotropic material. The electric conductivity is distributed in X, Y, Z direction for the material according to the electric conductivity vector value.

9.2.5 Go Step

There is no special specified setting for JMAG in Go step.

9.2.5.1 Running the Computation

When the Start button is pressed, JMAG is started with the options given in the Go step. If the Stop button is pressed, a stop-file "STOP" is created and JMAG will stop the next time it checks the presence of a stop file.

On the JMAG application side the file "mpcci_<jcf_filename>.log" contains the log of the co-simulation. The log file reflects the actions and state of the solver and the MpCCI adapter.

9.2.5.2 Post-Processing

After having solved a coupled simulation the results computed on the JMAG side (".jplot") may be opened with JMAG-Designer tool. You may visualize the coupled quantities sent to the third party code.

9.3 Code-Specific MpCCI Commands

The MpCCI subcommands available for JMAG are:

Usage:
```
mpcci JMAG [-]option
```

Synopsis:
```
'mpcci JMAG' is used to get information about JMAG.
```

Options:
```
-bbgcolor Display the JMAG Designer background color.
-diff <JCF1> <JCF2> Run the scanner on two .jcf files and print the differences.
-help This screen.
```
The subcommands \texttt{diff}, \texttt{info}, \texttt{releases} and \texttt{scan} are described in \ref{sec:common-subcommands}.

\begin{verbatim}
  mpcci jmag -bgcolor
\end{verbatim}

The \texttt{bgcolor} subcommand gets the JMAG Designer background color and prints it to stdout.

\begin{verbatim}
  mpcci jmag -solver <JCF file>
\end{verbatim}

The \texttt{solver} subcommand gets the used solver from the specified JCF file and prints it to stdout.

\section{9.4 Code Adapter Reference}

Within the MpCCI distribution the \texttt{"adapters"} directory contains the necessary software to connect the simulation programs to MpCCI depending on your license. The files are located within the subdirectory \texttt{"\$MpCCI\_home/codes/JMAG/adapters"}.

This subdirectory is further divided by several release subdirectories, e.g. \texttt{"11.0"}. The version directories are further divided by several architectures (e.g. \texttt{"linux32"}, \texttt{"linux64"}, \texttt{"win32"}, \texttt{"x64"}). There you find the library files of the JMAG adapter (e.g. \texttt{"libjmagmpcci.[so|dll]"}). The connection to MpCCI is established using these shared libraries which are loaded by the \texttt{JCoupled} application automatically.
10 MATLAB

10.1 Quick Information

<table>
<thead>
<tr>
<th>Physical Domains</th>
<th>System simulation, Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company name</td>
<td>MathWorks</td>
</tr>
<tr>
<td>Company homepage</td>
<td><a href="http://www.mathworks.com">www.mathworks.com</a></td>
</tr>
<tr>
<td>Support</td>
<td><a href="http://www.mathworks.com/support/">www.mathworks.com/support/</a></td>
</tr>
<tr>
<td>Tutorials</td>
<td>VII-11 Spring Mass System</td>
</tr>
</tbody>
</table>

10.1.1 Supported Coupling Methods

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Dimension</th>
<th>Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit</td>
<td>Implicit</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(V-3.1.1 Physical Domains)</td>
</tr>
</tbody>
</table>

| MATLAB | X | X | - | X | X | X | System simulation, Controller |

10.1.2 Supported Platforms and Versions

The following versions of MATLAB are supported by MpCCI:

<table>
<thead>
<tr>
<th>Code platform</th>
<th>Supported versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>R2016b</td>
<td>R2017a</td>
</tr>
<tr>
<td>lnx4_x64:</td>
<td>X</td>
</tr>
<tr>
<td>glnxa64</td>
<td></td>
</tr>
<tr>
<td>windows_x64:</td>
<td>X</td>
</tr>
<tr>
<td>win64</td>
<td></td>
</tr>
</tbody>
</table>

10.1.3 References

MATLAB Documentation is part of the MATLAB distribution or available online at www.mathworks.com/help.

10.1.4 Adapter Description

The MATLAB-MpCCI coupling is enabled by using the mex-function and is delivered as shared library. The MEX functions are called from a MATLAB script. The MATLAB-MpCCI code adapter is developed by Fraunhofer SCAI.

10.1.5 Prerequisites for a Coupled Simulation

To run a coupled simulation you need the following:

- Ordinary MATLAB installation.
- A MATLAB code adapter license token for MpCCI.
10.1.6 Supported MATLAB Modules

For now just the coupling with "*.m"-scripts is supported. However it is possible to use different MATLAB Toolboxes, which can be called from the "*.m"-Script.

10.2 Coupling Process

10.2.1 Model Preparation

As the user can create his own algorithms for the solution process, the user has to implement the MpCCI MEX function calls in the MATLAB script.

It is recommended to steer the simulation from a master "*.m"-script where all the different aspects of the simulation are managed. Let’s call it "model.m".

Using this approach the simulation stages are easier to be identified by the user. The solution process respectively the simulation must be implemented in a way which allows to exchange data with the MpCCI server at desired stages.

For a steady state simulation a stage reflects an iteration step for example. For a transient simulation a stage may be a time step or an inner iteration step from time step advancement in case of an implicit solver for example.

The user needs to create in the master "*.m"-script the solver structure to reflect the different stages according to the simulation type. For example a for-loop can be employed to divide a time interval in multiple steps and call an integration and an MpCCI data synchronization in each step.

Before starting a coupled simulation using MATLAB, the "model.m" script has to be prepared for MpCCI based on the assumption that the "model.m" has been prepared to integrate the MpCCI mex-function at the different stages.

The integration of the MpCCI mex-function is realized by inserting in the MATLAB script a comment line with the keyword $mpcci followed by a function name or variable type, for example:

```matlab
%$mpcci sync
```

Here are the different steps to follow in order to make the "model.m" script compatible for a co-simulation:

1. Declare the variables used for the coupling.
2. Insert the MpCCI co-simulation call.

10.2.1.1 MATLAB Variables Declaration

In order to let MpCCI recognize which variables should be used for the co-simulation, the user needs to declare in the master "model.m" script all the variables involved in the coupling.

The user should specify the type of each coupled variable. MpCCI provides the following supported types:

- **global**: Global variables represent information related to the code like physical time, time step size. (See 10.2.5 Regions Step IX for the list of supported global quantities)
- **ipoint**: Ipoint variables represent an integration point information. Theses variables are usually connected with a mesh interface.
- **cpoint**: Cpoint variables comprise point information which can be located in the space and basically interact with other point variables of the same dimension.
- **surface**: Surface variables represent mesh entities which contain values per elements, nodes.
- **volume**: Volume variables represent mesh entities which contain values per elements, nodes. Supports actually 2D surface volume.

Beside the declaration of the coupled variables, the user should declare the variable name used for
The iteration counter in case of a steady state solver: use the `iter` as type.

- the physical time, time step size, iteration counter in case of transient solver (explicit and implicit method): with resp. use the `time, dt, iter` as type.

- the current solution state to MpCCI. This state is used to get the convergence status of the solver and should be declared by using the `conv` type.

- the coupled job convergence status in case of an implicit coupling: use the `jobconv` as type. The user can control the status of the coupled job. A returned value greater than zero indicated that the current iteration should stop. All partners have stopped their iterations and are processing the next time step.

The variables declaration may be done at the begin of the "model.m" script for example. There is no restriction about the location of the co-simulation variables. It is recommended to group the declaration in order to better track them. Each MpCCI mex-function is explained in details in this section ▷ 10.2.2.1 Available Commands ◯.

```matlab
%$mpcci cpoint displacement,force
%$mpcci time cTime
%$mpcci dt GLOBALsyncTimeStep
%$mpcci iter cIter
%$mpcci conv cConv
%$mpcci jobconv coupledConv

c = [1.0, 1.0, 1.0]'*1.0e1;
displacement = [0.0, 0.0, 0.0]';
cTime = 0;
GLOBALsyncTimeStep = 3.1623e-02;
endTime = 1.0;
m = 10;
g = 9.81274;
```

The MATLAB-variables have to be column vectors.

MpCCI checks that
- all co-simulation variables are properly declared in the MATLAB script. Otherwise the variable is marked as "UNDECLARED" by the MpCCI scanner.
- all co-simulation variables are uniquely declared. During the scanner process the user will be informed if the variables are not unique. The user should fix the issue.

10.2.1.2 MATLAB Co-Simulation Declaration

The user has to place at the different stages an MpCCI call:
- At the end of the initialization phase of all the data, prior to the begin of the solving process the user should use the command `$mpcci init`. This call is therefore optional. This call requires that the iteration counter, time are correctly initialized too.
- During the iteration step or time stepping the user should choose where the MpCCI call should be placed: at the begin or at the end of the solving step (see ▷ V.3.4 Coupling Process ◯ for the difference in the coupling algorithm). Use the command `$mpcci sync`, or `$mpcci send`, `$mpcci recv`.
- At the end of the solution process, the user should disconnect the MpCCI server by inserting the call `$mpcci exit`. 
10.2 Coupling Process

You can check Figure 1 (part VIII) to visualize the possible stages for exchanging the data. A sample of the MATLAB script can be found at the next section > 10.2.2.2 Sample M Script <.

10.2.2 MpCCI MEX Function

In MATLAB, the coupling process is controlled via an "m" file which calls functions provided by MpCCI.

10.2.2.1 Available Commands

Each command is inserted as a comment `%` in the MATLAB script. This lets the user use the MATLAB script without co-simulation activation and test the proper run of the script.

Co-simulation variables declaration statement:

```matlab
%$mpcci global var1 var2
declare global var1 var2 are the name of the variables.
```  
The variables can be separated by a space or a comma.

```matlab
%$mpcci ipoint var1 var2
declare ipoint var1 var2 are the name of the variables.
```

The variables can be separated by a space or a comma.

```matlab
%$mpcci cpoint var1 var2
declare cpoint var1 var2 are the name of the variables.
```

The variables can be separated by a space or a comma.

If a cpoint is used as a moving sensor, the corresponding local coordinate of the sensor variable can be provided to MpCCI by associating the name of the variable var2 for example with the variable name containing the coordinates of this moving point, let’s say var2_coord.

In that case the complete declaration of var2 would look like:

```matlab
%$mpcci cpoint (var2 var2_coord) 
```

Basically you should pair the cpoint variable with its coordinates variable with parentheses.

```matlab
%$mpcci surface (var1 var1_elem) (var2 var2_elem)
declare surface var1 var1_elem var2 var2_elem are the name of the variables representing the mesh.
```

- The variable var1 contains the list of nodal ids and coordinates per row by respecting this syntax.

```
var1 = [id0,x0,y0,z0;id1,x1,x2,y2,z2;id3,x3,y3,z3;...]
```

- var1_elem, var2_elem are the name of the variable defining the element topology of the mesh var1. The variable is a matrix of size (number of elements * 9). Each row begins with the element id followed by the list of node ids defining this element.

MATLAB supports the following element types with MpCCI: linear, quadratic triangles and quadrilaterals. As the mesh may be composed of mixed elements, elements which don’t have 8 nodes should fill the rest of the row with the value -1. In that way MpCCI mex-function can recognize the element type.

```
var1_elem = [id0,n0,n1,n2,-1,-1,-1,-1;id1,n1,n4,n6,n8,-1,-1,-1,-1;...]
```

To store the quantities exchanged by the co-simulation based on this mesh, MpCCI mex-function can be using and expecting the declaration of following variable respecting this syntax:
The variable name is composed of the name of the mesh (\texttt{var1}) concatenated with an underscore and the name of the MpCCI quantity \texttt{quantityname}.

- The variable name must be in lower case.
- The size of the variable is for example equal to (number of nodes * 3) for a nodal vector quantity.
- Each mesh quantity variable must be correctly initialized.

For the following quantities \texttt{WallForce} and \texttt{RelWallForce} the force vector field may be computed automatically by MpCCI if MATLAB provides the pressure variable value using the naming convention: \texttt{<mesh\_prefix>\_pressure}. Otherwise MpCCI expects to find the variable \texttt{<mesh\_prefix>\_wallforce} or \texttt{<mesh\_prefix>\_relwallforce}.

\%$\texttt{mpcci time var1}$
Declare \texttt{var1} as \texttt{time} variable.
This corresponds to current simulation time. The value can not decrease. By a static simulation the value has to be equal -1 or not declared. If the variable is not declared MpCCI will set the value -1 for the data exchange call.

\%$\texttt{mpcci dt var1}$
Declare \texttt{var1} as time step size variable of the simulation.
The value can be set to -1 if not used, for e.g. a steady state simulation, or not declared. If the variable is not declared MpCCI will set the value 0 for the data exchange call.

\%$\texttt{mpcci iter var1}$
Declare \texttt{var1} as the iteration variable.
This defines the iteration number for a transient implicit simulation or for a steady state simulation. The value must be -1 for a transient explicit co-simulation or not declared. If the variable is not declared MpCCI will set the value -1 for the data exchange call. Depending of the coupling scheme selected in the MpCCI GUI (\texttt{Go step}), this value will be automatically ignored if Explicit-Transient is selected.

\%$\texttt{mpcci conv var1}$
Declare \texttt{var1} as the convergence variable.
This variable can be assigned with the following value:

- 1: notify a divergence status.
- 2: notify a stop status. The code can not continue the next iteration for the implicit coupling. This will force the partner code to proceed to the next time step.
- 3: notify a convergence status. The \texttt{MATLAB} solution has converged.
- 4: notify the continue status. \texttt{MATLAB} wants to continue the iteration.

If the variable is not declared \texttt{MpCCI} will set the value 0 (invalid state) for the data exchange call.

Co-simulation control statement:

\%$\texttt{mpcci init}$
Declare \texttt{MATLAB} connection with \texttt{MpCCI}.

\%$\texttt{mpcci sync}$
Declare a data exchange (send and receive) at this point.

\%$\texttt{mpcci send}$
Declare a data exchange (send only) at this point.

\%$\texttt{mpcci recv}$
Declare a data exchange (receive only) at this point.

\%$\texttt{mpcci exit}$
Disconnect from the \texttt{MpCCI} server and stop \texttt{MATLAB}. 

102 \textit{VI} \texttt{MpCCI 4.6.1-1}
%$mpcci quit
Disconnect from the MpCCI server and allow MATLAB to run.

%$mpcci error msg
Disconnect from the MpCCI server with an error message msg and stop MATLAB. This will terminate the complete co-simulation partners.

### 10.2.2.2 Sample M Script

The following example of the "*.m"-file is used to illustrate the definition of the requested information:

```matlab
clf=[1.0, 1.0, 1.0]'*1.0e1;
displacement=[0.0, 0.0, 0.0]';
GLOBALsyncTimeStep = 3.1623e-02;
endTime = 1.0;
m = 10; g = 9.81274;

% Setting control the solver type
iterative_solver = 0;
% Variable to evaluate the coupled job status
coupledConv = 0;

%$mpcci cpoint displacement,force
%$mpcci time cTime
%$mpcci dt GLOBALsyncTimeStep
%$mpcci iter cIter
%$mpcci conv cConv
%$mpcci jobconv coupledConv

if(iterative_solver == 1)
    iter_max = 10;
else
    iter_max = 0;
end

for i=0:endTime/GLOBALsyncTimeStep
    cTime = i*GLOBALsyncTimeStep;
    cConv = 2;
    % iterative coupling
    for cIter=0:iter_max
        if (iterative_solver == 1)
            if cIter == iter_max
                cConv = 2; % conv = stop signal
            else
                cConv = 4; % conv = continue signal
            end
        else
            cConv = 3; % state converged
        end
        %$mpcci sync

        force = -c.*displacement;
```

if (coupledConv > 0)
    disp('Receive stop signal: exit the iterative loop!');
    break;
end
end

if (coupledConv > 0)
    disp('Receive stop signal: exit the iterative loop!');
    break;
end
end

if (coupledConv > 0)
    disp('Receive stop signal: exit the iterative loop!');
    break;
end
end

%$mpcci exit

In the first comment we can see the definition of the coupling variables \texttt{displacement} and \texttt{force} as \texttt{cpoint}. The next four lines specify the auxiliary variables for the \texttt{MpCCI} server. This "*.m"-script demonstrates the iterative transient coupling. Therefore we can see two \texttt{for}-loops. The first loop increments the time variable and the second one implements the corrector iterations. There are two synchronization calls in the script. The first one communicates with the \texttt{MpCCI} server from the corrector iteration. The second synchronization call sets the convergence flag to 3, which means that the iterative calculation in \texttt{MATLAB} is finished and the last communication for the current time is requested. The comment in the last line stops the communication with the \texttt{MpCCI} server.

Before \texttt{MATLAB} starts \texttt{MpCCI} will create a new \texttt{MATLAB} script file and replace each \texttt{MpCCI} control statement with the real \texttt{mex} function call. Below is an example of the patched \texttt{MATLAB} script.

```matlab
addpath('/home/software/mpcci/codes/MATLAB/adapters/R2010b/glnxa64');
c=[1.0, 1.0, 1.0]'*1.0e1;
displacement=[0.0, 0.0, 0.0]';
GLOBALsyncTimeStep = 3.1623e-02;
endTime = 1.0;
m = 10; g = 9.81274;
%
Setting control the solver type
iterative_solver = 0;
%
Variable to evaluate the coupled job status
coupledConv = 0;

%$mpcci cpoint displacement,force
%$mpcci time cTime
%$mpcci dt GLOBALsyncTimeStep
%$mpcci iter cIter
%$mpcci conv cConv
%$mpcci jobconv coupledConv

if(iterative_solver == 1)
    iter_max = 10;
else
    iter_max = 0;
end

for i=0:endTime/GLOBALsyncTimeStep
    cTime = i*GLOBALsyncTimeStep;
    cConv = 2;
    %Iterative coupling
    for cIter=0:iter_max
```

104 VI MpCCI 4.6.1-1
if (iterative_solver == 1)
    if cIter == iter_max
        cConv = 2; % conv = stop signal
    else
        cConv = 4; % conv = continue signal
    end
else
    cConv = 3; % state converged
end
mpcci('SYNC', cTime, GLOBALsyncTimeStep, cIter, cConv, 'coupledConv');

force = -c.*displacement;

if (coupledConv > 0)
    disp('Receive stop signal: exit the iterative loop!');
    break;
end
end
mpcci('EXIT');
10.2.3 Models Step

In the Models step, the following options must be chosen:

**MATLAB release**  Select the release of MATLAB you want to use. latest (default) will select the latest version which is installed on your system.

**MATLAB file**  Select the MATLAB m file of your MATLAB project. The MpCCI scanner uses the MATLAB m file to extract model information.

**Unit system**  Select the unit system which was used in MATLAB (see ◀ 1.2 Unit Systems ◀).

10.2.4 Algorithm Step

For general options please refer to ◀ V-4.5.2 Code Specific Algorithm Settings ◀. There is no special specified setting for MATLAB.
### 10.2.5 Regions Step

The **MATLAB** adapter stores the quantities using a direct memory access to the **MATLAB** solver (“Direct”). The user has to use **MATLAB**-array with appropriate dimension for the coupling. If e.g. a force is received in **MATLAB**, the adapter expects a vector of dimension 3. All the vectors must be column vectors. **MATLAB** supports the following quantities for coupling:

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Dim.</th>
<th>Default Value</th>
<th>Integration Type</th>
<th>Coupling Dimension</th>
<th>Location</th>
<th>Send Option</th>
<th>Receive Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbsPressure</td>
<td>Scalar</td>
<td>0.0 N/m²</td>
<td>flux dens.</td>
<td>Face</td>
<td>Code, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>Acceleration</td>
<td>Vector</td>
<td>0.0 m/s²</td>
<td>field</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>AngularAccel</td>
<td>Vector</td>
<td>0.0 rad/s²</td>
<td>field</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>AngularCoord</td>
<td>Quaternion</td>
<td>0.0</td>
<td>mesh coordinate</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>AngularVel</td>
<td>Vector</td>
<td>0.0 rad/s</td>
<td>field</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>BodyForce</td>
<td>Vector</td>
<td>0.0 N/m³</td>
<td>flux dens.</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>CurrentDensity</td>
<td>Vector</td>
<td>0.0 A/m²</td>
<td>flux dens.</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>DeltaTime</td>
<td>Scalar</td>
<td>1.0 s</td>
<td>g-min</td>
<td>Global</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ElectrCond1</td>
<td>Scalar</td>
<td>0.0 S/m</td>
<td>field</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ElectrCond3</td>
<td>Vector</td>
<td>0.0 S/m</td>
<td>field</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ElectrCondX</td>
<td>Scalar</td>
<td>0.0 S/m</td>
<td>field</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ElectrCondY</td>
<td>Scalar</td>
<td>0.0 S/m</td>
<td>field</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ElectrCondZ</td>
<td>Scalar</td>
<td>0.0 S/m</td>
<td>field</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ElectricPot</td>
<td>Scalar</td>
<td>0.0 V</td>
<td>field</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ElectrRes1</td>
<td>Scalar</td>
<td>0.0 ohm m</td>
<td>field</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ElectrRes3</td>
<td>Vector</td>
<td>0.0 ohm m</td>
<td>field</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ElectrResX</td>
<td>Scalar</td>
<td>0.0 ohm m</td>
<td>field</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ElectrResY</td>
<td>Scalar</td>
<td>0.0 ohm m</td>
<td>field</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ElectrResZ</td>
<td>Scalar</td>
<td>0.0 ohm m</td>
<td>field</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>FilmTemp</td>
<td>Scalar</td>
<td>300.0 K</td>
<td>field</td>
<td>Face</td>
<td>Code, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>Force</td>
<td>Vector</td>
<td>0.0 N</td>
<td>flux integral</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>JouleHeat</td>
<td>Scalar</td>
<td>0.0 W/m³</td>
<td>flux dens.</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>LorentzForce</td>
<td>Vector</td>
<td>0.0 N/m³</td>
<td>flux dens.</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>MagneticFlux</td>
<td>Vector</td>
<td>0.0 T</td>
<td>flux dens.</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>MassFlowRate</td>
<td>Scalar</td>
<td>0.0 kg/s</td>
<td>flux integral</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>Quantity</td>
<td>Dim.</td>
<td>Value</td>
<td>Integration Type</td>
<td>Coupling Dimension</td>
<td>Location</td>
<td>Send Option</td>
<td>Receive Option</td>
</tr>
<tr>
<td>------------------</td>
<td>------------</td>
<td>-------------</td>
<td>------------------------</td>
<td>--------------------</td>
<td>--------------</td>
<td>-------------</td>
<td>---------------</td>
</tr>
<tr>
<td>MassFluxRate</td>
<td>Scalar</td>
<td>0.0 kg/m²s</td>
<td>flux dens.</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>NPosition</td>
<td>Vector</td>
<td>0.0 m</td>
<td>mesh coordinate Point</td>
<td>Code</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>PointPosition</td>
<td>Vector</td>
<td>0.0 m</td>
<td>mesh coordinate Point</td>
<td>Code</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>RealFlag</td>
<td>Scalar</td>
<td>0.0</td>
<td>g-max Global</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>SpecificHeat</td>
<td>Scalar</td>
<td>1.0 J/kg K</td>
<td>field Volume</td>
<td>Direct</td>
<td>Direct</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>Temperature</td>
<td>Scalar</td>
<td>300.0 K</td>
<td>field Point Volume</td>
<td>Direct</td>
<td>Direct</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ThermCond1</td>
<td>Scalar</td>
<td>0.0 W/m K</td>
<td>field Volume</td>
<td>Direct</td>
<td>Direct</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ThermCond3</td>
<td>Vector</td>
<td>0.0 W/m K</td>
<td>field Volume</td>
<td>Direct</td>
<td>Direct</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ThermCondX</td>
<td>Scalar</td>
<td>0.0 W/m K</td>
<td>field Volume</td>
<td>Direct</td>
<td>Direct</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ThermCondY</td>
<td>Scalar</td>
<td>0.0 W/m K</td>
<td>field Volume</td>
<td>Direct</td>
<td>Direct</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ThermCondZ</td>
<td>Scalar</td>
<td>0.0 W/m K</td>
<td>field Volume</td>
<td>Direct</td>
<td>Direct</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>Torque</td>
<td>Vector</td>
<td>0.0 N m</td>
<td>flux integral Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>TotalPressure</td>
<td>Scalar</td>
<td>0.0 N/m²</td>
<td>flux dens.</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>Velocity</td>
<td>Vector</td>
<td>0.0 m/s</td>
<td>field Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>VelocityMagnitude</td>
<td>Scalar</td>
<td>0.0 m/s</td>
<td>field Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>WallForce</td>
<td>Vector</td>
<td>0.0 N</td>
<td>flux integral Face</td>
<td>Code, Element</td>
<td>Direct</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>WallHTCoeff</td>
<td>Scalar</td>
<td>0.0 W/m² K</td>
<td>field Face</td>
<td>Code, Element</td>
<td>Direct</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>WallTemp</td>
<td>Scalar</td>
<td>300.0 K</td>
<td>field Face</td>
<td>Code, Element</td>
<td>Direct</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>
10.2.6 Go Step

MATLAB offers nothing but the common coupling configuration in the Go step (see Figure 2).

Define function call definition method

- none: MpCCI is just processing the master script and the MpCCI instructions in the m-file.
- nested: MpCCI uses the master script as the main and parent function. All other functions referenced from the master script will be included as nested function in a single MATLAB script.
- local: MpCCI uses the master script as function. All other functions referenced from the master script will be included as a local functions in a single MATLAB script.

10.2.6.1 Running the Computation

When the Start button is pressed, MATLAB is started with the options given in the Go step. The master script is patched for the co-simulation in a new file beginning with the mpcci, prefix followed by the master script name.

If the Stop button is pressed, a stop-file "job.stop" is created and MATLAB will stop the next time it checks the presence of a stop file.

10.3 Code-Specific MpCCI Commands

The MpCCI subcommands available for MATLAB are:

Usage:
mpcci MATLAB [-]option

Synopsis:
'mpcci MATLAB' is used to get information about MATLAB.

Options:
-files <main.m>
List all files involved starting with the main.m file.

-help
This screen.

-info
List verbose information about all MATLAB releases.

-join <main.m>
Join all files involved starting with the main.m file into a single .m file.

-releases
List all MATLAB releases which MpCCI can find.

-scan <main.m>
Run the scanner on the MATLAB script file and create a scanner output file.

The subcommands \textbf{info}, \textbf{releases} and \textbf{scan} are described in \S 1.1 Common MpCCI Subcommands for Simulation Codes.

\texttt{mpcci matlab -join <main.m>}

The \texttt{join} subcommand joins all files involved in the MATLAB run into a single ".m" file. It starts with the given main ".m" file.

### 10.4 Code Adapter Description

Within the \texttt{MpCCI} distribution the "adapters" directory contains the necessary software to connect the simulation programs to MpCCI depending on your license. The files are located within the subdirectory "\texttt{<MpCCI\_home>/codes/MATLAB/adapters}".

This subdirectory is further divided by several release subdirectories, e.g. "R2010b". The version directories are further divided by several architectures (e.g. "glnx64"). There you find the library files of the MATLAB adapter (e.g. "mpcci.[mexa64]").

With MATLAB the user can create his own algorithms for the solution process via an m-file script, therefore all coupling algorithms are possible.
11 MSC Adams

11.1 Quick Information

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Dimension</th>
<th>Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit</td>
<td>Implicit</td>
<td>(&lt; V-3.1.1 Physical Domains &lt;)</td>
</tr>
</tbody>
</table>

MSC Adams

11.1.1 Supported Coupling Methods

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Dimension</th>
<th>Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit</td>
<td>Implicit</td>
<td>mbdynamics</td>
</tr>
</tbody>
</table>

11.1.2 Supported Platforms and Versions

MSC Adams is supported on the following platforms:

<table>
<thead>
<tr>
<th>Code platform</th>
<th>Supported versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>win64:</td>
<td>X X X X X X X X X X X</td>
</tr>
</tbody>
</table>

For details on the platforms supported by MSC Adams see also the Platform Support page at www.mscsoftware.com/support/platform-support.

The template products MSC Adams/Car and MSC Adams/Chassis are supported.

11.1.3 References

MSC Adams Documentation The MSC Adams documentation is part of your MSC Adams installation. The most important parts for the co-simulation is the description of the MSC Adams/Solver.

11.1.4 Adapter Description

The code adapter for MSC Adams is developed and distributed by the Fraunhofer SCAI.

11.1.5 Prerequisites for a Coupled Simulation

To run a coupled simulation you need the following:

- Ordinary MSC Adams installation.
- Enough license tokens.
• Compiler, which is supported by MSC Adams, if you use MSC Adams/Car or MSC Adams/Chassis. For a list of required Fortran and C/C++ compilers, refer to the Hardware and Software specifications in the installation guide of your MSC Adams installation. Installation guides can be found at simcompanion.mscsoftware.com.

11.2 Coupling Process

Please read also ▷ IV-2 Setting up a Coupled Simulation◁.

11.2.1 Model Preparation

The MSC Adams model can be prepared with MSC Adams/View or as an input file. Please consider the following advice:

• The model can be defined in any of the consistent systems of units supported by MpCCI (see ▷ 1.2 Unit Systems◁). The basic unit system must be given in the Models step of the MpCCI GUI. Units of single quantities can be set in the Regions step.

• Please verify the simulation mode used by MSC Adams (e.g. STATICS, DYNAMICS) to be appropriate for the co-simulation partner.

• The MSC Adams model must contain a definition of coupling components. This can be one of the following elements:
  – GFORCE
  – MARKER
  – MOTION
  – VARIABLE

  The element which serves as coupling component can be selected in the Regions step of the MpCCI GUI. Also multiple elements can be selected.

• Code coupling can only run in one single step of a simulation.

⚠️ It is recommended to create a complete MSC Adams model first and test it separately without co-simulation. The quantities which will be transferred by the partner code can be simulated by appropriate loads or boundary conditions.

11.2.2 Dynamic or Kinematic Simulation

For a dynamic co-simulation the simulation mode on the MSC Adams side must be also set to DYNAMICS and the coupling scheme must be Explicit-Transient (see ▷ 11.2.7 Algorithm Step◁). The step sizes (e.g. HMAX) or any other options of the MSC Adams solver are not changed by MpCCI. MpCCI provides the data from the co-simulation to the MSC Adams/Solver as it is requested by the calls of the corresponding subroutines. The communication time points are not controlled by MpCCI. However it is possible to force MSC Adams to match certain time points. The option Synchronized history buffer update in conjunction with semi-implicit mode or partial derivatives support can be used for this purpose (see ▷ 11.2.7 Algorithm Step◁, ▷ 11.2.8.1 Semi-implicit Mode◁ and ▷ 11.2.8.2 Partial Derivatives in MSC Adams◁ for details).

Please be sure, that the co-simulation partner also uses a dynamic simulation or at least provide time values for the sent data.

The time steps of the co-simulation partners differ in general. To match the data values between requested and provided time points the server interpolates the quantities in time.

MSC Adams/Car and MSC Adams/Chassis simulations usually consist of static initialization analyses, followed by one or more dynamic or kinematic analyses. If Explicit is selected as the coupling scheme,
**11.2.3 Static Simulation**

For a static co-simulation the simulation mode on the **MSC Adams** side must be also set to **STATICS**. The setup of a quasi-static simulation corresponds to the setup of a dynamic simulation (see \( \ref{11.2.2 Dynamic or Kinematic Simulation} \)). All the parameters which you can use for the dynamic mode are also available. **MSC Adams/Car** and **MSC Adams/Chassis** simulations usually consist of static initialization analyses, followed by one or more dynamic or kinematic analyses. In steady-state simulations, **MpCCI** will only exchange data during the static analyses and quit the co-simulation before the dynamic or kinematic steps start, see also \( \ref{11.2.4 Multiple Statements} \).

**11.2.4 Multiple Statements**

You can use multiple **SIMULATE** statements in your **MSC Adams** simulation. In that case please ensure that the tags used for the exchanged quantities are unique. E.g. it is not possible to go back in time during a co-simulation.

If you switch between static and dynamic simulation (e.g. to compute a static equilibrium), please note that for an explicit co-simulation the data exchange is done only once for a time point. In this case the static equilibrium is calculated without co-simulation.

**11.2.5 MSC Adams Template Products**

**11.2.5.1 MSC Adams/Car**

The preparation of the co-simulation with **MSC Adams/Car** usually includes following steps:

- Creation of ".acf", ".adm" and ".xml" files from the **MSC Adams/Car** model. This can be done e.g. by running the desired analysis without co-simulation using the **Simulate** tab from within the **MSC Adams/Car** GUI.
- Adjustment of the model for the coupling (e.g. deactivation of some elements, introduction of new elements which serve as interfaces for the co-simulation or for the model consistency).
- Configuration of the co-simulation (see \( \ref{11.2.8 Regions Step} \))
- Selection of the co-simulation scheme. **MpCCI** interprets events defined by **CONTROL/FUNC=USER(...)** in the ".acf" file (see **MSC Adams/Chassis** documentation) as static co-simulation. Statements started using the [EventRunAll, EventRunFor, EventRunNext and EventRunUntil] command operate on a list of events read from an ".xml" file. These events consist of a set of dynamic maneuvers, possibly preceded by a static analysis.

⚠️ Please note, that for the static co-simulation the transient analysis, if included in the same ".acf" file, will not be treated right by **MpCCI** (e.g. an error arises). For the transient co-simulation there will be no transfer in the static events. It means, that the value of the co-simulation element keeps constant. This may cause difficulties with the convergence of the static solvers. So please be careful by setting up the co-simulation for large models.

**11.2.5.2 MSC Adams/Chassis**

The preparation of the co-simulation with **MSC Adams/Chassis** usually includes following steps:
• Creation of ".acf" and ".adm" files, and possibly of an ".xml" file from the MSC Adams/Chassis model. This can be done with the `build_model` command from the MSC Adams/Chassis GUI.

• Adjustment of the model for the coupling (e.g. deactivation of some elements, introduction of new elements which serve as interfaces for the co-simulation or for the model consistency).

• Configuration of the co-simulation (see > 11.2.8 Regions Step <)

• Selection of the co-simulation scheme. MpCCI interprets events defined by `CONTROL/FUNC=USER(...) in the ".acf" file (see MSC Adams/Chassis documentation) as static co-simulation. Events started by `SIMULATE` will lead to a transient co-simulation. Statements started using the `EventRunAll` command run a list of events read from an ".xml" file. These events consist of a set of dynamic maneuvers, possibly preceded by a static analysis.

⚠️ Please note, that for the static co-simulation the transient analysis, if included in the same ".acf" file, will not be treated right by MpCCI (e.g. an error arises). For the transient co-simulation there will be no transfer in the static events. It means, that the value of the co-simulation element keeps constant. This may cause difficulties with the convergence of the static solvers. So please be careful by setting up the co-simulation for large models.

### 11.2.6 Models Step

![Select code to couple](select_code.png)

Figure 1: MSC Adams options in the Models step

In the Models step, the following options must be chosen:

**Adams release** Select the MSC Adams release you wish to use. Only supported releases installed on your system are listed. The selection `latest` always refers to the latest supported version (default). The release should match the input file.

**Adams product type** This can be set to `solver` or any of the supported MSC Adams template products e.g. `acar` for MSC Adams/car.

**Adams input file** Select the MSC Adams input file ".adm" or ".acf". In the case you select an ".adm" file MpCCI assumes that the command file ".acf" has the same name. If you select an ".acf" MpCCI reads the ".adm" defined in the ".acf" file.

**Unit system** Select the unit system which was used in MSC Adams (see > 1.2 Unit Systems <). Default is set to `variable` for the unit system with `mm` for the grid length unit.
11.2.7 Algorithm Step

In the Algorithm step, some additional options are available in the code specific section. For general options please refer to V-4.5.2 Code Specific Algorithm Settings.<br><br>Following options can additionally be set in Solver settings (cf. Figure 2):

**Use semi-implicit interpolation** Check this option to use the semi-implicit interpolation. In this case MSC Adams locally mirrors the sent and received data. This saved information is then used to provide a local approximation (for the current co-simulation state) of the received quantities with respect to the sent quantities. Basically this feature aids the explicit co-simulation. In this case MSC Adams can update the received quantities for each iteration utilizing the available approximation. See 11.2.8.1 Semi-implicit Mode and VII-11 Spring Mass System for more details.

**Relaxation for the interpolation coefficients** For this option the values between 0.0 and 1.0 can be used. The adapter considers this parameter to relax the calculated coefficients (used with semi-implicit interpolation or partial derivatives). New values are then calculated from the actual approximation multiplied with the relaxation parameter and the old approximation multiplied with 1.0 minus the relaxation parameter.

**Number of saved states** The number of data points saved by the adapter to calculate the approximations for the semi-implicit mode or partial derivatives.

**Communication delay of the co-simulation** This parameter is used for the approximation of the semi-implicit mode or partial derivatives. This is the communication offset, introduced by the co-simulation. The reaction on the data, sent at some time points is received by the next communication. This is the general case for the parallel explicit co-simulation. For the implicit co-simulation the offset is 0 as the reaction is received at the same time point. Due to technical reasons, the offset by the co-simulation with the Abaqus adapter is to be set to 2. These are the communication scenarios which are supported by the MpCCI CouplingEnvironment for now. Wrong setting of this parameter results in errors by the calculation of approximation for the semi-implicit and partial derivatives modes.

**Synchronized history buffer update** Check this parameter for the MSC Adams adapter to update the saved states only at certain time points. This can be used e.g. if the co-simulation partner takes different time step sizes for the simulation. In this case, data provided as reaction and saved outgoing quantities have to be adjusted.

**Time step for buffer update** This option sets a constant step size for the updates. If this value is negative, the adapter tries to use a Time-Step-Size control variable (see 11.2.8 Regions Step).<br><br>**Initial time for saving states** With this option the adapter can skip some time interval at the beginning of the co-simulation and starts to save data after the specified time point is reached. This can help to avoid approximation errors for the semi-implicit mode or partial derivatives due to inconsistent initial conditions in the models.

**Use constant mass** Check this option to allow the calculation of the relation between incoming quantities (e.g. a force) to the outgoing acceleration or angular acceleration only at the start of the co-simulation. This option can help to determine the approximation coefficients with respect to acceleration very precisely. However, only if the initial values of the incoming quantities are close to zero or negligible, the approximation with respect to the other parameters (e.g. position) will still change with the co-simulation progress. Please see VII-11 Spring Mass System for more details.

**Value of the mass coefficient** Set the value of the mass coefficient, that will be used to relate incoming forces to outgoing accelerations or angular accelerations.

**Provide partials to MSC Adams** If this option is checked, MSC Adams will provide the partial derivatives of the received quantities with respect to the sent quantities. This helps the process of solving the nonlinear equations in the corrector iterations. See 11.2.8.2 Partial Derivatives
in MSC Adams and VII-11 Spring Mass System for more details.

**Minimum value for derivatives**  This is the minimum (negative!) value which can be used for the approximation of the partial derivatives. The numerical approximation of the value in the MSC Adams adapter will be restricted by this parameter. Values of the partial derivatives which are greater than 0.0 result in the interior instability of the MSC Adams model (for the applications, which utilize this option like e.g. controller design, iterative coupling must be used). Large negative values of the partial derivatives define very stiff components (e.g. GFORCE) and increase the numerical effort for the solution of the system. Please see VII-11 Spring Mass System for more details.

**Maximum value for derivatives**  This is the maximum (negative!) value which can be used for the approximation of the partial derivatives (see the option Minimum value for derivatives above for more details).

**Initial value for derivatives**  This parameter provides the initial value for the partial derivatives. This value is used before the MSC Adams adapter has enough data to make an approximation.

![Image](image)

Figure 2: Additional MSC Adams options in the Algorithm step for Explicit coupling scheme in transient studies

For Implicit coupling scheme following option is available in the code specific section:

**Analysis**

**Analysis type**  Select Transient.

⚠️ The Implicit option from Define the coupling scheme in Common Basics - Coupling analysis section should be selected.
MSC Adams uses a Newton-type solver and exchanges data in every iteration until the Maximum number of coupling step iterations is reached. Since MSC Adams adaptively changes its time step size, it is possible that Adams does several time steps before it reaches the next coupling point. In this case, MSC Adams only exchanges data in the last time step before the next coupling point and keeps the exchanged data constant in the preceding time steps.

11.2.8 Regions Step

MSC Adams supports the following quantities for coupling:

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Dim.</th>
<th>Default Value</th>
<th>Integration Type</th>
<th>Coupling Dimension</th>
<th>Location</th>
<th>Send Option</th>
<th>Receive Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acceleration</td>
<td>Vector</td>
<td>0.0 m/s²</td>
<td>field</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td>Usermemory</td>
</tr>
<tr>
<td>Angular Acceleration</td>
<td>Vector</td>
<td>0.0 rad/s²</td>
<td>field</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td>Usermemory</td>
</tr>
<tr>
<td>Angular Coordinate</td>
<td>Quaternion</td>
<td>0.0</td>
<td>mesh coordinate</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td>Usermemory</td>
</tr>
<tr>
<td>Angular Velocity</td>
<td>Vector</td>
<td>0.0 rad/s</td>
<td>field</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td>Usermemory</td>
</tr>
<tr>
<td>Delta Time</td>
<td>Scalar</td>
<td>1.0 s</td>
<td>g-min</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td>Usermemory</td>
</tr>
<tr>
<td>Force</td>
<td>Vector</td>
<td>0.0 N</td>
<td>flux integral</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td>Usermemory</td>
</tr>
<tr>
<td>Point Position</td>
<td>Vector</td>
<td>0.0 m</td>
<td>mesh coordinate</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td>Usermemory</td>
</tr>
<tr>
<td>Real Flag</td>
<td>Scalar</td>
<td>0.0</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td>Usermemory</td>
</tr>
<tr>
<td>Torque</td>
<td>Vector</td>
<td>0.0 N m</td>
<td>flux integral</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td>Usermemory</td>
</tr>
<tr>
<td>Velocity</td>
<td>Vector</td>
<td>0.0 m/s</td>
<td>field</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td>Usermemory</td>
</tr>
</tbody>
</table>

In this step user can select the elements of the MSC Adams-model and set the quantities to be exchanged by the co-simulation. Please note, that appropriate elements must be selected for certain quantities. E.g. the GFORCE element cannot receive any kinematic quantities like position or velocity. The following table shows the possible combinations between elements and quantities:

<table>
<thead>
<tr>
<th>Element</th>
<th>Receive</th>
<th>Send</th>
</tr>
</thead>
<tbody>
<tr>
<td>GFORCE</td>
<td>Force, Torque</td>
<td>PointPosition, Velocity, Acceleration,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Angular-Coordinates, -Velocity, -Acceleration</td>
</tr>
<tr>
<td>MARKER</td>
<td>Acceleration, Angular Acceleration</td>
<td>PointPosition, Velocity, Acceleration,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Angular-Coordinates, -Velocity, -Acceleration</td>
</tr>
<tr>
<td>MOTION</td>
<td>Value of the Variable (one-dimensional!)</td>
<td>Value for the Variable (one-dimensional!)</td>
</tr>
<tr>
<td>VARIABLE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In the case of GFORCE element the outgoing quantities are requested from the MARKER. Furthermore the time step size to control the simulation process can be received by MSC Adams. For this purpose an additional element [Time-Step-Size] is provided in the variables list, which can only receive the time step size.MpCCI uses this value to control calculation of partial derivatives (see 11.2.7 Algorithm Step<$>). If the user requires time step size to be sent by MSC Adams, the operation is ignored.
11.2.8.1 Semi-implicit Mode

The data exchange between MSC Adams and MpCCI server is done for every single time step in case of the dynamic explicit simulation or every iteration in case of a static simulation or dynamic implicit simulation. However the internal solution algorithms are in general iterative. As consequence, there are variations of the local state for each iteration at a single time point. These changes are not regarded in the case of dynamic explicit co-simulation. To improve this, the semi-implicit approach saves time history for the exchanged quantities. Those values are used to provide an approximation of the received quantities (e.g. force) with respect to the outgoing quantities (e.g. position). With this approximation it is then possible to consider the internal state changes also in case of the explicit dynamic co-simulation.

The user has to set an additional option (see § 11.2.7 Algorithm Step <) to allow the adapter to use the semi-implicit mode. Otherwise the incoming quantities stay constant for the different iterations of a single time point. The concept of the semi-implicit mode is very similar to the usage of the SYSFNC in MSC Adams.

The only elements, which may use this option are GFORCE and MOTION. However because of the MOTION statement introduces a constraint, it is not recommended to use the SYSFNC in this statement (see MSC Adams documentation for more details). That’s why the GFORCE element is the only one which uses the semi-implicit mode.

The relations between outgoing and incoming quantities can be adjusted in a very detailed manner. It is possible to introduce a dependency between any MARKER in the MSC Adams model and any GFORCE element. For this purpose the user can

- Copy the desired MARKER in the components selection list (press right mouse button and select copy).
- Introduce a mesh on which the MARKER sends some information (e.g. velocity).
- Add the copy of the MARKER to the mesh with the GFORCE.

Now the MSC Adams adapter knows that there is a relation between outgoing velocity on the MARKER and the incoming values of the GFORCE element. If the MARKER is not copied to the mesh with the GFORCE element, no dependency is defined. In this case the velocity is just sent to the partner code.

It is also possible to copy the GFORCE element itself and define multiple meshes with the same GFORCE element. The MSC Adams adapter collects all these definitions to a single coupling element. It is e.g. possible to copy the GFORCE and introduce two meshes. On the first one the GFORCE element will receive e.g. a force. On the second mesh the GFORCE element will send the acceleration (of the I-MARKER). This is necessary if the co-simulation partner provides two different elements for those quantities (e.g. two variables in MATLAB, see § VII-11 Spring Mass System <).

11.2.8.2 Partial Derivatives in MSC Adams

In MSC Adams the predictor-corrector-approach (see MSC Adams documentation) is implemented to solve the equation of motion. In corrector iterations a system of nonlinear equations is considered. For this purpose MSC Adams uses e.g. the Newton method. This algorithm utilizes partial derivatives with respect to generalized coordinates for solving nonlinear equations.

The MSC Adams adapter can provide the partial derivatives of the incoming quantities with respect to the outgoing quantities. To allow this the user must check the option Provide partials ((semi-)implicit mode only).

To provide the partial derivatives the MSC Adams adapter will store the incoming and outgoing quantities. These saved values are then used to calculate a numerical approximation of the partial derivatives. Different parameters can be adjusted to control the calculation (see § 11.2.7 Algorithm Step <).

The partial derivatives are only provided for the quantities which relate to each other within the scope of the co-simulation. The definition of the dependencies is described in § 11.2.8.1 Semi-implicit Mode <.
Please see ▶ VII-11 Spring Mass System ◄ for more details.

### 11.2.9 Go Step

In the Go step following options are general and can be set independent of the selected coupling scheme (cf. Figure 3):

**No. of parallel threads** Number of parallel threads for the MSC Adams execution. The maximum value is 8.

**Enter a job name** This name is used to concatenate the job name, which is saved by MSC Adams. With this option one can produce different co-simulation scenarios with different names for the same *Adams* model.

**Optional MSC Adams subroutines file** Provide a subroutine list file to include with the creation of an MSC Adams customized solver code. This file with suffix .lst contains the absolute pathname to the file to include per line. This could be a FORTRAN source file (".f") or an object file (".o" or ".obj").

![Figure 3: General MSC Adams options in the Go step](image)

For a Steady state analysis following options are additionally available in the Go step (cf. Figure 4):

**Choose set of allowed EQUILIBRIUM methods** Select this option to specify a set of particular methods to be used for the equilibrium solution. The selected methods are handed over to MSC Adams via the MSC_USE_ALTERNATE_SOLVERS variable. See MSC Adams documentation for a description of the methods in more detail. If no method is selected, ORIGINAL will be used as default method.

- ORIGINAL
- ORIGINAL+Krylov
- ORIGINAL+UMF
- Newton+Krylov
- Tensor-Krylov block-3
- Tensor-Krylov block-2+
- Broyden-Armijo
- Trust-Region
- Hooke-Jeeves
11.2.10 Running the Computation

With the **Start** button **MSC Adams** is started with the options given in the Go step.

In the explicit mode **MSC Adams** sends and receives data for each time step. In the implicit mode also for each iteration within the time step.

If the **Stop** button is pressed, **MSC Adams** receives a **STOP** command from the adapter and terminates the simulation.

During the **MSC Adams** run you can check for the **MSC Adams** log file. The messages of the co-simulation progress and of the **MSC Adams** simulation can be found there. In case of any issue with **MSC Adams**, please first check the **MSC Adams** log file for further information.

11.2.10.1 Batch Execution

**MSC Adams** always runs as a batch process, therefore no special settings are necessary for batch execution.

11.2.11 Post-Processing

Post-processing for the **MSC Adams** part of the results can be performed as in ordinary computations, e.g. with **MSC Adams/View**. The "<job name>.res" file can be found in the same directory as the input file.
11.3 Code-Specific MpCCI Commands

The MpCCI subcommands available for MSC Adams are:

Usage:

```
mpcci Adams [-]option
```

Synopsis:

'mpcci Adams' is used to get information about Adams.

Options:

- `-diff <ACF|ADM> <ACF|ADM>` Run the scanner on two <ACF|ADM> control files and print the differences.
- `-help` This screen.
- `-info` List verbose information about all Adams releases.
- `-products` List all available licensed Adams products.
- `-releases` List all Adams releases which MpCCI can find.
- `-scan <ACF|ADM>` Run the scanner on the Adams <ACF|ADM> control file and create a scanner output file.

The subcommands `diff`, `info`, `releases` and `scan` are described in > 1.1 Common MpCCI Subcommands for Simulation Codes.<

The subcommand `products` lists the available MSC Adams products, e.g. those for which licenses are available. The list looks like:

```
> mpcci adams -products
acar
carride
driveline
```
11.4 Code Adapter Reference

11.4.1 Data Exchange

<table>
<thead>
<tr>
<th>Coupling scheme</th>
<th>Data exchange</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit</td>
<td>at the end of the time step</td>
</tr>
<tr>
<td>Implicit</td>
<td>before each iteration and at the end of the time step</td>
</tr>
</tbody>
</table>

11.4.2 Patched Input File

Before a coupled simulation is started, MpCCI patches the MSC Adams model file. For the files "*.adm" and "*.acf" selected in the Models step, new files "mpcci_.adm" and "mpcci_.acf" are created. The "mpcci_.adm" contains the modified definition of the selected co-simulation elements, e.g. GFORCE. The original element statement is removed.

For the MOTION also the complete definition is replaced by MpCCI. Especially if only one quantity, e.g. position, is received on this element and the definition does not determine the complete set of the degree of freedom. In this case the further boundary conditions, e.g. the angular coordinates, are removed. The definition of the element then consists only of the constraints for the position. The complete MOTION statement is considered to be provided by MpCCI and therefore the angular coordinates are assumed to be zero. If this is not desired, the user can provide an additional MOTION element for the same marker, which is then selected for the co-simulation.

It is also possible to couple the VARIABLE statement in MSC Adams. One can provide the values of the VARIABLE to the partner code or receive any information. In both cases the values must be scalar. The special option is to receive the Time-Step-Size. One can assign the value to any VARIABLE for e.g. monitor purpose. However the value will be used internally by the MSC Adams adapter to adjust the communication step size.

In addition to the element statement, the patched "*.adm" file will contain some arrays which provide information requested by the adapter. This arrays are placed right after the element statements. The lines inserted by MpCCI may e.g. look as follows for the GFORCE statement:

```plaintext
adams_view_name='ActFrontLeft'
GFORCE/6
, I = 1023
, JFLOAT = 1025
, RM = 1024
,FUNCT = USER(3,6,0)
,ROUTINE = /.../libadamsmpcci.so::mpcci_c_gforce

adams_view_name='mpcci user function GFORCE-6::ActFrontLeft support array'
ARRAY/1133
, IC
, SIZE = 10
, NUMBERS = 1,7,3,6,898822979,7,1023,0,0,6
```

The settings in this block reflect the user’s choices in the MpCCI GUI. The support array contains information about outgoing and received quantities.

In the "*.acf" file the SIMULATE statement is replaced by the calls to CONSUB routine which is provided
by the adapter. The modified statement may look as follows:

```plaintext
CONTROL/FUNCTION = USER(2,3,5E-3,1E-3,1e-09,1E-3,1E-5,0.0)\
ROUTINE=/.../libadamsmpcci.so::mpcci_c_consub
```

In case **MSC Adams/Car** or **MSC Adams/Chassis** is used calls to

- EventRunAll
- EventRunNext
- EventRunFor
- EventRunUntil

are also replaced by the calls to the **MpCCI CONSUB** routine.

For **MSC Adams/Chassis**, simulations started by **CONTROL/FUNC=USER(...)** (see **MSC Adams/Chassis documentation**) will not be replaced by the **CONSUB** from **MpCCI**. But the co-simulation starts normally in the events defined by **CONTROL/FUNC=USER(...)** also in this case. However, **MpCCI** skips the transfers for those events if the coupling scheme Explicit for transient studies is selected.
12 MSC Marc

12.1 Quick Information

<table>
<thead>
<tr>
<th>Company name</th>
<th>MSC Software Corporation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company homepage</td>
<td><a href="http://www.mscsoftware.com/product/marc">www.mscsoftware.com/product/marc</a></td>
</tr>
<tr>
<td>Support</td>
<td>simcompanion.mscsoftware.com</td>
</tr>
</tbody>
</table>

Tutorials

▷ VII-2 Vortex-Induced Vibration of a Thin-Walled Structure
▷ VII-4 Elastic Flap in a Duct
▷ VII-5 Exhaust Manifold

12.1.1 Supported Coupling Methods

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Dimension</th>
<th>Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit</td>
<td>Implicit</td>
<td>FEM, SolidStructure, SolidThermal, SolidAcoustics</td>
</tr>
</tbody>
</table>

MSC Marc X - X X X X X FEM, SolidStructure, SolidThermal, SolidAcoustics

12.1.2 Supported Platforms and Versions

MSC Marc is supported on the following platforms:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>lnx4_x64:</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>lnx64:</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>windows_x64:</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>win64:</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

For more information on supported platforms and compilers see also the MSC Marc Release Guide.

12.1.3 References

MSC Marc, Volume A: Theory and User Information by MSC.Software Corporation. This part of the MSC Marc documentation contains general information on performing a simulation with MSC Marc.

MSC Marc, Volume D: User Subroutines and Special Routines This part of the MSC Marc documentation contains information on the user subroutines which are also used by the code adapter. See especially chapter 12.

MSC Marc, Volume C: Program Input for information on the COUPLING REGION option, which is used by MpcCCI to define the coupling region.

12.1.4 Adapter Description

The code adapter is provided in an object file. Similar to object files of user-defined functions these files are linked with MSC Marc before starting the computation. Therefore the appropriate compiler is needed to run a coupled simulation (see ▷ 12.1.5 Prerequisites for a Coupled Simulation).
12.1.5 Prerequisites for a Coupled Simulation

To run a coupled simulation you need the following:

- Ordinary MSC Marc installation.
- Appropriate compiler release in the user environment. As the MpCCI code adapter is based on user-defined methods which must be linked statically to the MSC Marc executable, the proper compiler is required for coupled simulations. The path to the compiler must be given in an appropriate shell variable, e.g. INTEL for the Intel compiler. MpCCI will complain if the variable is not set correctly.

It is recommended to first try to run an example which uses user subroutines to check whether the compiler is installed correctly.

For more information on compiler requirements see the list of supported platforms in the MSC Marc release guide, chapter 6.
12.2 Coupling Process

Please read also the corresponding ▷ IV-2 Setting up a Coupled Simulation ◁.

12.2.1 Model Preparation

MSC Marc does not use a fixed unit system, so any consistent unit system may be used to define the MSC Marc model.

The MSC Marc model can be defined like a normal model. For the coupled analysis, the surfaces which form the coupling region must be defined as user sets:

- sets of finite element edges or geometric curves, for 2-D surface coupling;
- sets of finite element faces or geometric surfaces, for 3-D surface coupling;
- sets of finite elements or contact bodies, for volume-based coupling in any dimension.

The sets will then be listed in the Regions step of the MpCCI GUI.

Coupling is only supported for the new input file format, so the option NEW-STYLE TABLE must be selected in the RUN JOB menu of MSC Marc Mentat (option tables in the input file). MpCCI has to insert boundary conditions into the input file and only supports the new format.

Instead of directly submitting the file from MSC Marc Mentat, simply write an input file. The analysis is started from the MpCCI GUI.

It is recommended to first start a stand-alone MSC Marc computation to check if the model is set up correctly.

12.2.2 Models Step

In the Models step the following options must be chosen:

MSC.Marc release Select the MSC Marc release you want to use. latest (default) selects the latest release which is installed on your system.

Multiple input file DDM job This option is needed for parallel runs of MSC Marc in which the domain decomposition is done in the GUI and multiple input files have been written. For single file DDM, the button must not be selected.

MSC.Marc input file Select the MSC Marc input file for the analysis.
Unit system Select the unit system which was used to define the MSC Marc model. MSC Marc has no units built into it, a self-consistent set of units should be used. In the Models step you can select from British, SI, cgs and variable (see > 1.2 Unit Systems ◄).

12.2.3 Algorithm Step

For general options please refer to > V-4.5.2 Code Specific Algorithm Settings ◄. There is no special specified setting for MSC Marc.

12.2.4 Regions Step

The following quantities are supported:

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Dim.</th>
<th>Default Value</th>
<th>Integration Type</th>
<th>Coupling Dimension</th>
<th>Location Option</th>
<th>Send Option</th>
<th>Receive Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbsPressure</td>
<td>Scalar</td>
<td>0.0 N/m²</td>
<td>flux dens.</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>DeltaTime</td>
<td>Scalar</td>
<td>1.0 s</td>
<td>g-min</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>FilmTemp</td>
<td>Scalar</td>
<td>300.0 K</td>
<td>field</td>
<td>Line, Face</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>NPosition</td>
<td>Vector</td>
<td>0.0 m</td>
<td>mesh coordinate</td>
<td>Line, Face, Volume</td>
<td>Node</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>OverPressure</td>
<td>Scalar</td>
<td>0.0 N/m²</td>
<td>flux dens.</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>RelWallForce</td>
<td>Vector</td>
<td>0.0 N</td>
<td>flux integral</td>
<td>Line, Face</td>
<td>Node</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>WallForce</td>
<td>Vector</td>
<td>0.0 N</td>
<td>flux integral</td>
<td>Line, Face</td>
<td>Node</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>WallHeatFlux</td>
<td>Scalar</td>
<td>0.0 W/m²</td>
<td>flux dens.</td>
<td>Line, Face, Element</td>
<td>Direct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WallHTCoeff</td>
<td>Scalar</td>
<td>0.0 W/m² K</td>
<td>field</td>
<td>Line, Face</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>WallTemp</td>
<td>Scalar</td>
<td>300.0 K</td>
<td>field</td>
<td>Line, Face</td>
<td>Node</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>

⚠️ If the time step size is received by MSC Marc, the option AUTO STEP must be selected in MSC Marc Mentat to enable time step adaptation in MSC Marc.

12.2.5 Go Step

In the Go step the following options can be chosen:

Enter a job name This is the name of the MSC Marc job, which is also used as base for the MSC Marc output files.

Select integer mode Select an integer mode to use which can be one of default, i4 and i8.

Optional restart file If you want to restart a previous computation, select the restart file here.

Optional post file If you want to restart a previous computation from a post file, select it here.

Optional view factor file Select a view factor file.

User Subroutine Select a file with the user subroutines.

Additional command line options If you need any further command line options for starting MSC Marc, enter them here. See also Volume C, Appendix B of the MSC Marc manual. Do not set command line options which are already set by the MpCCI GUI (e.g. -jid for the job name).

Run parallel Select this option for a parallel run. See > 12.2.6.1 Parallel Execution ◄ below.
12.2.6 Running the Computation

Instead of directly submitting the file from MSC Marc Mentat, simply write an input file. The analysis is started from the MpCCI GUI.

12.2.6.1 Parallel Execution

There is no limitation regarding parallel execution of MSC Marc in a coupled simulation. Both, automatic domain decomposition and manual decomposition are supported. The coupling region information is automatically passed to all subprocesses.

⚠️ If you use multiple input files, the option Multiple input file DDM job must be selected in the Models step.

If Run parallel is selected in the Go step of the MpCCI GUI, the following options can be selected for a parallel run of MSC Marc (cf. Figure 2):

- **No. of parallel domains**  Enter the number of parallel domains here. The number must correspond to the number of domain input files if multiple input files are used.

- **Optional 'host host ...' to be used**  Enter a list of host names for a simulation distributed on different computers.

- **Optional hostlist file**  Specify a hostfile, from which host names are extracted ▶ V-3.5.2 Hostlist File ◄.

- **Use default hostfile**  A default hostfile can be configured by setting MPCCI_HOSTLIST_FILE ▶ V-3.5.2 Hostlist File ◄.

Figure 2: MSC Marc options in the Go step
12.2.7 Post-Processing

Pressures or forces received by MSC Marc in a coupled simulation can be visualized in MSC Marc Mentat: Select External Force in the post-processing menu.
12.3 Code-Specific MpCCI Commands

The MpCCI subcommands available for MSC Marc are:

<table>
<thead>
<tr>
<th>Usage:</th>
<th>mpcci MSC.Marc [-]option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synopsis:</td>
<td>'mpcci MSC.Marc' is used to get information about MSC.Marc.</td>
</tr>
<tr>
<td>Options:</td>
<td>-align &lt;ARGS&gt; Do a coordinate transformation on all nodes of a .dat file based on a plane definition file and align the nodal coordinates for the coupling partner.</td>
</tr>
<tr>
<td></td>
<td>-help This screen.</td>
</tr>
<tr>
<td></td>
<td>-info List verbose information about all MSC.Marc releases.</td>
</tr>
<tr>
<td></td>
<td>-releases List all MSC.Marc releases which MpCCI can find.</td>
</tr>
<tr>
<td></td>
<td>-scan [-ddm] &lt;input-file&gt; Run the scanner and create a scanner output file. Specify -ddm if you have multiple input files.</td>
</tr>
</tbody>
</table>

The subcommands align, info, releases and scan are described in §1.1 Common MpCCI Subcommands for Simulation Codes.4.
12.4 Trouble Shooting, Open Issues and Known Bugs

Version:
From MSC Marc 2014.2 and later

Problem:
At start MSC Marc launch the compilation and link process to run the co-simulation. The linking process may fail with such error message:

```
ld: ACSI_MarcLib.a(Build_MarcLib.o): undefined reference to symbol '__cxa_pure_virtual@@CXXABI_1.3'
/usr/lib64/libstdc++.so.6: error adding symbols: DSO missing from command line
```

This issue usually appears when running MSC Marc on unsupported MSC Linux OS.

Workaround:
Modify the following file "include\linux64" located in MARC "install\directory\tools" directory. You will need to add the option -cxxlib to the variable SYSLIB like below:

```
SYSLIBS="-L${MPI_ROOT}/lib64 -lmpi_mt -lmppifort -lrt $OPENMP -threads -lpthread -shared-intel -cxxlib"
```

to

```
SYSLIBS="-L${MPI_ROOT}/lib64 -lmpi_mt -lmppifort -lrt $OPENMP -threads -lpthread -shared-intel -cxxlib"
```

Please check that you have set the right Intel compiler version for MSC Marc.
13 MSC NASTRAN

13.1 Quick Information

<table>
<thead>
<tr>
<th>Company name</th>
<th>MSC Software Corporation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company homepage</td>
<td><a href="http://www.mscsoftware.com/product/msc-nastran">www.mscsoftware.com/product/msc-nastran</a></td>
</tr>
<tr>
<td>Support</td>
<td>simcompanion.mscsoftware.com</td>
</tr>
<tr>
<td>Tutorials</td>
<td>▶ VII-2 Vortex-Induced Vibration of a Thin-Walled Structure ◀</td>
</tr>
<tr>
<td></td>
<td>▶ VII-3 Driven Cavity ◀</td>
</tr>
<tr>
<td></td>
<td>▶ VII-4 Elastic Flap in a Duct ◀</td>
</tr>
<tr>
<td></td>
<td>▶ VII-8 Pipe Nozzle ◀</td>
</tr>
</tbody>
</table>

13.1.1 Supported Coupling Methods

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Dimension</th>
<th>Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit</td>
<td>Implicit</td>
<td>✧</td>
</tr>
</tbody>
</table>

13.1.2 Supported Platforms and Versions

MSC NASTRAN is supported on the following platforms:

<table>
<thead>
<tr>
<th>OS</th>
<th>Code platform</th>
<th>Supported versions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>lnx4_x64:</td>
<td>X   X   X   X   X   X   X   X   X   X</td>
</tr>
<tr>
<td></td>
<td>lnx8664:</td>
<td>X   X   X   X   X   X   X   X   X   X</td>
</tr>
<tr>
<td></td>
<td>windows_x64:</td>
<td>X   X   X   X   X   X   X   X   X   X</td>
</tr>
<tr>
<td></td>
<td>win8664:</td>
<td>X   X   X   X   X   X   X   X   X   X</td>
</tr>
</tbody>
</table>

For details on the platforms supported by MSC NASTRAN see also the Platform Support page at www.mscsoftware.com/support/platform-support.

13.1.3 Adapter Description

The code adapter is provided as a dynamic library implementing the OpenFSI service.

13.1.4 Prerequisites for a Coupled Simulation

To run a coupled simulation you need the following:

- MSC NASTRAN installation including SOL400 and the necessary license token ('MD_Nonlinear', or 'NA_Nonlinear').
- Iterative coupling requires the usage of NLSTEP definition in the bulk data entries.
13.2 Coupling Process

Please read also the corresponding \textit{IV-2 Setting up a Coupled Simulation}.

13.2.1 Model Preparation

1. Create Structural model using \textit{SimXpert} (Figure 1).
   - Define appropriate properties for the structure.

2. Define Service for \textit{OpenFSI}.
   - Define CFD service under User Services \textit{OpenFSI}.
   - Provide a dummy name. This will be updated by \textit{MpCCI} before starting the computation.
   - Choose whether coupling should occur at each (Data Exchange Interval Figure 2):
     - solution time step (explicit): \textit{Time Steps}.
     - within the iteration loop (implicit): \textit{Nonlinear Iterations}.
     You can define the \textit{Frequency} of the data exchange within the time step.

3. Define \textit{OpenFSI} Boundary conditions (Figure 3).
• Applied loads to wetted surfaces under Loads OpenFSI.
• Select surfaces or element faces.
• Pick OpenFSI Service alias name.
• Wetted surfaces can point to more than one service.
• Enter Coupling Region tag name to identify the region.
• Scale factor "1" should be used.

4. Define MSC NASTRAN Job: Solution type must be “General Nonlinear Analysis (SOL400)”.
   • Define the Load Case: “Nonlinear Transient”, “Linear Static”, “Nonlinear Static”.
   • Define Load output Request for post processing of OpenFSI loads (forces).

13.2.2 Models Step

![Figure 4: MSC NASTRAN options in the Models step](image)

In the Models step the following options must be chosen:

**MSC NASTRAN release** Select the MSC NASTRAN release you want to use. `latest` (default) selects the latest release which is installed on your system.
MSC NASTRAN input file  Select the MSC NASTRAN input file for the analysis (".bdf" or ".dat" file)

Unit system  Select the unit system which was used to define the MSC NASTRAN model (see \> 1.2 Unit Systems\>).

13.2.3 Algorithm Step

For general options please refer to \> V-4.5.2 Code Specific Algorithm Settings\>. In the Algorithm step, following option is available in the code specific section in Coupling steps:

Number of inner iterations  This options appears only for implicit coupling scheme. Set up this option in order to configure the FSICTRL command. If the MSC NASTRAN .bdf file contains the keyword NLSTEP followed either by GENERAL, ADAPT or FIXED, the number of inner iterations and the time step size will be adjusted to match the settings here.

13.2.4 Regions Step

The following quantities are supported:

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Dim.</th>
<th>Default Value</th>
<th>Integration Type</th>
<th>Coupling Dimension</th>
<th>Location Option</th>
<th>Send Option</th>
<th>Receive Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeltaTime</td>
<td>Scalar</td>
<td>1.0 s</td>
<td>g-min</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>NPosition</td>
<td>Vector</td>
<td>0.0 m</td>
<td>mesh coordinate</td>
<td>Line, Face</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>RelWallForce</td>
<td>Vector</td>
<td>0.0 N</td>
<td>flux integral</td>
<td>Line, Face</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>Velocity</td>
<td>Vector</td>
<td>0.0 m/s</td>
<td>field</td>
<td>Line, Face</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>WallForce</td>
<td>Vector</td>
<td>0.0 N</td>
<td>flux integral</td>
<td>Line, Face</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>
13.2.5 Go Step

Figure 5: MSC NASTRAN options in the Go step
In the Go step the following options can be chosen:

**Do a cold start run** All ".IFPDAT", ".DBALL", ".MASTER" files will be deleted. Otherwise this will be considered as a restart job.

**Save previous copies of .f04, .f06... files** Save a previous copy of .f04, .f06... files using the MSC NASTRAN option -old=[yes/no].

**Delete database files at the job completion** Delete the database files at the job completion src=yes. The job could not be restarted if this is activated.

**Select Common Data Model for data processing** Activate the common data model processing sys444=1 for MSC NASTRAN.

**Additional command line options** Additional command line options for MSC NASTRAN can be given here, they will directly be used when MSC NASTRAN is started.

**No. of solver threads** The number of solver threads MSC NASTRAN should use for this run can be given here.

**Run parallel** Select this option for a parallel run. See \(\triangleright 13.2.6.1\) Parallel Execution below.

### 13.2.6 Running the Computation

When the [Start] button is pressed, MSC NASTRAN is started with the options given in the Go step in batch.

If the coupling scheme Implicit is selected, the FSICTRL from the ".bdf" file will be automatically set to IMPLICIT.

If the coupling scheme Explicit is selected, the FSICTRL from the ".bdf" file will be automatically set to EXPLICIT.

#### 13.2.6.1 Parallel Execution

![Figure 6: MSC NASTRAN options for parallel execution](image)

If Run parallel is selected in the Go step of the MpCCI GUI, the following options can be selected for a parallel run of MSC NASTRAN (cf. Figure 6):

- **No. of parallel processors** Enter the number of parallel processors here.
- **MPI vendor** Select the MPI vendor to use. If default is selected it will start MSC NASTRAN with your default installation setting.
- **Optional 'host host ...' to be used** Enter a list of host names for a simulation distributed on different computers.
- **Optional hostlist file** Specify a hostfile, from which host names are extracted.
- **Use default hostfile** A default hostfile can be configured by setting MPCCI_HOSTLIST_FILE \(\triangleright V-3.5.2\) Hostlist File.
13.2.6.2 Batch Execution

MSC NASTRAN is always run as a batch process, therefore no special settings are necessary for batch execution.

13.2.7 Post-Processing

Use SimXpert to post-process the results saved in a ".xdb" file.

13.3 Code-Specific MpCCI Commands

The MpCCI subcommands available for MSC NASTRAN are:

<table>
<thead>
<tr>
<th>Usage:</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpcci MSC.Nastran [-]option</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Synopsis:</th>
</tr>
</thead>
<tbody>
<tr>
<td>’mpcci MSC.Nastran’ is used to get information about MSC.Nastran.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Options:</th>
</tr>
</thead>
<tbody>
<tr>
<td>-align &lt;ARGS&gt;</td>
</tr>
<tr>
<td>Do a coordinate transformation on all nodes of a .bdf/.dat file based on a plane definition file and align the nodal coordinates for the coupling partner.</td>
</tr>
<tr>
<td>-bdfinfo [file file ..]</td>
</tr>
<tr>
<td>Extract and display coupling information on BDF files.</td>
</tr>
<tr>
<td>-help</td>
</tr>
<tr>
<td>This screen.</td>
</tr>
<tr>
<td>-info</td>
</tr>
<tr>
<td>List verbose information about all MSC.Nastran releases.</td>
</tr>
<tr>
<td>-releases</td>
</tr>
<tr>
<td>List all MSC.Nastran releases which MpCCI can find.</td>
</tr>
<tr>
<td>-scan &lt;input-file&gt;</td>
</tr>
<tr>
<td>Run the scanner and create a scanner output file.</td>
</tr>
</tbody>
</table>

The subcommands [align], [info], [releases] and [scan] are described in ▷ 1.1 Common MpCCI Subcommands for Simulation Codes ◁.

mpcci msc.nastran -bdfinfo [file file ...]

The bdfinfo subcommand extracts the coupling information on the given BDF files and prints them to stdout.
13.4 Code Adapter Reference

The MpCCI code adapter for MSC NASTRAN uses the OpenFSI Service definition to manage the data transfer. The code adapter is distributed as a dynamic library, which is located in "<MpCCI_home>/codes/MSC.Nastran/adapters/<MSCNastran_release>/<platform>/Apps".

MpCCI will set all necessary environment variable to locate the MpCCI- OpenFSI service.

MSC NASTRAN exchanges data during the time step initialization for the explicit scheme.

For implicit scheme MSC NASTRAN exchanges data during the time step initialization and during the iterative loop.

13.5 Trouble Shooting, Open Issues and Known Bugs

<table>
<thead>
<tr>
<th>Feature:</th>
<th>Face Coupling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version:</td>
<td>2010.1</td>
</tr>
<tr>
<td>Problem:</td>
<td>MSC NASTRAN could not load a model with QUAD8 elements for the wetted surface.</td>
</tr>
<tr>
<td>Workaround:</td>
<td>Replace the hexahedral element CHEXA20 to CHEXA8 which will create QUAD4 elements for the coupling.</td>
</tr>
</tbody>
</table>
14 OpenFOAM

14.1 Quick Information

<table>
<thead>
<tr>
<th></th>
<th>OpenCFD Limited</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company name</td>
<td></td>
</tr>
<tr>
<td>Company homepage</td>
<td><a href="http://www.openfoam.com">www.openfoam.com</a></td>
</tr>
<tr>
<td>Support</td>
<td><a href="http://www.openfoam.com/support/">www.openfoam.com/support/</a></td>
</tr>
<tr>
<td>Tutorials</td>
<td>▶ VII-3 Driven Cavity ◀</td>
</tr>
<tr>
<td></td>
<td>▶ VII-4 Elastic Flap in a Duct ◀</td>
</tr>
<tr>
<td></td>
<td>▶ VII-5 Exhaust Manifold ◀</td>
</tr>
<tr>
<td></td>
<td>▶ VII-9 Cube in a Duct Heater ◀</td>
</tr>
<tr>
<td></td>
<td>▶ VII-10 Y-Junction ◀</td>
</tr>
</tbody>
</table>

14.1.1 Supported Coupling Methods

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Dimension</th>
<th>Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit</td>
<td>Implicit</td>
<td></td>
</tr>
</tbody>
</table>

14.1.2 Supported Platforms and Versions

<table>
<thead>
<tr>
<th>MPCCI_ABCX:</th>
<th>Supported versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code platform</td>
<td></td>
</tr>
<tr>
<td>lnx64_x86:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

14.1.3 References

OpenFOAM User Guide is part of the OpenFOAM distribution.
OpenFOAM Programmer’s Guide is part of the OpenFOAM distribution.
OpenFOAM C++ Source Guide This is the source code documentation generated by Doxygen. It is available at www.openfoam.com/docs/.

14.1.4 Adapter Description

MpCCI uses the function object interface of OpenFOAM. This interface provides the opportunity to have the adapter library loaded by the solver and the data transfer triggered in each time step. Since MpCCI uses only OpenFOAM-intrinsic mechanisms, code-coupling works without any modification to the OpenFOAM solver.

For transient simulations with implicit coupling scheme, if constant time-stepping is chosen but the end time is not dividable by the time step size, the end time will not be reached in OpenFOAM’s calculations and it will stop calculating earlier than the given value. This might lead to co-simulation errors at the end of the coupling.
14.1.5 Prerequisites for a Coupled Simulation

To run a coupled simulation you need the following:

- Ordinary OpenFOAM installation.
- Before using MpCCI make sure the environment variable FOAM_INST_DIR is set properly. This means that FOAM_INST_DIR is the parent directory of any installed OpenFOAM version. e.g. 
  
  
  "<$FOAM_INST_DIR>/OpenFOAM-v2006"
- An MpCCI Grid Morpher license token if it will be used.
- Make sure that the appropriate environment of your OpenFOAM version is sourced.
- Make sure that the MpCCI code adapters for OpenFOAM are built as described in ▶ 14.5 Code Adapter Reference ◁.

14.2 Coupling Process

14.2.1 Model Preparation

Models can be prepared as usually. The coupling components must be defined as separate surfaces or volumes. Table 1 shows the list of surfaces defined in an OpenFOAM model, which then appear in the Coupling Step of the MpCCI GUI. MpCCI will detect the model to have a transient solution type if the OpenFOAM application contains one or more of the words pimple, piso, ico or DyM (e.g. icoFoam or pimpleDyMFoam); if the application name contains simple (e.g. simpleFoam) MpCCI assumes the solution type to be steady-state. If the OpenFOAM application name contains none of these keywords, the user has to set the coupling scheme – steady or transient – in the Go step.

If you exchange a relative pressure (e.g. RelWallForce or Overpressure), you must set the reference pressure to an appropriate value, which usually corresponds to the atmospheric pressure. (▶ 14.2.5 Go Step ◁)

For more information on relative and absolute pressures see the FSI section in ▶ V-3.1.2 Coupling Types ◁.

When receiving the wall temperature (WallTemp) in OpenFOAM set the boundary condition type of coupled walls from the file "0/T" to "fixedValue", otherwise the received wall temperature and heat flux are not taken into account, resp. not calculated for the computation.

When receiving mesh deformation (NPosition) in OpenFOAM set the boundary condition type of coupled walls from the file "0/U" to "movingWallVelocity" with value (0,0,0), otherwise the wall velocity is not taken into account, resp. not calculated for the computation.

Sample from the "0/T":

```plaintext
internalField uniform $temperature;
boundaryField
{
    couple-flap
    {
        type fixedValue;
        value uniform $temperature;
    }
    ....
}
```
All further options required for coupling can be set in the MpCCI GUI.

```
FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    object motionProperties;
}
// * * * * * * * * * * * * * * * //
5
{
    wall
    {
        type wall;
        nFaces 18640;
        startFace 296814;
    }
    outlet
    {
        type patch;
        nFaces 256;
        startFace 315454;
    }
    i3
    {
        type patch;
        nFaces 283;
        startFace 315710;
    }
    i2
    {
        type patch;
        nFaces 318;
        startFace 315993;
    }
    i1
    {
        type patch;
        nFaces 291;
        startFace 316311;
    }
}
// ****************************** //
```

Table 1: List of surfaces in the Regions step of the MpCCI GUI (right) and in OpenFOAM (left) defined in "<case directory>/constant/polymesh/boundary".

### 14.2.1.1 Setting function object

In order to have the function object loaded, MpCCI adds some lines to "<case directory>/system/controlDict" which instruct the solver to load the adapter library and to call the appropriate init- and transfer-functions.

```
libs ("libfoammpcci.so");

functions
{
    MpCCI_functionObject_adapter_for_OpenFOAM
    {
        type mpcci; // Type of functionObject
```
14.2.1.2 Deforming Meshes

In typical FSI problems, deformations are computed by the solid mechanics code and sent to OpenFOAM, i.e. OpenFOAM has to move boundaries and deform the mesh.

To make coupled simulations with deforming meshes you have to choose an OpenFOAM solver that is capable of handling dynamic meshes. Solvers which can handle deforming meshes are usually those that have a ...DyMFoam in its name. Alternatively you may write your own custom solver. Further you have to provide the file "<case directory>/constant/dynamicMeshDict" controlling how OpenFOAM performs mesh motion. Make sure to employ a motion solver operating with displacements e.g. displacementLaplacian or velocities velocityLaplacian. velocityLaplacian is not available for OpenFOAM versions below 2.4.

Here’s an example of a dynamicMeshDict:

FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    object motionProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * *

dynamicFvMesh dynamicMotionSolverFvMesh;

motionSolverLibs ("libfvMotionSolvers.so");
solver displacementLaplacian;

diffusivity inverseDistance (couple-flap);

// **************************************************************************** //

Currently it is not possible to remesh a coupling component itself during a computation.

14.2.2 Models Step

In the Models step, the following options must be chosen:

OpenFOAM option Select the OpenFOAM option from Opt, Debug.

OpenFOAM precision Select the OpenFOAM precision from DP for double precision or SP for single precision.

OpenFOAM release Select the release of OpenFOAM you want to use. The version must match your case. latest (default) will select the latest version which is installed on your system.

OpenFOAM case directory Select the case directory of your OpenFOAM model.
14.2.3 Algorithm Step

For general options please refer to \( \triangleright \) V-4.5.2 Code Specific Algorithm Settings \( \triangleright \).

In the Algorithm step, following additional option is available:

**Solver settings**

**Use alpha Courant interface**  
(Only for Implicit coupling scheme with an adaptive time step size negotiated with all codes (cf. \( \triangleright \) V-4.5.1 Common Basic Algorithm Settings \( \triangleright \)).) Check this option if the alpha Courant interface shall be used.

14.2.4 Regions Step

OpenFOAM supports the following quantities for coupling:

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Dim.</th>
<th>Default Value</th>
<th>Integration Type</th>
<th>Coupling Dimension</th>
<th>Location</th>
<th>Send Option</th>
<th>Receive Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbsPressure</td>
<td>Scalar</td>
<td>0.0 N/m²</td>
<td>flux dens.</td>
<td>Line, Face, Volume</td>
<td>Code</td>
<td>Dir</td>
<td></td>
</tr>
<tr>
<td>DeltaTime</td>
<td>Scalar</td>
<td>1.0 s</td>
<td>g-min</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>FilmTemp</td>
<td>Scalar</td>
<td>300.0 K</td>
<td>field</td>
<td>Line, Face</td>
<td>Code</td>
<td>Dir</td>
<td></td>
</tr>
<tr>
<td>HeatRate</td>
<td>Scalar</td>
<td>0.0 W</td>
<td>flux dens.</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td></td>
</tr>
<tr>
<td>MassFlowRate</td>
<td>Scalar</td>
<td>0.0 kg/s</td>
<td>flux integral</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>MassFluxRate</td>
<td>Scalar</td>
<td>0.0 kg/m²s</td>
<td>flux dens.</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>NPosition</td>
<td>Vector</td>
<td>0.0 m</td>
<td>mesh coordinate</td>
<td>Line, Face, Volume</td>
<td>Code</td>
<td>Dir</td>
<td></td>
</tr>
<tr>
<td>OverPressure</td>
<td>Scalar</td>
<td>0.0 N/m²</td>
<td>flux dens.</td>
<td>Line, Face, Volume</td>
<td>Code</td>
<td>Dir</td>
<td></td>
</tr>
<tr>
<td>RelWallForce</td>
<td>Vector</td>
<td>0.0 N</td>
<td>flux integral</td>
<td>Line, Face</td>
<td>Code</td>
<td>Dir</td>
<td></td>
</tr>
<tr>
<td>Temperature</td>
<td>Scalar</td>
<td>300.0 K</td>
<td>field</td>
<td>Line, Face, Volume</td>
<td>Code</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>TotalPressure</td>
<td>Scalar</td>
<td>0.0 N/m²</td>
<td>flux dens.</td>
<td>Line, Face, Volume</td>
<td>Code</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>Velocity</td>
<td>Vector</td>
<td>0.0 m/s</td>
<td>field</td>
<td>Line, Face, Volume</td>
<td>Code</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>WallForce</td>
<td>Vector</td>
<td>0.0 N</td>
<td>flux integral</td>
<td>Line, Face</td>
<td>Code</td>
<td>Dir</td>
<td></td>
</tr>
<tr>
<td>Quantity</td>
<td>Dim.</td>
<td>Default Value</td>
<td>Integration Type</td>
<td>Coupling Dimension</td>
<td>Location</td>
<td>Send Option</td>
<td>Receive Option</td>
</tr>
<tr>
<td>----------------</td>
<td>--------</td>
<td>---------------</td>
<td>------------------</td>
<td>--------------------</td>
<td>----------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>WallHeatFlux</td>
<td>Scalar</td>
<td>0.0 W/m²</td>
<td>flux dens.</td>
<td>Line, Face</td>
<td>Code</td>
<td>Dir</td>
<td></td>
</tr>
<tr>
<td>WallHTCoeff</td>
<td>Scalar</td>
<td>0.0 W/m² K</td>
<td>field</td>
<td>Line, Face</td>
<td>Code</td>
<td>Dir</td>
<td></td>
</tr>
<tr>
<td>WallTemp</td>
<td>Scalar</td>
<td>300.0 K</td>
<td>field</td>
<td>Line, Face</td>
<td>Code</td>
<td>Dir</td>
<td>Dir</td>
</tr>
</tbody>
</table>
14.2.5 Go Step

OpenFOAM offers a number of options in the Go panel, see Figure 2.

**OpenFOAM solver** Choose the OpenFOAM solver from the list of installed solvers. Select Auto-Select-by-Case to leave the decision to MpCCI. In the latter case MpCCI reads the solver name from the application entry in the "<case directory>/system/controlDict" file.

**Define a reference pressure [N/m^2]** If you exchange a relative pressure (e.g. RelWallForce or Overpressure), you must set the reference pressure to an appropriate value, which usually corresponds to the atmospheric pressure. For more information on relative and absolute pressures see the FSI section in V-3.1.2 Coupling Types.

**Define a reference density [kg/m^3]** This value is only relevant in incompressible fluid flow. The calculation of forces and heat flux involve the density, thus a reference density has to be specified. Beyond that some incompressible solvers' pressure field does not contain the plain pressure \( p \) itself but \( p/\rho \) which is actually the pressure divided by the density. Therefore the adapter needs a reference density to translate between the bare pressure \( p \) handled by MpCCI and \( p/\rho \) occurring in some OpenFOAM solvers.

**Define a reference specific heat [J/kg K]** This value is only relevant in incompressible fluid flow. The calculation of heat flux involves the specific heat, thus a reference value has to be specified.

**Additional command line options** Everything added here will be appended to the arguments of the starter command.

**Alternative field names** If your (customized) solver does not use the usual names for the quantities e.g. temperature instead of T, you may select this option to define alternative names for the density, pressure, velocity and temperature fields. If you are unsure about the names used by your solver you may define (whitespace separated) lists of names e.g. T temp temperature.

**Run parallel** Activate OpenFOAM in parallel mode. See 14.2.6.1 Parallel Execution.

**Use MpCCI Grid Morpher** Activate the MpCCI Grid Morpher and access to the MpCCI Grid Morpher configuration, see 14.3 Grid Morphing.
14.2.6 Running the Computation

Coupled simulations can be run using the Start button in the Go panel. No special steps need to be carried through to start the simulation.

14.2.6.1 Parallel Execution

For a parallel run of OpenFOAM (i.e. OpenFOAM itself uses several parallel processes) additional options can be selected in the Go step as shown in Figure 3.

Use external 3rd party MPI Check this box if you want to use an MPI runtime environment (e.g. mpirun) that is not part of your OpenFOAM distribution, see 14.2.6.2 External 3rd party MPI.

No. of parallel processes Select how many OpenFOAM processes you wish to start.

Select decomposition method Choose one out of three methods: simple, scotch ormetis.

Splits by direction In case of the simple decomposition method you have to define a list of three positive integer values whose product must equal the value specified in No. of parallel processes.

Optional processor weights In case of the scotch ormetis decomposition method you may enter a list of processor weights to specify a load balancing. Leave blank for default.

Decompose case MpCCI will automatically run the decomposer of OpenFOAM if the number of parallel processes changed since the last decomposition. Press this button to start the decomposer explicitly.

Reconstruct case Press this button to reconstruct the decomposed case.

Optional hostlist file Specify a hostfile, from which host names are extracted.

Use default hostfile A default hostfile can be configured by setting MPCCI_HOSTLIST_FILE, see V-3.5.2 Hostlist File.
14.2.6.2 External 3rd party MPI

By checking the Use external 3rd party MPI box you can use an MPI runtime environment that is not part of the OpenFOAM distribution. The MPI runtime environment should be already set up for OpenFOAM. Some following options will help to define a proper environment for starting the OpenFOAM application in parallel for third party MPI vendor.

**MPI vendor** Select the MPI vendor type to use.

There are two options:

- **default**: Start the default MPI executable command to launch the parallel execution. MpCCI searches in this sequence for the following program to use: `mpirun`, `mpiexec`. The standard MPI parallel options are passed to this executable.
- **sgi**: Start the SGI MPI executable command `mpiexec_mpt` with its specific options.
- **intelmpi**: Start the Intel MPI executable command `mpiexec` with its specific options.

**Use external foamExecMpcci launcher** Running OpenFOAM in parallel requires to export some environment variables defining the location of the co-simulation server for example. Some MPI vendors do not implement the export of environment variables by argument option. MpCCI provides the option to use a wrapper script "foamExecMpcci" to start the simulation. This script can be provided by the user or generated by MpCCI at runtime.

**Generate foamExecMpcci** MpCCI generates the script "foamExecMpcci" in the project directory to launch the application.

⚠️ If the script is provided by the user, the script should ensure to load the environment definition file "envMpcci" under Linux, "envMpcci.bat" under Microsoft Windows. The environment file definition is saved under the project directory.

**Define shell type for environment source** Set the shell type to define the list of environment to export. Under Microsoft Windows the syntax is automatically set to `batch` like syntax independent of the set value.

- **default** Default syntax option is `bash` like under Linux.
- **bash** Use the `bash` syntax to export the environment variable.
- **csh** Use the `csh` syntax to export the environment variable.

![Image](image_url)

Figure 4: OpenFOAM options for third party MPI

14.2.6.3 Batch Execution

OpenFOAM is always run as a batch process, therefore no special settings are necessary for batch execution.
14.2.6.4 Running on a Remote Machine

In case the required environment to start the simulation is not correctly set/exported on a remote machine one can help MpCCI to set the environment (see V-3.6.1.2 Configure a Simulation Code for Running on a Remote Machine). Example:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOAM_INST_DIR</td>
<td>~/OpenFOAM/OpenFOAM-v1712</td>
</tr>
<tr>
<td>PATH</td>
<td>~/OpenFOAM/OpenFOAM-v1712/platforms/linux64GccDPInt32O32pt/bin</td>
</tr>
<tr>
<td>LD_LIBRARY_PATH</td>
<td>/cluster/lib64</td>
</tr>
</tbody>
</table>

14.2.7 Post-Processing

Post-processing for the OpenFOAM part of the results can be performed as in ordinary computations, e.g. with paraFoam.

14.3 Grid Morphing

The grid morpher will smooth a grid based on the displacements of some boundary or interior vertices. A moving or morphing grid capability is always required with fluid-structure interactions. The user has the choice between the MpCCI Grid Morpher (14.3.1 MpCCI Grid Morpher) and the OpenFOAM Grid Morpher (14.3.2 OpenFOAM Grid Morpher).

14.3.1 MpCCI Grid Morpher

MpCCI offers a morphing method which allows vertices to float along semi-planar boundaries, e.g. symmetry planes. The morpher process may be monitored within an additional output terminal and a morpher log file in the OpenFOAM working directory ("mpcci_morpher_<case directory>.log").

The morpher controls the results of the morphing step, it may control the length of the edges, the shape and size of faces and also checks the quality of cells. There are options available to limit the compression and elongation of edges.

The following options (Figure 5) may be adjusted, whereas the default values are in many cases appropriate. The description of the MpCCI Grid Morpher is provided at V-8 MpCCI Grid Morpher.

14.3.2 OpenFOAM Grid Morpher

If the MpCCI Grid Morpher is not selected, OpenFOAM will use the solver type defined in the "dynamicMeshDict". In the case of the VII-4 Elastic Flap in a Duct tutorial, the displacementLaplacian has been used.

You can observe the log output of OpenFOAM in Figure 6. See also 14.2.1.2 Deforming Meshes.
Figure 5: OpenFOAM Options for MpCCI Grid Morpher
Create time

Create mesh for time = 0

Selecting dynamicFvMesh dynamicMotionSolverFvMesh
Selecting motion solver: displacementLaplacian
Selecting motion diffusion: inverseDistance

Figure 6: OpenFOAM output for displacement laplacian solver
14.4 Code-Specific MpCCI Commands

The MpCCI subcommands available for OpenFOAM are:

Usage:
mpcci OpenFOAM [-]option

Synopsis:
‘mpcci OpenFOAM’ is used to get information about OpenFOAM.

Options:
- allapps [CASEDIR] [SP|DP] [VOpt|VDebug] [RELEASE]
  List the solver used by this case and all OpenFOAM applications.
- caseapp [CASEDIR]
  Show the solver used by this case.
- cleanTimeDir
  Mark time directories (serial and parallel) of an OpenFOAM case containing only empty functionObjectProperties files with <time>_DELETE.
- decompose [OPTIONS]
  Decompose the serial OpenFOAM case into a parallel case. Use -f option to overwrite existing decomposition.
- diff <CASEDIR1> <CASEDIR2>
  Run the scanner on two cases and print the differences.
- env [RELEASE]
  Print the OpenFOAM specific environment.
- envmpcci [RELEASE]
  Print the OpenFOAM MpCCI specific environment.
- gmdmake [CASEDIR]
  Make an MpCCI grid morpher data file from the OpenFOAM case.
- help
  This screen.
- info [SP|DP] [VOpt|VDebug]
  List verbose information about all OpenFOAM releases.
- magic [CASEDIR]
  Print the magic tokens of the MpCCI grid morpher data .gmd file or all gmd files in the current directory.
- reconstruct [CASEDIR] [RELEASE]
  Reconstruct a serial OpenFOAM case from the decomposed case.
-releases [SP|DP] [VOpt|VDebug]
    List all OpenFOAM releases which MpCCI can find.
-scan <CASEDIR>
    Run the scanner on the case and create a scanner output file.

The subcommands `diff`, `info`, `releases` and `scan` are described in > 1.1 Common MpCCI Subcommands for Simulation Codes <.

`mpcci openfoam -allapps [CASEDIR] [SP|DP] [VOpt|VDebug] [RELEASE]`

The `allapps` subcommand gets the solver used by the given case as well as all OpenFOAM basic and user local applications and prints them to stdout line by line.

`mpcci openfoam -caseapp [CASEDIR]`

The `caseapp` subcommand gets the solver used by the given case and prints it to stdout.

`mpcci openfoam -cleanTimeDir`

The `cleanTimeDir` subcommand marks time directories (serial and parallel) of an OpenFOAM case containing only empty functionObjectProperties as "<time>_DELETE".

`mpcci openfoam -decompose [OPTIONS]`

The `decompose` subcommand decomposes the serial OpenFOAM case into a parallel case. It offers following options:

Usage:

```
mpcci OpenFOAM decompose [-]options
```

Synopsis:

'`mpcci OpenFOAM decompose'` is used to decompose an OpenFOAM case.

Options:

- `case <casedir>` Defines the case directory.
- `force` Force a new decomposition.
- `help` This screen.
- `method <simple|scotch>` Defines the decomposition method.
- `nprocs <nprocesses>` Defines the no. of processes.
- `pweights <weights>` Defines the processor weights.
- `release <release>` Define the release to be used.
- `splits <x y z>` Defines the split in each direction.

Use `force` option to overwrite an existing decomposition.

`mpcci openfoam -env [RELEASE]`

The `env` subcommand gets the OpenFOAM specific environment variables for the specified release and prints them to stdout. If no release is specified the latest installed release will be taken.
The `envmpcci` subcommand gets the OpenFOAM MpCCI specific environment variables for the specified release and prints them to stdout. If no release is specified the latest installed release will be taken.

The `gmdmake` subcommand makes an MpCCI grid morpher data file from the OpenFOAM case. CASEDIR specifies the OpenFOAM case directory. Default is the current directory.

The `magic` subcommand gets the magic tokens of the MpCCI grid morpher data gmd file in the specified case directory i.e. "<CASEDIR>/mpcci_morpher.gmd" or of all gmd files in the current directory if CASEDIR is not specified and prints them to stdout.

The `reconstruct` subcommand reconstructs a serial OpenFOAM case from the decomposed case in the given CASEDIR and for the given RELEASE. The current directory and latest installed release will be taken as default.

### 14.5 Code Adapter Reference

- Volumetric data can only be transferred unidirectional—namely from OpenFOAM to another code.
- The MpCCI code adapter for OpenFOAM uses so called function objects to manage the data transfer. Whether exchange is done before or after the solution depends on the solver employed to solve your problem. The standard solvers of OpenFOAM usually perform an exchange before solution. The code adapter is distributed as dynamic libraries, which are located below "<MpCCI_home>/codes/OpenFOAM/adapters/".

Depending on version and platform the binaries are placed in further subdirectories:

"./<WM_PROJECT_VERSION>/libfoampcci.so".

There’s also an auxiliary library used in case of dynamic mesh motion; this is "./<WM_PROJECT_VERSION>/libmpccimotionsolver.so".

If there’s no adapter available for your particular version and platform, you can simply build the desired libraries on your own. Just call "./Allwmake" located in "<MpCCI_home>/codes/OpenFOAM/adapters/src/<MPCCI_FOAM_VERSION>".

<MPCCI_FOAM_VERSION> corresponds to the following folder name:
- "v1.7-2.1" Adapter code compatible with OpenFOAM 1.7 to 2.1
- "v2.2.-2.3" Adapter code compatible with OpenFOAM 2.2 to 2.4
- "v3.0.1" Adapter code compatible with OpenFOAM 3.0.1
- "v1606+" Adapter code compatible with OpenFOAM v1606+
- "v1612++v1712" Adapter code compatible with OpenFOAM v1612+,v1706,v1712.

- If the MpCCI Grid Morpher is used you will require an auxiliary tool "mpcci_foam2gmd.exe" which is located below "<MpCCI_home>/codes/OpenFOAM/bin/".

Depending on version and platform the binaries are placed in further subdirectories:

"./<WM_PROJECT_VERSION>/mpcci_foam2gmd.exe".
This tool is used to convert the OpenFOAM mesh to the MpCCI Grid Morpher format.
To build the tool you should:

1. Go to the directory "<MpCCI_home>/codes/OpenFOAM/foam2gmd"
2. Call ./Allwmake

The tool "mpcci_foam2gmd.exe" executable will be automatically placed under the "bin" directory.

Please make sure that the appropriate environment of your OpenFOAM version is sourced.
15 SIMPACK

15.1 Quick Information

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Dimension</th>
<th>Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit</td>
<td>Implicit</td>
<td>X X</td>
</tr>
</tbody>
</table>

15.1.1 Supported Coupling Methods

15.1.2 Supported Platforms and Versions

SIMPACK is supported on the following platforms:

<table>
<thead>
<tr>
<th>Platform</th>
<th>Supported versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>x86_64</td>
<td>9.7</td>
</tr>
<tr>
<td>x86_64</td>
<td>9.8.1</td>
</tr>
<tr>
<td>windows_x64</td>
<td>X</td>
</tr>
<tr>
<td>win64</td>
<td>X</td>
</tr>
</tbody>
</table>

For details on the platforms supported by SIMPACK see also the Platform Support page at www.simpack.com/platforms.html.

15.1.3 References

SIMPACK Documentation Your SIMPACK installation contains the SIMPACK Assistant.

15.1.4 Adapter Description

The code adapter for SIMPACK is developed and distributed by the Fraunhofer SCAI.

15.1.5 Prerequisites for a Coupled Simulation

To run a coupled simulation you need the following:

- Ordinary SIMPACK installation.
- Enough license tokens.

15.2 Coupling Process

Please read also IV-2 Setting up a Coupled Simulation.
15.2.1 Model Preparation

The SIMPACK model can be prepared with SIMPACK GUI. Please consider the following advices:

- The model can be defined in any of the consistent systems of units supported by MpCCI (see \[1.2 Unit Systems\]). The basic unit system must be given in the Models step of the MpCCI GUI. Units of single quantities can be set in the Regions step.

- The SIMPACK model must contain a definition of a force element of type user force 20 as a coupling component. Also multiple elements of this type can be included, to introduce different coupling points. The element which serves as coupling component can be selected in the Regions step of the MpCCI GUI. Also multiple elements can be selected.

It is recommended to create a complete SIMPACK model first and test it separately without co-simulation. The quantities which will be transferred by the partner code can be simulated by appropriate loads or boundary conditions.

15.2.2 Simulation

The only SIMPACK simulation mode, supported by MpCCI for now is a dynamic simulation. The step sizes or any other options of the SIMPACK solver are not changed by MpCCI. The data from the co-simulation will be provided to SIMPACK/Solver as it is requested by the calls of the corresponding uforce subroutine. The communication time points are not controlled by MpCCI. However it is possible to restrict the communication to a fixed time steps scheme. Please be sure, that the co-simulation partner also uses a dynamic simulation or at least provide time values for the sent data.

The time steps of the co-simulation partners differ in general. To match the data values between requested and provided time points the server interpolates the quantities in time. Therefore multiple buffers are used per default with a history size of 4. If the time step size is much smaller than that from the partner code you may need to increase the history size (See \[V-4.8.2 Job\] and \[V-3.4.5.3 Coupling with Non-Matching Time Steps\]).

15.2.3 Models Step

![SIMPACK options in the Models step](image)

In the Models step, the following options must be chosen:

**SIMPACK release** Select the SIMPACK release you wish to use. Only supported releases installed on your system are listed. The selection latest always refers to the latest supported version (default). The release should match the input file.
**SIMPACK file for scanning**  Select the SIMPACK input file "*.spck" to provide the simpack model definition to MpCCI.

**Unit system**  Select the unit system which was used in SIMPACK (see ▶ 1.2 Unit Systems ◄). Default is set to variable for the unit system with m for the grid length unit.

### 15.2.4 Algorithm Step

For general options please refer to ▶ V-4.5.2 Code Specific Algorithm Settings ◄.

In the Algorithm step, following additional options are available:

**Solver settings**

- **Provide partials**  If this option is checked, SIMPACK will provide the partial derivatives of the received quantities with respect to the sent quantities. This helps the process of solving the non-linear equations in the corrector iterations. See ▶ 15.2.5.1 Semi-implicit Mode and Partial Derivatives ◄ for more details.

- **Communication delay of the co-simulation**  This parameter is used for the approximation of the semi-implicit mode or partial derivatives. This is the communication offset, introduced by the co-simulation. The reaction on the data, sent at some time points is received by the next communication. This is the general case for the parallel explicit co-simulation. Due to technical reasons, the offset by the co-simulation is to be set to 1. Those are the communication scenarios which are for now supported by MpCCI. Wrong setting of this parameter results in errors by the calculation of approximation for the semi-implicit and partial derivatives modes.

- **Synchronized history buffer update**  Check this parameter for the SIMPACK adapter to update the saved states only at certain time points. This can be used e.g. if the co-simulation partner takes different time step sizes for the simulation. In this case, data provided as reaction and saved outgoing quantities have to be adjusted. With the option **Time step for approximation update** a constant step size for the updates can be set. As well a Time-Step-Size control variable can be used (see ▶ 15.2.5 Regions Step ◄).

- **Time step for approximation update**  Sets a constant step size for the synchronized history buffer updates. If this value is negative, the adapter tries to use the Time-Step-Size control variable.
15.2.5 Regions Step

**SIMPACK** supports the following quantities for coupling:

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Dim.</th>
<th>Default Value</th>
<th>Integration Type</th>
<th>Coupling Dimension</th>
<th>Location</th>
<th>Send Option</th>
<th>Receive Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>AngularCoordinate</td>
<td>Quaternion</td>
<td>0.0</td>
<td>mesh coordinate</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>AngularVelocity</td>
<td>Vector</td>
<td>0.0 rad/s</td>
<td>field</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>DeltaTime</td>
<td>Scalar</td>
<td>1.0 s</td>
<td>g-min</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>Force</td>
<td>Vector</td>
<td>0.0 N</td>
<td>flux integral</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>PointPosition</td>
<td>Vector</td>
<td>0.0 m</td>
<td>mesh coordinate</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>RealFlag</td>
<td>Scalar</td>
<td>0.0</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>Torque</td>
<td>Vector</td>
<td>0.0 N m</td>
<td>flux integral</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>Velocity</td>
<td>Vector</td>
<td>0.0 m/s</td>
<td>field</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>

In this step the user can select the elements of the **SIMPACK**-model and set the quantities to be exchanged by the co-simulation.

The outgoing quantities are requested from the **From Marker**.

Furthermore the time step size to control the approximation process can be received by **SIMPACK**. For this purpose an additional element **Time-Step-Size** is provided in the global variables list, which can only receive the time step size. **MpCCI** uses this value to control the calculation of partial derivatives (see $\triangleright$ 15.2.6 Go Step $\triangleright$). If the user requires time step size to be sent by **SIMPACK**, the operation is ignored.

15.2.5.1 Semi-implicit Mode and Partial Derivatives

In **SIMPACK** the predictor-corrector-approach (see **SIMPACK** documentation) is implemented to solve the equation of motion. In corrector iterations a system of non-linear equations is considered. For this purpose **SIMPACK** uses e.g. the Newton approach. This algorithm utilizes partial derivatives with respect to generalized coordinates for solving non linear equations.

The **SIMPACK** adapter can provide the partial derivatives of the incoming quantities with respect to the outgoing quantities. To allow this the user must check the option **Provide partials** (see $\triangleright$ 15.2.6 Go Step $\triangleright$). To provide the partial derivatives the **SIMPACK** adapter will store the incoming and outgoing quantities. Those saved values are then used to calculate a numerical approximation of the partial derivatives. Different parameters can be adjusted to control the calculation (see $\triangleright$ 15.2.6 Go Step $\triangleright$).

Normally, the data exchange between **SIMPACK** and **MpCCI** server is done for every single time step. However, the internal solution algorithms are in general iterative. As consequence, there are variations of local state for each iteration at single time point. With the partial derivatives we use the semi-implicit approach in **SIMPACK**, to improve the local accuracy of solution and provide an approximation of the received quantities (e.g. force) with respect to the outgoing quantities (e.g. position) between the communication points. The semi-implicit mode is automatically activated with the **Provide partials**.

The relations between outgoing and incoming quantities are adjusted automatically. **MpCCI** defines dependencies between appropriate kinematic and dynamic quantities. The force quantity is linked to the position, velocity or acceleration. The torque quantity is linked to the angular position, angular velocity or angular acceleration.

15.2.6 Go Step

In the **Go** step, the following options can be selected:
Additional command line options  Additional command line options for SIMPACK can be given here, they will directly be used when SIMPACK is started.

User library configuration  Expand this section to access the additional user subroutine configuration.

Use default uforce20  Check this option if you want to use the provided user library based on uforce20 force type element. If this option is unchecked, MpCCI supposes that the user provides an MpCCI-compatible user library for SIMPACK, which already includes MpCCI calls.

Please select the SIMPACK file for a restart  If you want to restart a previous computation, select a restart file here.

15.2.7 Running the Computation

With the button SIMPACK is started with the options given in the Go step.

In the explicit mode SIMPACK sends and receives data for each time step.

If the button is pressed SIMPACK receives a command from the adapter and terminates the simulation.

You can check during the SIMPACK run for the SIMPACK log file. The messages of the co-simulation progress and of the SIMPACK simulation can be found there. In case of any issue with SIMPACK, please first check the SIMPACK log file for further information.

15.2.7.1 Batch Execution

SIMPACK always runs as a batch process, therefore no special settings are necessary for batch execution.

15.2.8 Post-Processing

Post-processing for the SIMPACK part of the results can be performed as in ordinary computations, e.g. with SIMPACK-post. The "<job name>.sbr" file can be found in the SIMPACK model directory.
15.3 Code-Specific MpCCI Commands

The MpCCI subcommands available for SIMPACK are:

<table>
<thead>
<tr>
<th>Usage:</th>
<th>mpcci SIMPACK [-]option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synopsis:</td>
<td>'mpcci SIMPACK' is used to get information about SIMPACK.</td>
</tr>
<tr>
<td>Options:</td>
<td>-help</td>
</tr>
<tr>
<td></td>
<td>This screen.</td>
</tr>
<tr>
<td></td>
<td>-info</td>
</tr>
<tr>
<td></td>
<td>List verbose information about all SIMPACK releases.</td>
</tr>
<tr>
<td></td>
<td>-releases</td>
</tr>
<tr>
<td></td>
<td>List all SIMPACK releases which MpCCI can find.</td>
</tr>
<tr>
<td></td>
<td>-scan [-r release] &lt;spck file&gt;</td>
</tr>
<tr>
<td></td>
<td>Run the scanner and create a scanner output file.</td>
</tr>
</tbody>
</table>

The subcommands [info, releases] and [scan] are described in >1.1 Common MpCCI Subcommands for Simulation Codes <.
16 STAR-CCM+

16.1 Quick Information

<table>
<thead>
<tr>
<th>Company name</th>
<th>Siemens PLM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company homepage</td>
<td><a href="https://www.plm.automation.siemens.com">https://www.plm.automation.siemens.com</a></td>
</tr>
<tr>
<td>Support</td>
<td><a href="https://www.plm.automation.siemens.com/global/de/support/">https://www.plm.automation.siemens.com/global/de/support/</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tutorials</th>
<th>VII-2 Vortex-Induced Vibration of a Thin-Walled Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VII-4 Elastic Flap in a Duct</td>
</tr>
<tr>
<td></td>
<td>VII-5 Exhaust Manifold</td>
</tr>
<tr>
<td></td>
<td>VII-8 Pipe Nozzle</td>
</tr>
<tr>
<td></td>
<td>VII-9 Cube in a Duct Heater</td>
</tr>
<tr>
<td></td>
<td>VII-10 Y-Junction</td>
</tr>
</tbody>
</table>

### 16.1.1 Supported Coupling Methods

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Dimension</th>
<th>Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit</td>
<td>Implicit</td>
<td>□</td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td>X</td>
<td>-</td>
</tr>
</tbody>
</table>

### 16.1.2 Supported Platforms and Versions

<table>
<thead>
<tr>
<th>MPCCI_ARCH: Code platform</th>
<th>Supported versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>(windows</td>
<td>lnx4)</td>
</tr>
<tr>
<td>lib</td>
<td>X</td>
</tr>
</tbody>
</table>

### 16.1.3 References

STAR-CCM+ Documentation is part of the STAR-CCM+ distribution (e.g. STAR-CCM+ User Guide).

### 16.1.4 Adapter Description

For STAR-CCM+ the MpCCI plugin library "libstarccmpcci.jar" is retrieved from MpCCI installation and dynamically loaded by STAR-CCM+ at runtime and all data transfer and management is carried out over MpCCI Java macro routines ([16.5 Code Adapter Reference](#)).

The viscous and viscoelastic solvers in STAR-CCM+ are supported for the co-simulation.
16.1.5 Prerequisites for a Coupled Simulation

To run a coupled simulation you need the following:

- Ordinary STAR-CCM+ installation.
- The command `starccm+` should be set in the PATH under Linux.

16.2 Coupling Process

Please read also ▶ IV-2 Setting up a Coupled Simulation ◀.

16.2.1 Model Preparation

The STAR-CCM+ model can be prepared with STAR-CCM+ GUI. Please consider the following approach for model preparation:

- The STAR-CCM+ solver internally operates only in SI units. The geometrical dimensions should best be defined in meters. However the user can use in STAR-CCM+ GUI a preferred unit system. In that case you should remember the unit setting in order to provide it at the Models step in MpCCI GUI.
- The STAR-CCM+ model must contain a definition of the coupling domains and boundary regions (e.g. couple-wall).
- To enable a coupled simulation with an unprepared model several modifications of the model are required. For popular coupling types (▶ V-3.1.2 Coupling Types ◀) the modifications will be carried out by MpCCI automatically.
- MpCCI supports following STAR-CCM+ Motion Specification for FSI (▶ 16.4 Grid Morphing ◀):
  - Morphing: The Morpher method Displacement is used in this case.
  - DFBI Morphing: The Six DOF morphing method Six DOF body plus Displacement is used.
- Model using Overset Mesh method is also supported for FSI application.
- MpCCI supports the different heat transfer coefficient type of STAR-CCM+ for the thermal co-simulation model. Therefore if the convection surface load is defined as
  - heat transfer coefficient (`WallHTCoeff`), ambient temperature (`FilmTemp`) or,
  - heat rate (`HeatRate`), ambient temperature (`FilmTemp`) the heat transfer coefficient properties of STAR-CCM+ should be configured according to your application. Following heat coefficient type of STAR-CCM+ are available to select in the MpCCI GUI (see ▶ 16.2.5 Go Step ◀):
    - `HeatTransferCoefficientUserYPlus` (default) is the specified y+ heat transfer coefficient and will be combined with specified y+ heat transfer reference temperature for `FilmTemp`.
    - `LocalHeatTransferCoefficient` is the local heat transfer coefficient and will be combined with the local heat transfer reference temperature for `FilmTemp`.
    - `HeatTransferCoefficient` is the heat transfer coefficient and will be combined with the constant reference temperature defined in this field function for `FilmTemp`. For this purpose a user field function `HeatTransferReferenceTemperature` will be built and filled with the reference value.

⚠️ It is recommended to create a complete STAR-CCM+ model first and test it separately without co-simulation. The quantities, which will be later on transferred by the partner code, can be simulated by appropriate loads or boundary conditions if desired.
16.2.2 Models Step

In the Models step, the following options must be selected: (Figure 1):

**STAR-CCM+ release**  Select the release of STAR-CCM+ you would like to use, latest (default) will select the latest installed STAR-CCM+ version for which an MpCCI adapter is also installed. Please make sure that the model file was not created with a STAR-CCM+ version newer than the STAR-CCM+ version used with MpCCI.

**STAR-CCM+ simulation file**  Select the model file (*.sim") of your STAR-CCM+ model.

**Unit system**  Select the unit system which was used in STAR-CCM+ (see ▶ 1.2 Unit Systems ◄).

16.2.3 Algorithm Step

For general options please refer to ▶ V-4.5.2 Code Specific Algorithm Settings ◄.

In the Algorithm step, following additional options are available:

**Solver settings**

**Type of solver step size**  (*Only available for the STAR-CCM+ implicit unsteady solver*)

- Defined by model Select this option in order to used the time step from the model file.
- Constant Select this option in order to overwrite the setting from the model file by providing a new constant time step size value.

**Solver time step size**  (*Only available with Constant Type of solver step size*)  Provide the solver time step size to use.

**Maximum number of inner iterations**  Provide the maximum number of inner iterations to perform for a solver time step.

**Select type of heat transfer coefficient**  If WallHTCoeff, HeatRate, FilmTemp are coupled, define here the heat coefficient type of STAR-CCM+, available are:

- **HeatTransferCoefficientUserYPlus** (default) is the specified y+ heat transfer coefficient and will be combined with specified y+ heat transfer reference temperature for FilmTemp.
- **LocalHeatTransferCoefficient** is the local heat transfer coefficient and will be combined with the local heat transfer reference temperature for FilmTemp.
- **HeatTransferCoefficient** is the heat transfer coefficient and will be combined with the constant reference temperature defined in this field function for FilmTemp. For this purpose a user field function **HeatTransferReferenceTemperature** will be built and filled with the reference value.
For STAR-CCM+, the coupling period should be limited via Coupling duration panel in the Common Basics settings.

### 16.2.4 Regions Step

STAR-CCM+ supports the following quantities for coupling:

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Dim.</th>
<th>Default Value</th>
<th>Integration Type</th>
<th>Coupling Dimension</th>
<th>Location</th>
<th>Send Option</th>
<th>Receive Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbsPressure</td>
<td>Scalar</td>
<td>0.0 N/m²</td>
<td>flux dens.</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>AngularVelocity</td>
<td>Vector</td>
<td>0.0 rad/s</td>
<td>field</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>DeltaTime</td>
<td>Scalar</td>
<td>1.0 s</td>
<td>g-min</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>FilmTemp</td>
<td>Scalar</td>
<td>300.0 K</td>
<td>field</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>Force</td>
<td>Vector</td>
<td>0.0 N</td>
<td>flux integral</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>HeatRate</td>
<td>Scalar</td>
<td>0.0 W</td>
<td>flux dens.</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>IterationNo</td>
<td>Scalar</td>
<td>0</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>MassFlowRate</td>
<td>Scalar</td>
<td>0.0 kg/s</td>
<td>flux integral</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>MassFluxRate</td>
<td>Scalar</td>
<td>0.0 kg/m²s</td>
<td>flux dens.</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>NPosition</td>
<td>Vector</td>
<td>0.0 m</td>
<td>mesh coordinate</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>OverPressure</td>
<td>Scalar</td>
<td>0.0 N/m²</td>
<td>flux dens.</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>PhysicalTime</td>
<td>Scalar</td>
<td>0.0 s</td>
<td>g-min</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>RefPressure</td>
<td>Scalar</td>
<td>1.12e5 N/m²</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>RelWallForce</td>
<td>Vector</td>
<td>0.0 N</td>
<td>flux integral</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>StaticPressure</td>
<td>Scalar</td>
<td>0.0 N/m²</td>
<td>flux dens.</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>Temperature</td>
<td>Scalar</td>
<td>300.0 K</td>
<td>field</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>TimeStepNo</td>
<td>Scalar</td>
<td>0</td>
<td>g-max</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>Torque</td>
<td>Vector</td>
<td>0.0 N</td>
<td>flux integral</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>TotalPressure</td>
<td>Scalar</td>
<td>0.0 N/m²</td>
<td>flux dens.</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>Velocity</td>
<td>Vector</td>
<td>0.0 m/s</td>
<td>field</td>
<td>Point, Face</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>WallForce</td>
<td>Vector</td>
<td>0.0 N</td>
<td>flux integral</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>WallHeatFlux</td>
<td>Scalar</td>
<td>0.0 W/m²</td>
<td>flux dens.</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>WallHTCoeff</td>
<td>Scalar</td>
<td>0.0 W/m² K</td>
<td>field</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>WallTemp</td>
<td>Scalar</td>
<td>300.0 K</td>
<td>field</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>

### 16.2.5 Go Step

In the Go step, the following options can be selected:

**Run in batch**  STAR-CCM+ runs in batch mode.

**Auto start with the default template macro**  Select this option to use the MpCCI provided Java macro template. Depending on the selected coupling scheme one of the following template files will be copied into the working directory containing the ".sim" file:

"mpcci_runjob_steady.java" or "mpcci_runjob_unsteady.java".

Following option is available if the default Java macro template shall be used (see Figure 3):

**Do a cold start run**  Delete the current solution from the ".sim" file before starting the simulation.

**STAR-CCM+ Java macro file**  STAR-CCM+ macro file to steer the computation. This option is displayed and required if the Auto start with the default template macro is not activated (see Figure 2 on the left).

**Do not shut down server after batch file execution**  STAR-CCM+ won't shut down after batch file has been executed.
Additional command line options
It is possible to invoke STAR-CCM+ with additional command line options (STAR-CCM+ Run Options). Please refer to the STAR-CCM+ User Guide for possible options.

Additional user macro classpath
It is possible to provide to STAR-CCM+ an additional path containing a user library. This classpath is appended to the MpCCI path.

Boundary export mode (optional)
Define how the boundary information is exported.
- single (default) The code adapter will use a single file for exporting all the coupled regions.
- individual The code adapter will use an individual file for exporting each coupled region.

License options
See 16.2.5.1 License Setting Details.

Run parallel
Run STAR-CCM+ in parallel mode, see 16.2.6.1 Parallel Execution.

Use remeshing
Activate the MpCCI remeshing module, see 16.2.6.2 MpCCI Remeshing Module.

16.2.5.1 License Setting Details

License options
Define the STAR-CCM+ license options (see Figure 4).
- Power session: use power session license model.
- Lite session: use lite session license model.
- PoD session: for this license option you should additionally provide the Server name in the form portnumber@host and the PoD key.

If no additional license option is provided the standard default method to check out a license is used by STAR-CCM+.
16.2.6 Running the Computation

When the [Start] button is pressed, STAR-CCM+ is started with the options given in the Go step. If the [Stop] button is pressed, an ABORT-file is created and STAR-CCM+ will stop the next time it checks the presence of a stop file. This requires that the stop criterion for the "ABORT" file is activated and that the user Java macro script exits if the computation is aborted. The provided MpCCI Java macro script implements such exit on "ABORT" file.

When STAR-CCM+ is started, MpCCI will update the classpath in the STAR-CCM+ setting file if STAR-CCM+ is running with GUI otherwise the new classpath is passed via a command line option to STAR-CCM+.

16.2.6.1 Parallel Execution

If the Run parallel check button is activated the following options are raised (Figure 5):

- **No. of parallel processes**  Select the number of parallel STAR-CCM+ processes to be launched.
- **MPI driver used for communication**  Select the MPI implementation.
- **Optional 'host host...' to be used**  Enter host names for parallel execution of STAR-CCM+.
- **Optional hostlist file**  Specify a hostfile from which host names are extracted.
- **Use default hostfile**  A default hostfile can be configured by setting MPCCI_HOSTLIST_FILE $\text{\textasciitilde}V-3.5.2$ Hostlist File.$\text{\textasciitilde}$

In case that a hostlist and a hostfile are defined, MpCCI adds up the hosts and passes as much hosts as defined processes to STAR-CCM+. The order of the host definitions in the MpCCI GUI defines the priority of the employed hosts (1st hostlist, 2nd hostfile and 3rd default hostfile).
16.2.6.2 MpCCI Remeshing Module

Prerequisites for MpCCI remeshing module:

- The model must have a valid meshing module.
- The remeshing module only works for 3D models because based on the STAR-CCM+ meshing tools.
- The model must contain unique boundary names.
- The model must contain unique feature curve names.

The MpCCI remeshing module will basically create a new mesh based by considering the meshing option values from the mesh continua and from the boundaries if available.

The user may activate the module by checking the Use remeshing option (see Figure 6). Once the remeshing module is activated this one is only applied on regions with morphing motion and if the region will be deformed by MpCCI.

![Figure 6: STAR-CCM+ options for remeshing](image)

**Remeshing setting description:**

**Use remeshing**  Activate the remeshing module.

**Remeshing method** The user can choose to apply a global remeshing on the model (option: all regions) or to limit the remeshing on the coupled regions (option: coupled regions).

**Use surface remesher**  Turn on surface mesh refinement.

**Use higher stencil**  Turn on higher-order stencil for interpolation for the mesh replacement instead of closest point. The option is only available if the remeshing is applied on coupled regions only.

ån The usage of higher-order may consume more CPU time on large regions.
**Feature angle** Set the sharp edge angle value for the creation of feature curves.

**Skewness minimum value** Set the minimum value of the skewness angle function.

**Skewness maximum value** Set the maximum value of the skewness angle function.

**Skewness coefficient limit** Set the skewness coefficient limit.

The skewness coefficient is defined as the ratio between the current number of cells in the threshold skewness and the last number of cells in the threshold skewness from the last remeshing. If this value is larger than the specified limit the remeshing condition is satisfied.

**Quality minimum value (0 = worst)** Set the minimum value for the cell quality function.

**Quality maximum value (1 = best)** Set the maximum value for the cell quality function.

**Cell quality coefficient limit** Set the cell quality coefficient limit.

The cell quality coefficient is defined as the ratio between the current number of cells in the threshold cell quality and the last number of cells in the threshold from the last remeshing. If this value is larger than the specified limit the remeshing condition is satisfied.

**Minimum cell quality (worst 0 - 1 best)** Set the minimum cell quality value with 0 being the worst and 1 being perfect.

If the current minimum cell quality is smaller than the specified value, the remeshing condition is satisfied.

**Minimum volume change (worst 0 - 1 best)** Set the minimum volume change value with 0 being the worst and 1 being perfect.

If the current minimum volume change is smaller than the specified value, the remeshing condition is satisfied.

**Maximum cell skewness (worst 90 - 0 best)** Set the maximum cell skewness value with 90 being the worst and 0 being perfect.

If the current maximum cell skewness is larger than the specified value, the remeshing condition is satisfied.

**Minimum face validity (worst 0 - 1 best)** Set the minimum face validity value with 0 being the worst and 1 being perfect.

If the current minimum face validity is smaller than the specified value, the remeshing condition is satisfied.

**Minimum cell aspect ratio (worst 0 - 1 best)** Set the minimum cell aspect ratio value with 0 being the worst and 1 being perfect.

If the current cell aspect ratio is smaller than the specified value, the remeshing condition is satisfied.

If the remeshing module is activated, it will be checked before every MpCCI data exchange.

### 16.2.6.3 Batch Execution

Activate the option Run in batch (see Figure 2 on the left) to run STAR-CCM+ as a batch process.

### 16.2.7 Post-Processing

Post-processing for the STAR-CCM+ part of the results can be performed as in ordinary computations, e.g. with STAR-CCM+ GUI. The results "<job name>.sim" file can be found in the same directory as the input file.
16.3 Code-Specific MpCCI Commands

The MpCCI subcommands available for STAR-CCM+ are:

**Usage:**
```
mpcci STAR-CCM+ [-]option
```

**Synopsis:**
Use `mpcci STAR-CCM+` to get information about STAR-CCM+, and more...

Most of the required steps are automatically done within the MpCCI GUI. The commandline version may be helpful in case of problems and for debugging.

**Options:**
- `-diff <sim1> <sim2>`
  Run the scanner on two sim files and print the differences.
- `-getmacro <steady|unsteady>`
  Copy the MpCCI Java macro template to the current directory.
- `-help`
  This screen.
- `-info`
  List verbose information about all STAR-CCM+ releases.
- `-releases`
  List all STAR-CCM+ releases which MpCCI can find.
- `-scan <sim>`
  Run the scanner and create a scanner output file.

The subcommands `-diff`, `-info`, `-releases` and `-scan` are described in §1.1 Common MpCCI Subcommands for Simulation Codes.

```
mpcci star-ccm+ -getmacro <steady|unsteady>
```

The `getmacro` subcommand provides a local copy of the MpCCI Java macro template. This allows the user to append own additional functions to the template.

16.4 Grid Morphing

The grid morpher will smooth a grid based on the displacements of some boundary or interior vertices. A moving or morphing grid capability is always required with fluid-structure interactions. The user should activate the morpher model for the simulation and configure the boundaries that should be morphed.

MpCCI supports the following STAR-CCM+ moving mesh models:
- Mesh morphing
- Dynamic fluid body interaction 6DOF
The boundary conditions properties for these morpher models will be automatically updated resp. to
- total displacement method
- 6 DOF body plus displacement

16.5 Code Adapter Reference

The necessary plug-ins can be found in
"<MpCCI_home>/codes/STAR-CCM+/adapters/<starccmrelease>/lib"

The data transfer and management is handled over Java macros plug-ins, which are implemented in the MpCCI-STAR-CCM+ adapter library.

MpCCI has a limited ability to analyze the input in the MpCCI GUI and then decide what modifications have to be done to enable the model for a coupled simulation. These will be carried out when starting the simulation in the Go step (▷ 16.2.5 Go Step ◁) and when STAR-CCM+ connects to the MpCCI server.

16.5.1 Java Macro Script

In STAR-CCM+, the coupling process is controlled via a Java macro script, which calls functions provided by MpCCI.

MpCCI provides a set of Java macro scripts ready for the coupling under the directory:
"<MpCCI_home>/codes/STAR-CCM+/adapters/macro"
- "mpcci_runjob_steady.java": for steady state analysis.
- "mpcci_runjob_unsteady.java": for transient analysis.

One of the macro scripts is automatically used if the Auto start with the default template macro is activated. If the user provides its own Java macro script file, this one should contain an import statement to use the MpCCI-STAR-CCM+ adapter library.

```java
import mpcci.starccm.*;
import mpcci.client.*;
```

The user has the possibility to get a copy of the MpCCI Java macro template in order to update it to his convenience. Therefor use the command mpcci star-ccm+ -getmacro [steady|unsteady].

The `execute()` function could be implemented to start the simulation computation with some MpCCI commands to enable the data exchange.

16.5.1.1 Available Commands

The command for MpCCI calls within STAR-CCM+ is available after having instantiated the MpCCI adapter
```java
new Adapter(simulation);
```
The available routines have the following functionality:

**Adapter mpcci = new Adapter(simulation);**
- Instantiates the MpCCI adapter.
- The adapter functions are accessible by the variable `mpcci`.

**mpcci.initCouplingInfo();**
- Read MPCCI\_TINFO environment variable containing the coupling setting information.
- Part of these settings contain information settings from the duration control, subcycle.

**JMpCCIClient.MPCCI\_TINFO tinfo = mpcci.getTinfo();**
- Get the variable `tinfo` to access the information from MPCCI\_TINFO.

**mpcci.mpcciTinfoHasDuration();**
- Query if the Use duration control has been activated.
- For a steady state simulation you can access through `tinfo` variable from above the following information:
  - `tinfo.iter_cbeg`: Iteration No. for starting the coupling.
  - `tinfo.iter_cend`: Iteration No. for ending the coupling.
  - `tinfo.iter_pend`: No. of iterations after the coupling.
- After reading these information you must remove the iteration control from the MpCCI coupling manager by calling `mpcci.mpcciTinfoRemoveDurationCtl()`.
- For a transient simulation you can access through `tinfo` variable from above the following information:
  - `timeCplStart`: Time of coupling start.
  - `timeCplPost`: Time after the coupling.
  - `timeStepSize`: Solver time step size if provided from MpCCI GUI.
  - `maxInnerIterations`: Maximum number of iterations per time step.

**mpcci.mpcciTinfoRemoveDurationCtl();**
- Remove the control duration from the MpCCI coupling manager as this is implemented in the Java macro script.

**mpcci.mpcciTinfoHasSubcycle();**
- Query if the Use subcycling has been activated.
- The number of subcycle steps can be read from `tinfo.sub_step`. After reading this information you must remove the iteration control from the MpCCI coupling manager by calling `mpcci.mpcciTinfoRemoveSubcycleCtl()`.

**mpcci.mpcciTinfoRemoveSubcycleCtl();**
- Remove the subcycle control from the MpCCI coupling manager as this is implemented in the Java macro script. It avoids to subcycle twice.

**String coldStart = System.getenv("MPCCI\_STARCCM\_COLDSTART");**
- Get information if a cold start should be done. In that case the solution should be cleared before starting the computation.

**mpcci.init();**
- Initializes the MpCCI process.
- During that process the STAR-CCM+ model will be modified automatically for the coupling.

**mpcci.exchange();**
- Performs a transfer operation, first SEND/ followed by RECEIVE.
  - In case of a transient simulation using a grid morpher, the data exchange should be performed prior to the time step in order to provide updated values for the grid morpher.
mpcci.exit()

Performs an MpCCI quit and quits the MpCCI process within STAR-CCM+.

16.5.1.2 MpCCI Template Java Script: Steady State Simulation

This sample is extracted from the "<MpCCI_home>/codes/STAR-CCM+/adapters/macro/mpcci_runjob_steady.java" file.
This sample script is divided into 5 sections:

1. There are some steering variables for running the simulation:
   - nbCoupling: defines the number of data exchanges.
   - initIteration: defines the number of iterations before the coupling begins.
   - nbIteration: defines the number of iterations to be done between each coupling step.
   - postIteration: defines the number of iterations after the coupling.
   - lastIteration: defines the iteration number of the final coupling.

   These variables are initialized by the values set from the tinfo.

2. The setup of the simulation stop criteria is adjusted according to the steering variables.

3. The simulation will be initialized and the pre iterations executed.

4. The co-simulation phase begins:
   - The connection to the MpCCI server is established.
   - The solution will exchange data then iterate. This will loop for nbCoupling times.
   - The connection to the MpCCI server is closed.

5. The simulation will terminate with some post iterations.

```java
package macro;

import java.util.*;
import java.io.File;
import star.common.*;
import star.base.neo.*;
import star.vis.*;
import mpcci.starccm.*;
import mpcci.client.*;

public class mpcci_runjob_steady extends StarMacro {
    public boolean isImplicitUnsteadyModel(Simulation simulation) {
        try {
            simulation.getSolverManager().getSolver(ImplicitUnsteadySolver.class);
            return true;
        } catch (Exception e) {
            return false;
        }
    }

    public boolean isExplicitUnsteadyModel(Simulation simulation) {
        try {
            simulation.getSolverManager().getSolver(ExplicitUnsteadySolver.class);
            return true;
        } catch (Exception e) {
```
return false;
}

public boolean isSteadyModel(Simulation simulation) {
    return (!isImplicitUnsteadyModel(simulation)
        && !isExplicitUnsteadyModel(simulation));
}

public void execute() {
    // number of data exchanges
    int nbCoupling = 1;
    // number of initial iterations before the coupling begins
    int initIteration = 0;
    // number of iterations between coupling steps
    int nbIteration = 1;
    // number of iterations after the coupling
    int postIteration = 0;
    // last of iterations ending the coupling
    int lastIteration = 0;
    boolean abort = false;
    boolean removeInitialSolution = false;

    Simulation simulation = getActiveSimulation();
    if(!isSteadyModel(simulation)) {
        simulation.println("** ERROR ** The solution is not a steady state model:");
        simulation.println(" The MpCCI Java macro is only valid for a steady state model!");
        simulation.kill();
    }

    // instantiate MpCCI Co-simulation for CCM+
    Adapter mpcci = new Adapter(simulation);
    mpcci.initCouplingInfo();
    JMpCCIClient.MPCCI_TINFO tinfo = mpcci.getTinfo();

    String coldStart = System.getenv("_MPCCI_STARCCM_COLDSTART");

    if (mpcci.mpcciTinfoHasDuration()) {
        if (tinfo.iter_cbeg > 0) initIteration = tinfo.iter_cbeg;
        if (tinfo.iter_cend > 0) lastIteration = tinfo.iter_cend;
        if (tinfo.iter_pend > 0) postIteration = tinfo.iter_pend;
        mpcci.mpcciTinfoRemoveDurationCtl();
    }

    if (mpcci.mpcciTinfoHasSubcycle()) {
        if (mpcci.mpcciTinfoSubcycleType() == JMpCCIClient.MPCCI_TMODE_SUBCYCLE_CONST
            && tinfo.sub_step > 0)
        {
            }
nbIteration = tinfo.sub_step;
}
mpcci.mpcciTinfoRemoveSubcycleCtl();
}

if (coldStart != null && coldStart.length() > 0)
{
    removeInitialSolution = Boolean.parseBoolean(coldStart);
}

if (!mpcci.mpcciTinfoHasSubcycle() || mpcci.mpcciTinfoSubcycleType() == JMpCCIClient.MPCCI_TMODE_SUBCYCLE_CONST)
{
    int iterCount = 0;
    nbCoupling = (lastIteration - initIteration) / nbIteration;
    iterCount = (initIteration + (nbIteration * nbCoupling));
    if (iterCount < lastIteration)
    {
        postIteration = postIteration + (lastIteration - iterCount);
        lastIteration = iterCount;
    }
}

int currentIteration = simulation.getSimulationIterator().getCurrentIteration();

// total number of iterations for the solver
int totalIteration = lastIteration + postIteration;
if (!removeInitialSolution)
{
    totalIteration += currentIteration;
    lastIteration += currentIteration;
}

simulation.println("*** MpCCI Co-simulation setup for steady state analysis:");
if (mpcci.mpcciTinfoSubcycleType() == JMpCCIClient.MPCCI_TMODE_SUBCYCLE_CONST)
{
    simulation.println("*** Number of data exchange : " + nbCoupling);
    simulation.println("*** Number of sub-iteration : " + nbIteration);
}
else
{
    simulation.println("*** Number of sub-iteration       : VARIABLE");
}

simulation.println("*** Number of pre-iteration   : " + initIteration);
simulation.println("*** Number of post-iteration   : " + postIteration);
simulation.println("*** Number of existing iteration: " + currentIteration);
simulation.println("*** Total number of iteration: " + totalIteration);
simulation.println("*** Clear initial solution: " + removeInitialSolution);

// Get the maximum steps set from the sim project file and modify it
StepStoppingCriterion stepStoppingCriterion = null;
AbortFileStoppingCriterion abortFileStoppingCriterion = null;
Collection<Object> v = simulation.getSolverStoppingCriterionManager().getChildren();
Iterator<Object> it = v.iterator();
while(it.hasNext())
{
    Object stopingCriterion = it.next();
    if (stopingCriterion instanceof AbortFileStoppingCriterion)
    {
        abortFileStoppingCriterion = (AbortFileStoppingCriterion) stopingCriterion;
    } else
    if (stopingCriterion instanceof StepStoppingCriterion)
    {
        stepStoppingCriterion = (StepStoppingCriterion)stopingCriterion;
    }
}
if (stepStoppingCriterion != null) // reset the maximun steps for the model
{
    stepStoppingCriterion.setMaximumNumberSteps(totalIteration);
} else
{
    simulation.println("*** No Maximum number of step criterion has been found!");
}
if (abortFileStoppingCriterion != null) // activate stop file
{
    abortFileStoppingCriterion.setFilePath("ABORT");
    abortFileStoppingCriterion.setIsUsed(true);
    abortFileStoppingCriterion.setInnerIterationCriterion(true);
} else
{
    simulation.println("*** No stop file criterion has been found!
    An ABORT will not be possible");
}

File abortFile = new File(simulation.getSessionDir()+File.separator +"ABORT");

Solution solution = simulation.getSolution();
if (removeInitialSolution)
    solution.clearSolution();
// initialize the solver
if (!solution.isInitialized())
    solution.initializeSolution();
abort = abortFile.exists();
if (!abort && initIteration > 0)
{
    // Start a pre-computation without coupling
    simulation.println("Starting pre iterations...");
    simulation.getSimulationIterator().step(initIteration);
abort = abortFile.exists();
}

// Initialize MpCCI co-simulation
if (!abort && mpcci.init())
{
    currentIteration = simulation.getSimulationIterator().getCurrentIteration();
    int couplingStepNo = 0;
    while (currentIteration < lastIteration)
    {
        // Exchange the data to MpCCI
        mpcci.exchange();
        if (mpcci.mpcciTInfoSubcycleType() == JMpCCIClient.MPCCI_TMODE_SUBCYCLE_TABLE)
        {
            nbIteration = mpcci.mpcciTinfoGetSubcycleSizeAt(couplingStepNo);
            if ((nbIteration + currentIteration) >= lastIteration)
            {
                nbIteration = lastIteration - currentIteration;
                simulation.println("Reducing the number of sub-iterations to meet the maximum iterations criterion...");
            }
            simulation.println("Starting sub-iterations cycle #" + (couplingStepNo+1) + " with " + nbIteration + " iterations...");
        }
        else
        {
            simulation.println("Starting sub-iterations cycle " + (couplingStepNo+1) + "/" + nbCoupling + "...");
        }
        // iterate nbIteration before exchange
        simulation.getSimulationIterator().step(nbIteration);
        currentIteration = simulation.getSimulationIterator().getCurrentIteration();
        couplingStepNo++;
        abort = abortFile.exists();
        if (abort)
        {
            simulation.println("Aborting sub-iterations...");
            break;
        }
    }
    // Try last data exchange to MpCCI
    mpcci.exchange();
    // Finalize the co-simulation: close the communication with the server
    mpcci.exit();
}

// Terminate the CFD solution by performing some iterations: endIterations
// sim is automatically saved
if (!abort)
{

if (postIteration > 0)
{
    simulation.println("Starting post iterations");
    simulation.getSimulationIterator().step(postIteration);
}
else
{
    simulation.println("Solution has been aborted.");
    abortFile.delete();
}
}

16.5.1.3 MpCCI Template Java Script: Transient Simulation

This sample is extracted from the
"<MpCCI_home>/codes/STAR-CCM+/adapters/macro/mpcci_runjob_unsteady.java" file.
This sample script is divided into 5 sections:

1. There are some steering variables for running the simulation:
   - timeCplStart: defines time before starting the coupling.
   - timeCplEnd: defines time for ending the coupling.
   - maxInnerIterations: defines the number of iterations for a time step.
   - timeCplPost: defines time after the coupling.
   - changeTimeStepSizeSetting: possibility to change the initial time step size.
   - timeStepSize: time step size to use.
   - nbStep: defines the number of time steps without coupling (subcycle).
   - maxTotalTime: defines the total simulation time.

   These variables are initialized by the values set from the tinfo and the list of getenv calls.

2. The setup of the simulation stop criteria is adjusted according to the steering variables.

3. The co-simulation phase begins:
   - The connection to the MpCCI server is established.
   - The solution will exchange data then performs nbStep time steps. This will loop for the duration of maxTotalTime. This section takes in account the time before starting the coupling and the time after the coupling.
   - The connection to the MpCCI server is closed.

package macro;
import java.util.*;
import java.beans.*;
import java.io.File;
import star.common.*;
import star.base.neo.*;
import star.vis.*;
import mpcci.starccm.*;
import mpcci.client.*;

/**
* NOTE:
* Java Macro file valid for an implicit unsteady solver.
*/
public class mpcci_runjob_unsteady extends StarMacro{
    public boolean isImplicitUnsteadyModel(Simulation simulation) {
        try {
            simulation.getSolverManager().getSolver(ImplicitUnsteadySolver.class);
            return true;
        } catch (Exception e) {
        }
        return false;
    }
    public boolean isExplicitUnsteadyModel(Simulation simulation) {
        try {
            simulation.getSolverManager().getSolver(ExplicitUnsteadySolver.class);
            return true;
        } catch (Exception e) {
        }
        return false;
    }
    public boolean isSteadyModel(Simulation simulation) {
        return (!isImplicitUnsteadyModel(simulation)
            && !isExplicitUnsteadyModel(simulation));
    }
    public void execute() {
        // Time before starting the coupling
        double timeCplStart = 0.0;
        // Time for ending the coupling
        double timeCplEnd = 0.0;
        // Time after the coupling
        double timeCplPost = 0.0;
        // Number of time steps between coupling steps
        int nbStep = 1;

        // Parameter for the Stopping Criteria of the implicit unsteady solver:
        // flag to modify the setting in the sim file,
        // if false the setting from the sim file is used.
        boolean changeTimeStepSizeSetting = false;
        // time step size in seconds
        double timeStepSize = 0.00025;
        // number of maximum inner iterations for a time step
        int maxInnerIterations = 30;
        double currentTimeStepSize = 0.0;
        boolean abort = false;
        boolean removeInitialSolution = false;

        Simulation simulation = getActiveSimulation();
        if(isSteadyModel(simulation))
```java
{ 
    simulation.println("** ERROR ** The solution is not a unsteady model:");
    simulation.println("The MpCCI Java macro is only valid for a unsteady model!");
    simulation.kill();
}

// instantiate MpCCI Co-simulation for CCM+
Adapter mpcci = new Adapter(simulation);
mpcci.initCouplingInfo();
JMpCCIClient.MPCCI_TINFO tinfo = mpcci.getTinfo();

String timePend = System.getenv("_MPCCI_STARCCM_TIMEPEND");
String dt = System.getenv("_MPCCI_STARCCM_TIMEDT");
String maxIter = System.getenv("_MPCCI_STARCCM_TIMEITER");
String coldStart = System.getenv("_MPCCI_STARCCM_COLDSTART");

if (mpcci.mpcciTinfoHasDuration())
{
    if (tinfo.time_cbeg > 0.0)
    {
        timeCplStart = tinfo.time_cbeg;
    }
    if (tinfo.time_cend > 0.0)
    {
        timeCplEnd = tinfo.time_cend;
    }
}

// Check coupling time.
if (!(timeCplEnd - timeCplStart > 0))
{
    simulation.println("*** No coupling time has been specified in the
    MpCCI coupling configuration.");
    simulation.println("*** Simulation will exit after one coupled time step.");
}

if (timePend != null && timePend.length() > 0 )
{
    timeCplPost = Double.parseDouble(timePend);
}

// Set number of substeps.
if (mpcci.mpcciTinfoHasSubcycle())
{
    if (mpcci.mpcciTinfoSubcycleType() == JMpCCIClient.MPCCI_TMODE_SUBCYCLE_CONST
        && tinfo.sub_step > 0)
    {
        nbStep = tinfo.sub_step;
    }
}

// Set maximum inner iterations.
if (maxIter != null && maxIter.length() > 0)
{
```java
{  
    maxInnerIterations = Integer.parseInt(maxIter);  
}

// Set time step size.  
if (dt != null && dt.length() > 0 && Double.parseDouble(dt) > 0)  
{  
    changeTimeStepSizeSetting = true;  
    timeStepSize = Double.parseDouble(dt);  
}

// Set whether to remove the initial solution.  
if (coldStart != null && coldStart.length() > 0)  
{  
    removeInitialSolution = Boolean.parseBoolean(coldStart);  
}

// Set current simulation time.  
double currentTime = simulation.getSimulationIterator().getElapsedTime();

// Set total maximum time.  
double maxTotalTime = timeCplEnd + timeCplPost;  
if (!removeInitialSolution)  
{  
    maxTotalTime += currentTime;  
}

simulation.println("*** MpCCI Co-simulation setup for transient analysis:");  
simulation.println("*** Coupling time : " + (timeCplEnd - timeCplStart));  
simulation.println("*** Time before coupling : " + timeCplStart);  
if (mpcci.mpcciTinfoSubcycleType() == JMpCCIClient.MPCCI_TMODE_SUBCYCLE_CONST)  
{  
    simulation.println("*** Number of time steps without coupling: " + nbStep);  
}  
else  
{  
    simulation.println("*** Number of time steps without coupling: VARIABLE");  
}

simulation.println("*** Number of inner iteration/time step : " + maxInnerIterations);  
simulation.println("*** Time post coupling : " + timeCplPost);  
simulation.println("*** Existing time : " + currentTime);  
simulation.println("*** Total time : " + maxTotalTime);  
if (isImplicitUnsteadyModel(simulation))  
{  
    ImplicitUnsteadySolver implicitUnsteadySolver = ((ImplicitUnsteadySolver)  
        simulation.getSolverManager().getSolver(ImplicitUnsteadySolver.class));

    if (changeTimeStepSizeSetting) // Set the initial time step size  
    {  
        implicitUnsteadySolver.getTimeStep().setValue(timeStepSize);  
        simulation.println("*** New time step size value : " + timeStepSize);  
    }
```

currentTimeStepSize = implicitUnsteadySolver.getTimeStep().getSIValue();
simulation.println("*** Current time step size value : "
  + currentTimeStepSize);
simulation.println("*** Unsteady model : implicit");
} else {
  simulation.println("*** Unsteady model : explicit");
}
simulation.println("*** Clear initial solution : " + removeInitialSolution);

// Get the maximum time set from the sim project file and modify it
StepStoppingCriterion stepStoppingCriterion = null;
AbortFileStoppingCriterion abortFileStoppingCriterion = null;
InnerIterationStoppingCriterion innerIterationStoppingCriterion = null;
PhysicalTimeStoppingCriterion physicalTimeStoppingCriterion = null;
Collection<Object> v = simulation.getSolverStoppingCriterionManager().getChildren();
Iterator<Object> it = v.iterator();
while(it.hasNext()) {
  Object stoppingCriterion = it.next();
  if (stoppingCriterion instanceof AbortFileStoppingCriterion) {
    abortFileStoppingCriterion = (AbortFileStoppingCriterion) stoppingCriterion;
  } else if (stoppingCriterion instanceof StepStoppingCriterion) {
    stepStoppingCriterion = (StepStoppingCriterion) stoppingCriterion;
  } else if (stoppingCriterion instanceof InnerIterationStoppingCriterion) {
    innerIterationStoppingCriterion = (InnerIterationStoppingCriterion) stoppingCriterion;
  } else if (stoppingCriterion instanceof PhysicalTimeStoppingCriterion) {
    physicalTimeStoppingCriterion = (PhysicalTimeStoppingCriterion) stoppingCriterion;
  }
}

// Reset maximum physical time stop criteria.
if (physicalTimeStoppingCriterion == null) {
  simulation.println("*** No maximum time criterion has been found, set own.");
  // Need a name for the stopping criterion
  physicalTimeStoppingCriterion = simulation.getSolverStoppingCriterionManager()
      .createSolverStoppingCriterion(PhysicalTimeStoppingCriterion.class,
      "maxTotalTime");
}
physicalTimeStoppingCriterion.setMaximumTime(maxTotalTime);
physicalTimeStoppingCriterion.setIsUsed(true);
if (stepStoppingCriterion != null) // Deactivate the maximum steps for the model.
{
    stepStoppingCriterion.setIsUsed(false);
}

if (abortFileStoppingCriterion != null) // activate stop file
{
    abortFileStoppingCriterion.setFilePath("ABORT");
    abortFileStoppingCriterion.setIsUsed(true);
    abortFileStoppingCriterion.setInnerIterationCriterion(true);
}
else
{
    simulation.println("*** No stop file criterion has been found!
                             An ABORT will not be possible");
}

if (innerIterationStoppingCriterion != null)
// set the maximum number of inner iterations for the time step
{
    innerIterationStoppingCriterion.setIsUsed(true);
    innerIterationStoppingCriterion.setMaximumNumberInnerIterations(maxInnerIterations);
}
else
{
    simulation.println("*** No inner iteration criterion has been found!");
}

File abortFile = new File(simulation.getSessionDir()+File.separator +"ABORT");

Solution solution = simulation.getSolution();

if (removeInitialSolution)
    solution.clearSolution();
// initialize the solver
if (!solution.isInitialized())
    solution.initializeSolution();

abort = abortFile.exists();

SimulationIterator simulationIterator = simulation.getSimulationIterator();

// Initialize MpCCI co-simulation
if (!abort && mpcci.init())
{
    // Count number of coupling steps for info output and getting
    // number of substeps in case of subcycle table.
    int cstep = 0;
    boolean subcycleTable = mpcci.mpcciTinfoSubcycleType()
        == JMpCCIClient.MPCCI_TMODE_SUBCYCLE_TABLE;

// Perform data exchanges until stopping criterion is satisfied.
while (!physicalTimeStoppingCriterion.getIsSatisfied())
{
    // Exchange the data to MpCCI
    mpcci.exchange();
    simulation.println("Starting coupling cycle \"+ (cstep + 1) + \" at \" + solution.getPhysicalTime() + \"s\" + maxTotalTime + \"s \...\"\);
    if (subcycleTable)
    {
        nbStep = mpcci.mpcciTinfoGetSubcycleSizeAt(cstep);
    }
    simulation.println("Running \" + nbStep + \" time step(s)\...");
    simulationIterator.run(nbStep);

    // Check abort.
    abort = abortFile.exists();
    if (abort)
    {
        simulation.println("Aborting time step cycle\...");
        break;
    }
    // Next coupling step.
    cstep++;
}
// Finalize the co-simulation: close the communication with the server.
mpcci.exit();

if (abort)
{
    simulation.println("Solution has been aborted." );
    abortFile.delete();
}
}
16.6 Trouble Shooting, Open Issues and Known Bugs

- Under Microsoft Windows please avoid the usage of white space in your working directory containing the STAR-CCM+ model file. The loading of the adapter library would fail.
17 TAITherm

17.1 Quick Information

<table>
<thead>
<tr>
<th>Company name</th>
<th>ThermoAnalytics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company homepage</td>
<td><a href="http://www.thermoanalytics.com">www.thermoanalytics.com</a></td>
</tr>
<tr>
<td>Support</td>
<td><a href="http://www.thermoanalytics.com/support">www.thermoanalytics.com/support</a></td>
</tr>
<tr>
<td>Tutorials</td>
<td>⊳ VII-9 Cube in a Duct Heater ⊳</td>
</tr>
</tbody>
</table>

17.1.1 Supported Coupling Methods

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Dimension</th>
<th>Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit</td>
<td>Implicit</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Radiation, Radiosity</td>
</tr>
</tbody>
</table>

17.1.2 Supported Platforms and Versions

The following versions of TAITherm are supported by MpCCI:

<table>
<thead>
<tr>
<th>MPCCI ARCH:</th>
<th>Supported versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code platform</td>
<td>120.0</td>
</tr>
<tr>
<td>inx4_x64:</td>
<td>X</td>
</tr>
<tr>
<td>x86_64:</td>
<td>X</td>
</tr>
<tr>
<td>windows_x64:</td>
<td>X</td>
</tr>
</tbody>
</table>

17.1.3 References

TAITherm Documentation is part of the TAITherm distribution.

17.1.4 Adapter Description

The code adapter for TAITherm is based on a shared library "libradthermmpci.so|dll" which is loaded by TAITherm and it includes the necessary interface functions.

17.1.5 Prerequisites for a Coupled Simulation

To run a coupled simulation you need the following:

- Ordinary TAITherm installation.

17.2 Coupling Process

Please read also ⊳ IV-2 Setting up a Coupled Simulation ⊳.
17.2.1 Model Preparation

Models can be prepared as usually. The view factors file may be already computed by TAITherm before starting the coupled simulation because it may take some time until the view factors file is created. You can execute the following TAITherm command to run the view factors computation separately:

`taitherm -runviewfactors <file.tdf>`

If the view factors have not been pre-computed before the coupled simulation start, TAITherm will perform this calculation as first step before solving the thermal problem. The coupling components must be defined as separate parts.

17.2.2 Models Step

In the Models step, the following options must be chosen:

**TAITherm program family**  Select the TAITherm program family from

- muses
- musesprome
- radtherm, (default)
- radthermir
- radthermrt

**TAITherm release**  Select the release of TAITherm you want to use. latest (default) will select the latest version which is installed on your system.

**.tdf file**  Select the "TAITherm *.tdf" file of your TAITherm model.

The MpCCI TAITherm scanner reads the "*.tdf" file and extracts all the parts by checking the following rules:

- the TAITherm part must not be empty. Parts having no elements are not scanned. For example Fluid part is not listed, because it has a single node and no geometry. This is not actually supported.
- TAITherm part having a rear side is automatically suffixed with "-backside" in the part name.
- The following part types are accepted for the coupling:
  - STANDARD: standard two sided part, standard (1-Layer) insulated part.
  - STANDARD_ASSIGNED: standard part with assigned temperature part.
  - MULTILAYER: multilayer part.
  - HIGH_CONDUCTIVE: high conductive part.
  - ENGINE: engine part.
- face: for backward compatibility reason.

- Duplicated part names will be automatically suffixed with the part ID.

17.2.3 Algorithm Step

For general options please refer to △V-4.5.2 Code Specific Algorithm Settings ◄.

In the Algorithm step, following additional options are available partly depending on the analysis type:

Solver settings

Convergence criteria The convergence criteria define the stopping criteria to apply.

- Fixed number of loops
  This option sets a maximum number of iterations within a subcycling step (cf. coupling steps setting) in case of a Steady state analysis type or the maximum number of iterations per time step for an Transient analysis type.

- Tolerance slope
  This option complements the Fixed number of loops by considering a tolerance slope value within the subcycling step in case of a Steady state analysis type or within the iterations per time step for an Transient analysis type.

According to the selected Convergence criteria, different options will be proposed in the different setting sections. This allows to configure the solver behaviour when coupling with subcycling is activated (Figure 2) or for the transient analysis type.

17.2.3.1 Steady State Analysis Type

Solver settings

Tolerance slope (K) (only with Tolerance slope option) Set the tolerance slope to be used.

Tolerance slope refers to the rate at which the tolerance is changing. The tolerance refers to the maximum change in temperature for any one element in K (SI units) between the current and previous iteration.

The Tolerance slope is evaluated at each solver iteration step within the coupling step.

Current values Shows the current information from the TAITherm model:

- Solver tolerance slope (K) Displays the current solver setting used by the current model file for the tolerance slope.
- Maximum number of solver iterations Displays the current solver setting used by the current model file for the maximum value of iterations.
- Solver total runtime (s) Displays the current solver setting used by the current model file for the duration.

Coupling steps

Constant maximum coupling step size (solver steps) (only with Tolerance slope option) Defines the maximum number of iterations without coupling to perform.

Minimum coupling step size (solver steps) (only with Tolerance slope option) Defines the minimum number of iterations without coupling to perform at least before activating the selected Tolerance slope convergence criterion.
17.2.3.2 Transient Analysis Type

Solver settings

Tolerance slope (K) \textit{(only with Tolerance slope option)} Set the tolerance slope to be used. Tolerance slope refers to the rate at which the tolerance is changing. The tolerance refers to the maximum change in temperature for any one element in K (SI units) between the current and previous iteration. The Tolerance slope is evaluated at each solver inner iteration step within the time step.

Type of solver step size Define the solver time step type to use:
- Defined by model Use the time step size defined in the input file. This allows you to use a curve definition for the time step size.
- Constant Overwrite the time step size setting with a new constant time step size value provided in the MpCCI GUI.

Solver step size (s) \textit{only with Constant option} Define the solver time step size to use.

Number of inner iterations \textit{(only with Fixed number of loops option)} Specifies the value for a fixed number of inner iterations during a solver step.

Maximum number of inner iterations \textit{(only with Tolerance slope option)} Specifies the value for a maximum number of inner iterations during a solver step.

Current values Groups the options for starting and ending the coupling. The additional information for TAITherm solver settings are:

Solver tolerance slope (K) Displays the current solver setting used by the current model file for the tolerance slope.

Solver step size (s) Displays the current solver setting used by the current model file for the time step size.

Maximum number of solver iterations Displays the current solver setting used by the current model file for the maximum number of solver iterations.

Solver total runtime (s) Displays the current solver setting used by the current model file for the duration.

17.2.4 Regions Step

The TAITherm adapter only stores quantities directly (“Direct”). TAITherm supports the following quantities for coupling:
### 17.2.5 Go Step

In the Go step the following options can be chosen:

**Initialization parameters** Groups the options which can be set for initialization (cf. Figure 3).

**Current model initialization method** The initialization method used by the current model file. This value is delivered by the scanner.

**Tdf model initialization** Choosable initialization types for the tdf model are (additional information about these options can be found in TAITherm manual):

- **PART_INITIAL_TEMPERATURES** This option will use the initial temperature that was set for each part to start the steady state solution.
- **SEED_STEADY_STATE** This option allows you to specify the initial temperature used by the solution to come from the results of a previously computed solution. This feature can provide faster steady state convergence in new models.
  - If you choose SEED_STEADY_STATE and do not browse to a file, it will automatically try to seed from the current file (self-seed).
  - If the seed model does not have results, the initialization method falls back to PART_INITIAL_TEMPERATURES.
- **TRANSIENT_RESTART** This option allows you to continue a transient thermal solution with a new file.

---

**Figure 3:** TAITherm options for Initialization parameters in the Go step.
**TRANSIENT_INITIALIZATION**  Similar to **TRANSIENT_RESTART**, **TRANSIENT_INITIALIZATION** allows you to continue a transient thermal solution with a new file. The only difference between the two options is that the tdf file with the transient results does not have to exist when selecting **TRANSIENT_INITIALIZATION**.

**Tdf model initialization file**  The initialization file used for the tdf model. Not available for tdf model initialization type **PART_INITIAL_TEMPERATURES**. This value is delivered by the scanner.

**Initialization from another tdf file**  Choose another tdf file for initialization. Not available for tdf model initialization type **PART_INITIAL_TEMPERATURES**.

**Use absolute path for the initialization file**  The selected initialization file will be saved in the current tdf file with the absolute path reference. Not available for tdf model initialization type **PART_INITIAL_TEMPERATURES**.

**View factors file status**  Indicates the status of the view factors file which is delivered by the scanner. Default is **Missing** meaning it is not delivered by the scanner.

<table>
<thead>
<tr>
<th>Write frequency</th>
<th>1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Write intermediate results</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Iteration interval</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Modify solution output frequency</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Iteration frequency</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Define baffle thickness location</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Additional command line options</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Figure 4: Output and start settings.](image)

**Write frequency**  Defines the frequency (in time step) to store the results (cf. Figure 4). For steady state simulation the results are saved at the end of the calculation.

- For **Explicit-SteadyState** coupling this value is set to 1.

**Write intermediate results**  If this option is enabled, an additional tdf file will be written at each iteration interval during the thermal solution. Each tdf file will contain the model data and the thermal results from that point in the simulation.

**Iteration interval**  Defines the iteration interval at which the tdf file will be saved.

**Modify solution output frequency**  If this option is enabled, you can define the output frequency of the solver residual.

**Iteration frequency**  Defines the iteration frequency to output the solver residual.

**Define baffle thickness location**  If the option **BaffleShift** from ▶ V-4.8.2 Job ◅ is activated, this option **Define baffle thickness location** will be displayed. The user should provide the thickness information about how the TAITherm part having layer properties has been defined. This defines the reference layer used for the thickness definition. The user has the following options:

- **None**  The front and rear side will not be shifted. (default)
- **Backplane**  The rear side is used as reference location to shift the front side.
- **Frontplane**  The front side is used as reference location to shift the rear side.
- **Middle**  Both sides are shifted from the middle of the thickness value.
The front and rear sides are virtually separated in the MpCCI server in the geometric space if one of the options Backplane, Frontplane or Middle is selected.

Additional command line options Additional command line options for TAITherm can be given here, they will directly be used when TAITherm is started.

Run in batch mode If you want to run TAITherm in batch mode, select this option.

Figure 5: TAITherm Parallel processing options.

Parallel processing options Display configuration for the parallel processing (Figure 5):

- **Processing model** List of available parallel processing model
  - **Serial** Run the thermal solution using one processor.
  - **Shared memory parallel** This option is applicable to a single system.
    - **No. of solver threads** Set the number of threads to use on a multi-threaded system based.
  - **Distributed memory parallel** This option is useful for a single system or a series of connected machines.
    - **No. of processes** Set the number of processes that the solver will use. Each process will be configured with one thread.
    - **Use rsh to spawn the processes** Specify RSH method for spawning the processes over the list of machines. Default is SSH.
    - **Option 'host host...' to be used** Enter host names for the parallel execution.
    - **Optional hostlist file** Specify a hostfile from which the host names are extracted.
    - **Use default hostfile** A default hostfile can be configured by setting MPCCI_HOSTLIST_FILE

17.2.6 Checking the Computation

TAITherm provides a configuration checker as described in V-4.9.2 Checking the Configuration which
- checks if the initialization file is not omitted in case of transient restart or initialization method.
- checks the convergence criteria and solution parameters settings. For example:
  - checks if the duration value is greater or equal the time for ending the coupling.
  - checks if the maximum value of iterations is large enough according to the settings made in convergence criteria.
17.2 Coupling Process

– checks if the Minimum no. of steps without coupling does not exceed the Maximum no. of steps without coupling.
– checks the variable subcycling table definition.

• checks that there is no mixing part in the coupled regions: front side and rear side parts are not allowed to be selected in the same coupled region.
• provides a coupling configuration summary.

17.2.7 Running the Computation

By clicking on the Start button from MpCCI GUI, MpCCI

• checks that there is no mixing part in the coupled regions: front side and rear side parts are not allowed to be selected in the same coupled region.
• starts a tool to prepare the "tdf" file for co-simulation (see ▶ 17.2.7.1 Hook Functions ◄).
• does a local copy of the MpCCI TAITherm adapter library before the code starts. It facilitates the port of the TAITherm "tdf" file.

The TAITherm computation is started, when using the GUI mode (non-batch mode), by clicking the Run button in section Analyse of the TAITherm GUI, otherwise TAITherm runs in batch mode.

When running a steady state simulation, TAITherm will save the results when either the total number of Maximum # iterations or the Tolerance Slope (K) criterion is satisfied. The Maximum # iterations is adjusted in the tdf file by MpCCI before the simulation starts. In MpCCI GUI the fields Iteration no. for ending the coupling and No. of iterations after the coupling are used to calculate this criterion. The Tolerance slope(K) is also adjusted in the tdf with the defined value from MpCCI GUI if the Convergence criteria is selected to Tolerance slope.

If TAITherm gets disconnected from the coupling server, the TAITherm solution will continue until the defined convergence criterion is reached.

When running a transient simulation, TAITherm saves the data at specified Write frequency value.

If TAITherm gets disconnected from the coupling server, the TAITherm solution will abort the current time step where the incident occurred and the results should be saved by TAITherm.

Using the MpCCI GUI Stop button executes a stopper script which sends a stop signal to TAITherm. Receiving this signal, TAITherm is able to terminate and save the tdf results.

17.2.7.1 Hook Functions

By starting the TAITherm computation the model will be automatically set up by MpCCI.

• MpCCI automatically hooks the Solution Start, Time Step Start, Iteration Start, Iteration End and Solution End TAITherm hook functions. The MpCCI TAITherm adapter decides which hook function to use according to the analysis type (Steady state or Transient).

• TAITherm model is automatically checked: the part convection type will be adjusted to TAITherm HandTfluid property for the co-simulation with the following default values:
  – \( \frac{W}{m^2K} \) for heat transfer coefficient.
  – 273 K for the film temperature.

Running TAITherm with GUI allows you to see the changes applied to the model.

You can check the hooks setting by selecting the Analyse→Params section in the TAITherm GUI in order to access the hook functions setup Figure 6. By clicking on the button HookFunctions... the setup window will appear. You have to set up these functions for the following available TAITherm hooks:
17 TAITherm

17.2.7.2 Parallel Execution

There are two ways to launch TAITherm in parallel:

**In batch mode**  By activating the batch mode you may specify the number of parallel solver threads.

**In TAITherm GUI**  Before starting the TAITherm computation you need to check the number of processors to use set by MpCCI GUI, see

- Edit $\rightarrow$ Preferences $\rightarrow$ Application for older release than TAITherm 11.2 Figure 7
- Edit $\rightarrow$ Preferences $\rightarrow$ Parallel Processing Figure 8.
17.2.8 Post-Processing

After a coupled simulation has finished the results computed on the TAITherm side may be visualized by using the TAITherm post-processing tool.
17 TAITherm

17.3 Code-Specific MpCCI Commands

The MpCCI subcommands available for TAITherm are:

Usage:
    mpcci TAITherm [-]option

Synopsis:
    'mpcci TAITherm' is used to get information about TAITherm.

Options:
    -clean [-r release] <tdf file>
        Clean up MpCCI hooks from tdf file.

    -diff <tdf1> <tdf2>
        Run the scanner on two tdf files and print the differences.

    -help
        This screen.

    -importCFD [-r release] [-tcd tcd_filename] <-csp project> <-out tdf_output>
        Generate the file setting and execute the command for the importCFD option.

    -info
        List verbose information about all TAITherm releases.

    -releases
        List all TAITherm releases which MpCCI can find.

    -scan [-r release] <tdf file>
        Run the scanner and create a scanner output file.

The subcommands [diff, info, releases] and scan are described in 1.1 Common MpCCI Subcommands for Simulation Codes.
mpcci taitherm -clean [-r release] <tdf file>

The `clean` subcommand cleans up the MpCCI hooks from the given tdf file. All parts of type assigned temperature with a convection type HandTfluid will be set back without convection (value none for the convection type). If no release is specified, the latest one will be taken.

mpcci taitherm -importCFD [-r release] [-tcd tcd_file] -csp <project> -out <tdf output>

This option allows the user to import convection boundary conditions from an external CFD program using a transient convection data file (-tcd). The file to be used may be passed to the option -tcd. The .tcd file is a free-formatted file that holds fluid temperatures and convection coefficients from CFD analysis (see TAITherm User Manual for Transient Convection Data for details of the TCD format).

The `importCFD` subcommand generates a file setting and executes the command for the `importCFD` option from TAITherm. The file setting contains following options:

- the mesh mapping option is set to Off.
- The append option is to
  - Off for a steady state calculation or the first import of a transient calculation.
  - On for a transient calculation.

The option value depends on the analysis type (Steady state or Transient) set during the co-simulation.

- the imported values are set on convection coefficients and film temperature.
- The list of parts is generated according to the list of coupled components defined in the MpCCI project file -csp. The option for Front, Back, Both is automatically set.
- The CFD results to import is provided by the option -tcd to add only one results file or by omitting this option the selection of the tcd files is automatically done based on the analysis type performed during the co-simulation:
  - select the latest tcd file for a steady state from the model directory.
  - select all tcd files for a transient calculation from the model directory.

The TAITherm setup with imported CFD data is saved in an new tdf file by the option -out. The tdf file to setup is automatically extracted from the MpCCI csp file. The directory containing the tdf file is used as model directory and will contain the file setting "mpcci_cfdimport_setting_INDEX.txt" and the new tdf file. INDEX corresponds to the import step number. The import step number zero has will always remove all existing records from the tdf file. You may keep track of the import process executed by MpCCI by executing by yourself the command: taitherm -importCFD "mpcci_cfdimport_setting_INDEX.txt" -save new.tdf

The parts of type assigned will not be included in the import setting file if they have been used during the co-simulation.
17.4 Code Adapter Reference

- Within the MpCCI distribution the "adapters" directory contains the necessary software to connect the simulation programs to MpCCI. The files are located within the subdirectory 
  
  
  
  
  "<MpCCI_home>/codes/TAITherm/adapters".

This subdirectory is further divided by several release subdirectories, e.g. "10.5.0" and "11.0.0". The version directories are further divided by several architectures (e.g. "x86_64", "win64"). There you find the library files of the TAITherm adapter (e.g. "libradthermmppcci.so"). The connection to MpCCI is established using these shared libraries.

MpCCI adapter recognizes parts with layer definition and automatically defines the method to shift the front and rear side part to the MpCCI server.

Only constant thickness over the part is defined and supported.

- To prepare and check the "TDF" model file MpCCI provides a tool based on the TAITherm TDFIO library to process the model file. The files are located within the subdirectory 
  
  
  
  "<MpCCI_home>/codes/TAITherm/bin".

This subdirectory is further divided by several release subdirectories, e.g. "10.5.0" and "11.0.0". The version directories are further divided by several architectures (e.g. "x86_64", "win64"). There you find the executable of the MpCCI TAITherm utility tool "mpcci_radthermutil.exe".

17.4.1 Quantity Handling

Following paragraph describes the default rules applied by the code adapter for the following quantities prescribed to TAITherm:

- Negative wall temperature values are reset to zero.
- Negative wall heat transfer coefficient values are reset to zero.
- Negative film temperature values are reset to zero.
- Negative wall heat flux values are inverted.
Part VII

Tutorial

Version 4.6.1
VII Tutorial – Contents

1 Introduction 8

2 Vortex-Induced Vibration of a Thin-Walled Structure 11
  2.1 Problem Description .................................................. 11
  2.2 Model Preparation .................................................. 11
    2.2.1 Fluid Model .................................................. 12
    2.2.2 Solid Model .................................................. 13
  2.3 Setting Up the Coupled Simulation with MpCCI GUI .......... 14
    2.3.1 Models Step .................................................. 14
    2.3.2 Algorithm Step .................................................. 17
    2.3.3 Regions Step .................................................. 18
    2.3.4 Go Step .................................................. 19
  2.4 Running the Computation .......................................... 21
  2.5 Discussion of Results ............................................. 21

3 Driven Cavity 23
  3.1 Problem Description .................................................. 23
  3.2 Model Preparation .................................................. 23
    3.2.1 Fluid Model .................................................. 24
    3.2.2 Solid Model .................................................. 24
  3.3 Setting Up the Coupled Simulation with MpCCI GUI .......... 24
    3.3.1 Models Step .................................................. 25
    3.3.2 Algorithm Step .................................................. 26
    3.3.3 Regions Step .................................................. 27
    3.3.4 Settings Step .................................................. 29
    3.3.5 Go Step .................................................. 29
  3.4 Running the Computation .......................................... 30
  3.5 Discussion of Results ............................................. 30
    3.5.1 Coupling with Adaptive Time Step ........................ 32

4 Elastic Flap in a Duct 34
  4.1 Problem Description .................................................. 34
  4.2 Model Preparation .................................................. 34
    4.2.1 Solid Model .................................................. 34
    4.2.2 Fluid Model .................................................. 36
  4.3 Setting Up the Coupled Simulation with MpCCI GUI .......... 37
    4.3.1 Models Step .................................................. 37
    4.3.2 Algorithm Step .................................................. 40
### Contents VII Tutorial

4.3.3 Regions Step ............................................. 41
4.3.4 Go Step .................................................. 43

4.4 Running the Computation ................................... 45
4.4.1 Starting the Simulation ................................ 45
4.4.2 End of the Simulation .................................. 46

4.5 Discussion of Results ....................................... 46

5 Exhaust Manifold ............................................. 48
5.1 Problem Description ........................................ 48

5.2 Model Preparation .......................................... 48
5.2.1 Solid Model ............................................. 49
5.2.2 Fluid Model ............................................. 50
5.2.3 Uncoupled Flow Simulation ............................ 51
5.2.4 Prepare Models for Coupled Simulation ............... 51

5.3 Setting Up the Coupled Simulation with MpCCI GUI ............... 51
5.3.1 Models Step ............................................. 51
5.3.2 Algorithm Step .......................................... 53
5.3.3 Regions Step ............................................ 55
5.3.4 Go Step ................................................ 56

5.4 Running the Computation ................................... 58

5.5 Post-Processing ............................................. 58

6 Busbar System ................................................. 60
6.1 Problem Description ........................................ 60

6.2 Model Preparation .......................................... 61
6.2.1 Fluid Model ............................................. 61
6.2.2 Electromagnetic Model ................................ 61

6.3 Setting Up the Coupled Simulation with MpCCI GUI ............... 62
6.3.1 Models Step ............................................. 62
6.3.2 Algorithm Step .......................................... 63
6.3.3 Regions Step ............................................ 64
6.3.4 Settings Step ............................................ 66
6.3.5 Go Step ................................................ 66

6.4 Running the Computation ................................... 67

6.5 Discussion of Results ....................................... 67
7 Three Phase Transformer
7.1 Problem Description
7.2 Model Preparation
7.2.1 Fluid Model
7.2.2 Electromagnetic Model
7.3 Setting Up the Coupled Simulation with MpCCI GUI
7.3.1 Models Step
7.3.2 Algorithm Step
7.3.3 Regions Step
7.3.4 Go Step
7.4 Running the Computation
7.5 Discussion of Results

8 Pipe Nozzle
8.1 Problem Description
8.2 Model Preparation
8.2.1 Fluid Model
8.2.2 Solid Model
8.3 Setting Up the Coupled Simulation with MpCCI GUI
8.3.1 Models Step
8.3.2 Algorithm Step
8.3.3 Regions Step
8.3.4 Go Step
8.4 Running the Computation
8.5 Discussion of Results

9 Cube in a Duct Heater
9.1 Problem Description
9.2 Model Preparation
9.2.1 Radiation Model
9.2.2 Fluid Model
9.3 Setting Up the Coupled Simulation with MpCCI GUI
9.3.1 Models Step
9.3.2 Algorithm Step
9.3.3 Regions Step
9.3.4 Go Step
9.4 Running the Computation
9.4.1 Starting the Simulation
9.5 Discussion of Results
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3.2 Algorithm Step</td>
<td>139</td>
</tr>
<tr>
<td>12.3.3 Regions Step</td>
<td>139</td>
</tr>
<tr>
<td>12.3.4 Settings Step</td>
<td>141</td>
</tr>
<tr>
<td>12.3.5 Go Step</td>
<td>141</td>
</tr>
<tr>
<td>12.4 Running the Computation</td>
<td>141</td>
</tr>
<tr>
<td>12.5 Discussion of Results</td>
<td>142</td>
</tr>
</tbody>
</table>
## 1 Introduction

The tutorial is a collection of examples. The files needed to run these examples are included in the *MpCCI* distribution in the directory `~<MpCCI_home>/tutorial`. All analysis steps are explained in detail. Please keep in mind that the descriptions are written to demonstrate the usage of *MpCCI*, they are not very useful if you want to learn how to use the simulation codes.

### Overview of Tutorials

<table>
<thead>
<tr>
<th>Example</th>
<th>Physical domains</th>
<th>Codes</th>
<th>Quantities</th>
<th>Analysis type</th>
<th>Special topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Fluid-Structure Interaction</td>
<td>Fluid Mechanics: FINE/Open, FINE/Turbo, FLUENT, STAR-CCM+</td>
<td>NPosition, RelWallForce</td>
<td>Transient with pre-computed flow field</td>
<td>Different unit systems</td>
</tr>
<tr>
<td>4</td>
<td>Fluid-Structure Interaction</td>
<td>Solid Mechanics: Abaqus, MSC NASTRAN</td>
<td>DeltaTime, NPosition, RelWallForce</td>
<td>Transient</td>
<td>Exchange of time step size, grid morphing</td>
</tr>
<tr>
<td>5</td>
<td>Thermal Coupling Fluid-Solid</td>
<td>Solid Mechanics: Abaqus, ANSYS, MSC Marc</td>
<td>FilmTemp, WallHTCoeff, WallTemp</td>
<td>Steady-state with pre-computed flow field</td>
<td>Subcycling of fluid solver</td>
</tr>
</tbody>
</table>
6 Busbar System
- Physical domains: Electrothermal Analysis
- Codes: Fluid Mechanics: FLUENT
  Electromagnetism: ANSYS, JMAG
- Quantities: JouleHeat, Temperature
- Analysis type: Steady-state flow field calculation and
  magnetic calculation in frequency domain

7 Three Phase Transformer
- Physical domains: Electrothermal Analysis
- Codes: Fluid Mechanics: FLUENT
  Electromagnetism: JMAG
- Quantities: JouleHeat, Temperature
- Analysis type: Steady-state flow field calculation and
  magnetic calculation in frequency domain

8 Pipe Nozzle
- Physical domains: Fluid-Structure Interaction
- Codes: Fluid Mechanics: FLUENT, STAR-CCM+
  Solid Mechanics: ANSYS, MSC NASTRAN
- Quantities: RelWallForce
- Analysis type: Steady-state unidirectional exchange
- Special topics: Axisymmetry, different unit systems

9 Cube in a Duct Heater
- Physical domains: Thermal Coupling Fluid
- Codes: Radiation: TAITherm
  Fluid Mechanics: FLUENT, OpenFOAM, STAR-CCM+
- Quantities: FilmTemp, WallHTCoeff, WallTemp
- Analysis type: Steady-state radiative heat transfer
- Special topics: Subcycling

10 Y-Junction
- Physical domains: Fluid Mechanics and Fluid Pipe System
- Codes: Network: FloMASTER
  Fluid Mechanics: FLUENT, OpenFOAM, STAR-CCM+
- Quantities: MassFluxRate, TotalPressure, MassFlowRate
- Analysis type: Steady-state
11 Spring Mass System

Physical domains: Spring-Mass-System with a split mass (analytical solution is known)
Codes: Model A: SimulatorA (simple C executable), MATLAB
Model B: Abaqus, MATLAB, MSC Adams, SIMPACK, SimulatorB (simple C executable).
Quantities: Force, PointPosition, additionally Acceleration for coupling with MSC Adams
Analysis type: Iterative transient
Special topics: Iterative coupling, non-matching time step sizes, comparison of iterative to explicit coupling

12 Periodic Rotating Fan Model

Physical domains: Fluid-Structure Interaction
Codes: Fluid Mechanics: FLUENT
Solid Mechanics: Abaqus
Quantities: NPosition, RelWallForce
Analysis type: Steady-state
Special topics: Periodic models with different section shapes
2 Vortex-Induced Vibration of a Thin-Walled Structure

2.1 Problem Description

![Figure 1: Vortex-induced vibration: Geometry and boundary conditions, the dimensions are given in cm.](image)

Topics of this Tutorial
- Fluid-Structure Interaction (FSI)
- Models with different unit systems (structural model: cgs units, fluid model: SI units).
- 2D-model

Simulation Codes
- Fluid Mechanics: FINE/Open, FINE/Turbo, FLUENT, STAR-CCM+
- Solid Mechanics: Abaqus, ANSYS, MSC Marc, MSC NASTRAN

Description
The example is taken from an article by Walhorn et al. [2002] and is originally based on a computation by Wall [1999].

The computation consists of two parts. First, only the flow field is computed until a Von Kármán vortex street develops behind the square. This first step is computed without coupling, the flexible structure remains rigid. After this initial calculation, a coupled analysis is carried through, the structure is now flexible. This yields vibration of the structure with increasing amplitude.

2.2 Model Preparation

The simulation couples a solid mechanics model with a fluid mechanics model. The files which you need for the simulation are included in the MpCCI distribution. Create a new directory and copy the particular subdirectories from "<MpCCI home>/tutorial/VortexVibration" which correspond to the simulation codes you want to use.
2.2.1 Fluid Model

The fluid domain comprises the whole rectangular area in Figure 1 except for the rigid square and the flexible structure.

The fluid has a constant density of $\rho = 1.18 \cdot 10^{-3} \frac{g}{cm^3}$ and a viscosity $\mu = 1.82 \cdot 10^{-4} \frac{g}{cm\cdot s}$.

The information about mesh properties and the steps to provide the initial solution can be found below, depending on your chosen CFD code:

**FINE/Open**

The mesh has been created by HEXPRESS and is built as nonconformal mesh. The surface of the flexible structure is defined by the components “VortexVibration_group_6”, “VortexVibration_group_7” and “VortexVibration_group_8”.

In order to calculate the Von Kármán vortex street serving as initial condition for the coupled simulation perform the following steps:

- Open the FINE/Open GUI and select the project vortexVibration.iec
- Make the computation preComputation active. It will run for 2.16 s.
- Save the run file and start the computation. This computation is used as initial solution for the coupled simulation.
- Select the serial computation name and replace the initial solution file by selecting the "run" file from the computation preComputation.
- Save the run file.
- Close the FINE/Open GUI and proceed with the MpCCI GUI.

The computation project name serial has been set for a 2D co-simulation analysis. The parameter iMpCCIzSize is set to 0.002 which is the length of the cell in z-direction. This parameter can be set from the Local parameters in the Advanced menu of the Control Variables in the Computation Control section.

**FINE/Turbo**

The mesh has been created by IGG and is built as structured mesh. The surface of the flexible structure is defined by the components “Beam_1”, “Beam_2” and “Beam_3”.

In order to calculate the Von Kármán vortex street serving as initial condition for the coupled simulation perform the following steps:

- Open the FINE/Turbo GUI and select the project vortex.iec
- Make the computation init_ustd active. It will run for 2.16 s.
- Save the run file and start the computation. This computation is used as initial solution for the coupled simulation.
- Select the cpl_ustd computation and reset the initial solution file by selecting the "run" file from the computation init_ustd.
- Save the run file.
- Close the FINE/Turbo GUI and proceed with the MpCCI GUI.

**FLUENT**

The mesh was generated with Gambit. Around the flexible structure, the elements are smaller than those at the outer boundaries. The surface of the flexible structure is defined as a separate boundary named “flexible-struct”.

---

*VII Tutorial*
The Von Kármán vortex street serving as initial condition for the coupled simulation is pre-computed and is provided in terms of the file "FLUENT/plate-2160.dat".

**STAR-CCM+**

The mesh corresponds to the mesh which is used in FLUENT. The surface of the flexible structure is defined as a separate boundary named “fluid:flexible-struct”. This boundary region will later be addressed by the STAR-CCM+ grid morpher.

The Von Kármán vortex street serving as initial condition for the coupled simulation is pre-computed and is included in the simulation setup file "STAR-CCM+/plate2160.sim".

2.2.2 Solid Model

The solid part corresponds to a cantilevered transverse beam as shown in Figure 2.

![Figure 2: Vortex-induced vibration: flexible solid model and meshing parameters.](image)

The material is linear elastic with density \( \rho = 2.0 \text{ g/cm}^3 \), elastic modulus \( E = 2.0 \cdot 10^6 \text{ g/cm}^2 \text{s}^2 \) and Poisson’s ratio \( \nu = 0.35 \).

The mesh consists of 128 \times 6 linear 4-node plane-strain elements. All nodes at the left end are fixed, i.e. they cannot move in \( x \) and \( y \)-direction. The whole surface except for the fixed end is defined as an element set, which can be selected as coupling component.

All solid models are created using the cgs unit system, i.e. the units given above.

The solid model receives forces from the fluid model, which are computed with an out-of-plane thickness \( t = 1 \text{ m} \). Therefore the solid models must have the same thickness \( t = 100 \text{ cm} \).

**Abaqus**

In Abaqus, CPE4 elements are used. The meshing and modeling was performed in Abaqus/CAE.

**ANSYS**

In ANSYS, element PLANE42 is used, the coupling surface is covered with BEAM3 elements, which are deselected for the solution.

ANSYS does not provide the possibility to use a plane strain model with a given thickness. Therefore a plane stress model is used instead, which yields a softer structure.

**MSC NASTRAN**

In MSC NASTRAN CQUAD4 elements are used with shell property. The meshing and modeling was performed in SimXpert.

**MSC Marc**

In MSC Marc the element type 11 is used, i.e. plane-strain full integration quadrilateral elements. The meshing and modeling was performed in MSC Marc Mentat.
2.3 Setting Up the Coupled Simulation with MpCCI GUI

See \(\text{IV-2 Setting up a Coupled Simulation} \) and \(\text{V-4 Graphical User Interface} \) for a detailed description on how to use the MpCCI GUI. Following are the substantive rules to be set in the MpCCI GUI in order to run this tutorial.

At first start the MpCCI GUI by running the command \(\text{mpcci gui} \).

2.3.1 Models Step

In the Models step select and configure the codes to be coupled (cf. \(\text{IV-2.4 Models Step – Choosing Codes and Model Files} \)). Further information on the offered code parameters in the Models step can be looked up in the respective code section of the Codes Manual.

1. Choose your fluid mechanics code as first code to couple:

<table>
<thead>
<tr>
<th><strong>Option</strong></th>
<th><strong>Action</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select FINE/Open.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see (\text{VI-5 FINE/Open} )).</td>
</tr>
<tr>
<td>Model file</td>
<td>Select &quot;vortexVibration_serial.run&quot; in the subdirectory &quot;FINEOpen/vortexVibration_serial/&quot;.</td>
</tr>
<tr>
<td></td>
<td>The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Transient is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Option</strong></th>
<th><strong>Action</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select FINE/Turbo.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see (\text{VI-6 FINE/Turbo} )).</td>
</tr>
<tr>
<td>Run file</td>
<td>Select the run file &quot;vortex_cpl_ustd.run&quot; in the subdirectory &quot;FINETurbo/vortex_cpl_ustd&quot;.</td>
</tr>
<tr>
<td></td>
<td>The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Transient is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>
### FLUENT

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select FLUENT.</td>
</tr>
<tr>
<td>Version</td>
<td>The model is two-dimensional, thus select the FLUENT version 2ddp.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see ▶ VI-8 FLUENT ◀).</td>
</tr>
<tr>
<td>Case file</td>
<td>Select the case file &quot;FLUENT/plate.cas&quot;. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Transient is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>

### STAR-CCM+

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select STAR-CCM+.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see ▶ VI-16 STAR-CCM+ ◀).</td>
</tr>
<tr>
<td>Simulation file</td>
<td>Select the simulation file &quot;STAR-CCM+/plate2160.sim&quot;. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Transient is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>

... 

2. Choose your solid mechanics code as second code to couple:

### Abaqus

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select Abaqus.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see ▶ VI-2 Abaqus ◀).</td>
</tr>
<tr>
<td>Input file</td>
<td>Select input file &quot;Abaqus/plate.inp&quot;. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Transient is displayed in this newly added option as a result of the scan process.</td>
</tr>
<tr>
<td>Unit system</td>
<td>Select cgs to ensure that dimensions and quantities are transferred correctly. The Abaqus model was created using the cgs unit system, i.e. the values given in Figure 1 were directly entered in Abaqus/CAE and the geometry was created in cm.</td>
</tr>
</tbody>
</table>
### ANSYS

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select ANSYS.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see VI-3 ANSYS).</td>
</tr>
<tr>
<td>Product</td>
<td>You can select any ANSYS product which can be used for structural analysis, e.g. ansys.</td>
</tr>
<tr>
<td>Database file</td>
<td>Select database file &quot;ANSYS/plate.db&quot;.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Undefined is displayed in this newly added option as a result of the scan process.</td>
</tr>
<tr>
<td>Unit system</td>
<td>Select cgs to ensure that dimensions and quantities are transferred correctly. The ANSYS model was created using the cgs unit system, i.e. the values given in Figure 1 were directly entered in ANSYS and the geometry was created in cm.</td>
</tr>
</tbody>
</table>

### MSC NASTRAN

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select MSC NASTRAN.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see VI-13 MSC NASTRAN).</td>
</tr>
<tr>
<td>Input file</td>
<td>Select input file &quot;MSC.Nastran/vortexvibration.bdf&quot;.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Transient is displayed in this newly added option as a result of the scan process.</td>
</tr>
<tr>
<td>Unit system</td>
<td>Select cgs to ensure that dimensions and quantities are transferred correctly. The MSC NASTRAN model was created using the cgs unit system, i.e. the values given in Figure 1 were directly entered in SimXpert and the geometry was created in cm.</td>
</tr>
</tbody>
</table>

### MSC Marc

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select MSC Marc.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see VI-12 MSC Marc).</td>
</tr>
<tr>
<td>Input file</td>
<td>Select input file &quot;MSC.Marc/plate.dat&quot;.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Transient is displayed in this newly added option as a result of the scan process.</td>
</tr>
<tr>
<td>Unit system</td>
<td>Select cgs to ensure that dimensions and quantities are transferred correctly. The MSC Marc model was created using the cgs unit system, i.e. the values given in Figure 1 were directly entered in MSC Marc Mentat and the geometry was created in cm.</td>
</tr>
</tbody>
</table>

Proceed to the Algorithm step by pressing Algorithm.
2.3.2 Algorithm Step

In the Algorithm step define the coupling algorithm (cf. IV-2.5 Algorithm Step – Defining the Coupling Algorithm). The settings should be as follows. If no action is mentioned, keep the default option.

1. For the Common Basics panel:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Coupling analysis</td>
<td></td>
</tr>
<tr>
<td>Define the coupling scheme</td>
<td>Set to Explicit.</td>
</tr>
<tr>
<td>▼ Coupling algorithm</td>
<td></td>
</tr>
<tr>
<td>Algorithm type</td>
<td>Set to Serial.</td>
</tr>
<tr>
<td>Leading code</td>
<td>Set to solid mechanics code.</td>
</tr>
<tr>
<td>▼ Coupling duration</td>
<td></td>
</tr>
<tr>
<td>Set total coupling time</td>
<td>Check this option.</td>
</tr>
<tr>
<td>Total coupling time</td>
<td>Set to 10 s.</td>
</tr>
</tbody>
</table>

2. For the fluid mechanics code:

**FINE/Open**

No code specific options need to be set.

**FINE/Turbo**

No code specific options need to be set.

**FLUENT**

No code specific options need to be set.

**STAR-CCM+**

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Solver settings</td>
<td></td>
</tr>
<tr>
<td>Type of solver step size</td>
<td>Set to Constant.</td>
</tr>
<tr>
<td>Solver time step size</td>
<td>Set to 0.001.</td>
</tr>
<tr>
<td>Maximum number of iterations per time step</td>
<td>Set to 30.</td>
</tr>
</tbody>
</table>

3. For your solid mechanics code:
Abaqus

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>** Solver settings**</td>
<td>Deselect this option.</td>
</tr>
<tr>
<td>Use subcycling</td>
<td></td>
</tr>
<tr>
<td>** Coupling steps**</td>
<td></td>
</tr>
<tr>
<td>Type of coupling step size</td>
<td>Set to Constant.</td>
</tr>
<tr>
<td>Constant coupling step size</td>
<td>Set to 0.001, which equals the time step size in the Abaqus input file.</td>
</tr>
</tbody>
</table>

ANSYS

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>** Analysis**</td>
<td></td>
</tr>
<tr>
<td>Analysis type</td>
<td>Select Transient.</td>
</tr>
</tbody>
</table>

MSC NASTRAN

No code specific options need to be set.

MSC Marc

No code specific options need to be set.

Press the [Regions] button at the bottom of the window to get to the Regions step.

2.3.3 Regions Step

In the Regions step build the coupling regions, define quantities to be exchanged and assign the defined quantities to the built regions (cf. IV-2.6 Regions Step – Defining Coupling Regions and Quantities).

1. At first select the Mesh tab to get access to the mesh based element components.

2. Select Build Regions tab and here the Setup tab. Now choose the components to be coupled. In this example the coupling region corresponds to the surface of the flexible structure. MpCCI treats this region as a “Face” because it represents a 2D structure. Therefore select the Face tab (▲) in the components list where appropriate and select the coupling components by dragging them into the Coupled boxes.

Lookup your codes in the following table and select the components to be coupled:
### 2.3 Setting Up the Coupled Simulation with MpCCI GUI

#### 2.3.3 For region Region_1

<table>
<thead>
<tr>
<th>For fluid mechanics codes</th>
<th>Select coupling component(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FINE/Open</td>
<td>▲ VortexVibration_group_6</td>
</tr>
<tr>
<td></td>
<td>▲ VortexVibration_group_7</td>
</tr>
<tr>
<td></td>
<td>▲ VortexVibration_group_8</td>
</tr>
<tr>
<td>FINE/Turbo</td>
<td>▲ Beam_1</td>
</tr>
<tr>
<td></td>
<td>▲ Beam_2</td>
</tr>
<tr>
<td></td>
<td>▲ Beam_3</td>
</tr>
<tr>
<td>FLUENT</td>
<td>▲ flexible-struct</td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td>▲ flexible-struct</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>For solid mechanics codes</th>
<th>Select coupling component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abaqus</td>
<td>▲ ASSEMBLY_OUTSIDE</td>
</tr>
<tr>
<td>ANSYS</td>
<td>▲ OUTSIDE</td>
</tr>
<tr>
<td>MSC NASTRAN</td>
<td>▲ struct</td>
</tr>
<tr>
<td>MSC Marc</td>
<td>▲ outside</td>
</tr>
</tbody>
</table>

3. Select Define Quantity Sets tab. As currently no quantities are assigned to “EmptyQuantitySet” (created by default), a warning symbol is displayed in front of it. Now choose the quantities to be exchanged.

<table>
<thead>
<tr>
<th>For quantity set</th>
<th>Select quantities</th>
<th>Set sender</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmptyQuantitySet</td>
<td>NPosition</td>
<td>Solid mechanics code</td>
</tr>
<tr>
<td></td>
<td>RelWallForce</td>
<td>Fluid mechanics code</td>
</tr>
</tbody>
</table>

The resulting name for the quantity set will be NPosition<<Solid mechanics code | RelWallForce <<Fluid mechanics code.

4. Select Assign Quantity Sets tab and assign the defined quantity set to the built region.

<table>
<thead>
<tr>
<th>For selected region</th>
<th>Check quantity set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Region_1</td>
<td>NPosition&lt;&lt;Solid mechanics code</td>
</tr>
</tbody>
</table>

No changes are required in the Monitors and Settings steps. You may proceed to the Go step by pressing [Go].

#### 2.3.4 Go Step

In the Go step configure the application start-up and start server and codes (cf. ▶IV-2.8 Go Step – Configuring the Application Startup and Running the Coupled Simulation ◁).

The code panels are described for each code below. If nothing special is mentioned keep the default settings. Further information on the offered code parameters in the Go step can be looked up in the respective code section of the Codes Manual.

1. For your fluid mechanics code:
### FINE/Open

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expert mode</td>
<td>Check to access further options.</td>
</tr>
<tr>
<td>Mesh deformation</td>
<td>Select Radial basis functions.</td>
</tr>
</tbody>
</table>

### FINE/Turbo

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expert mode</td>
<td>Check to access further options.</td>
</tr>
<tr>
<td>Mesh deformation</td>
<td>Select Radial basis functions.</td>
</tr>
<tr>
<td>Save inverse matrix of radial basis functions</td>
<td>Select Compute and use inverse matrix.</td>
</tr>
</tbody>
</table>

### FLUENT

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data file (optional)</td>
<td>Choose the data file &quot;FLUENT/plate-2160.dat&quot; because FLUENT does a restart from pre-computed flow field.</td>
</tr>
</tbody>
</table>

### STAR-CCM+

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run in batch</td>
<td>Check this option.</td>
</tr>
<tr>
<td>Auto start with the default template macro</td>
<td>Check this option.</td>
</tr>
<tr>
<td>License options</td>
<td>Set according to your license type (cf. ▷ VI-16.2.5 Go Step ◄).</td>
</tr>
</tbody>
</table>

2. For your solid mechanics code:

#### Abaqus

No code specific options need to be set.

#### ANSYS

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gui option</td>
<td>Keep -b to start ANSYS in batch mode.</td>
</tr>
<tr>
<td>APDL input script</td>
<td>Select the input script &quot;ANSYS/runplate.ans&quot;.</td>
</tr>
</tbody>
</table>

#### MSC NASTRAN

No code specific options need to be set.

#### MSC Marc

No code specific options need to be set.

Now you may run the computation.
2.4 Running the Computation

Save the MpCCI project file with name "plate.csp" via the MpCCI GUI menu File→Save Project As.
Press the [Start] button in the Go step.
Firstly a warning message from the application checker is displayed. It should be noted that no extrapolation in time is supported by MpCCI. However, the coupling step size is set to the same value for both codes, so that no extrapolation is required. This warning can therefore be acknowledged.
Then the simulation codes should start.
Some codes require additional actions:

**FINE/Open**
Because a flow field is loaded from "FINEOpen/vortexVibration_preComputation/vortexVibration_preComputation.run", the computation does not need to be initialized. The time step has been set to 0.001 seconds and the maximum number of time steps has been set to 10000 in the project file.

**FINE/Turbo**
Because a flow field is loaded from "FINETurbo/vortex_init_ustd/vortex_init_ustd.run", the computation does not need to be initialized. The time step has been set to 0.001 seconds and the maximum number of time steps has been set to 10000 in the project file.

**FLUENT**
The computation does not need to be initialized as a flow field is loaded from "FLUENT/plate-2160.dat". Select (Solve|Solution|Solving)—Run Calculation section. Set the Number of Time Steps to 10000 and press Calculate to start the simulation.
During the simulation, FLUENT will display the residual, the resulting moment of the forces on the coupling surface and the pressure distribution in the fluid domain. You can also see the deformation of the flexible structure in the pressure distribution window.

**STAR-CCM+**
The computation does not need to be initialized as a flow field is loaded from "STAR-CCM+/plate2160.sim". In the java macro file "mpcci_runjob_steady.java", the initialize() function for the solution is not executed because the Do a cold start run option is not activated. The time step has been initialized to 0.001 seconds and the Total coupling time is read from the MpCCI GUI setting which was defined to 10 s.

2.5 Discussion of Results

The vortex street yields pressure changes at the surface of the flexible structure, a typical pressure distribution is shown in Figure 3. The resulting forces accelerate the structure and cause a vibration with increasing amplitude.
However, as the structural models are not perfectly identical – the Abaqus model is a little stiffer than the MSC Marc model. The ANSYS model is much weaker as the other two, because the ANSYS model is based on a plane stress assumption while the others use plane strain. Therefore the natural frequencies of the models differ, which yields different responses as shown in Figure 4.
Figure 3: Vortex-induced vibration: Part of the flow field in FLUENT with velocity vectors and pressure distribution (red=high pressure, blue=low pressure).

Figure 4: Vortex-induced vibration: Vertical deflection of the tip.
3 Driven Cavity

3.1 Problem Description

![Figure 1: Driven cavity: Geometry of the two-dimensional flowed through cavity with moving top face and flexible bottom.](image)

Topics of this Tutorial

- Fluid-Structure Interaction (FSI)
- Iterative Transient Coupling with Quasi-Newton Method
- Adaptive Time Step Size

Simulation Codes

- Fluid Mechanics: FLUENT, OpenFOAM
- Solid Mechanics: Abaqus, MSC NASTRAN

Description

The example is taken from Koch [2016], which was originally set up by Mok [2001]. The low structural stiffness leads to a strong added mass effect. Only iterative coupling leads to a stable simulation. The Quasi-Newton relaxation is used in order to accelerate the convergence in each time step. In this tutorial, both two- and three-dimensional simulations can be performed, except for OpenFOAM, as it is only capable of simulating 3D models. 2D simulations are generally simpler and faster. However, simulations are more realistic for three-dimensional cases. At the end in \( \triangleright 3.5.1 \) Coupling with Adaptive Time Step \(<\), it is shown how to perform this tutorial with adaptive time step size.

3.2 Model Preparation

The simulation couples a solid mechanics model with a fluid mechanics model. The files which you need for the simulation are included in the MpCCI distribution. Create a new directory and copy the particular subdirectories from "\(<MpCCI\_home>/tutorial/DrivenCavity\)".
3.2.1 Fluid Model

The fluid domain comprises the quadratic area (1 m × 1 m) and the cube (1 m × 1 m × 1 m) for the two- and three-dimensional models, respectively. The two-dimensional model is shown in Figure 1, which is also the 2D projection of the three-dimensional model. The top face is oscillating vertically with a frequency of 0.2 Hz. The flow inlet and outlet are situated left and right near the top of the domain and are 0.2 m wide. The bottom of the cavity is flexible, which is modelled in the structural simulation.

The fluid is incompressible with density $\rho_F = 1 \text{ kg/m}^3$ and a viscosity of $\nu_F = 0.001 \text{ m}^2/\text{s}$.

The time steps in the solid and fluid code are set to the same fixed values. This is necessary for iterative transient coupling schemes. The time step size for this example is $\Delta t = 0.1 \text{ s}$ and the total time is $t = 40.0 \text{ s}$.

**FLUENT**

For FLUENT both the two- and three-dimensional cases are provided. An adaptive mesh is used. Overall, the 2D and 3D meshes consist of $24 \times 24$ quadrilateral cells and $24 \times 26 \times 24$ Hexahedron cells, respectively. The wetted surface of the flexible wall is defined as a boundary named “bottom”.

**OpenFOAM**

For OpenFOAM only the three-dimensional case is provided. An adaptive mesh is used. Overall, the mesh consists of $24 \times 26 \times 24$ Hexahedron cells. The wetted surface of the flexible wall is defined as a boundary named “bottom”.

3.2.2 Solid Model

The solid part corresponds to the flexible wall at the bottom. Its thickness is 0.002 m and it is fixed at the left and the right side. A control node $A$ in the middle is selected for evaluation of the results.

The material is linear elastic with density $\rho_S = 250 \text{ kg/m}^3$, elastic modulus $E = 250 \text{ kg/m/s}^2$, and Poisson’s ratio $\nu_S = 0$.

The time step size is set to $\Delta t = 0.1 \text{ s}$ to match the step size of the computational fluid dynamics code.

**Abaqus**

For Abaqus both the two- and three-dimensional cases are provided. In the 2D-model, the flexible wall is meshed by three layers of 80 CPS4 elements. Whereas in 3D-model, it is meshed by 1200 C3D8 elements. In order to define the region, where quantities are exchanged, a surface named “COUPLINGSURF” is built from the top of the flexible structure.

**MSC NASTRAN**

Both the two- and three-dimensional cases are provided for MSC NASTRAN as well. In the 2D-model, the flexible wall is meshed by three layers of 80 CQUAD4 elements. For the 3D case, a shell model with thickness 0.002 m in y-direction is discretized by a $30 \times 30$ mesh, resulting in 900 CQUAD4 elements. The coupled surface is called “surf” for both models.

3.3 Setting Up the Coupled Simulation with MpCCI GUI

See▷ IV-2 Setting up a Coupled Simulation◁ and▷ V-4 Graphical User Interface◁ for a detailed description on how to use the MpCCI GUI. Following are the substantive rules to be set in the MpCCI GUI in order to
run this tutorial.
At first start the MpCCI GUI by running the command \texttt{mpcci gui}.

### 3.3.1 Models Step

In the Models step select and configure the codes to be coupled (cf. IV-2.4 Models Step – Choosing Codes and Model Files). Further information on the offered code parameters in the Models step can be looked up in the respective code section of the Codes Manual.

1. Choose your fluid mechanics code as first code to couple:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select FLUENT.</td>
</tr>
<tr>
<td>Version</td>
<td>For a 2D simulation select the FLUENT version 2ddp.</td>
</tr>
<tr>
<td></td>
<td>For a 3D simulation select the FLUENT version 3ddp.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see VI-8 FLUENT).</td>
</tr>
<tr>
<td>Case file</td>
<td>For a 2D simulation select input file &quot;FLUENT/cavity2d.cas&quot;.</td>
</tr>
<tr>
<td></td>
<td>For a 3D simulation select input file &quot;FLUENT/cavity3d.cas&quot;.</td>
</tr>
<tr>
<td></td>
<td>The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Transient is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>

2. Choose your solid mechanics code as second code to couple:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select OpenFOAM.</td>
</tr>
<tr>
<td>Option</td>
<td>Select the OpenFOAM option Opt or Debug.</td>
</tr>
<tr>
<td>Precision</td>
<td>Select the OpenFOAM precision: DP for double or SP for single precision.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see VI-14 OpenFOAM).</td>
</tr>
<tr>
<td>Case directory</td>
<td>Select the case directory &quot;OpenFOAM/&lt;version&gt;&quot; fitting your OpenFOAM version (e.g. &quot;OpenFOAM/v1806-v2012&quot; for use with OpenFOAM v1806 to v2012). Please note that the provided files are only for 3D simulation. The files will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Transient is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>
3 Driven Cavity Tutorial

### Abaqus

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select Abaqus</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see (\triangleright) VI-2 Abaqus (\triangleleft)).</td>
</tr>
<tr>
<td>Input file</td>
<td>For a 2D simulation select input file &quot;ABAQUS/body2d.inp&quot;.</td>
</tr>
<tr>
<td></td>
<td>For a 3D simulation select input file &quot;ABAQUS/body3d.inp&quot;.</td>
</tr>
<tr>
<td></td>
<td>The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Transient is displayed in this newly added option as a result of the scan process.</td>
</tr>
<tr>
<td>Unit system</td>
<td>Select the SI unit system (which is the standard).</td>
</tr>
</tbody>
</table>

### MSC NASTRAN

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select MSC NASTRAN.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see (\triangleright) VI-13 MSC NASTRAN (\triangleleft)).</td>
</tr>
<tr>
<td>Input file</td>
<td>For a 2D simulation select input file &quot;MSC.Nastran/body2d.bdf&quot;.</td>
</tr>
<tr>
<td></td>
<td>For a 3D simulation select input file &quot;MSC.Nastran/shell3d.bdf&quot;.</td>
</tr>
<tr>
<td></td>
<td>The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Transient is displayed in this newly added option as a result of the scan process.</td>
</tr>
<tr>
<td>Unit system</td>
<td>Select the SI unit system (which is the standard).</td>
</tr>
</tbody>
</table>

⚠️ Please make sure to select the model files with matching dimensions.

Proceed to the Algorithm step by pressing Algorithm.

### 3.3.2 Algorithm Step

In the Algorithm step define the coupling algorithm (cf. \(\triangleright\) IV-2.5 Algorithm Step – Defining the Coupling Algorithm \(\triangleleft\)).

The settings should be as follows. If no action is mentioned, keep the default option.

1. For the Common Basics panel:
3.3 Setting Up the Coupled Simulation with MpCCI GUI

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Coupling analysis</td>
<td></td>
</tr>
<tr>
<td>Define the coupling scheme</td>
<td>Set to Implicit.</td>
</tr>
<tr>
<td>▼ Solver settings</td>
<td></td>
</tr>
<tr>
<td>Type of solver step size</td>
<td>Set to Constant.</td>
</tr>
<tr>
<td>Constant solver step size</td>
<td>Set to 0.1 s.</td>
</tr>
<tr>
<td>▼ Coupling algorithm</td>
<td></td>
</tr>
<tr>
<td>Algorithm type</td>
<td>Set to Serial.</td>
</tr>
<tr>
<td>Leading code</td>
<td>Set to solid mechanics code.</td>
</tr>
<tr>
<td>▼ Coupling duration</td>
<td></td>
</tr>
<tr>
<td>Maximum number of coupling step iterations</td>
<td>Set to 20.</td>
</tr>
<tr>
<td>Set total coupling time</td>
<td>Check this option.</td>
</tr>
<tr>
<td>Total coupling time</td>
<td>Set to 40 s.</td>
</tr>
</tbody>
</table>

2. For the fluid mechanics code:

**FLUENT**

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Coupling steps</td>
<td></td>
</tr>
<tr>
<td>Number of inner iterations</td>
<td>Set to 25.</td>
</tr>
</tbody>
</table>

**OpenFOAM**

No code specific options need to be set.

3. For your solid mechanics code:

**Abaqus**

No code specific options need to be set.

**MSC NASTRAN**

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Coupling steps</td>
<td></td>
</tr>
<tr>
<td>Number of inner iterations</td>
<td>Set to 1, as the residuum in MSC NASTRAN after 1 inner iteration is much smaller than the coupling residuum.</td>
</tr>
</tbody>
</table>

Press the **Regions** button at the bottom of the window to get to the Regions step.

### 3.3.3 Regions Step

In the Regions step build the coupling regions, define quantities to be exchanged and assign the defined quantities to the built regions (cf. ▷ IV-2.6 Regions Step – Defining Coupling Regions and Quantities ◄).
1. At first select the **Mesh** tab to get access to the mesh based element components.

2. Select **Build Regions** tab and here the **Setup** tab. Now choose the components to be coupled. In this example, the coupling region corresponds to the wetted surface of the flexible structure. **MpCCI** treats this region as a “Face” because it represents the surface of a 2D structure. So, select the **Face** tab (▲) in the components list and select the coupling components by dragging them into the **Coupled** boxes.

   Lookup your codes in the following table and select the components to be coupled:

<table>
<thead>
<tr>
<th>For fluid mechanics code</th>
<th>Select coupling component</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLUENT</td>
<td>▲ bottom</td>
</tr>
<tr>
<td>OpenFOAM</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>For solid mechanics code</th>
<th>Select coupling component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abaqus</td>
<td>▲ COUPLINGSURF</td>
</tr>
<tr>
<td>MSC NASTRAN</td>
<td>▲ surf</td>
</tr>
</tbody>
</table>

3. Select **Define Quantity Sets** tab. As currently no quantities are assigned to “EmptyQuantitySet” (created by default), a warning symbol is displayed in front of it. Now choose the quantities to be exchanged.

<table>
<thead>
<tr>
<th>For quantity set</th>
<th>Select quantities</th>
<th>Set sender</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmptyQuantitySet</td>
<td>NPosition</td>
<td>Solid mechanics code</td>
</tr>
<tr>
<td></td>
<td>WallForce</td>
<td>Fluid mechanics code</td>
</tr>
</tbody>
</table>

   The resulting name for the quantity set will be **NPosition<–Solid mechanics code | WallForce<–Fluid mechanics code**.

4. Add a relaxation operator to quantity **WallForce** (cf. ▷ 4.6.7.4 Applying Operators to Quantities of Mesh Based Components ◄).

   For a stable and accelerated computation, the Quasi-Newton relaxation is used in conjunction with transient iterative coupling. A more detailed description of Quasi-Newton methods is given in ▷ 3.3.3.4 Quasi-Newton Method ◄.

   Apply the relaxation operator to the **WallForce** quantity in quantity set **NPosition<–Solid mechanics code | WallForce<–Fluid mechanics code** as follows:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantity</td>
<td>Mark the already selected quantity <strong>WallForce</strong> by clicking on its name.</td>
</tr>
<tr>
<td>Operator</td>
<td>Select <strong>Relaxation</strong>.</td>
</tr>
</tbody>
</table>

   The relaxation operator is a post processor which will be applied to the receiver of the quantity (here the solid mechanics code). Configure the relaxation parameters on the basis of the above description as follows:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relaxation method</td>
<td>Select <strong>Quasi-Newton</strong>.</td>
</tr>
<tr>
<td>Type of Anderson Mixing method</td>
<td>Select <strong>Inverse</strong>.</td>
</tr>
<tr>
<td>Omega(ω)</td>
<td>Set to 0.1 which is the default.</td>
</tr>
<tr>
<td>Number of reused levels</td>
<td>Set to 1 which is the default.</td>
</tr>
</tbody>
</table>
5. Add a convergence check operator to quantity WallForce (cf. V-4.6.7.4 Applying Operators to Quantities of Mesh Based Components).

To terminate the coupling process a convergence tolerance for the inner iterations can be entered.

Apply the convergence check operator to the WallForce quantity in quantity set NPosition<–Solid mechanics code | WallForce<–Fluid mechanics code as follows:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantity</td>
<td>Mark the already selected quantity WallForce by clicking on its name.</td>
</tr>
<tr>
<td>Operator</td>
<td>Select ConvergenceCheck.</td>
</tr>
</tbody>
</table>

This is a pre processor and will be applied to the sender of the quantity (here the fluid mechanics code). Configure the convergence parameters as follows:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tolerance value</td>
<td>Set to 1e – 4 which is appropriate for this simulation.</td>
</tr>
</tbody>
</table>

6. Select Assign Quantity Sets tab and assign the defined quantity set to the built region.

<table>
<thead>
<tr>
<th>For selected region</th>
<th>Check quantity set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Region_1</td>
<td>NPosition&lt;–Solid mechanics code</td>
</tr>
</tbody>
</table>

No changes are required in the Monitors step. You may proceed to the Settings step by pressing [Settings].

### 3.3.4 Settings Step

In the Settings step the plots of the peak (min and max) and mean values for the inner iterations – which are used to check the convergence – have to be activated. To achieve this:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor.Peaks</td>
<td>Set selected.</td>
</tr>
</tbody>
</table>

Proceed to the Go step by pressing [Go].

### 3.3.5 Go Step

In the Go step configure the application start-up and start server and codes (cf. IV-2.8 Go Step – Configuring the Application Startup and Running the Coupled Simulation).

In the server panel, no changes are necessary.

The code panels are described for each code below. If nothing special is mentioned keep the default settings. Further information on the offered code parameters in the Go step can be looked up in the respective code section of the Codes Manual.

1. For your fluid mechanics code:

No code specific options need to be set.
2. For your solid mechanics code:

**Abaqus**

No code specific options need to be set.

**MSC NASTRAN**

No code specific options need to be set.

Now you may run the computation.

### 3.4 Running the Computation

Save the MpCCI project file with name "cavity.csp" via the MpCCI GUI menu File→Save Project As. Press the Start button in the Go step and the simulation codes should start.

Some codes require additional actions:

**FLUENT**

Select (Solve|Solution|Solving)→Run Calculation... to open the Run Calculation panel.

Check that the time step size is fixed to 0.1 s.

The iterative coupling settings done in the MpCCI GUI will be automatically reported in the Max Iterations/Time Step to fit the maximum number of iterations. For this example maximum 500 iterations per time step are needed – for 25 iterations between the data exchanges and at most 20 coupling steps per time step defined in MpCCI GUI.

The computation needs to be initialized by selecting Solve→Initialization and pressing the Initialize button.

Select MpCCI→MpCCI Run FSI. You should select 400 Number of Time Steps, to cover a simulation time of 40.0 s at least and press Run to start the simulation.

During the simulation, FLUENT will display the residuals.

... 

### 3.5 Discussion of Results

The results of the different simulations can be compared using the displacement of the central node in the solid mechanics code. To display the motion of the central node in y-direction proceed as follows:

**Abaqus**

Open the output database "abaqus_run.odb" in Abaqus/CAE to plot the displacement of the control point:
Select **Result → History Output** from the menu bar to open the History Output panel. Select U2 at Node 41 in NSET CONTROL-NODE and U2 at Node 26938 in NSET CONTROL-NODE for 2D and 3D models, respectively. Then press **Plot** to plot the displacement in y-direction.

**MSC NASTRAN**

In SimXpert or PATRAN attach the results file "mpcci_body2d.xdb" or "mpcci_shell3d.xdb" for 2D and 3D models, respectively. You can create a chart for plotting the displacement in y direction for the node 248 for the 2D and 481 for the 3D model.

It should be noted that an explicit transient coupling scheme leads to divergence, as shown in Figure 2. This is due to a known instability problem named the strong added mass effect. However, a robust co-simulation can be performed by an iterative coupling. The complete oscillation of control node A is plotted in Figure 3. For these results, implicit coupling scheme with Quasi-Newton relaxation is used. It should be noted that the different response between two- and three-dimensional models is due to the slight difference in the nature of 2D and 3D simulations, as it is assumed for 2D models there is no gradient in the z-direction.

![Figure 2: 2D driven cavity: Computed y-displacement of control node A with an explicit (divergent) and an iterative (convergent) transient coupling scheme.](image1)

![Figure 3: Oscillation of control node A for two- and three-dimensional models.](image2)
If the iterative coupling is used without any relaxation, the number of necessary coupling step iterations until convergence \((tol = 1e^{-4})\) is roughly 30 per time step on average. The Quasi-Newton relaxation, which is chosen in this tutorial, leads to a computational cost saving by a factor of 7, while the results of both iterative methods coincide. This reduction is attributed to the small number of necessary coupling step iterations (average 5), cf. Figure 4a. The oscillation of the iterations is completely eliminated by the Quasi-Newton relaxation, cf. Figure 4b. A comparison between performance of Quasi-Newton with different Number of reused levels values is tabulated in Table 1, where 4 is the best found value and 0 is the worst one for this configuration. However, it should be noted that the value of Number of reused levels is very problem dependent. These results are obtained from the 2D coupling between FLUENT and Abaqus.

Table 1: Performance of Quasi-Newton with different Number of reused levels for two-dimensional driven cavity.

<table>
<thead>
<tr>
<th>Number of reused levels</th>
<th>0</th>
<th>1</th>
<th>4</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average number of iterations</td>
<td>6.9</td>
<td>5.9</td>
<td>5.4</td>
<td>5.6</td>
</tr>
</tbody>
</table>

Figure 4: 2D driven cavity: Comparison of iterative coupling schemes with respect to the number of necessary coupling step iterations to converge the time steps.

The oscillation of the flexible wall at the bottom of the cavity is shown in Figure 5 for the two-dimensional model.

### 3.5.1 Coupling with Adaptive Time Step

In order to reduce the taken time for the whole simulation, one may increase the time step. However, this might lead to instability in the solvers as well as the coupling. This problem can be solved by consideration of solver limits for choosing an appropriate time step, e.g. CFL condition. MpCCI is capable of performing
VII Tutorial

3.5 Discussion of Results

Figure 5: 2D driven cavity: Simulation results.

co-simulations with time step negotiation between the codes (cf. V-3.4.5.2 Coupling with Exchange of Time Step Size).

In Algorithm step, adaptive time step can be enabled from the Solver settings panel of Common Basics. In this panel, Type of solver step size should be set to Negotiation with 0.1 s for both Initial solver step size and Minimum solver step size. This reduces the number of performed time steps, while the final results still match.

⚠️ In order to perform coupling with time step negotiation, the models files must be able to vary the time step size.

⚠️ MpCCI does not support adaptive time step for FLUENT and MSC NASTRAN.
4 Elastic Flap in a Duct

4.1 Problem Description

![Figure 1: Elastic flap: Geometry [m] and boundary conditions](image)

Topics of this Tutorial

- Fluid-Structure Interaction (FSI)
- Coupling of time step (sent by computational fluid dynamics code)
- Usage of the MpCCI Grid Morpher for OpenFOAM
- 3D-model

Simulation Codes

- Solid Mechanics: Abaqus, ANSYS, MSC Marc, MSC NASTRAN
- Fluid Mechanics: FINE/Open, FINE/Turbo, FLUENT, OpenFOAM, STAR-CCM+

4.2 Model Preparation

The simulation couples a solid mechanics model with a fluid mechanics model. The files which you need for the simulation are included in the MpCCI distribution. Create a new directory and copy the subdirectories from "<MpCCI_home>/tutorial/ElasticFlap" which correspond to the simulation codes you want to use.

4.2.1 Solid Model

The solid part corresponds to a rectangle (Figure 2):

The material is linear elastic with density $\varrho = 1000$ kg/m³, elastic modulus $E = 1.0 \times 10^8$ Pa and Poisson’s ratio $\nu = 0.49$.

The mesh consists of $20 \times 20 \times 2$ brick elements with 20 nodes each. All nodes at the top are fixed, i.e. they cannot move in any direction.
The solid mechanics code is set up with an adaptive time step setting with an initial time step size $\Delta t = 0.25\text{ ms}$ and a total time $t = 0.2\text{ s}$.

A control-point/node is selected in each solid model for evaluation of the results, see 4.5 Discussion of Results.<

**Abaqus**

C3D20R elements are used. The whole surface except for the fixed end is defined as an element set, which can be selected as coupling component.

**ANSYS**

The standard 20-node brick element type SOLID95 - 3D 20-Node Structural Solid is used. In ANSYS, the time stepping scheme is set up in the APDL script by:

```plaintext
ptime = ptime0 + DeltaTime
ptime0= ptime
time, ptime
solve
```

ANSYS receives the time step size from the component DeltaTime and the total simulation time is set with the `time, ptime`.

**MSC Marc**

The fully integrated 20-node brick elements 21 are used. The solver has been set up with an adaptive time step.

**MSC NASTRAN**

The standard CHEXA8 elements are used. The whole surface except for the fixed end is defined as a wetted surface, which can be selected as coupling component. The wetted surface has QUAD4 elements defined for applying the FSI load.

The control node in MSC NASTRAN is the node number 1. The solver has been set up with an adaptive time step.
4.2.2 Fluid Model

The fluid domain comprises the whole rectangular area in Figure 3 except for the flexible elastic flap structure.

The computational fluid dynamics code is set up with a fixed time step configuration $\Delta t = 0.25$ ms.

![Elastic flap: Partial mesh of the fluid domain](image)

**Figure 3:** Elastic flap: Partial mesh of the fluid domain

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FINE/Open</td>
<td>The mesh was generated with HEXPRESS and consists of hexahedral elements. The surface of the flexible structure is defined as separate solid boundaries.</td>
</tr>
<tr>
<td>FINE/Turbo</td>
<td>The mesh was generated with IGG and consists of hexahedral elements. The surface of the flexible structure is defined as separate solid boundaries.</td>
</tr>
<tr>
<td>FLUENT</td>
<td>The mesh was generated with Gambit and consists of tetrahedral elements. The surface of the flexible structure is defined as a separate boundary named “couple-flap”.</td>
</tr>
<tr>
<td>OpenFOAM</td>
<td>The mesh corresponds to the mesh which is used in FLUENT. The surface of the flexible structure is defined as a separate boundary named “couple-flap”.</td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td>The mesh corresponds to the mesh which is used in FLUENT. The boundary regions in the middle (named air-remesh) are subdivided, because they will later be addressed by the STAR-CCM+ grid morpher as sliding boundaries.</td>
</tr>
</tbody>
</table>
4.3 Setting Up the Coupled Simulation with MpCCI GUI

See IV-2 Setting up a Coupled Simulation and V-4 Graphical User Interface for a detailed description on how to use the MpCCI GUI. Following are the substantive rules to be set in the MpCCI GUI in order to run this tutorial.

At first start the MpCCI GUI by running the command `mpcci gui`.

### 4.3.1 Models Step

In the Models step select and configure the codes to be coupled (cf. IV-2.4 Models Step – Choosing Codes and Model Files). Further information on the offered code parameters in the Models step can be looked up in the respective code section of the Codes Manual.

1. Choose your CSM code (CSM = Computational Structural Mechanics) as first code to couple:

#### Abaqus

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select Abaqus.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to <code>latest</code> or a version supported by MpCCI (see VI-2 Abaqus).</td>
</tr>
<tr>
<td>Input file</td>
<td>Select input file &quot;Abaqus/elastic_flap.inp&quot;. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td><code>Transient</code> is displayed in this newly added option as a result of the scan process.</td>
</tr>
<tr>
<td>Unit system</td>
<td>Select the SI unit system (which is the standard).</td>
</tr>
</tbody>
</table>

#### ANSYS

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select ANSYS.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to <code>latest</code> or a version supported by MpCCI (see VI-3 ANSYS).</td>
</tr>
<tr>
<td>Product</td>
<td>You can select any ANSYS product which can be used for structural analysis, e.g. <code>ansys</code>.</td>
</tr>
<tr>
<td>Database file</td>
<td>Select database file &quot;ANSYS/elastic_flap.db&quot;. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td><code>Undefined</code> is displayed in this newly added option as a result of the scan process.</td>
</tr>
<tr>
<td>Unit system</td>
<td>Select the SI unit system (which is the standard).</td>
</tr>
</tbody>
</table>
MSC Marc

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select MSC Marc.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see VI-12 MSC Marc).</td>
</tr>
<tr>
<td>Input file</td>
<td>Select input file &quot;MSC.MARC/elastic_flap_job1.dat&quot;. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Transient is displayed in this newly added option as a result of the scan process.</td>
</tr>
<tr>
<td>Unit system</td>
<td>Select the SI unit system (which is the standard).</td>
</tr>
</tbody>
</table>

MSC NASTRAN

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select MSC NASTRAN.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see VI-13 MSC NASTRAN).</td>
</tr>
<tr>
<td>Input file</td>
<td>Select input file &quot;MSC.Nastran/elastic_flap.bdf&quot;. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Transient is displayed in this newly added option as a result of the scan process.</td>
</tr>
<tr>
<td>Unit system</td>
<td>Select the SI unit system (which is the standard).</td>
</tr>
</tbody>
</table>

... 

2. Choose your CFD code (CFD = Computational Fluid Dynamics) as second code to couple:

FINE/Open

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select FINE/Open.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see VI-5 FINE/Open).</td>
</tr>
<tr>
<td>Model file</td>
<td>Select &quot;elasticFlap_serial.run&quot; in the subdirectory &quot;FINEOpen/elasticFlap_serial/&quot;. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Transient is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>
### FINE/Turbo

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select FINE/Turbo.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see VI-6 FINE/Turbo).</td>
</tr>
<tr>
<td>Run file</td>
<td>Select the run file &quot;flap_model.run&quot; in the subdirectory &quot;FINETurbo/flap_model&quot;. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Transient is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>

### FLUENT

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select FLUENT.</td>
</tr>
<tr>
<td>Version</td>
<td>The model is three-dimensional, thus select the FLUENT version 3ddp.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see VI-8 FLUENT).</td>
</tr>
<tr>
<td>Case file</td>
<td>Select the case file &quot;FLUENT/elastic_flap.cas&quot;. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Transient is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>

### OpenFOAM

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select OpenFOAM.</td>
</tr>
<tr>
<td>Option</td>
<td>Select the OpenFOAM option Opt or Debug.</td>
</tr>
<tr>
<td>Precision</td>
<td>Select the OpenFOAM precision: DP for double or SP for single precision.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see VI-14 OpenFOAM).</td>
</tr>
<tr>
<td>Case directory</td>
<td>Select the case directory &quot;OpenFOAM/&lt;version&gt;&quot; fitting your OpenFOAM version (e.g. &quot;OpenFOAM/v1806-v2012&quot; for use with OpenFOAM v1806 to v2012). The files will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Transient is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>
### 4 Elastic Flap in a Duct

#### 4.3.2 Algorithm Step

In the Algorithm step define the coupling algorithm (cf. ▷ IV-2.5 Algorithm Step – Defining the Coupling Algorithm ◄).

The settings should be as follows. If no action is mentioned, keep the default option.

1. For the Common Basics panel:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Coupling analysis</td>
<td></td>
</tr>
<tr>
<td>Define the coupling scheme</td>
<td>Set to Explicit.</td>
</tr>
<tr>
<td>▼ Coupling steps</td>
<td></td>
</tr>
<tr>
<td>Type of coupling step size</td>
<td>Set to Sent by code because the time step is coupled.</td>
</tr>
<tr>
<td>▼ Coupling algorithm</td>
<td></td>
</tr>
<tr>
<td>Algorithm type</td>
<td>Set to Serial.</td>
</tr>
<tr>
<td>Leading code</td>
<td>Set to CSM code.</td>
</tr>
<tr>
<td>▼ Coupling duration</td>
<td></td>
</tr>
<tr>
<td>Set total coupling time</td>
<td>Check this option.</td>
</tr>
<tr>
<td>Total coupling time</td>
<td>Set to 0.2 s.</td>
</tr>
</tbody>
</table>

2. For your solid mechanics code:

**Abaqus**

No code specific options need to be set.

**ANSYS**

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Analysis</td>
<td></td>
</tr>
<tr>
<td>Analysis type</td>
<td>Select Transient.</td>
</tr>
</tbody>
</table>
3. For your fluid mechanics code:

<table>
<thead>
<tr>
<th>Code</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>FINE/Open</td>
<td>No code specific options need to be set.</td>
</tr>
<tr>
<td>FINE/Turbo</td>
<td>No code specific options need to be set.</td>
</tr>
<tr>
<td>FLUENT</td>
<td>No code specific options need to be set.</td>
</tr>
<tr>
<td>OpenFOAM</td>
<td>No code specific options need to be set.</td>
</tr>
</tbody>
</table>

### STAR-CCM+

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Solver settings</td>
<td></td>
</tr>
<tr>
<td>Type of solver step size</td>
<td>Set to Defined by model.</td>
</tr>
<tr>
<td>Maximum number of inner iterations</td>
<td>Set to 30.</td>
</tr>
</tbody>
</table>

Press the [Regions] button at the bottom of the window to get to the Regions step.

### 4.3.3 Regions Step

In the Regions step build the coupling regions, define quantities to be exchanged and assign the defined quantities to the built regions (cf. ▷ IV-2.6 Regions Step – Defining Coupling Regions and Quantities ◄).

In this example the time steps of the two solvers are coupled, whereas the CFD code will send the time step to the CSM code.

1. At first select the Global tab to get access to the global variables.
2. Choose the variables to be coupled by dragging them into the Coupled boxes:

Lookup your codes in the following table and select the components to be coupled:
3. Select **DeltaTime** as quantity to be transferred and configure as follows:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sender</td>
<td>Select CFD code.</td>
</tr>
<tr>
<td>Default value</td>
<td>Set to 2.5E-4.</td>
</tr>
</tbody>
</table>

Furthermore the coupling region has to be defined.

1. At first select the **Mesh** tab to get access to the mesh based element components.

2. Select **Build Regions** tab and here the **Setup** tab. Now choose the components to be coupled. In this example the coupling region corresponds to the surface of the flexible structure. **MpCCI** treats this region as a “Face” because it represents a 2D structure. Therefore select the **Face** tab (▲) in the components list where appropriate and select the coupling components by dragging them into the **Coupled** boxes.

Lookup your codes in the following table and select the components to be coupled which characterize the surface of the flap:
For region Region_1

<table>
<thead>
<tr>
<th>For CSM codes</th>
<th>Select coupling component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abaqus</td>
<td>◯ ASSEMBLY_WALL_WALL-SURFACE</td>
</tr>
<tr>
<td>ANSYS</td>
<td>◯ WALLSURFACE</td>
</tr>
<tr>
<td>MSC Marc</td>
<td>◯ wallsurface</td>
</tr>
<tr>
<td>MSC NASTRAN</td>
<td>◯ flap</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>For CFD codes</th>
<th>Select coupling component(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FINE/Open</td>
<td>◯ Elastic_Flap_group_0</td>
</tr>
<tr>
<td></td>
<td>◯ Elastic_Flap_group_1</td>
</tr>
<tr>
<td></td>
<td>◯ Elastic_Flap_group_2</td>
</tr>
<tr>
<td></td>
<td>◯ Elastic_Flap_group_3</td>
</tr>
<tr>
<td></td>
<td>◯ Elastic_Flap_group_4</td>
</tr>
<tr>
<td>FINE/Turbo</td>
<td>◯ flap_front</td>
</tr>
<tr>
<td></td>
<td>◯ flap_back</td>
</tr>
<tr>
<td>FLUENT</td>
<td>◯ couple-flap</td>
</tr>
<tr>
<td>OpenFOAM</td>
<td>◯ couple-flap</td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td>◯ couple-flap</td>
</tr>
</tbody>
</table>

3. Select Define Quantity Sets tab. As currently no quantities are assigned to “EmptyQuantitySet” (created by default), a warning symbol is displayed in front of it. Now choose the quantities to be exchanged.

<table>
<thead>
<tr>
<th>For quantity set</th>
<th>Select quantities</th>
<th>Set sender</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmptyQuantitySet</td>
<td>NPosition</td>
<td>CSM code</td>
</tr>
<tr>
<td></td>
<td>RelWallForce</td>
<td>CFD code</td>
</tr>
</tbody>
</table>

The resulting name for the quantity set will be NPosition<–CSM code | RelWallForce<–CFD code.

4. Select Assign Quantity Sets tab and assign the defined quantity set to the built region.

<table>
<thead>
<tr>
<th>For selected region</th>
<th>Check quantity set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Region_1</td>
<td>NPosition&lt;–CSM code</td>
</tr>
</tbody>
</table>

No changes are required in the Monitors and Settings steps. You may proceed to the Go step by pressing [Go].

### 4.3.4 Go Step

In the Go step configure the application start-up and start server and codes (cf. ▷ IV-2.8 Go Step – Configuring the Application Startup and Running the Coupled Simulation ◀).
In the server panel, no changes are necessary.
The code panels are described for each code below. If nothing special is mentioned keep the default settings. Further information on the offered code parameters in the Go step can be looked up in the respective code section of the Codes Manual.

1. For your solid mechanics code:

<table>
<thead>
<tr>
<th>Code</th>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abaqus</td>
<td></td>
<td>No code specific options need to be set.</td>
</tr>
<tr>
<td>ANSYS</td>
<td>Gui option</td>
<td>Keep -b to start ANSYS in batch mode.</td>
</tr>
<tr>
<td></td>
<td>APDL input script</td>
<td>Select the input script &quot;ANSYS/runflap.ana&quot;.</td>
</tr>
<tr>
<td>MSC NASTRAN</td>
<td></td>
<td>No code specific options need to be set.</td>
</tr>
<tr>
<td>MSC Marc</td>
<td></td>
<td>No code specific options need to be set.</td>
</tr>
</tbody>
</table>

2. For your fluid mechanics code:

<table>
<thead>
<tr>
<th>Code</th>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>FINE/Open</td>
<td>Define a reference pressure</td>
<td>Set pressure to 101325 N/m².</td>
</tr>
<tr>
<td></td>
<td>Expert mode</td>
<td>Check and access further options.</td>
</tr>
<tr>
<td></td>
<td>Mesh deformation</td>
<td>Select Radial basis functions.</td>
</tr>
<tr>
<td>FINE/Turbo</td>
<td>Set the requested memory</td>
<td>Check this option because for the coupling the memory has to be increased.</td>
</tr>
<tr>
<td></td>
<td>Number of reals (Mb)</td>
<td>Set to 100.</td>
</tr>
<tr>
<td></td>
<td>Number of ints (Mb)</td>
<td>Set to 10.</td>
</tr>
<tr>
<td></td>
<td>Expert mode</td>
<td>Check and access further options.</td>
</tr>
<tr>
<td></td>
<td>Mesh deformation</td>
<td>Select Radial basis functions.</td>
</tr>
<tr>
<td></td>
<td>Save inverse matrix of radial basis functions</td>
<td>Select Compute and use inverse matrix.</td>
</tr>
<tr>
<td>FLUENT</td>
<td></td>
<td>No code specific options need to be set.</td>
</tr>
</tbody>
</table>
4.4 Running the Computation

4.4.1 Starting the Simulation

Save the MpCCI project file with name "elastic_flap.csp" over the MpCCI GUI menu File→Save Project As.

Press the [Start] button in the Go step and the simulation codes should start.

Some codes require additional actions:

**FINE/Open**

FINE/Open will perform 800 time steps and will run in batch mode.

**FINE/Turbo**

FINE/Turbo will perform 800 time steps and will run in batch mode.
Initialize the flow field by selecting **Solve** → **Initialization** to open the initialization panel. And press the button **Initialize**.

Select (**Solve** | **Solution** | **Solving**) → **Run Calculation**... to open the **Run Calculation** panel as shown on the left.

As the time step size is fixed to 0.00025 s, you should select a large **Number of Time Steps**, e.g., 800 to cover a simulation time of 0.2 s at least.

Finally press **Calculate** to start the simulation.

During the simulation, **FLUENT** will display the residuals and a plane section through the flap to visualize the deformation.

**STAR-CCM+**

**STAR-CCM+** will perform 800 time steps to cover a total coupling time of 0.2 seconds. The time step of 0.00025 s will be set in the java macro file "mpcci_runjob_unsteady.java" which will be automatically copied to the working directory. **STAR-CCM+** will run in batch mode.

**4.4.2 End of the Simulation**

The CFD codes determine the time step size which is constant $\Delta t = 0.00025$ s. Therefore only they have the information when to finish the simulation. In this case it is almost impossible to obtain a "clean exit" of both codes.

If the simulation has reached the end time of $t = 0.2$ s you can kill the remaining codes with the **Kill** button. All of the codes used in this example still write complete result files, therefore no data loss should be expected and it is possible to obtain the results described below.

**4.5 Discussion of Results**

To compare the results of the different simulations, a control point was selected in the solid mechanics codes, see **Figure 2**. To display the motion of this point in x-direction similar to **Figure 4** proceed as follows:
Abaqus

Open the output database "abaqus_run.odb" in Abaqus/CAE to plot the displacement of the control point:
Select Result—History Output from the menu bar to open the History Output panel.
Select U1 at Node 1 in NSET CONTROL-NODE and press [Plot] to plot the displacement in x-direction.

ANSYS

Start ANSYS and select TimeHist PostPro, open the file "file.rst". In the Time History Variables panel, press the + button and select Nodal Solution → DOF Solution → X-Component of displacement and select node 366. Press the Graph Data button to plot the graph.

MSC Marc

In MSC Marc Mentat select [RESULTS] and open the "marc_run[i4|i8].t16" file. Switch to [HISTORY PLOT], create a plot for the node-set "control-node" over all increments. Select the following data variables “Time” (x-axis) and “Displacement X” (y-axis) to plot the curve. Finally press [FIT] to fit the plot limits, otherwise the curve is displayed very small.

MSC NASTRAN

In SimXpert or PATRAN attach the results file "mpcci_elastic_flap.xdb". You can create a chart for plotting the displacement in x direction for the node 1 for example.

![Graph of Displacement](image.png)

Figure 4: Elastic flap: Displacement of the control point for Abaqus, ANSYS, MSC Marc and MSC NASTRAN coupled with FLUENT.
5 Exhaust Manifold

5.1 Problem Description

Figure 1: Manifold: Geometry [m] and boundary conditions

Topics of this Tutorial

- Thermal coupling fluid and solid
- Steady state
- Incompressible fluid
- Fluid solver subcycling
- 3D-model

Simulation Codes

- Solid Mechanics: Abaqus, ANSYS, MSC Marc
- Fluid Mechanics: FLUENT, OpenFOAM, STAR-CCM+

5.2 Model Preparation

The simulation couples a solid mechanics model with a fluid mechanics model. The files which you need for the simulation are included in the MpCCI distribution. Create a new directory "ExhaustManifold" and copy the subdirectories from "<MpCCI_home>/tutorial/ExhaustManifold" which correspond to the simulation codes you want to use.
5.2.1 Solid Model

The solid part of the manifold has the following properties (Figure 2):

![Figure 2: Manifold: Mesh of the solid domain](image)

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conductivity</td>
<td>55.0 $\frac{W}{m \cdot K}$</td>
</tr>
<tr>
<td>Film temperature (outside)</td>
<td>300.0 $K$</td>
</tr>
<tr>
<td>Heat coefficient (outside)</td>
<td>50.0 $\frac{W}{m^2 \cdot K}$</td>
</tr>
</tbody>
</table>

Table 1: Manifold: Boundary conditions for solid model

### Abaqus

In Abaqus, the mesh consists of 15244 DC3D8 eight-node brick elements.

### ANSYS

In ANSYS, the mesh consists of 15244 D3-SOLID70 eight-node brick elements. Additional D3-SHELL57 dummy surface elements are defined for quantity exchange as described in § VI-3.2.1 Model Preparation.

### MSC Marc

In MSC Marc, the mesh consists of 3764 shell elements with 2 mm thickness. Two surfaces are defined on each side of the shell.
5.2.2 Fluid Model

The fluid domain comprises the inner part of the pipe (Figure 3).

Figure 3: Manifold: Mesh of the fluid domain

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Density</td>
<td>1.225 $\frac{kg}{m^3}$</td>
</tr>
<tr>
<td>Viscosity</td>
<td>$1.7894E-05$ $\frac{kg}{m \cdot s}$</td>
</tr>
<tr>
<td>Conductivity</td>
<td>0.0242 $\frac{W}{m \cdot K}$</td>
</tr>
<tr>
<td>Specific heat</td>
<td>1006.43 $\frac{j}{kg \cdot K}$</td>
</tr>
<tr>
<td>Inlet Velocity</td>
<td>6.0 $\frac{m}{s}$</td>
</tr>
<tr>
<td>Temperature</td>
<td>600.0 $K$</td>
</tr>
<tr>
<td>Turbulent Intensity</td>
<td>0.03</td>
</tr>
<tr>
<td>Turbulent Length</td>
<td>0.003 $m$</td>
</tr>
<tr>
<td>Outlet Relative pressure</td>
<td>0 $Pa$</td>
</tr>
<tr>
<td>Turbulent Intensity</td>
<td>0.03</td>
</tr>
<tr>
<td>Turbulent Length</td>
<td>0.003 $m$</td>
</tr>
</tbody>
</table>

Table 2: Manifold: Boundary conditions for fluid model

The fluid mesh consists of 102236 hexahedral elements.

**FLUENT**

A standard $k - \epsilon$ model with enhanced wall treatment is applied.
A standard \( k - \epsilon \) model with standard wall treatment is applied.

A standard \( k - \epsilon \) model with enhanced wall treatment is applied.

### 5.2.3 Uncoupled Flow Simulation

Prior to a coupled simulation, it is advisable to carry out an uncoupled flow simulation with simplified boundary conditions. The temperature of the later coupled interface of fluid and solid is set to constant 600 K.

#### FLUENT

Start FLUENT in double precision and read the case file "FLUENT/exhaust_manifold.cas". Carry out a flow simulation with 200 iterations and save the result in "FLUENT/exhaust_manifold.dat".

#### OpenFOAM

The pre-calculation is controlled by the Runtime control option Define coupling start in the Algorithm step.

#### STAR-CCM+

Start STAR-CCM+ and read the model file "STAR-CCM+/exhaust_manifold.sim". Carry out a flow simulation with 200 iterations. The result is saved in "STAR-CCM+/exhaust_manifold.sim".

### 5.2.4 Prepare Models for Coupled Simulation

After the uncoupled simulations have been completed, some input files need modifications to accomplish the coupled simulation using the uncoupled results for a restart.

#### FLUENT

No modifications have to be done, as we later carry out a GUI based coupled simulation.

### 5.3 Setting Up the Coupled Simulation with MpCCI GUI

See IV-2 Setting up a Coupled Simulation and V-4 Graphical User Interface for a detailed description on how to use the MpCCI GUI. Following are the substantive rules to be set in the MpCCI GUI in order to run this tutorial.

At first start the MpCCI GUI by running the command `mpcci gui`.

#### 5.3.1 Models Step

In the Models step select and configure the codes to be coupled (cf. IV-2.4 Models Step – Choosing Codes and Model Files). Further information on the offered code parameters in the Models step can be looked up in the respective code section of the Codes Manual.
1. Choose your solid mechanics code as first code to couple:

**Abaqus**

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select Abaqus.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see VI-2 Abaqus).</td>
</tr>
<tr>
<td>Input file</td>
<td>Select input file &quot;Abaqus/exhaust_manifold.inp&quot;. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Steady state is displayed in this newly added option as a result of the scan process.</td>
</tr>
<tr>
<td>Unit system</td>
<td>Select the SI unit system (which is the standard).</td>
</tr>
</tbody>
</table>

**ANSYS**

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select ANSYS.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see VI-3 ANSYS).</td>
</tr>
<tr>
<td>Product</td>
<td>You can select any ANSYS product which can be used for structural analysis, e.g. ansys.</td>
</tr>
<tr>
<td>Database file</td>
<td>Select database file &quot;ANSYS/manifold.db&quot;. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Undefined is displayed in this newly added option as a result of the scan process.</td>
</tr>
<tr>
<td>Unit system</td>
<td>Select the SI unit system (which is the standard).</td>
</tr>
</tbody>
</table>

**MSC Marc**

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select MSC Marc.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see VI-12 MSC Marc).</td>
</tr>
<tr>
<td>Input file</td>
<td>Select input file &quot;MSC.Marc/exhaust_manifold.dat&quot;. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Steady state is displayed in this newly added option as a result of the scan process.</td>
</tr>
<tr>
<td>Unit system</td>
<td>Select the SI unit system (which is the standard).</td>
</tr>
</tbody>
</table>

2. Choose your fluid mechanics code as second code to couple:
### FLUENT

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select FLUENT.</td>
</tr>
<tr>
<td>Version</td>
<td>The model is three-dimensional, thus select the FLUENT version <code>3ddp</code>.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to <code>latest</code> or a version supported by MpCCI (see VI-8 FLUENT).</td>
</tr>
<tr>
<td>Case file</td>
<td>Select the case file &quot;FLUENT/exhaust_manifold.cas&quot;.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Steady state is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>

### OpenFOAM

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select OpenFOAM.</td>
</tr>
<tr>
<td>Option</td>
<td>Select the OpenFOAM option Opt or Debug.</td>
</tr>
<tr>
<td>Precision</td>
<td>Select the OpenFOAM precision: DP for double or SP for single precision.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to <code>latest</code> or a version supported by MpCCI (see VI-14 OpenFOAM).</td>
</tr>
<tr>
<td>Case directory</td>
<td>Select the case directory &quot;OpenFOAM/&lt;&lt;version&gt;&gt;&quot; fitting your OpenFOAM version (e.g. &quot;OpenFOAM/v1606-v2012&quot; for use with OpenFOAM v1606 to v2012). The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Steady state is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>

### STAR-CCM+

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select STAR-CCM+.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to <code>latest</code> or a version supported by MpCCI (see VI-16 STAR-CCM+).</td>
</tr>
<tr>
<td>Simulation file</td>
<td>Select the simulation file &quot;STAR-CCM+/exhaust_manifold.sim&quot;.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Steady state is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>

Proceed to the Algorithm step by pressing [Algorithm].

### 5.3.2 Algorithm Step

In the Algorithm step define the coupling algorithm (cf. IV-2.5 Algorithm Step – Defining the Coupling Algorithm).

The settings should be as follows. If no action is mentioned, keep the default option.

1. For the Common Basics panel:
Option | Action
--- | ---
▼ Coupling analysis
Define the coupling scheme | This option is automatically set to Explicit.
▼ Coupling algorithm
Algorithm type | Set to Parallel.
Use initialization | Check this option.
Initializing code | Set to fluid mechanics code.
▼ Coupling duration
Set maximum number of coupling steps | Check this option.
Maximum number of coupling steps | Set to 20.

2. For the solid mechanics code:

**Abaqus**

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Solver settings</td>
<td></td>
</tr>
<tr>
<td>Use subcycling</td>
<td>Deselect this option as there is no subcycling of Abaqus included.</td>
</tr>
<tr>
<td>▼ Coupling steps</td>
<td></td>
</tr>
<tr>
<td>Constant coupling step size</td>
<td>Set to 1 for one increment size for the steady state analysis.</td>
</tr>
</tbody>
</table>

When starting the simulation, Abaqus will receive data from the fluid mechanics code and accomplish a steady state heat transfer analysis. After this analysis, data is exchanged with the partner code and another analysis is started.

**ANSYS**

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Analysis</td>
<td></td>
</tr>
<tr>
<td>Analysis type</td>
<td>Select Steady state.</td>
</tr>
</tbody>
</table>

**MSC Marc**

No code specific options need to be set.

3. For the fluid mechanics code:

**FLUENT**

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Coupling steps</td>
<td></td>
</tr>
<tr>
<td>Type of coupling step size</td>
<td>Set to Constant.</td>
</tr>
<tr>
<td>Constant coupling step size</td>
<td>Set to 10 solver steps without coupling for subcycling.</td>
</tr>
</tbody>
</table>
5.3 Setting Up the Coupled Simulation with MpCCI GUI

### OpenFOAM

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Coupling steps</td>
<td>Set to Constant.</td>
</tr>
<tr>
<td>Type of coupling step size</td>
<td>Set to 10 solver steps without coupling for subcycling.</td>
</tr>
<tr>
<td>Constant coupling step size</td>
<td>Set to 10 solver steps without coupling for subcycling.</td>
</tr>
<tr>
<td>▼ Runtime control</td>
<td></td>
</tr>
<tr>
<td>Define coupling start</td>
<td>Set to Specify iteration to enable a pre-calculation.</td>
</tr>
<tr>
<td>Iteration number for coupling start</td>
<td>Set to 200 to activate a pre-calculation of 200 iterations.</td>
</tr>
</tbody>
</table>

### STAR-CCM+

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Coupling steps</td>
<td>Set to Constant.</td>
</tr>
<tr>
<td>Type of coupling step size</td>
<td>Set to 10 solver steps without coupling for subcycling.</td>
</tr>
<tr>
<td>Constant coupling step size</td>
<td>Set to 10 solver steps without coupling for subcycling.</td>
</tr>
</tbody>
</table>

Press the [Regions] button at the bottom of the window to get to the Regions step.

### 5.3.3 Regions Step

In the Regions step build the coupling regions, define quantities to be exchanged and assign the defined quantities to the built regions (cf. ▶ IV-2.6 Regions Step – Defining Coupling Regions and Quantities ▷).

1. At first select the Mesh tab to get access to the mesh based element components.

2. Select Build Regions tab and here the Setup tab. Now choose the components to be coupled. In this example the coupling region corresponds to the inner surface of the pipe. MpCCI treats this region as a “Face” because it represents a 2D structure. Therefore select the Face tab (▲) in the components list where appropriate and select the coupling components by dragging them into the Coupled boxes.

Lookup your codes in the following table and select the components to be coupled:

For region Region 1

<table>
<thead>
<tr>
<th>For solid mechanics code</th>
<th>Select coupling component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abaqus</td>
<td>▲ ASSEMBLY_MANIFOLD_INNER_SURFACE</td>
</tr>
<tr>
<td>ANSYS</td>
<td>▲ INNER_SURFACE</td>
</tr>
<tr>
<td>MSC Marc</td>
<td>▲ INNER_SURFACE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>For fluid mechanics code</th>
<th>Select coupling component</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLUENT</td>
<td></td>
</tr>
<tr>
<td>OpenFOAM</td>
<td>▲ wall</td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td></td>
</tr>
</tbody>
</table>

3. Select Define Quantity Sets tab. As currently no quantities are assigned to “EmptyQuantitySet” (created by default), a warning symbol is displayed in front of it. Now choose the quantities to be
exchanged. For this use a quantity set predefined by MpCCI (cf. \textit{IV-4.6.8 Assigning Preconfigured Quantity Settings}): \begin{center}
\begin{tabular}{|l|l|}
\hline
For quantity set & Option & Action \\
\hline
EmptyQuantitySet & Select Coupling Type & For coupling with \textbf{Abaqus 2018 and newer} \\
 & & \textbf{ANSYS} \\
 & & \textbf{MSC Marc} \\
 & select & \textbf{Steady state surface heat transfer} \\
 & & \textbf{Abaqus 2016-2017} \\
 & select & \textbf{Steady state concentrated heat transfer} \\
 & & and click \textbf{Set}. \\
\hline
\end{tabular}
\end{center}

Now the quantities \texttt{FilmTemp}, \texttt{WallHTCoeff} for coupling with \texttt{Abaqus 2018 and newer}, \texttt{ANSYS} or \texttt{MSC Marc} resp. \texttt{HeatRate} for coupling with \texttt{Abaqus 2016-2017} only and \texttt{WallTemp} are activated and MpCCI already set the appropriate sender for them.

The resulting name for the quantity set will be \texttt{FilmTemp<Fluid mechanics code | WallHTCoeff<Fluid mechanics code | WallTemp<Solid mechanics code} resp.

\texttt{FilmTemp<Fluid mechanics code | HeatRate<Fluid mechanics code | WallTemp<Solid mechanics code}.

4. Select \textit{Assign Quantity Sets} tab and assign the defined quantity set to the built region.

\begin{center}
\begin{tabular}{|l|l|}
\hline
For selected region & Check quantity set \\
\hline
Region_1 & For coupling with \textbf{Abaqus 2018 and newer} \\
 & \textbf{ANSYS} \\
 & \textbf{MSC Marc} \\
 & \texttt{FilmTemp<Fluid mechanics code | WallHTCoeff<Fluid mechanics code | WallTemp<Solid mechanics code} \\
 & \textbf{Abaqus 2016-2017} \\
 & \texttt{FilmTemp<Fluid mechanics code | HeatRate<Fluid mechanics code | WallTemp<Solid mechanics code} \\
\hline
\end{tabular}
\end{center}

No changes are required in the \textit{Monitors} and \textit{Settings} steps. You may proceed to the \textit{Go} step by pressing \textit{Go}.

\subsection*{5.3.4 Go Step}

In the \textit{Go} step configure the application start-up and start server and codes (cf. \textit{IV-2.8 Go Step – Configuring the Application Startup and Running the Coupled Simulation}).

The code panels are described for each code below. If nothing special is mentioned keep the default settings. Further information on the offered code parameters in the \textit{Go} step can be looked up in the respective code section of the \textit{Codes Manual}.

1. For your solid mechanics code:
Abaqus

No code specific options need to be set.

MpCCI will run the simulation not on the original model file, but rather will use a copy with the prefix "abaqus_run".

ANSYS

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gui option</td>
<td>Keep -b to start ANSYS in batch mode.</td>
</tr>
</tbody>
</table>
| APDL input script       | Select the input script "ANSYS/startjob.ans".

MSC Marc

No code specific options need to be set.

MpCCI will run the simulation not on the original model file, but rather will use a copy with the prefix "marc_run_[i4|i8]".

FLUENT

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto read case data</td>
<td>Be sure to keep this option selected to automatically read file &quot;FLUENT/exhaust_manifold.dat&quot; the result of the uncoupled flow simulation.</td>
</tr>
<tr>
<td>Auto set MDM zones</td>
<td>Deselect.</td>
</tr>
</tbody>
</table>

OpenFOAM

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define a reference density</td>
<td>Set density to 1.225 kg/m³.</td>
</tr>
<tr>
<td>Define a reference specific heat</td>
<td>Set specific heat to 1006.43 J/kgK.</td>
</tr>
</tbody>
</table>

STAR-CCM+

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run in batch</td>
<td>Check this option.</td>
</tr>
<tr>
<td>Auto start with the default template macro</td>
<td>Check this option.</td>
</tr>
<tr>
<td>License options</td>
<td>Set according to your license type (cf. &gt; VI-16.2.5 Go Step&lt;).</td>
</tr>
</tbody>
</table>

Now you may run the computation.
5.4 Running the Computation

Save the MpCCI project file with name "exhaust_manifold.csp" via the MpCCI GUI menu File→Save Project As.
Press the [Start] button in the Go step and the simulation codes should start.
Some codes require additional actions:

**FLUENT**

1. FLUENT will load automatically the data file resulting from the initial computation.
2. Open the (Solve|Solution|Solving)→Run Calculation and start 200 iterations. FLUENT will do 10 iterations before each data exchange with MpCCI.

In order to survey the convergence of the simulation, FLUENT will display the residual and the integral total surface heat flux on the coupling surface.

**OpenFOAM**

OpenFOAM will run for 200 iterations. The solution needs not to be initialized as a flow field is loaded from simulation file previously computed.

**STAR-CCM+**

STAR-CCM+ will run for 200 iterations. The solution needs not to be initialized as a flow field is loaded from simulation file previously computed.

...  

5.5 Post-Processing

Use the following files for the evaluation of the results.

**MpCCI**

MpCCI tracefile "mpccirun-0000.ccvx" from the tracefile folder "mpccirun_<TIMESTAMP>_.ccvx" can be opened by the command mpcci visualize which one can trace the exchange process of the coupled surface.

**Abaqus**

Abaqus result file name is equal to the defined Abaqus job name parameter "VI-2.2.6 Go Step" followed by the suffix ".odb", e.g. "abaqus_run.odb" and temperatures for node set nodes evaluate on "abaqus_run.dat".

**ANSYS**

ANSYS result file "thermal.rth".

**MSC Marc**

MSC Marc result file "marc_run_[i4][i8].t16".

**FLUENT**

FLUENT result file "exhaust_manifold.dat".
**OpenFOAM**

Start `paraFoam` from the case directory to visualize the results of the OpenFOAM solver.

**STAR-CCM+**

STAR-CCM+ result file "exhaust_manifold@00400.sim".

---

Figure 4: Manifold: Temperature distribution on coupled surface
6 Busbar System

6.1 Problem Description

Topics of this Tutorial
- Magneto-Thermal
- 3D-model
- Volume coupling
- Several coupling regions
- Fluid solver subcycling

Simulation Codes
- Fluid Mechanics: FLUENT
- Electromagnetism: ANSYS, JMAG

Description
The example is taken from an article by Lyttle et al. [2006]. The task is to solve the electrothermal coupled problem of a busbar system carrying a three-phase current where the current leads to power losses due to the finite resistivity of the conductor. The busbars are made of copper, the RMS current level is 1600 A at a frequency of 60 Hz. Because of the power losses the conductor is heated up as well as the surrounding air leading to free convection around the busbars. To model the convection process the Boussinesq approximation for the density of air is used. A linear rise of resistivity dependent on the temperature of the conductor is also modeled.

The electromagnetic calculation of power-losses is considering:
- eddy currents,
- skin effect,
VII Tutorial 6.2 Model Preparation

- proximity effect,
- temperature dependent rise of resistivity.

The thermal calculation is considering:
- heat conduction (fluid and solid),
- heat convection (including buoyancy effects),
- laminar flow.

6.2 Model Preparation

The simulation couples a electromagnetic model with a fluid model. The files you need for the simulation are included in the MpCCI distribution. Create a new directory and copy the subdirectories from "<MpCCI_home>/tutorial/Busbar" corresponding to the simulation codes you want to use.

6.2.1 Fluid Model

On the front and back side of the fluid cabinet the conductors enter the fluid domain. Here are adiabatic wall boundary conditions set. On the left and right side of the cabinet a wall boundary condition with fixed temperature of 293.15 K is defined. On the bottom and top of the cabinet air can enter and leave the domain on pressure boundary conditions.

FLUENT

No modifications have to be done, as we later carry out a GUI based coupled simulation.

6.2.2 Electromagnetic Model

A busbar conductor corresponds to three rectangles each with a dimension of 12.7 mm $\times$ 76.2 mm $\times$ 250 mm. The magnetic property of copper is linear isotropic with a relative permeability of 1. The temperature dependent electrical resistivity is calculated in this way:

$$\varrho(T) = \varrho_0 (1 + \alpha (T - T_{ref}))$$

At reference temperature $T_{ref} = 300$ K the value of resistivity is $\varrho_0 = 1.7241 \times 10^{-8}$ $\Omega \cdot m$. The temperature coefficient is $\alpha = 0.004$ $K^{-1}$. The calculation of the current flow and magnetic field is done in frequency domain.

ANSYS

The "ANSYS/busbar.db" contains all finite element information. Beside the model file an ANSYS input file "ANSYS/startjob.ans" is used to define all boundary conditions, loads and to control the coupling process. At the front and back side of the conductors the voltage degree of freedom will be coupled and on one side for every conductor the voltage degree of freedom is set to 0. At the front side of the conductors a nodal load amps is defined separately for every conductor with real and imaginary part depending on the phasing. The elements are grouped into element components named "phase-a", "phase-b" and "phase-c" to make it possible for the MpCCI ANSYS adapter to detect the elements. For coupling these commands are used:

- \texttt{~mpcci, init, 3D}: Initialize the connection to MpCCI.
- \texttt{~mpcci, exchange}: Exchange data from the partner code and wait until the data exchange is done.
6.3 Setting Up the Coupled Simulation with MpCCI GUI

See ▶ IV-2 Setting up a Coupled Simulation ◀ and ▶ V-4 Graphical User Interface ◀ for a detailed description on how to use the MpCCI GUI. Following are the substantive rules to be set in the MpCCI GUI in order to run this tutorial.

At first start the MpCCI GUI by running the command `mpcci gui`.

### 6.3.1 Models Step

In the Models step select and configure the codes to be coupled (cf. ▶ IV-2.4 Models Step – Choosing Codes and Model Files ◀). Further information on the offered code parameters in the Models step can be looked up in the respective code section of the Codes Manual.

1. Choose your EMAG code (EMAG = electromagnetic) as first code to couple:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select ANSYS.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see ▶ VI-3 ANSYS ◀).</td>
</tr>
<tr>
<td>Product</td>
<td>You can select any ANSYS product which can be used for electromagnetic analysis, e.g. ane3fl or emag.</td>
</tr>
<tr>
<td>Database file</td>
<td>Select database file &quot;ANSYS/busbar.db&quot;.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Undefined is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>
### VII Tutorial

#### 6.3 Setting Up the Coupled Simulation with MpCCI GUI

##### J MAG

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select J MAG.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see ▶ VI-9 J MAG ◄).</td>
</tr>
<tr>
<td>JCF file</td>
<td>Select file &quot;J MAG/busbar_system.jcf&quot; as input file. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Steady state is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>

2. Choose your CFD code (CFD = Computational Fluid Dynamics) as second code to couple:

##### FL UENT

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select FLUENT.</td>
</tr>
<tr>
<td>Version</td>
<td>The model is three-dimensional, thus select the FLUENT version 3ddp.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see ▶ VI-8 FLUENT ◄).</td>
</tr>
<tr>
<td>Case file</td>
<td>Select the case file &quot;FLUENT/busbar.cas&quot;. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Steady state is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>

Proceed to the Algorithm step by pressing [Algorithm].

### 6.3.2 Algorithm Step

In the Algorithm step define the coupling algorithm (cf. ▶ IV-2.5 Algorithm Step – Defining the Coupling Algorithm ◄).

The settings should be as follows. If no action is mentioned, keep the default option.

1. For the Common Basics panel:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Coupling analysis</td>
<td>This option is automatically set to Explicit.</td>
</tr>
<tr>
<td>Define the coupling scheme</td>
<td>This option is automatically set to Explicit.</td>
</tr>
<tr>
<td>▼ Coupling algorithm</td>
<td>Set to Serial.</td>
</tr>
<tr>
<td>Leading code</td>
<td>Set to electromagnetism code.</td>
</tr>
<tr>
<td>▼ Coupling duration</td>
<td>Check this option.</td>
</tr>
<tr>
<td>Set maximum number of coupling steps</td>
<td>Set to 10 for the frequency response analysis simulation.</td>
</tr>
</tbody>
</table>
2. For the electromagnetism code:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Analysis</td>
<td></td>
</tr>
<tr>
<td>Analysis type</td>
<td>Select Steady state.</td>
</tr>
</tbody>
</table>

No code specific options need to be set.

3. For the fluid mechanics code:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Coupling steps</td>
<td></td>
</tr>
<tr>
<td>Type of coupling step size</td>
<td>Set to Constant.</td>
</tr>
<tr>
<td>Constant coupling step size</td>
<td>Set to 20 solver steps without coupling for subcycling.</td>
</tr>
</tbody>
</table>

Press the [Regions] button at the bottom of the window to get to the Regions step.

### 6.3.3 Regions Step

In the Regions step build the coupling regions, define quantities to be exchanged and assign the defined quantities to the built regions (cf. ▷ 4.6.1-1 Regions Step – Defining Coupling Regions and Quantities ▷).

1. At first select the Mesh tab to get access to the mesh based element components.
2. Select Build Regions tab and here the Setup tab. Now choose the components to be coupled. In this example the coupling region corresponds to the solid conductors.

MpCCI treats this region as a “Volume” because it represents a 3D structure. Therefore select the Volume tab (●) in the components list where appropriate and select the coupling components by dragging them into the Coupled boxes.

There are 3 conductors to couple and for each conductor from the EMAG and CFD code a couple region will be created. This will increase the performance of the neighborhood search.

Lookup your codes in the following table and select the components to be coupled for each region:

- Build the first region.

<table>
<thead>
<tr>
<th>For EMAG code</th>
<th>Select coupling component</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>phase-a</td>
</tr>
<tr>
<td>J MAG</td>
<td>busbar a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>For CFD code</th>
<th>Select coupling component</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLUENT</td>
<td>phase_a-solid</td>
</tr>
</tbody>
</table>
• Click on button **Add** to add new region **Region_2**.

<table>
<thead>
<tr>
<th>For EMAG code</th>
<th>Select coupling component</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>phase-b</td>
</tr>
<tr>
<td>J MAG</td>
<td>busbar b</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>For CFD code</th>
<th>Select coupling component</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLUENT</td>
<td>phase_b-solid</td>
</tr>
</tbody>
</table>

• Click on button **Add** to add new region **Region_3**.

<table>
<thead>
<tr>
<th>For EMAG code</th>
<th>Select coupling component</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>phase-c</td>
</tr>
<tr>
<td>J MAG</td>
<td>busbar c</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>For CFD code</th>
<th>Select coupling component</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLUENT</td>
<td>phase_c-solid</td>
</tr>
</tbody>
</table>

3. Select **Define Quantity Sets** tab. As currently no quantities are assigned to “EmptyQuantitySet” (created by default), a warning symbol is displayed in front of it. Now choose the quantities to be exchanged.

<table>
<thead>
<tr>
<th>For quantity set <strong>EmptyQuantitySet</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Select quantities</strong></td>
</tr>
<tr>
<td><strong>Option</strong></td>
</tr>
<tr>
<td>Set sender</td>
</tr>
<tr>
<td><strong>Code specific settings</strong></td>
</tr>
<tr>
<td>ANSYS</td>
</tr>
<tr>
<td>Send method</td>
</tr>
<tr>
<td>index</td>
</tr>
<tr>
<td>Location</td>
</tr>
<tr>
<td>J MAG</td>
</tr>
</tbody>
</table>

| Orphans settings | Select Extrapolate to extrapolate values into the orphaned regions where this is possible. |

<table>
<thead>
<tr>
<th>Temperature</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Set sender</td>
<td>Select CFD code.</td>
</tr>
<tr>
<td><strong>Code specific settings</strong></td>
<td></td>
</tr>
<tr>
<td>ANSYS</td>
<td></td>
</tr>
<tr>
<td>Location</td>
<td>Select node.</td>
</tr>
<tr>
<td>J MAG</td>
<td></td>
</tr>
</tbody>
</table>

| Orphans settings | Select Extrapolate to extrapolate values into the orphaned regions where this is possible. |
The resulting name for the quantity set will be JouleHeat<EMAG code | Temperature<CFD code.

4. Select Assign Quantity Sets tab and assign the defined quantity set to the built regions.

<table>
<thead>
<tr>
<th>For selected regions</th>
<th>Check quantity set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Region_1</td>
<td>JouleHeat&lt;EMAG code</td>
</tr>
<tr>
<td>Region_2</td>
<td></td>
</tr>
<tr>
<td>Region_3</td>
<td></td>
</tr>
</tbody>
</table>

No changes are required in the Monitors step. You may proceed to the Settings step by pressing Settings.

### 6.3.4 Settings Step

**JMAG**

The JMAG model presents some light differences.

For visualizing the orphan information set the following parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor.Orphans</td>
<td>Set selected.</td>
</tr>
</tbody>
</table>

No further changes are required in the Settings step. Proceed to the Go step by pressing Go.

### 6.3.5 Go Step

In the Go step configure the application start-up and start server and codes (cf. IV-2.8 Go Step – Configuring the Application Startup and Running the Coupled Simulation).

In the server panel, no changes are necessary.

The code panels are described for each code below. If nothing special is mentioned keep the default settings. Further information on the offered code parameters in the Go step can be looked up in the respective code section of the Codes Manual.

1. For your electromagnetism code:

<table>
<thead>
<tr>
<th>ANSYS</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gui option</td>
<td>Keep -b to start ANSYS in batch mode.</td>
</tr>
<tr>
<td>APDL input script</td>
<td>Select the input script &quot;ANSYS/startjob.ans&quot;.</td>
</tr>
</tbody>
</table>

**JMAG**

No code specific options need to be set.

2. For your fluid mechanics code:

**FLUENT**

No code specific options need to be set.
Now you may run the computation.

### 6.4 Running the Computation

Save the MpCCI project file with name "busbar.csp" via the MpCCI GUI menu File→Save Project As. Press the **Start** button in the Go step and the simulation codes should start.

Some codes require additional actions:

<table>
<thead>
<tr>
<th><strong>ANSYS</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS reads the APDL script file &quot;startjob.ans&quot; and starts the computation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>JMAG</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>The JMAG reads the model file &quot;busbar_system.jcf&quot; and starts the computation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>FLUENT</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Open (Solve</td>
</tr>
<tr>
<td>2. Press <strong>Initialize</strong>.</td>
</tr>
<tr>
<td>3. Open (Solve</td>
</tr>
<tr>
<td>4. Press <strong>Calculate</strong>.</td>
</tr>
</tbody>
</table>

### 6.5 Discussion of Results

In this busbar simulation it is seen that the coupling takes ten steps to reach a maximum temperature.

Exemplarily the temperature distribution in the middle plane of the cabinet from the FLUENT solution is given in the next picture (**Figure 2**).

Exemplarily the temperature average for each busbar [K] and the summarized power losses [W] from the coupling between JMAG-FLUENT can be plotted. We can observe the correlation between the temperature and joule heat values (**Figure 3**).
Figure 2: FLUENT results: temperature distribution [K]

Figure 3: JMAG FLUENT results: temperature average [K] and total power losses [W]
7 Three Phase Transformer

7.1 Problem Description

The transformer geometry model is taken from J MAG application notes # 146.
The tutorial covers an electrothermal coupled problem of a transformer carrying a three-phase current.
Losses include copper losses in the coils and iron loss in the core. We have used a step down three phase transformer with a turn ratio of 10 to 1 with the following characteristics:

- The primary voltage of 141.42 V is operating at 60 Hz.
- The connection pattern in the transformer and load sides are Delta-Delta and Y connection respectively.
- The coil resistance for primary winding is 0.031 Ω/phase and 0.00156 Ω/phase for secondary winding.
- External load resistance is 0.06 Ω/phase.
• The core material is 50JN270 (manufacturer JFE steel) with laminating factor of 98.
• The dimensions of the tank are 0.6 m * 0.6 m * 0.3 m.
• The cooling liquid is n-heptane.

Magnetic field analysis handles the phenomena that produce magnetic flux and eddy currents in transformer’s core, when current flows through the coil. A linear rise of resistivity dependent on the temperature of the conductor is also modeled.

The electromagnetic calculation of power-losses is considering:
• eddy currents,
• temperature dependent rise of resistivity.

The thermal calculation is considering:
• heat conduction (fluid and solid),
• heat convection (including buoyancy effects),
• laminar flow.

### 7.2 Model Preparation

The simulation couples an electromagnetic model with a fluid model. The files you need for the simulation are included in the MpCCI distribution. Create a new directory and copy the subdirectories from 
"<MpCCI home>/<tutorial/Transformer" corresponding to the simulation codes you want to use.

#### 7.2.1 Fluid Model

The fluid model is composed of the tank, the core and coils. The core and coils are immersed in a cooling fluid: the n-heptane liquid. A symmetry condition is considered for the model. The inlet for the cooling liquid is located at the bottom of the tank and the outlet is located at the top of the tank. The inlet velocity of the liquid is set to 0.001 m/s. The liquid leaves the domain on a pressure boundary condition. The coils are modeled as conductors based on copper material and the core uses steel material. The tank wall is set to an adiabatic wall boundary condition.

**FLUENT**

No modifications have to be done, as we later carry out a GUI based coupled simulation.

### 7.2.2 Electromagnetic Model

The magnetic property of copper is linear isotropic with a relative permeability of 1. The temperature dependent electrical resistivity is calculated in this way:

\[ \varrho(T) = \varrho_0(1 + \alpha(T - T_{ref})) \]

At reference temperature \( T_{ref} = 300 \) K the value of resistivity is \( \varrho_0 = 1.7241 \times 10^{-8} \) Ωm. The temperature coefficient is \( \alpha = 0.004 \) K\(^{-1}\). The calculation of the current flow and magnetic field is done in frequency domain.

**JMAG**

All necessary model information is prepared in the "transformer.jcf", no further action is needed for generating the jcf file.
• The Coupled Analysis is already activated.
• Electric Properties of the material is set up to be temperature dependent.

An electric circuit is defined as shown in Figure 4. The three-phase voltage source provides an amplitude of 141.42 V. The condition for windings is set to FEM coil which assumes a uniform current distribution in the cross section of the wires. Because of existing symmetry in the structure we construct a half model and analyze this model which has a lower computational cost. This can be done by setting the symmetry boundary condition in JMAG. The tank has not been modeled in JMAG.

In the JMAG directory you have the "transformer_standalone.jcf" file which corresponds to the setup for a standalone simulation.

...
7.3 Setting Up the Coupled Simulation with MpCCI GUI

See IV-2 Setting up a Coupled Simulation and V-4 Graphical User Interface for a detailed description on how to use the MpCCI GUI. Following are the substantive rules to be set in the MpCCI GUI in order to...
run this tutorial.

At first start the MpCCI GUI by running the command \texttt{mpcci gui}.

### 7.3.1 Models Step

In the Models step select and configure the codes to be coupled (cf. \ref{IV-2.4 Models Step – Choosing Codes and Model Files}). Further information on the offered code parameters in the Models step can be looked up in the respective code section of the Codes Manual.

1. Choose your EMAG code (EMAG = electromagnetic) as first code to couple:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select JMAG.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to \texttt{latest} or a version supported by MpCCI (see \ref{VI-9 JMAG}).</td>
</tr>
<tr>
<td>JCF file</td>
<td>Select file &quot;JMAG/transformer.jcf&quot; as input file. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>\texttt{Steady state} is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>

2. Choose your CFD code (CFD = Computational Fluid Dynamics) as second code to couple:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select FLUENT.</td>
</tr>
<tr>
<td>Version</td>
<td>The model is three-dimensional, thus select the FLUENT version \texttt{3ddp}.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to \texttt{latest} or a version supported by MpCCI (see \ref{VI-8 FLUENT}).</td>
</tr>
<tr>
<td>Case file</td>
<td>Select the case file &quot;FLUENT/transformer.cas.gz&quot;. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>\texttt{Steady state} is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>

Proceed to the Algorithm step by pressing \texttt{Algorithm}.

### 7.3.2 Algorithm Step

In the Algorithm step define the coupling algorithm (cf. \ref{IV-2.5 Algorithm Step – Defining the Coupling Algorithm}).

The settings should be as follows. If no action is mentioned, keep the default option.

1. For the Common Basics panel:
### Option | Action
--- | ---
▼ Coupling analysis
Define the coupling scheme | This option is automatically set to Explicit.
▼ Coupling algorithm
Algorithm type | Set to Serial.
Leading code | Set to electromagnetism code.
▼ Coupling duration
Set maximum number of coupling steps | Check this option.
Maximum number of coupling steps | Set to 20 for the frequency response analysis simulation.

2. For the electromagnetism code:

**JMAG**

No code specific options need to be set.

3. For the fluid mechanics code:

**FLUENT**

| Option | Action |
--- | --- |
▼ Coupling steps
Type of coupling step size | Set to Constant.
Constant coupling step size | Set to 20 solver steps without coupling for subcycling.

Press the **Regions** button at the bottom of the window to get to the **Regions** step.

### 7.3.3 Regions Step

In the **Regions** step build the coupling regions, define quantities to be exchanged and assign the defined quantities to the built regions (cf. ▶ IV-2.6 Regions Step – Defining Coupling Regions and Quantities ◄).

1. At first select the **Mesh** tab to get access to the mesh based element components.

2. Select **Build Regions** tab and here the **Setup** tab. Now choose the components to be coupled. In this example the coupling region corresponds to the coils and core. **MpCCI** treats this region as a “Volume” because it represents a 3D structure. Therefore select the **Volume** tab (Ĩ) in the components list where appropriate.

There are 6 conductors (coils) and one core to couple and for each conductor from the EMAG and CFD code a couple region will be created. This will increase the performance and quality of the neighborhood search.

Following table gives an overview of the components to couple for each code in each region:
VII Tutorial

7.4 Running the Computation

<table>
<thead>
<tr>
<th>J MAG</th>
<th>FLUENT</th>
<th>Region name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary windings</td>
<td></td>
<td></td>
</tr>
<tr>
<td>☐ Coil 1_U</td>
<td>☐ p_u</td>
<td>Region_1</td>
</tr>
<tr>
<td>☐ Coil 1_V</td>
<td>☐ p_v</td>
<td>Region_2</td>
</tr>
<tr>
<td>☐ Coil 1_W</td>
<td>☐ p_w</td>
<td>Region_3</td>
</tr>
<tr>
<td>Secondary windings</td>
<td></td>
<td></td>
</tr>
<tr>
<td>☐ Coil 2_U</td>
<td>☐ s_u</td>
<td>Region_4</td>
</tr>
<tr>
<td>☐ Coil 2_V</td>
<td>☐ s_v</td>
<td>Region_5</td>
</tr>
<tr>
<td>☐ Coil 2_W</td>
<td>☐ s_w</td>
<td>Region_6</td>
</tr>
<tr>
<td>Core</td>
<td>☐ core</td>
<td>Region_7</td>
</tr>
</tbody>
</table>

- For each region lookup your codes in the table above and select the components to be coupled by dragging them into the Coupled boxes.
- After building one region, click on button Add to add a new region and fill it in the same way as the one before.
- After all regions have been built, assign them a more understandable name as follows:
  - Select all regions.
  - Point with the mouse on the regions and do a right click.
  - From the popup menu choose Rename automatically and select a code from the list. The components from this code will be used to rename the regions.
  - Selecting e.g. FLUENT for renaming would lead to regions p_u, p_v, p_w, s_u, s_v, s_w and core.

3. Select Define Quantity Sets tab. As currently no quantities are assigned to “EmptyQuantitySet” (created by default), a warning symbol is displayed in front of it. Now choose the quantities to be exchanged.

<table>
<thead>
<tr>
<th>For quantity set</th>
<th>Select quantities</th>
<th>Set sender</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmptyQuantitySet</td>
<td>JouleHeat</td>
<td>EMAG code</td>
</tr>
<tr>
<td></td>
<td>Temperature</td>
<td>CFD code</td>
</tr>
</tbody>
</table>

The resulting name for the quantity set will be JouleHeat<-EMAG code | Temperature<-CFD code.

4. Select Assign Quantity Sets tab and assign the defined quantity set to the built regions.

<table>
<thead>
<tr>
<th>For selected regions</th>
<th>Check quantity set</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;all regions&gt; e.g.</td>
<td></td>
</tr>
<tr>
<td>p_u</td>
<td>JouleHeat&lt;-EMAG code</td>
</tr>
<tr>
<td>p_v</td>
<td></td>
</tr>
<tr>
<td>p_w</td>
<td></td>
</tr>
<tr>
<td>s_u</td>
<td></td>
</tr>
<tr>
<td>s_v</td>
<td></td>
</tr>
<tr>
<td>s_w</td>
<td></td>
</tr>
<tr>
<td>core</td>
<td></td>
</tr>
</tbody>
</table>

No changes are required in the Monitors and Settings steps. You may proceed to the Go step by pressing Go.

**7.3.4 Go Step**

No changes are required in the Monitors step. Now you may run the computation.
7.4 Running the Computation

Save the MpCCI project file with name "transformer.csp" over the MpCCI GUI menu File→Save Project As. Press the Start button in the Go step and the simulation codes should start.

Some codes require additional actions:

<table>
<thead>
<tr>
<th>J MAG</th>
</tr>
</thead>
<tbody>
<tr>
<td>JMAG reads the model file &quot;transformer.jcf&quot; and starts the computation automatically.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FLUENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Open (Solve</td>
</tr>
<tr>
<td>2. Press Initialize.</td>
</tr>
<tr>
<td>3. Open (Solve</td>
</tr>
<tr>
<td>4. Press Calculate.</td>
</tr>
</tbody>
</table>

7.5 Discussion of Results

In this transformer simulation it is seen that the coupling takes 20 steps to reach a minimum temperature of 306 K and a maximum temperature of 320 K.

Exemplarily the temperature distribution in the transformer from the FLUENT solution is given in the next picture (Figure 5).

The MpCCI co-simulation thermal analysis will be compared with a standalone JMAG magneto-thermal analysis. The same JMAG magnetic model "transformer_standalone_MG.jcf" is used to coupled with a JMAG thermal analysis "transformer_standalone_TH.jcf" model.

The thermal analysis model has following properties:

- Use the losses calculation results from the JMAG magnetic simulation. Each coil has a heat source boundary defined.
- Heat transfer boundary for the transformer faces. A constant heat transfer coefficient of 10 W/m².K is used.
- Average temperature condition is set up for each coil.

The JMAG coupled analysis is carried out by JMAG itself.

The Joule loss density distribution [W/m³] in the transformer from the coupling between JMAG-FLUENT can be plotted (Figure 6). The results on the left side correspond to a standalone simulation without consideration of the cooling effect. The maximum Joule loss density reported is about 3.19e+05 W/m³. The coupled simulation on the right side reports a maximum Joule loss density value of 2.22e+04 W/m³.

The temperature distribution [K] in the transformer from the coupling between JMAG-FLUENT can be plotted (Figure 7). The results on the left side correspond to a standalone simulation with consideration of the cooling effect assumptions. The standalone analysis shows a minimum temperature of 293.15 K and maximum temperature of 307.15 K.
Figure 5: FLUENT results: temperature distribution [K]

Figure 6: Joule loss density without cooling (left) and with cooling effect (right) [W/m³] (logarithmic view)
Figure 7: Thermal analysis standalone (left) and with cooling effect (right) [T]
8 Pipe Nozzle

8.1 Problem Description

Topics of this Tutorial
- Fluid-Structure Interaction (FSI)
- Axisymmetric model
- Steady state one-way force mapping
- Use of UDF-functions in FLUENT
- Use of Field Functions in STAR-CCM+
- Models with different unit systems.

Simulation Codes
- Fluid Mechanics: FLUENT, STAR-CCM+
- Solid Mechanics: ANSYS, MSC NASTRAN

8.2 Model Preparation

Figure 1: Pipe Nozzle: Section sketch of the axisymmetric model.
The model is axisymmetric, thus only a section of the model is created and meshed. The axis of revolution is different for different simulation codes. MpCCI uses the vertical $y$-axis. For simulation codes which use different axes, the data is converted in the code adapter.

The solid material is a simple linear elastic material with elastic modulus $E = 2000$ MPa and $\nu = 0.3$. The fluid is modeled as an ideal gas with properties of air.

The fluid is streaming into the nozzle at the pressure inlet as shown in Figure 1. During the simulation the pressure is slowly increased with a user-defined function up to a value of $p_i = 1$ MPa.

The simulation is a steady state simulation, the surface forces are only transferred once from the fluid model to the solid model, where they are applied as boundary conditions to compute the stress distribution.

### 8.2.1 Fluid Model

**FLUENT**

FLUENT uses the horizontal $x$-axis as axis of revolution, thus the model is created as shown in Figure 1. A user-defined function is used to slowly increase the inlet pressure in order to ensure convergence of the FLUENT solver.

Before starting the computation, the "libudf" library for FLUENT must be built:

- Change to the "FLUENT" directory, start fluent 2d and read the casefile "nozzle.cas". FLUENT will print an error message `open_udf_library: No such file or directory`, which indicates that the UDF library is still missing. The error message should not appear if the library was built and is present in the corresponding "libudf" directory.
- Select `Define → User-Defined → Functions → Compiled...` to open the Compiled UDFs panel shown on the left.
- Press the left `Add...` button and select the file "inpressure.c".
- Press [Build] to compile the "libudf" library.
- Finally exit the panel with [Load] to load the library.
- Quit FLUENT, you do not need to save changes.

**STAR-CCM+**

STAR-CCM+ uses the horizontal $x$-axis as axis of revolution, thus the model is created as shown in Figure 1. A user field function is used to slowly increase the inlet pressure in order to ensure convergence of the STAR-CCM+ solver.

Before starting the computation, the user field function $p_{\text{inlet}}$ has been created. This function is hooked to the boundary fluid:inlet as Total Pressure.
8.2.2 Solid Model

Figure 2: Pipe Nozzle: Section sketch of the axisymmetric model.

**ANSYS**

ANSYS uses the vertical \( y \)-axis as axis of revolution in axisymmetric models. Therefore the geometry must be flipped by swapping \( x \)- and \( y \)-coordinates, which yields the configuration shown in Figure 2. In ANSYS, PLANE183 elements are used.

**MSC NASTRAN**

MSC NASTRAN uses the vertical \( y \)-axis as axis of revolution in axisymmetric models. Therefore the geometry must be flipped by swapping \( x \)- and \( y \)-coordinates, which yields the configuration shown in Figure 2. In MSC NASTRAN, CTRIA3X elements are used.

Only one iteration step is carried out.

8.3 Setting Up the Coupled Simulation with MpCCI GUI

See \( \triangleright \) IV-2 Setting up a Coupled Simulation \( \triangleleft \) and \( \triangleright \) V-4 Graphical User Interface \( \triangleleft \) for a detailed description on how to use the MpCCI GUI. Following are the substantive rules to be set in the MpCCI GUI in order to run this tutorial.

Before starting the coupled simulation, ensure that you have built the udf library for FLUENT as described above.

At first start the MpCCI GUI by running the command `mpcci gui`.
8.3.1 Models Step

In the Models step select and configure the codes to be coupled (cf. IV-2.4 Models Step – Choosing Codes and Model Files). Further information on the offered code parameters in the Models step can be looked up in the respective code section of the Codes Manual.

1. Choose your fluid mechanics code as first code to couple:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select FLUENT.</td>
</tr>
<tr>
<td>Version</td>
<td>The model is axisymmetric, i.e. a 2d model. Thus select the FLUENT version 2ddp.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see VI-8 FLUENT).</td>
</tr>
<tr>
<td>Case file</td>
<td>Select the case file &quot;FLUENT/nozzle.cas&quot;.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Steady state is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>

2. Choose your solid mechanics code as second code to couple:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select ANSYS.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see VI-3 ANSYS).</td>
</tr>
<tr>
<td>Product</td>
<td>You can select any ANSYS product which can be used for structural analysis, e.g. ansys.</td>
</tr>
<tr>
<td>Database file</td>
<td>Select database file &quot;ANSYS/nozzle.db&quot;.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Undefined is displayed in this newly added option as a result of the scan process.</td>
</tr>
<tr>
<td>Unit system</td>
<td>Select the SI unit system, because the ANSYS model file was created using these units.</td>
</tr>
</tbody>
</table>
8.3 Setting Up the Coupled Simulation with MpCCI GUI

**MSC NASTRAN**

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select MSC NASTRAN.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see VI-13 MSC NASTRAN).</td>
</tr>
<tr>
<td>Input file</td>
<td>Select input file &quot;MSC.Nastran/nozzle.bdf&quot;. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Steady state is displayed in this newly added option as a result of the scan process.</td>
</tr>
<tr>
<td>Unit system</td>
<td>Select the mm-t-s unit system, because the MSC NASTRAN model was created using these units.</td>
</tr>
</tbody>
</table>

Proceed to the Algorithm step by pressing [Algorithm].

### 8.3.2 Algorithm Step

In the Algorithm step define the coupling algorithm (cf. IV-2.5 Algorithm Step – Defining the Coupling Algorithm).

The settings should be as follows. If no action is mentioned, keep the default option.

1. For the Common Basics panel:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Coupling analysis</td>
<td>This option is automatically set to Explicit.</td>
</tr>
<tr>
<td>Define the coupling scheme</td>
<td>Set to Parallel.</td>
</tr>
<tr>
<td>▼ Coupling algorithm</td>
<td>Check this option.</td>
</tr>
<tr>
<td>Algorithm type</td>
<td>Set to Fluid mechanics code.</td>
</tr>
<tr>
<td>Use initialization</td>
<td>Check this option.</td>
</tr>
<tr>
<td>Initializing code</td>
<td>Set to Fluid mechanics code.</td>
</tr>
<tr>
<td>▼ Coupling duration</td>
<td>Check this option.</td>
</tr>
<tr>
<td>Set maximum number of coupling steps</td>
<td>Set to 1.</td>
</tr>
</tbody>
</table>

2. For the fluid mechanics code:

   No code specific options need to be set.
8 Pipe Nozzle Tutorial

**STAR-CCM+**

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime control</td>
<td></td>
</tr>
<tr>
<td>Define coupling start</td>
<td>Set to Specify iteration to enable a pre-calculation.</td>
</tr>
<tr>
<td>Iteration number for coupling start</td>
<td>Set to 120 to activate a pre-calculation of 120 iterations.</td>
</tr>
</tbody>
</table>

3. For the solid mechanics code:

**ANSYS**

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis</td>
<td></td>
</tr>
<tr>
<td>Analysis type</td>
<td>Select Steady state.</td>
</tr>
</tbody>
</table>

**MSC NASTRAN**

No code specific options need to be set.

... Press the Regions button at the bottom of the window to get to the Regions step.

8.3.3 Regions Step

In the Regions step build the coupling regions, define quantities to be exchanged and assign the defined quantities to the built regions (cf. IV-2.6 Regions Step – Defining Coupling Regions and Quantities).

1. First, select the Mesh tab to get access to the mesh based element components.
2. Select Build Regions tab and here the Setup tab. Now choose the components to be coupled. In this example MpCCI treats the coupling region as a “Face” because it represents a 2D structure. Therefore select the Face tab (▲) in the components list where appropriate and select the coupling components by dragging them into the Coupled boxes.

Lookup your codes in the following table and select the components to be coupled:

<table>
<thead>
<tr>
<th>For fluid mechanics code</th>
<th>Select coupling component</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLUENT</td>
<td>▲ pipesurface</td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>For solid mechanics code</th>
<th>Select coupling component</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>▲ PIPESURFACE</td>
</tr>
<tr>
<td>MSC NASTRAN</td>
<td>▲ nozzle</td>
</tr>
</tbody>
</table>

3. Select Define Quantity Sets tab. As currently no quantities are assigned to “EmptyQuantitySet” (created by default), a warning symbol is displayed in front of it. Now choose the quantities to be exchanged.

<table>
<thead>
<tr>
<th>For quantity set</th>
<th>Select quantity</th>
<th>Set sender</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmptyQuantitySet</td>
<td>RelWallForce</td>
<td>Fluid mechanics code</td>
</tr>
</tbody>
</table>
The resulting name for the quantity set will be **RelWallForce**.

4. Select **Assign Quantity Sets** tab and assign the defined quantity set to the built region.

| Region 1 | RelWallForce |

No changes are required in the **Monitors** and **Settings** steps. You may proceed to the **Go** step by pressing **GO**.

### 8.3.4 Go Step

In the **Go** step configure the application start-up and start server and codes (cf. ▷ IV-2.8 Go Step – Configuring the Application Startup and Running the Coupled Simulation ◁).

In the **server** panel, no changes are necessary.

The code panels are described for each code below. If nothing special is mentioned keep the default settings. Further information on the offered code parameters in the **Go** step can be looked up in the respective code section of the **Codes Manual**.

1. For your **fluid mechanics** code:

   **FLUENT**

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto hook functions</td>
<td>Deselect to keep FLUENT from sending data in each iteration.</td>
</tr>
</tbody>
</table>

   **STAR-CCM+**

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run in batch</td>
<td>Check this option.</td>
</tr>
<tr>
<td>Auto start with the default template macro</td>
<td>Check this option.</td>
</tr>
<tr>
<td>License options</td>
<td>Set according to your license type (cf. ▷ VI-16.2.5 Go Step ◁).</td>
</tr>
</tbody>
</table>

2. For your **solid mechanics** code:

   **ANSYS**

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gui option</td>
<td>Keep -b to start ANSYS in batch mode.</td>
</tr>
<tr>
<td>APDL input script</td>
<td>Select the input script “ANSYS/nozzle.ans”.</td>
</tr>
</tbody>
</table>

   **MSC NASTRAN**

   No code specific options need to be set.

Now you may run the computation.
8.4 Running the Computation

Save the MpCCI project file with name "nozzle.csp" over the MpCCI GUI menu File→Save Project As.
Press the Start button in the Go step and the simulation codes should start.
Some codes require additional actions:

**FLUENT**

- In the FLUENT window, select (Solve|Solution|Solving)→Initialization and press the Initialize button to initialize the FLUENT solution.

- Open the FLUENT Run Calculation panel by selecting (Solve|Solution|Solving)→Run Calculation.... Set the number of iterations to 120 and press the Calculate button. You should see a plot of the FLUENT results during the iterations.

- Open the MpCCI panel which you find under FLUENT menubar MpCCI→MpCCI Control....

- In the MpCCI panel, press the Initialize button in the On Demand Init and Exit panel, which will start the neighborhood search in both codes – you should notice according output in the code windows.

- To send the surface forces from FLUENT to the partner code, press the Send button once. The partner code will now compute the stresses and quit. Close any appearing message windows.

- Press the Finalize button in the MpCCI panel of FLUENT, which disconnects FLUENT from MpCCI and exit FLUENT.

- Quit MpCCI, the simulation is finished.

**STAR-CCM+**

The java macro file "mpcci_runjob_steady.java" will carry the following steps automatically:

- Initializing the solution.

- Run 120 iterations.

- Connecting STAR-CCM+ to MpCCI server.

- Perform the data exchange, which is only a SEND operation.

- Run 1 iteration.

- Finalize the coupled simulation.

- Results are saved in "nozzle@00121.sim".

Have a look at the stress results in the solid mechanics code and compare them to those shown below.
The stress distribution in the **ANSYS** model is shown on the left.

The stress distribution in the **MSC NASTRAN** model is shown on the left.
9 Cube in a Duct Heater

9.1 Problem Description

![Figure 1: Duct Heater: Geometry [m] and boundary conditions section view](image)

Topics of this Tutorial
- Radiative Heat Transfer
- Steady State
- Forced convection
- 3D-model
- Subcycling

Simulation Codes
- Radiation: TAITherm
- Fluid Mechanics: FLUENT, OpenFOAM, STAR-CCM+

Description
The example case presented in this tutorial describes the forced convection. For this purpose a cube is inserted in an rectangular duct, which is heated up by a heater at the bottom of the duct. Additionally the cube is cooled by air passing through the rectangular duct.

9.2 Model Preparation

The simulation couples a radiation model with a fluid mechanics model. The files which you need for the simulation are included in the MpCCI distribution. Create a new directory "DuctHeater" and copy the subdirectories from "<MpCCI_home>/tutorial/DuctHeater" which correspond to the simulation codes you want to use.
9.2.1 Radiation Model

**TAITherm**

All values at standards settings use temperature calculated except for the heater. The heater is set on a constant surface temperature identical (T=673K) to the CFD settings.

The maximum iterations is set to 499.

---

9.2.2 Fluid Model

The fluid domain comprises the inner part of the duct heater (Table 1.)

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Density</td>
<td>1.205 $\frac{kg}{m^3}$</td>
</tr>
<tr>
<td>Conductivity</td>
<td>0.0237 $\frac{W}{m \cdot K}$</td>
</tr>
<tr>
<td>Specific heat</td>
<td>1006 $\frac{J}{kg \cdot K}$</td>
</tr>
<tr>
<td>Inlet</td>
<td></td>
</tr>
<tr>
<td>Velocity</td>
<td>0.5 $\frac{m}{s}$</td>
</tr>
<tr>
<td>Temperature</td>
<td>293.0 K</td>
</tr>
<tr>
<td>Turbulent Intensity</td>
<td>0.1</td>
</tr>
<tr>
<td>Turbulent Length</td>
<td>0.02 m</td>
</tr>
<tr>
<td>Outlet</td>
<td></td>
</tr>
<tr>
<td></td>
<td>standard setting</td>
</tr>
<tr>
<td>Cube</td>
<td></td>
</tr>
<tr>
<td>wall heat</td>
<td>fixed</td>
</tr>
<tr>
<td>Temperature</td>
<td>293.0 K</td>
</tr>
<tr>
<td>Duct</td>
<td></td>
</tr>
<tr>
<td>wall heat</td>
<td>fixed</td>
</tr>
<tr>
<td>Temperature</td>
<td>293.0 K</td>
</tr>
<tr>
<td>Heater</td>
<td></td>
</tr>
<tr>
<td>wall heat</td>
<td>fixed</td>
</tr>
<tr>
<td>Temperature</td>
<td>673.0 K</td>
</tr>
</tbody>
</table>

Table 1: Duct Heater: Boundary conditions for fluid model

The setting used for fluid model are in the following Table 2

**FLUENT**

A standard $k - \epsilon$ model with enhanced wall treatment is applied.

**STAR-CCM+**

A standard $k - \epsilon$ turbulence model is applied with wall treatment.

In general, STAR-CCM+ needs an existing solution for the thermal coupling when STAR-CCM+ should send some data first. A cold start would fail in this case. In the Algorithm step STAR-CCM+ will be configured to calculate an initial solution before the coupling begins.


<table>
<thead>
<tr>
<th>Time Domain</th>
<th>Steady state</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gravity</td>
<td>$-9.81 \frac{m}{s^2}$</td>
</tr>
<tr>
<td>Thermal option</td>
<td>All settings off</td>
</tr>
<tr>
<td>Buoyancy for AIR</td>
<td>On</td>
</tr>
</tbody>
</table>

Table 2: Duct Heater: Parameter setting for fluid model

---

**OpenFOAM**

A standard $k-\varepsilon$ turbulence model with wall treatment is applied. The solver `buoyantBoussinesqSimpleFoam` is used.

---

### 9.3 Setting Up the Coupled Simulation with MpCCI GUI

See [IV-2 Setting up a Coupled Simulation](#) and [V-4 Graphical User Interface](#) for a detailed description on how to use the MpCCI GUI. Following are the substantive rules to be set in the MpCCI GUI in order to run this tutorial.

At first start the MpCCI GUI by running the command `mpcci gui`. 
### 9.3.1 Models Step

In the Models step select and configure the codes to be coupled (cf. IV-2.4 Models Step – Choosing Codes and Model Files). Further information on the offered code parameters in the Models step can be looked up in the respective code section of the Codes Manual.

1. Choose your radiation code as first code to couple:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select TAITherm.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see VI-17 TAITherm).</td>
</tr>
<tr>
<td>&quot;.tdf&quot; file</td>
<td>Select &quot;radtherm_model.tdf&quot; as input file. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Steady state is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>

2. Choose your fluid mechanics code as second code to couple:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select FLUENT.</td>
</tr>
<tr>
<td>Version</td>
<td>The model is three-dimensional, thus select the FLUENT version 3ddp.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see VI-8 FLUENT).</td>
</tr>
<tr>
<td>Case file</td>
<td>Select the case file &quot;fluent_model.cas&quot;. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Steady state is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select OpenFOAM.</td>
</tr>
<tr>
<td>Option</td>
<td>Select the OpenFOAM option Opt or Debug.</td>
</tr>
<tr>
<td>Precision</td>
<td>Select the OpenFOAM precision: DP for double or SP for single precision.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see VI-14 OpenFOAM).</td>
</tr>
<tr>
<td>Case directory</td>
<td>Select the case directory &quot;OpenFOAM/.VERSION&quot; fitting your OpenFOAM version (e.g. &quot;OpenFOAM/v1606-v2012&quot; for use with OpenFOAM v1606 to v2012). The files will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Steady state is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>
STAR-CCM+

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select STAR-CCM+.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see VI-16 STAR-CCM+).</td>
</tr>
<tr>
<td>Simulation file</td>
<td>Select the simulation file &quot;STAR-CCM+/ductheater.sim&quot;. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Steady state is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>

Proceed to the Algorithm step by pressing [Algorithm].

### 9.3.2 Algorithm Step

In the Algorithm step define the coupling algorithm (cf. IV-2.5 Algorithm Step – Defining the Coupling Algorithm).

The settings should be as follows. If no action is mentioned, keep the default option.

1. For the Common Basics panel:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Coupling analysis</td>
<td></td>
</tr>
<tr>
<td>Define the coupling scheme</td>
<td>This option is automatically set to Explicit.</td>
</tr>
<tr>
<td>▼ Coupling algorithm</td>
<td></td>
</tr>
<tr>
<td>Algorithm type</td>
<td>Set to Parallel.</td>
</tr>
<tr>
<td>Use initialization</td>
<td>Check this option.</td>
</tr>
<tr>
<td>Initializing code</td>
<td>Set to radiation code.</td>
</tr>
<tr>
<td>▼ Coupling duration</td>
<td></td>
</tr>
<tr>
<td>Set maximum number of coupling steps</td>
<td>Check this option.</td>
</tr>
<tr>
<td>Maximum number of coupling steps</td>
<td>Set to 10.</td>
</tr>
</tbody>
</table>

2. For the radiation code:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Coupling steps</td>
<td></td>
</tr>
<tr>
<td>Type of coupling step size</td>
<td>Set to Constant.</td>
</tr>
<tr>
<td>Constant coupling step size</td>
<td>Set to 50 solver steps without coupling for subcycling.</td>
</tr>
</tbody>
</table>

3. For the fluid mechanics code:
9.3 Setting Up the Coupled Simulation with MpCCI GUI

### FLUENT

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Coupling steps</td>
<td></td>
</tr>
<tr>
<td>Type of coupling step size</td>
<td>Set to Constant.</td>
</tr>
<tr>
<td>Constant coupling step size</td>
<td>Set to 50 solver steps without coupling for subcycling.</td>
</tr>
</tbody>
</table>

### OpenFOAM

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Coupling steps</td>
<td></td>
</tr>
<tr>
<td>Type of coupling step size</td>
<td>Set to Constant.</td>
</tr>
<tr>
<td>Constant coupling step size</td>
<td>Set to 50 solver steps without coupling for subcycling.</td>
</tr>
</tbody>
</table>

### STAR-CCM+

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Coupling steps</td>
<td></td>
</tr>
<tr>
<td>Type of coupling step size</td>
<td>Set to Constant.</td>
</tr>
<tr>
<td>Constant coupling step size</td>
<td>Set to 50 solver steps without coupling for subcycling.</td>
</tr>
<tr>
<td>▼ Runtime control</td>
<td></td>
</tr>
<tr>
<td>Define coupling start</td>
<td>Set to Specify iteration to enable a pre-calculation.</td>
</tr>
<tr>
<td>Iteration number for coupling start</td>
<td>Set to 5 to activate a pre-calculation of 5 iterations. STAR-CCM+ needs a solution before sending the quantities.</td>
</tr>
</tbody>
</table>

Press the **Regions** button at the bottom of the window to get to the **Regions** step.

#### 9.3.3 Regions Step

In the **Regions** step build the coupling regions, define quantities to be exchanged and assign the defined quantities to the built regions (cf. ▶ IV-2.6 Regions Step – Defining Coupling Regions and Quantities ◀).

1. At first select the **Mesh** tab to get access to the mesh based element components.

2. Select **Build Regions** tab and here the **Setup** tab. Now choose the components to be coupled. In this example the coupling region corresponds to the surface of the duct and the rectangle. **MpCCI** treats these regions as a “Face” because they represent a 2D structure. Therefore select the **Face** tab (▲) in the components list where appropriate and select the coupling components.

Because in this example the coupling components for all codes are named by a special prefix, you may choose an easier way to select all coupling components by using rules (cf. **Figure 19 (part V)**). We want to couple all components starting with the strings `cube_` or `duct_`, but not those ending in `-backside`. The latter ones are automatically generated by the **TAItherm** scanner and are actually not involved in the coupling. So we will create three rules, one for generating an unused region with the `-backside` components a second for the `cube_` components and a third for the `duct_` components. The last two rules can be combined in a rules list to generate a joined region. (By the way, you may choose another way e.g. by creating only one rule with a more complex regular expression, but here we won’t make a depression on regular expressions but would like to show the power of rules and rules lists.)
To create the rule for the unused backside components press the \textbf{New} button in the \textit{Rules} tab and set following properties:

\begin{tabular}{|l|l|}
\hline
Option & Action  \\
\hline
Rule name & Type in unusedBackside.  \\
\hline
Region criterion & Select Matching pattern and type in .*-backside  \\
\hline
Region type & Set Unused.  \\
\hline
Region naming & Select Matching pattern.  \\
\hline
\end{tabular}

Press \textbf{Ok} button to save the rule.

Generate unused region .*-backside with backside components by selecting unusedBackside in the rule list and pressing the \textbf{Generate} button.

To create the rule for the components starting with \textit{cube}, press the \textbf{New} button again and set following properties:

\begin{tabular}{|l|l|}
\hline
Option & Action  \\
\hline
Rule name & Type in cube.  \\
\hline
Region criterion & Select Matching pattern and type in cube_.*  \\
\hline
Region type & Keep Unused deselected.  \\
\hline
Region naming & Select Matching pattern.  \\
\hline
\end{tabular}

Press \textbf{Ok} button to save the rule.

To create the rule for the components starting with \textit{duct}, press the \textbf{New} button again and set following properties:

\begin{tabular}{|l|l|}
\hline
Option & Action  \\
\hline
Rule name & Type in duct.  \\
\hline
Region criterion & Select Matching pattern and type in duct_.*  \\
\hline
Region type & Keep Unused deselected.  \\
\hline
Region naming & Select Matching pattern.  \\
\hline
\end{tabular}

Press \textbf{Ok} button to save the rule.

Now select the \textit{Rules lists} tab and press the \textbf{New} button to create a rules list combining the just created rules. Set following properties:

\begin{tabular}{|l|l|}
\hline
Option & Action  \\
\hline
Rules list name & Type in cubeDuct.  \\
\hline
Region criterion & Select Generate one merged region and by join.  \\
\hline
Rules list & Add cube from Available rules.  \\
\hline
& Add duct from Available rules.  \\
\hline
\end{tabular}

Press \textbf{Ok} button to save the rules list.

Generate coupling region cubeDuct by selecting cubeDuct in the rules list and pressing the \textbf{Generate} button.

3. Select \textit{Define Quantity Sets} tab. As currently no quantities are assigned to “EmptyQuantitySet” (created by default), a warning symbol is displayed in front of it. Now choose the quantities to be exchanged. For this use a quantity set predefined by MpCCI (cf. \textit{III} V-4.6.8 Assigning Preconfigured
Quantity Settings:<br><br>For quantity set | Option | Action<br>--- | --- | ---<br>EmptyQuantitySet | Select Coupling Type | Select Steady state radiative heat transfer and click Set.<br><br>Now the quantities FilmTemp, WallHTCoeff and WallTemp are activated and MpCCI already set the appropriate sender for them.<br><br>The resulting name for the quantity set will be FilmTemp←Fluid mechanics code | WallHTCoeff←Fluid mechanics code | WallTemp←Radiation code.<br><br>4. Select Assign Quantity Sets tab and assign the defined quantity set to the built region.<br><br>For selected region | Check quantity set<br>--- | ---<br>cubeDuct | FilmTemp←Fluid mechanics code | WallHTCoeff←Fluid mechanics code | WallTemp←Radiation code<br><br>No changes are required in the Monitors and Settings steps. You may proceed to the Go step by pressing Go.<br><br>9.3.4 Go Step<br><br>In the Go step configure the application start-up and start server and codes (cf. IV-2.8 Go Step – Configuring the Application Startup and Running the Coupled Simulation).<br><br>In the server panel, no changes are necessary.<br><br>The code panels are described for each code below. If nothing special is mentioned keep the default settings. Further information on the offered code parameters in the Go step can be looked up in the respective code section of the Codes Manual.<br><br>1. For your radiation code:<br><br>**TAITherm**<br><br>Option | Action<br>--- | ---<br>Run in batch | Check this option.<br><br>2. For your fluid mechanics code:<br><br>**FLUENT**<br><br>Run in batch | Check this option.<br>Optional journal files | Select the journal file "run.jou".<br><br>**OpenFOAM**<br><br>Option | Action<br>--- | ---<br>Select OpenFOAM solver | Leave the default Auto-Select-by-Case. The solver setting from the "controlDict" file (buoyantBoussinesqSimpleFoam) will be used.<br>Define a reference density | Set density to 1.205 kg/m³.<br>Define a reference specific heat | Set specific heat to 1006 J/kgK according to the fluid properties from Table 1.
9.4 Running the Computation

9.4.1 Starting the Simulation

Save the MpCCI project file with name "duct.heater.csp" over the MpCCI GUI menu
File→Save Project As.

Press the Start button in the Go step and the simulation codes should start.

Some codes require additional actions:

**TAITherm**

TAITherm calculation is started in batch mode. The tdf file has been prepared to run 500 iterations.

**FLUENT**

The FLUENT calculation is started in batch mode. The maximum number of iterations has been set to 500 in the journal file and at the end of the solution the case and dat file are saved.

**STAR-CCM+**

The STAR-CCM+ calculation is started in batch mode. The maximum number of iterations has been set to 505 by the java macro file as stopping criteria.

**OpenFOAM**

The OpenFOAM calculation is started in batch mode. The maximum number of iterations has set to 500 in the "controlDict" file.

... 

9.5 Discussion of Results

**TAITherm**

Temperature results may be visualized in TAITherm GUI. (Figure 2)

**FLUENT**

FLUENT result file "fluent.model.dat" may be post processed by FLUENT.

**STAR-CCM+**

STAR-CCM+ result file "ductheater.sim" may be post processed with STAR-CCM+ GUI. At the end of the computation STAR-CCM+ may saved the simulation under the name "ductheater@505.sim"
OpenFOAM

OpenFOAM results may be post processed with paraFoam.

... 

By using the MpCCI Visualizer you may visualize the coupled values from the "mpccirun-0000.ccvx". You may open the tools from the MpCCI GUI **Tools → Visualizer**. This will open automatically the "mpccirun-0000.ccvx" file.

![Duct Heater: Temperature section plot](image)

Figure 2: Duct Heater: Temperature section plot
Figure 3: Duct Heater: FLUENT wall temperature contour plot
Figure 4: Duct Heater: \textsc{FLUENT} total temperature contour plot
10 Y-Junction

10.1 Problem Description

Figure 1: Y-Junction: Fluid geometry and network boundaries

Topics of this Tutorial
- 1D network model
- 3D-model
- Steady State

Simulation Codes
- Network: FloMASTER
- Fluid Mechanics: FLUENT, OpenFOAM, STAR-CCM+

10.2 Model Preparation

The simulation couples a network model with a fluid mechanics model. The files which you need for the simulation are included in the MpCCI distribution. Create a new directory "Y-Junction" and copy the subdirectories from "<MpCCI_home>/tutorial/Y-Junction" which correspond to the simulation codes you want to use.
10.2.1 Network Model

FloMASTER

The FloMASTER model (Figure 2) is composed of three networks. Each network will prescribe a flow (inlet/outlet) or pressure boundaries to the fluid model. The boundaries in a fluid model are represented within a FloMASTER network by a source.

Figure 2: Y-Junction: FloMASTER network

Network Upstream Components Settings

- Liquid type is water.
- The pressure source component “12” has pressure value set to $4 \times 10^5$ Pa.
- The pressure source component “13” has no specific pressure value set.

Network First Downstream Branch Settings

- Liquid type is water.
- The pressure source component “14” has pressure value set to $1.5 \times 10^5$ Pa.
- The flow source component “15” has an initial volumetric flow rate value set to $0.0005 \, \text{m}^3/\text{s}$. 
Network Second Downstream Branch Settings

- Liquid type is water.
- The pressure source component “27” has pressure value set to 1.2e+5 Pa.
- The flow source component “26” has an initial volumetric flow rate value set to 0.0004 m³/s.

Extract the FloMASTER Network
1. Start FloMASTER then you need to log on the FloMASTER database.
2. Unpack the appropriate "Y_Junction.FMpk" pack file from the 
"<MpcCI_home>/tutorial/Y-Junction/FloMASTER " folder.
   Unpack the file directly under the root of your project tree.
Configure the FloMASTER Network for the Co-Simulation

Each source boundary has to activate its External Model Boundary property. For the following boundary IDs “13”, “15”, “26” you have to follow these instructions:

1. Select the boundary component.

2. Edit the boundary with the Edit mouse menu context.

3. Select in the property list the External Model Boundary item.

4. Click on the Sub Form... button.

5. Change the property Boundary Active value to Yes.

6. Click on the button Return.

Run a standalone incompressible steady state (SS) FloMASTER analysis to test the configuration.

Export the FloMASTER Network Co-Simulation Information

- In the Network Views select the Data tab option.
- Click on the icon button that Generate a network data or analysis results report.
- Select the MpCCI Link File report type and click on the button OK.
10 Y-Junction Tutorial

- From the list of Active Boundary Components, select each item and click on the button Add >.
- All active boundary components are in the Selected Boundaries.
- Select a boundary and provide a description by filling the text field Add/Edit Boundary Description.
- For the boundary “13”, enter the name “mass-in”.
- For the boundary “15”, enter the name “p-out1”.
- For the boundary “26”, enter the name “p-out2”.
- Click on the button to select an output directory. (e.g. "<MpCCI_home>/tutorial/Y-Junction/FloMASTER")
- In the File Name field provide the name "Y_Junction.fmlink".
- Click on the button [Create ASCII File...]
- The file has been created, the dialog can be closed.
- Exit the FloMASTER graphical interface.

Do not renumber any Network used for co-simulation as component IDs are used to identify boundaries in MpCCI link file.

10.2.2 Fluid Model

The fluid model is a 3D model using the following settings:
- Turbulence model: k-epsilon RNG
- Time: Steady State
- Fluid: water
- Initial velocity 20 m/s
- Mass flow inlet boundary has an initial mass flux rate of 24950 kg m⁻² s⁻¹
- Pressure outlet boundary has an initial pressure of 1.5e+5 Pa and 1.2e+5 Pa.
- Reference Pressure is 101325 Pa.
Figure 3: Y-Junction: mesh
10.3 Setting Up the Coupled Simulation with MpCCI GUI

See \( \triangleleft \) IV-2 Setting up a Coupled Simulation \( \triangleleft \) and \( \triangleleft \) V-4 Graphical User Interface \( \triangleleft \) for a detailed description on how to use the MpCCI GUI. Following are the substantive rules to be set in the MpCCI GUI in order to run this tutorial.

At first start the MpCCI GUI by running the command \texttt{mpcci gui}.

10.3.1 Models Step

In the Models step select and configure the codes to be coupled (cf. \( \triangleleft \) IV-2.4 Models Step – Choosing Codes and Model Files \( \triangleleft \)). Further information on the offered code parameters in the Models step can be looked up in the respective code section of the Codes Manual.

1. Choose your network code as first code to couple:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select FloMASTER.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see ( \triangleleft ) VI-7 FloMASTER ( \triangleleft )).</td>
</tr>
<tr>
<td>Data file</td>
<td>Select data file &quot;FloMASTER/Y.Junction.fmlink&quot;. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Undefined is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>

2. Choose your fluid mechanics code as second code to couple:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select FLUENT.</td>
</tr>
<tr>
<td>Version</td>
<td>The model is three-dimensional, thus select the FLUENT version 3ddp.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see ( \triangleleft ) VI-8 FLUENT ( \triangleleft )).</td>
</tr>
<tr>
<td>Case file</td>
<td>Select the case file &quot;FLUENT/Y-junction.cas.gz&quot;. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Steady state is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>
10.3 Setting Up the Coupled Simulation with MpCCI GUI

### OpenFOAM

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select OpenFOAM.</td>
</tr>
<tr>
<td>Option</td>
<td>Select the OpenFOAM option Opt or Debug.</td>
</tr>
<tr>
<td>Precision</td>
<td>Select the OpenFOAM precision: DP for double or SP for single precision.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see VI-14 OpenFOAM).</td>
</tr>
<tr>
<td>Case directory</td>
<td>Select the case directory &quot;OpenFOAM/&lt;version&gt;&quot; fitting your OpenFOAM version (e.g. &quot;OpenFOAM/v1606-v2012&quot; for use with OpenFOAM v1606 to v2012). The files will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Steady state is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>

### STAR-CCM+

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select STAR-CCM+.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see VI-16 STAR-CCM+).</td>
</tr>
<tr>
<td>Simulation file</td>
<td>Select the simulation file &quot;STAR-CCM+/yjunction.sim&quot;. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Steady state is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>

A warning window will appear and complains about the model dimensions. You can continue by clicking on **Continue**.

Proceed to the Algorithm step by pressing **Algorithm**.

### 10.3.2 Algorithm Step

In the Algorithm step define the coupling algorithm (cf. IV-2.5 Algorithm Step – Defining the Coupling Algorithm).

The settings should be as follows. If no action is mentioned, keep the default option.

1. For the Common Basics panel:
### Option: Coupling analysis

**Define the coupling scheme**

This option is automatically set to **Explicit**.

**Algorithm type**

Set to **Serial**.

**Leading code**

Set to **network code**.

### Option: Coupling duration

**Set maximum number of coupling steps**

Check this option.

**Maximum number of coupling steps**

Set to **50**.

2. For the **network code**:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Analysis</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Analysis type</strong></td>
<td><strong>Select Steady state.</strong></td>
</tr>
</tbody>
</table>

3. For the **fluid mechanics code**:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Coupling steps</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Type of coupling step size</strong></td>
<td><strong>Set to Constant.</strong></td>
</tr>
<tr>
<td><strong>Constant coupling step size</strong></td>
<td><strong>Set to 20 for number of steps without coupling.</strong></td>
</tr>
<tr>
<td><strong>Runtime control</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Define coupling start</strong></td>
<td><strong>Set to Specify iteration to enable a pre-calculation.</strong></td>
</tr>
<tr>
<td><strong>Iteration number for coupling start</strong></td>
<td><strong>Set to 200 to activate a pre-calculation of 200 iterations.</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OpenFOAM</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Coupling steps</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Type of coupling step size</strong></td>
<td><strong>Set to Constant.</strong></td>
</tr>
<tr>
<td><strong>Constant coupling step size</strong></td>
<td><strong>Set to 20 for number of steps without coupling.</strong></td>
</tr>
<tr>
<td><strong>Runtime control</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Define coupling start</strong></td>
<td><strong>Set to Specify iteration to enable a pre-calculation.</strong></td>
</tr>
<tr>
<td><strong>Iteration number for coupling start</strong></td>
<td><strong>Set to 200 to activate a pre-calculation of 200 iterations.</strong></td>
</tr>
</tbody>
</table>
10.3 Setting Up the Coupled Simulation with MpCCI GUI

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Coupling steps</td>
<td></td>
</tr>
<tr>
<td>Type of coupling step size</td>
<td>Set to Constant.</td>
</tr>
<tr>
<td>Constant coupling step size</td>
<td>Set to 20 for number of steps without coupling.</td>
</tr>
<tr>
<td>▼ Runtime control</td>
<td></td>
</tr>
<tr>
<td>Define coupling start</td>
<td>Set to Specify iteration to enable a pre-calculation.</td>
</tr>
<tr>
<td>Iteration number for coupling start</td>
<td>Set to 200 to activate a pre-calculation of 200 iterations.</td>
</tr>
</tbody>
</table>

Press the Regions button at the bottom of the window to get to the Regions step.

### 10.3.3 Regions Step

In the Regions step build the coupling regions, define quantities to be exchanged and assign the defined quantities to the built regions (cf. ▷ IV-2.6 Regions Step – Defining Coupling Regions and Quantities ◁).

1. At first select the Mesh tab to get access to the mesh based element components.

2. Select Build Regions tab and here the Setup tab. Now choose the components to be coupled. In this example the coupling region corresponds to the surface of the inlet and outlet boundaries. For the network code MpCCI treats this region as a zero-dimensional “Integration Point”. For the fluid mechanics codes MpCCI treats this region as a “Face” because it represents a 2D structure. Therefore select the appropriate tab (Integration Point (●) resp. Face (▲)) in the components list where appropriate and select the coupling components by dragging them into the Coupled boxes.

Lookup your codes in the following tables and select the components to be coupled:

- Build the first region.

<table>
<thead>
<tr>
<th>For network code</th>
<th>Select coupling component</th>
</tr>
</thead>
<tbody>
<tr>
<td>FloMASTER</td>
<td>● mass-in</td>
</tr>
<tr>
<td>For fluid mechanics code</td>
<td>Select coupling component</td>
</tr>
<tr>
<td>FLUENT</td>
<td>▲ mass-in</td>
</tr>
<tr>
<td>OpenFOAM</td>
<td></td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td></td>
</tr>
</tbody>
</table>

- Click on button Add to add new region Region_2.
• For region Region_2

<table>
<thead>
<tr>
<th>For network code</th>
<th>Select coupling component</th>
</tr>
</thead>
<tbody>
<tr>
<td>FloMASTER</td>
<td>▲ p-out1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>For fluid mechanics code</th>
<th>Select coupling component</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLUENT</td>
<td>▲ p-out1</td>
</tr>
<tr>
<td>OpenFOAM</td>
<td></td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td></td>
</tr>
</tbody>
</table>

• For region Region_3

<table>
<thead>
<tr>
<th>For network code</th>
<th>Select coupling component</th>
</tr>
</thead>
<tbody>
<tr>
<td>FloMASTER</td>
<td>▲ p-out2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>For fluid mechanics code</th>
<th>Select coupling component</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLUENT</td>
<td>▲ p-out2</td>
</tr>
<tr>
<td>OpenFOAM</td>
<td></td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td></td>
</tr>
</tbody>
</table>

3. Select Define Quantity Sets tab. As currently no quantities are assigned to “EmptyQuantitySet” (created by default), a warning symbol is displayed in front of it. Now choose the quantities to be exchanged.

<table>
<thead>
<tr>
<th>For quantity set EmptyQuantitySet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select quantities</td>
</tr>
<tr>
<td>MassFluxRate</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>MassFluxRate</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>TotalPressure</td>
</tr>
</tbody>
</table>

The resulting name for the quantity set will be MassFluxRate<–Network code | TotalPressure<–Fluid mechanics code.

4. Apply an operator to MassFluxRate to get a stable computation (cf. V-4.6.7.4 Applying Operators to Quantities of Mesh Based Components). Under-relaxation will be needed. Therefore mark the wanted quantity by clicking on its name and configuring the listed operator as follows:

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Operator</th>
<th>Operator option</th>
<th>Operator action</th>
</tr>
</thead>
<tbody>
<tr>
<td>MassFluxRate</td>
<td>Select Relaxation</td>
<td>Relaxation method</td>
<td>Select Fixed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Relaxation factor</td>
<td>Set to 0.3 for under-relaxation.</td>
</tr>
</tbody>
</table>

The relaxation operator is a post processor which will be applied to the receiver of the quantity.

5. Click on button Add to add new quantity set EmptyQuantitySet.
For quantity set EmptyQuantitySet

<table>
<thead>
<tr>
<th>Select quantities</th>
<th>Make quantity settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option</td>
<td>Action</td>
</tr>
<tr>
<td>MassFlowRate</td>
<td>Set sender</td>
</tr>
<tr>
<td>TotalPressure</td>
<td>Set sender</td>
</tr>
</tbody>
</table>

**Code specific settings**

---

The resulting name for the quantity set will be `MassFlowRate<Fluid mechanics code | TotalPressure<Network code`.

6. Apply an operator to `TotalPressure` to get a stable computation (cf. V-4.6.7.4 Applying Operators to Quantities of Mesh Based Components). Under-relaxation will be needed. Therefore mark the wanted quantity in quantity set `MassFlowRate<Fluid mechanics code | TotalPressure<Network code` by clicking on its name and configuring the listed operator as follows:

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Operator</th>
<th>Operator option</th>
<th>Operator action</th>
</tr>
</thead>
<tbody>
<tr>
<td>TotalPressure</td>
<td>Select Relaxation.</td>
<td>Relaxation method</td>
<td>Select Fixed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Relaxation factor</td>
<td>Set to 0.3 for under-relaxation.</td>
</tr>
</tbody>
</table>

The relaxation operator is a post processor which will be applied to the receiver of the quantity.

7. Select **Assign Quantity Sets** tab and assign the defined quantity sets to the built regions.

For selected regions | Check quantity sets |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Region_1</td>
<td>MassFluxRate&lt;Network code</td>
</tr>
<tr>
<td>Region_2</td>
<td>MassFlowRate&lt;Fluid mechanics code</td>
</tr>
<tr>
<td>Region_3</td>
<td></td>
</tr>
</tbody>
</table>

Proceed to the **Monitors** step by pressing [Monitors].

### 10.3.4 Monitors Step

To monitor the total pressure distribution on the uncoupled pipe surface of the fluid mechanics code an additional component has to be selected.

- Select the **Mesh** tab to get access to the mesh based element components.
- Lookup your code and configure the component to be monitored.

<table>
<thead>
<tr>
<th>For fluid mechanics code</th>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLUENT</td>
<td>Component Dimension</td>
<td>Select Face (▲)</td>
</tr>
<tr>
<td>OpenFOAM</td>
<td>Monitored</td>
<td>Select component ▲ wall</td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td>Quantity</td>
<td>Check TotalPressure</td>
</tr>
</tbody>
</table>

No changes are required in the **Settings** step. You may proceed to the **Go** step by pressing [Go].
10.3.5 Go Step

In the Go step configure the application start-up and start server and codes (cf. IV-2.8 Go Step – Configuring the Application Startup and Running the Coupled Simulation). 

In the server panel, no changes are necessary.

The code panels are described for each code below. If nothing special is mentioned keep the default settings. Further information on the offered code parameters in the Go step can be looked up in the respective code section of the Codes Manual.

1. For your network code:

<table>
<thead>
<tr>
<th>FloMASTER</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Option</td>
<td>Action</td>
</tr>
<tr>
<td>User name</td>
<td>Enter the user name to log in the database.</td>
</tr>
<tr>
<td>Working project name</td>
<td>Change the name of the working project corresponding to the root of your project list view in FloMASTER if necessary.</td>
</tr>
<tr>
<td>Analysis type</td>
<td>Select SS for the analysis type.</td>
</tr>
</tbody>
</table>

2. For your fluid mechanics code:

<table>
<thead>
<tr>
<th>FLUENT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Option</td>
<td>Action</td>
</tr>
<tr>
<td>Optional journal file</td>
<td>Select journal file &quot;FLUENT/job.jou&quot;.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OpenFOAM</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Option</td>
<td>Action</td>
</tr>
<tr>
<td>Define a reference pressure</td>
<td>Set pressure to 101325 N/m².</td>
</tr>
<tr>
<td>Define a reference density</td>
<td>Set density to 998 kg/m³.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STAR-CCM+</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Option</td>
<td>Action</td>
</tr>
<tr>
<td>Run in batch</td>
<td>Check this option.</td>
</tr>
<tr>
<td>Auto start with the default template macro</td>
<td>Check this option.</td>
</tr>
<tr>
<td>License options</td>
<td>Set according to your license type (cf. VI-16.2.5 Go Step).</td>
</tr>
</tbody>
</table>

Now you may run the computation.

10.4 Running the Computation

10.4.1 Starting the Simulation

Save the MpCCI project file with name "y_junction.csp" over the MpCCI GUI menu File→Save Project As. Press the Start button in the Go step and the simulation codes should start.
10.5 Discussion of Results

**FloMASTER**
You can open the FloMASTER GUI and open the Y-Junction project. You can select some last results and plot a chart of the pressure or mass flow for one boundary component for example.

**FLUENT**
After FLUENT finished the 1000 iterations, you could save the data and post process the results with FLUENT.

**OpenFOAM**
OpenFOAM results may be visualized with paraFoam.

**STAR-CCM+**
STAR-CCM+ result file "yjunction@01000.sim" may be post processed.

On the fluid model we expect that the pressure at the inlet and outlet is closed to the prescribed pressure values imposed by the network model.
At the inlet boundary “mass-in” a pressure of $4 \times 10^5$ Pa was prescribed on the network model and on the fluid a total pressure value around $2.3 \times 10^5$ Pa is expected.
At the outlet boundary “p-out1” and “p-out2” a total pressure value of $1.5 \times 10^5$ Pa respectively $1.2 \times 10^5$ Pa is expected.
By using the MpCCI Visualizer you may visualize the coupled values from the
tracefile directory "mpccirun_<TIMESTAMP>_ccvx" by executing the command from the tracefile directory:

mpcci visualize mpccirun-0000.ccvx

Figure 4: Y-Junction: Fluent Residuals plot
Figure 5: Y-Junction: Fluent Total Pressure plot
Figure 6: Y-Junction: Fluent Velocity vector section plot

Figure 7: Y-Junction: STAR-CCM+ Velocity vector section plot
Figure 8: Y-Junction: OpenFOAM Velocity contour section plot
11 Spring Mass System

11.1 Problem Description

![Double mass-spring system](image)

Figure 1: Double mass-spring system

Topics of this Tutorial
- Iterative Coupling
- Coupling with non-matching time steps
- Verification model

Simulation Codes
- Model A: SimulatorA (simple C executable for spring mass example), MATLAB
- Model B: Abaqus, MATLAB, MSC Adams, SIMPACK, SimulatorB (simple C executable for spring mass example)

11.2 Model Preparation

This tutorial uses a quite simple example – a dual mass-spring system – to show the possibilities MpCCI offers for iterative coupling and non-matching time step sizes.

The files which you need for the simulation are included in the MpCCI distribution. Create a new directory and copy the subdirectories from 

```
<MpCCI home>/tutorial/Spring
```

correspond to the simulation codes you want to use.

11.2.1 Model Description

Initially, a one mass system, with the analytical solution \( x(t) = -\frac{gm}{k}(1 - \cos(\sqrt{\frac{k}{m}}t)) \) for the position, is partitioned by splitting the mass into two parts. The applied-force coupling in this example is of the type force/displacement coupling, where a subsystem provides kinematic coupling data (position, velocity, acceleration) while the other system provides a force.

The sort of partitioning and the coupling type have an effect on the stability of the simulation. Partitioning by applied-force coupling will guarantee the null stability, i.e. numerical methods applied on these systems converge with infinitesimal coupling step size.

**Figure 1** shows two spring-mass systems tied together and initially at rest. A gravity load \( g = 10\text{m/s}^2 \) is imposed at \( t = 0 \). The displacements are constrained, resulting in both masses moving together. The constraint force between the systems is defined as \( \lambda \). This results in the following equations for the double mass-spring system:
m_a \ddot{x}_a + k_a x_a = -m_a g - \lambda \\
m_b \ddot{x}_b + k_b x_b = -m_b g - \lambda \\
\dot{x}_a = \dot{x}_b \\
at t = 0 : \quad x_a = x_b = 0 \quad \text{and} \quad \dot{x}_a = \dot{x}_b = 0 

For the mass and the stiffness of the spring-mass systems “stiffness factors” $\alpha$ and $\beta$ are introduced. These factors define the relative distribution of the single system mass and stiffness values to the total system:

\[
\begin{align*}
  k_a &= \alpha k \\
  k_b &= (1 - \alpha) k \\
  m_a &= \beta m \\
  m_b &= (1 - \beta) m 
\end{align*}
\]

Using these factors the previous equations can be written as:

\[
\begin{align*}
  \ddot{x}_a + \frac{\alpha}{\beta} \omega^2 x_a &= -g - \frac{\lambda}{\beta m} \\
  \ddot{x}_b + \frac{1 - \alpha}{1 - \beta} \omega^2 x_b &= -g - \frac{\lambda}{(1 - \beta) m} 
\end{align*}
\]

The two equations will be solved separately by a (second order) time integration scheme implemented in a simple C program. The exchange of the quantity information between the two codes will be handled by MpCCI.

SimulatorA – solving the first equation – sends the computed force to SimulatorB, which solves the second equation and then sends the displacement to SimulatorA.

Additionally, MATLAB can be used to solve this simple problem monolithically. Simply execute "runSimA-SimB.m" to simulate this example monolithically in MATLAB.

This file can be found in "mpcci/tutorial/Spring/Matlab/monolithic".

A co-simulation setup with MpCCI is described below.

Model A can be realized with SimulatorA or MATLAB while model B besides SimulatorB also can be realized with a simple Abaqus simulation consisting of one mass and a spring or by a MATLAB, MSC Adams or SIMPACK model.

### 11.2.2 Model A

**MATLAB**

The MATLAB files for Model A are located in "MATLAB/SimA/". Several files for different coupling types and handling of quantity Acceleration are provided:

- "runSimA_explicit.m" for an explicit coupling with computation of acceleration quantity.
- "runSimA_implicit.m" for an iterative coupling with computation of acceleration quantity.
- "runSimA_importAcceleration_explicit.m" for an explicit coupling with acceleration as coupled quantity which will be exchanged.
- "runSimA_importAcceleration_implicit.m" for an iterative coupling with acceleration as coupled quantity which will be exchanged.
The MATLAB codes for Model A do basically the same as the C source code for the SimulatorA. They compute the acceleration and then solve for the interface force $\lambda_a$.

If the acceleration is a coupled quantity, the computation of the acceleration is deactivated by resetting the `calculatedACC` to zero in the m-file:

```matlab
% Flag for using acceleration information from coupled system
% set to 0 to use the coupled value if this one is exchanged
calculatedACC = 0;
```

Please use one of the following m-files with the preset variable value in this case: "runSimA_importAcceleration_explicit.m" or "runSimA_importAcceleration_implicit.m" depending on the coupling scheme.

At the beginning of the m-files MpCCI tags for the co-simulation setup have to be written as a MATLAB annotation `%` with a leading `$mpcci`-declaration. In this case the coupling point (also called coupling variable) `cpoint` is declared in the definition section of the m-file codes as follows:

```matlab
% Definition of the co-simulation:
% $mpcci time t
% $mpcci dt dt
% $mpcci iter i
% $mpcci conv cConv
% $mpcci cpoint disp1,lambda,acc1
%
```

In this model three possible coupling variables have been declared: `disp1,lambda,acc1`.

Each coupling variable is associated with a physical quantity. The variable `acc1` is declared when the code imports the acceleration quantity.

### SimulatorA

The executable for the SimulatorA is part of the MpCCI distribution (`"<MPCCI_HOME>/codes/SimulatorA/bin/<version>/<MPCCI_ARCH>/Mpcci_SimulatorA.exe"`).

SimulatorA receives the displacements $x_a$ (e.g. from SimulatorB), computes the acceleration and then solves for the interface force $\lambda_a$:

$$\ddot{x}_a + \frac{\alpha}{\beta}\omega^2 x_a = -g - \frac{\lambda_a}{\beta m}$$

The interface force $\lambda_a$ is then exported.

### 11.2.3 Model B

#### Abaqus

The directory "Spring/Abaqus" contains an Abaqus model for the part modeled by SimulatorB: it is a very simple model consisting of a spring and an attached mass.

The coupled point in Abaqus is called NQA. Additional instances of this point have been created, NQA_DUP and NQA_DUP2, for coupling with a code like MATLAB which defines coupling variables (i.e. signals) for each quantity.
MSC Adams

In MSC Adams the files "Adams/ModelB.adm" and "Adams/sim_command.acf" represent the part B of the spring mass system. It computes the position of the coupled mass $x_b$ and the acceleration $\ddot{x}_b$. The model has been created using mm as length unit.

No additional preparation steps are required.

The MSC Adams model consists of a mass and a spring with one degree of freedom in the x-direction. The gravity is also defined in the x-direction.

The solver setting uses an adaptive time step model which provides some limitations for a scenario with non-matching time step sizes coupling.

MATLAB

The MATLAB files for Model B are located in "MATLAB/SimB/".

To get the displacements of the submodel B an ODE is solved. This is done by Newmark time integration method (as in the C source code for SimulatorB as described before). Select one of the m-files to choose the solver method and coupling type to use:

- "runSimB_newmark_explicit.m" for an integration by Newmark’s method and explicit coupling.
- "runSimB_newmark_implicit.m" for an integration by Newmark’s method and iterative coupling.

SIMPACK

The file "ModelB.spck" defines the part B of the spring mass system. SIMPACK computes the position of the mass $x_b$.

No additional preparation steps are required. The model consists of a mass and a spring with one degree of freedom in the x-direction. The gravity is also defined in the x-direction.

For SIMPACK iterative coupling is not supported.

SimulatorB

The executable for the SimulatorB is part of the MpCCI distribution ("<MPCCI_HOME>/codes/SimulatorB/bin/<version>/Mpcci_SimulatorB.exe"). SimulatorB receives the interface force $\lambda_b$ (e.g. from SimulatorA) and then uses the Newmark time integration method to compute the acceleration, velocity and displacements:

$$\ddot{x}_b + \frac{1 - \alpha}{1 - \beta}\dot{x}_b = -g - \frac{\lambda_b}{(1 - \beta)m}$$

The displacements are exported to its partner code.

11.3 Setting Up the Coupled Simulation with MpCCI GUI

See ▶ IV-2 Setting up a Coupled Simulation ◀ and ▶ V-4 Graphical User Interface ◀ for a detailed description on how to use the MpCCI GUI. Following are the substantive rules to be set in the MpCCI GUI in order to run this tutorial.

At first start the MpCCI GUI by running the command mpcci gui.
11.3.1 Models Step

In the Models step select and configure the codes to be coupled (cf. IV-2.4 Models Step – Choosing Codes and Model Files). Further information on the offered code parameters in the Models step can be looked up in the respective code section of the Codes Manual.

1. Choose your code for Model A as first code to couple:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select MATLAB.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see VI-10 MATLAB).</td>
</tr>
</tbody>
</table>
| MATLAB file     | Select the MATLAB file from directory "Matlab/SimA/" depending on the used coupling type and whether the quantity Acceleration will be coupled or computed:  

- "runSimA_implicit.m" for iterative coupling and computed acceleration.
- "runSimA_explicit.m" for explicit coupling and computed acceleration.
- "runSimA_importAcceleration_implicit.m" for iterative coupling and coupled acceleration.
- "runSimA_importAcceleration_explicit.m" for explicit coupling and coupled acceleration.  

If MATLAB is coupled with MSC Adams select the file with the coupled Acceleration quantity. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly. Solution type is displayed in this newly added option as a result of the scan process. |

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimulatorA file</td>
<td>Select the SimulatorA file &quot;SimulatorA/model.empty&quot;.</td>
</tr>
</tbody>
</table>

This file is empty. The information is generated from the "Scanner" and all settings can be handled in the Go step. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly. Solution type is displayed in this newly added option as a result of the scan process. |

2. Choose your code for Model B as second code to couple:
### Abaqus

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select Abaqus</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see ▶ VI-2 Abaqus ◄).</td>
</tr>
<tr>
<td>Input file</td>
<td>Select input file &quot;Abaqus/modelB.inp&quot;. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Transient is displayed in this newly added option as a result of the scan process.</td>
</tr>
<tr>
<td>Unit system</td>
<td>Select the SI unit system (which is the standard).</td>
</tr>
</tbody>
</table>

### MATLAB

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select MATLAB.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see ▶ VI-10 MATLAB ◄).</td>
</tr>
</tbody>
</table>
| MATLAB file    | Select the MATLAB file from directory "Matlab/SimB/" depending on the used solver and coupling type:  
                   For integration by Newmark’s method  
                   "runSimB_newmark_implicit.m" for iterative coupling.  
                   "runSimB_newmark_explicit.m" for explicit coupling.  
                   The file will be scanned immediately and a green check mark should appear which means everything is set up correctly. |
| Solution type  | Transient is displayed in this newly added option as a result of the scan process. |

### MSC Adams

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select MSC Adams</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see ▶ VI-11 MSC Adams ◄).</td>
</tr>
</tbody>
</table>
| MSC Adams file | Select MSC Adams file "Adams/sim_command.acf".  
                   The file will be scanned immediately and a green check mark should appear which means everything is set up correctly. |
| Solution type  | Undefined is displayed in this newly added option as a result of the scan process. |
| Unit system    | Select the variable unit system (which is the default). |
| Grid length unit | Select mm as unit for the grid length (which is the default). |
**SIMPACK**

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select SIMPACK</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see § VI-15 SIMPACK ◁).</td>
</tr>
<tr>
<td>SIMPACK file</td>
<td>Select SIMPACK file &quot;Simpack/ModelB.spck&quot;. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Transient is displayed in this newly added option as a result of the scan process.</td>
</tr>
<tr>
<td>Unit system</td>
<td>Select the variable unit system (which is the default).</td>
</tr>
<tr>
<td>Grid length unit</td>
<td>Select m as unit for the grid length (which is the default).</td>
</tr>
</tbody>
</table>

**SimulatorB**

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select SimulatorB.</td>
</tr>
<tr>
<td>Release</td>
<td>Set to latest.</td>
</tr>
<tr>
<td>SimulatorB file</td>
<td>Select the SimulatorB file &quot;SimulatorB/model.empty&quot;. This file is empty. The information is generated from the &quot;Scanner&quot; and all settings can be handled in the Go step. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Transient is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>

Proceed to the Algorithm step by pressing [Algorithm].

### 11.3.2 Algorithm Step

In the Algorithm step define the coupling algorithm (cf. ▶ IV-2.5 Algorithm Step – Defining the Coupling Algorithm ◁).

The settings should be as follows. If no action is mentioned, keep the default option.

1. For the Common Basics panel:

   **Iterative Coupling:**
### 11.3 Setting Up the Coupled Simulation with MpCCI GUI

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Coupling analysis</td>
<td></td>
</tr>
<tr>
<td>Define the coupling scheme</td>
<td>Set to Implicit.</td>
</tr>
<tr>
<td>▼ Solver settings</td>
<td></td>
</tr>
<tr>
<td>Type of solver step size</td>
<td>Set to Constant.</td>
</tr>
<tr>
<td>Constant solver step size</td>
<td>Set to 2.655e-3 s.</td>
</tr>
<tr>
<td>▼ Coupling algorithm</td>
<td></td>
</tr>
<tr>
<td>Algorithm type</td>
<td>Set i.e. to Parallel. Depending on this setting a Jacobi or Gauss-Seidel implicit coupling is applied (cf. §V-3.4.5.1 Implicit Coupling).</td>
</tr>
<tr>
<td>▼ Coupling duration</td>
<td></td>
</tr>
<tr>
<td>Maximum number of coupling step iterations</td>
<td>Set to 10.</td>
</tr>
<tr>
<td>Set total coupling time</td>
<td>Check this option.</td>
</tr>
<tr>
<td>Total coupling time</td>
<td>Set to 0.25 s.</td>
</tr>
</tbody>
</table>
Explicit Coupling:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Coupling analysis</td>
<td></td>
</tr>
<tr>
<td>Define the coupling scheme</td>
<td>Set to Explicit.</td>
</tr>
<tr>
<td>▼ Coupling steps</td>
<td></td>
</tr>
<tr>
<td>Type of coupling step size</td>
<td>Set Defined by each code.</td>
</tr>
<tr>
<td>Order of interpolation</td>
<td>Select one of Higher order, First order or Zero order. If coupling with non-matching time step sizes is used, different interpolation methods may lead to different results (cf. 11.5.2 Non-Matching Time Step Sizes).</td>
</tr>
<tr>
<td>▼ Coupling algorithm</td>
<td></td>
</tr>
<tr>
<td>Algorithm type</td>
<td>Set to Parallel. Serial coupling with non-matching time step sizes should be avoided (cf. V-3.4.5.3 Coupling with Non-Matching Time Steps).</td>
</tr>
<tr>
<td>▼ Coupling duration</td>
<td></td>
</tr>
<tr>
<td>Set total coupling time</td>
<td>Check this option.</td>
</tr>
<tr>
<td>Total coupling time</td>
<td>Set to 0.25 s.</td>
</tr>
</tbody>
</table>

2. For your Model A code:

Iterative Coupling:

MATLAB
No code specific options need to be set.

SimulatorA
No code specific options need to be set.

Explicit Coupling:

MATLAB
No code specific options need to be set.

SimulatorA
No code specific options need to be set.

3. For your Model B code:
Iterative Coupling:

**Abaqus**

No code specific options need to be set.

**MATLAB**

No code specific options need to be set.

⚠️ The MATLAB file selected in the Models step must fit to implicit coupling and handling of acceleration quantity.

**MSC Adams**

No code specific options need to be set.

**SIMPACK**

No code specific options need to be set.

**SimulatorB**

No code specific options need to be set.

... Explicit Coupling:

**Abaqus**

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Coupling steps</td>
<td></td>
</tr>
<tr>
<td>Type of coupling step size</td>
<td>Set to Constant.</td>
</tr>
<tr>
<td>Constant coupling step size</td>
<td>Set to 2.655e-03. This results in a coupling with matching time steps. Alternatively different time steps can be used for each code (cf. 11.5.2 Non-Matching Time Step Sizes). You can either enter another fixed coupling time step or let Abaqus use its own adaptive time increment values by deactivating Constant coupling time step box.</td>
</tr>
</tbody>
</table>

**MATLAB**

No code specific options need to be set.

⚠️ The MATLAB file selected in the Models step must fit to explicit coupling and handling of acceleration quantity.
### MSC Adams

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Analysis</td>
<td></td>
</tr>
<tr>
<td>Analysis type</td>
<td>Select Transient.</td>
</tr>
<tr>
<td>▼ Solver settings</td>
<td></td>
</tr>
<tr>
<td>Use semi-implicit interpolation</td>
<td>Check and access further options (see ▶ VI-11.2.8.1 Semi-implicit Mode for details).</td>
</tr>
<tr>
<td>Communication delay of the co-simulation</td>
<td>Set to 1 because we have explicit coupling.</td>
</tr>
<tr>
<td>Synchronized history buffer update</td>
<td>Check and access further options.</td>
</tr>
<tr>
<td>Time step for buffer update</td>
<td>Set to 2.655e-03.</td>
</tr>
<tr>
<td>Use constant mass</td>
<td>Check and access further options.</td>
</tr>
<tr>
<td>Value of the mass coefficient</td>
<td>Set to 0.25. This value corresponds to the mass of 2.5 kg used for the simulation multiplied by $\beta$ and is the mass on the Model B side.</td>
</tr>
<tr>
<td>Provide partials to MSC Adams</td>
<td>Check and access further options (see ▶ VI-11.2.8.2 Partial Derivatives in MSC Adams for details).</td>
</tr>
<tr>
<td>Minimum value for derivatives</td>
<td>Set to -100e6.</td>
</tr>
</tbody>
</table>

See ▶ VI-11.2.7 Algorithm Step for details on the settings in the MSC Adams Algorithm step.

### SIMPACK

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Solver settings</td>
<td></td>
</tr>
<tr>
<td>User library configuration</td>
<td>Check and access further options.</td>
</tr>
<tr>
<td>Use default uforce20</td>
<td>Set selected.</td>
</tr>
<tr>
<td>SIMPACK runtime configuration</td>
<td>Check and access further options.</td>
</tr>
<tr>
<td>Provide partials</td>
<td>Check to provide the partial derivatives of the received quantities with respect to the sent quantities (see ▶ VI-15.2.5.1 Semi-implicit Mode and Partial Derivatives for details).</td>
</tr>
<tr>
<td>Communication delay of the co-simulation</td>
<td>Set to 1 because we have explicit coupling.</td>
</tr>
<tr>
<td>Synchronized history buffer update</td>
<td>Check and access further options.</td>
</tr>
<tr>
<td>Time step for approximation update</td>
<td>Set to the step size of the co-simulation partner, e.g. 2.655e-3.</td>
</tr>
</tbody>
</table>
11.3 Setting Up the Coupled Simulation with MpCCI GUI

**SimulatorB**

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solver settings</td>
<td>Set to e.g. 2.655e-3. Different time steps can be used for the simulators (cf. ∇11.5.2 Non-Matching Time Step Sizes ◄).</td>
</tr>
</tbody>
</table>

Press the Regions button at the bottom of the window to get to the Regions step.

### 11.3.3 Regions Step

In the Regions step build the coupling regions, define quantities to be exchanged and assign the defined quantities to the built regions (cf. ∇IV-2.6 Regions Step – Defining Coupling Regions and Quantities ◄).

1. At first select the Mesh tab to get access to the mesh based element components.

2. Select Build Regions tab and here the Setup tab. Now choose the components to be coupled. In this example the coupling region corresponds to points. Therefore select the Mesh tab and where appropriate the Point tab ( *) in the components list where appropriate.

The build regions setup differs depending on the used simulation codes. Therefore the setup is split into the parts “Coupling with MATLAB ” and “Coupling without MATLAB ”. Lookup your setup in the appropriate part.

**Coupling without MATLAB** Build the coupling regions. Therefore select the coupling components by dragging them into the Coupled boxes.

Lookup your codes and their components to be coupled in the following table:

<table>
<thead>
<tr>
<th>For region Region_1</th>
<th>Select coupling component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abaqus</td>
<td>NQA</td>
</tr>
<tr>
<td>MSC Adams</td>
<td>GFORCE-1::GFORCE_1</td>
</tr>
<tr>
<td>SIMPACK</td>
<td>$F_{Spring}</td>
</tr>
<tr>
<td>SimulatorA</td>
<td>BearingPoint</td>
</tr>
<tr>
<td>SimulatorB</td>
<td></td>
</tr>
</tbody>
</table>

**Coupling with MATLAB** Because MATLAB can couple only one quantity per point, the MATLAB model consists of three separate points, one for each quantity. In order to be able to create a region for each coupled quantity it is necessary to copy a part from the partner code where the partner code provides only one component for coupling.

- Therefore lookup your partner code in the list below and copy their components twice as described in ∇V-4.6.2.1 Copying Components ◄.

<table>
<thead>
<tr>
<th>Partner code</th>
<th>Select component for copying</th>
<th>Copies</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSC Adams</td>
<td>GFORCE-1::GFORCE_1</td>
<td>GFORCE-1::GFORCE_1:1</td>
</tr>
<tr>
<td>SIMPACK</td>
<td>$F_{Spring}$</td>
<td>$F_{Spring}:1$</td>
</tr>
<tr>
<td>SimulatorA</td>
<td>BearingPoint</td>
<td>BearingPoint:1</td>
</tr>
<tr>
<td>SimulatorB</td>
<td></td>
<td>BearingPoint:2</td>
</tr>
</tbody>
</table>

Because SIMPACK doesn’t support to couple quantity Acceleration only one copied component is needed for coupling with MATLAB.
Because the copy function is not available for Abaqus, the Abaqus model already provides the necessary parts (NQA, NQA_DUP, NQA_DUP2).

MATLAB may be coupled with itself and needs no extra component preparation for this case.

- Build the coupling regions. Therefore select the coupling components by dragging them into the Coupled boxes.

For this lookup your codes and their components to be coupled in the following tables:

- Build the first region.

<table>
<thead>
<tr>
<th>For code</th>
<th>Select coupling component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abaqus</td>
<td>NQA</td>
</tr>
<tr>
<td>MATLAB</td>
<td>lambda</td>
</tr>
<tr>
<td>MSC Adams</td>
<td>GFORCE-1::GFORCE_1</td>
</tr>
<tr>
<td>SIMPACK</td>
<td>$F_{Spring}</td>
</tr>
<tr>
<td>SimulatorA</td>
<td>BearingPoint</td>
</tr>
<tr>
<td>SimulatorB</td>
<td></td>
</tr>
</tbody>
</table>

- Click on button **Add** to add new region **Region_2**.

<table>
<thead>
<tr>
<th>For code</th>
<th>Select coupling component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abaqus</td>
<td>NQA_DUP</td>
</tr>
<tr>
<td>MATLAB</td>
<td>disp1</td>
</tr>
<tr>
<td>MSC Adams</td>
<td>GFORCE-1::GFORCE_1::1</td>
</tr>
<tr>
<td>SIMPACK</td>
<td>$F_{Spring}::1</td>
</tr>
<tr>
<td>SimulatorA</td>
<td>BearingPoint::1</td>
</tr>
<tr>
<td>SimulatorB</td>
<td></td>
</tr>
</tbody>
</table>

- If MATLAB is coupled with MSC Adams you need to create a third coupled region for exchanging the Acceleration quantity.

For the other codes, except SIMPACK which doesn’t support the Acceleration quantity, the exchange of Acceleration is optional. Doing so you may observe an improvement of the force progression on the monitor plot.

For adding a third region **Region_3** click on button **Add**.

<table>
<thead>
<tr>
<th>For code</th>
<th>Select coupling component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abaqus</td>
<td>NQA_DUP2</td>
</tr>
<tr>
<td>MATLAB</td>
<td>acc1</td>
</tr>
<tr>
<td>MSC Adams</td>
<td>GFORCE-1::GFORCE_1::2</td>
</tr>
<tr>
<td>SimulatorA</td>
<td>BearingPoint::2</td>
</tr>
<tr>
<td>SimulatorB</td>
<td></td>
</tr>
</tbody>
</table>

- After all regions have been built, assign them a more understandable name as follows:
  * Select all regions.
  * Point with the mouse on the regions and do a right click.
  * From the popup menu choose Rename automatically and select MATLAB from the code list. The components from MATLAB will be used to rename the regions which leads to regions lambda, disp1 and acc1.
3. Select Define Quantity Sets tab. As currently no quantities are assigned to “EmptyQuantitySet” (created by default), a warning symbol is displayed in front of it. Now choose the quantity to be exchanged.

<table>
<thead>
<tr>
<th>For quantity set</th>
<th>Select quantity</th>
<th>Set sender</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmptyQuantitySet</td>
<td>Force</td>
<td>Code for Model A</td>
</tr>
</tbody>
</table>

The resulting name for the quantity set will be \textit{Force}<-\textit{Model A code}.

**For an iterative coupling** it is advisable to define a convergence tolerance for the inner iterations. Therefore add a convergence check operator (cf. \textit{V-4.6.7.4 Applying Operators to Quantities of Mesh Based Components}) to the \textit{Force} quantity. For this example a value of 0.01 is small enough to produce an accurate solution.

Configure the operator as follows:

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Operator</th>
<th>Operator option</th>
<th>Operator action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Force</td>
<td>ConvergenceCheck</td>
<td>Tolerance value</td>
<td>Set to 0.01.</td>
</tr>
</tbody>
</table>

The convergence check operator is a pre processor and will be applied to the sender of the quantity.

**For explicit coupling** no more quantity settings need to be done for this quantity set.

4. Click on button \textit{Add} to add new quantity set \textit{EmptyQuantitySet}.

<table>
<thead>
<tr>
<th>For quantity set</th>
<th>Select quantity</th>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmptyQuantitySet</td>
<td>PointPosition</td>
<td>Set sender</td>
<td>Select code for Model B.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code specific settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSC Adams</td>
</tr>
</tbody>
</table>

| Unit | Set to mm. |

The resulting name for the quantity set will be \textit{PointPosition}<-\textit{Model B code}.

5. For coupling with MSC Adams you need to create a third quantity set for exchanging the \textit{Acceleration} quantity. For the other codes, but SIMPACK which doesn’t support the \textit{Acceleration} quantity, the exchange of \textit{Acceleration} is optional. Doing so you may observe an improvement of the force progression on the monitor plot.

In order to couple acceleration click on button \textit{Add} to add new quantity set \textit{EmptyQuantitySet}.

<table>
<thead>
<tr>
<th>For quantity set</th>
<th>Select quantity</th>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmptyQuantitySet</td>
<td>Acceleration</td>
<td>Set sender</td>
<td>Select code for Model B.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code specific settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSC Adams</td>
</tr>
</tbody>
</table>

| Unit | Set to mm/s². |

The resulting name for the quantity set will be \textit{Acceleration}<-\textit{Model B code}.

6. Select Assign Quantity Sets tab and assign the defined quantity sets to the built regions.

**For coupling without MATLAB**
For selected region  | Check quantity sets
--- | ---
Region_1  | Force<--Model A code
| PointPosition<--Model B code
| If coupling acceleration
| Acceleration<--Model B code

For coupling with MATLAB

<table>
<thead>
<tr>
<th>For selected regions</th>
<th>Check quantity sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>lambda</td>
<td>Force&lt;--Model A code</td>
</tr>
<tr>
<td>disp1</td>
<td>PointPosition&lt;--Model B code</td>
</tr>
<tr>
<td></td>
<td>If coupling acceleration</td>
</tr>
<tr>
<td>accl</td>
<td>Acceleration&lt;--Model B code</td>
</tr>
</tbody>
</table>

No changes are required in the Monitors step. You may proceed to the Settings step by pressing **Settings**.

### 11.3.4 Settings Step

If iterative coupling is used and the relative difference between the inner iterations should be monitored during the coupled simulation, the monitor option `Inorm` has to be selected:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor.Inorm</td>
<td>Set selected.</td>
</tr>
</tbody>
</table>

No further changes are required in the Settings step. Proceed to the Go step by pressing **Go**.

### 11.3.5 Go Step

No changes are required in the Go step. Now you may run the computation.

### 11.4 Running the Computation

#### 11.4.1 Starting the Simulation

Save the MpCCI project file with name "springmass.csp" over the MpCCI GUI menu **File**→**Save Project As**.

Press the **Start** button in the Go step and the simulation codes should start.

No additional actions are required.

### 11.5 Discussion of Results

#### 11.5.1 Iterative Coupling Compared to Explicit Coupling

Figure 2 shows the different results for an iterative and an explicit coupling algorithm. While the amplitude of the oscillating point displacements remains constant for the iterative coupling scheme, the explicit solution shows a decreasing amplitude.
For other parameter values the explicit coupling algorithm also produces the stable solution with a constant amplitude. Results of explicit coupling for small values $\beta$ and small time steps are fine. Only for larger values of $\beta$ and big time step values the instability shown in Figure 2 can be observed.

Iterative coupling results in a stable solution even for large values for $\beta$ and larger time steps. For the results in Figure 2 a time step value of 2.655E-3 and default values for all other parameters were used.

Figure 3 (left) shows the displacement of the coupled point for the iterative coupling scheme in more detail. At the end of each time step the displacement is plotted for every inner iteration, distributed over the time step. The relatively big changes during the first iterations can be seen; then – depending on the stopping criteria used – the solution remains more or less constant and the next time step is started.

If the adaptive stopping criterion is set to a smaller value the displacement plot will change: the displacement value will reach the correct values earlier in the time step, and then remain more or less constant for the rest of the inner iterations in the time step. The force plots show the same behavior.

On the right hand side of Figure 3 the relative variation for the inner iterations – which is used to judge the convergence of the iterative scheme – is plotted. At the start of each new time step there is a peak, which gets smaller during the following inner iterations. This plot helps to adjust the convergence tolerance to a sensible and problem adapted value.
11.5.2 Non-Matching Time Step Sizes

Using explicit coupling algorithms this example can be used to demonstrate the flexibility of MpCCI when dealing with non-matching time step sizes (cf. V-3.4.5.3 Coupling with Non-Matching Time Steps). If the codes each use their own time step size, MpCCI automatically interpolates the coupled quantity values to match the respective time steps as closely as possible.

Depending on the available quantity information from previous time steps and the user’s interpolation method selection in the Algorithm step, a constant, linear or higher order interpolation (using up to four different time steps for the interpolation) is used.

Because of the oscillatory displacement, this example is very suitable for showing the differences between the interpolation methods.

Using different time step sizes and linear interpolation the oscillatory displacement cannot be recovered resulting in a wrong solution of the coupled simulation. For the plots shown in Figure 4 SimulatorA used a time step of 2.3E-3 and SimulatorB used the default time step value 2.655E-3 setting a parallel algorithm without initialization.

If both time steps are small enough and quite close to each other, a linear interpolation is able to reproduce the oscillations.

![Figure 4: Non-matching time steps (2.3E-3 for SimulatorA and 2.655E-3 for SimulatorB) and different interpolation methods for the spring mass example: displacement for SimulatorA using linear interpolation (left) and higher order interpolation (right).](image-url)
12 Periodic Rotating Fan Model

12.1 Problem Description

Topics of this Tutorial

- Coupling of periodic models
- Ramping
- Fluid-Structure Interaction (FSI) for steady state analysis
- 2D-model

Simulation Codes

- Solid Mechanics: Abaqus
- Fluid Mechanics: FLUENT

Description

In this tutorial we show the concept of periodic models in MpCCI through the example of a steady fluid structure interaction of a two dimensional rotating fan model in a circular channel. Here the periodic section in the CFD computation is geometrically different to the periodic section in the structural mechanics simulation, cf. Figure 1.

The example demonstrates how to use periodic models in an MpCCI coupling. The user can define different section shapes of the coupled models, even use a full model. The only limitation is that the virtual full models coincide (without respect to units and slight geometrical differences), cf. Figure 2. This feature is available for all codes supported by MpCCI.
12.2 Model Preparation

This simulation couples a structural mechanics model with a fluid mechanics model, both considered as 2D. The files which you need for the simulation are included in the MpCCI distribution. Create a new directory and copy the subdirectories from 

```
<MpCCI_home>/tutorial/Periodic
```

The following model files provide enough simulation period in order to cover the configured coupled simulation. In that case the **Set maximum number of coupling steps** option is not needed to limit the coupling period as it will end at the end of the total simulation period set in the model files or provided in the user interface.

12.2.1 Fluid Model

The fluid domain is a periodic section of a 2D circular channel, where the outer circle is a stationary wall (radius 2 m). In the center lies a 1/4 cross which rotates (as a moving reference frame) with speed 100 rad/s around the z-axis lying in zero. The rotating and stationary fluid parts are connected using an interface (at radius 1.2 m), see Figure 3.

The fluid is considered as an ideal gas with a viscosity of $\mu = 1.7894 \cdot 10^{-5}$ kg m s$^{-1}$. A steady state solution is to be computed.

**FLUENT**

The mesh was created by ANSA and consists of first order tria elements. The surface of the flexible structure is defined as a separate boundary named “turbo”. The periodic boundary conditions were defined at the section’s boundaries. Since the mesh will deform during the coupling “Dynamic Mesh” is activated where the periodic boundaries are kept stationary. **FLUENT** total number of iterations will be set to 5000.
12.2.2 Solid Model

The solid part corresponds to a quarter of the cross, where the position of the periodic boundaries are selected in a different way than those chosen in the fluid model, see Figure 4. The thickness of the cross arms is 0.1 m, their length is 0.5 m. The model is fixed in the center of the cross and a centrifugal load of 100 rad/s is applied. Here an artificial linear elastic material has been used with density $\rho = 78$ kg/m$^3$, elastic modulus 0.5 MPa and Poisson’s ratio $\nu = 0.3$. 

Figure 3: Periodic fluid mesh

Figure 4: Periodic solid mesh
Abaqus

The mesh was created by ANSA and consists of first order quadrilateral elements (CPE4 element type). The surface of the flexible structure is defined as a 2D-SURFACE named “blade-wall”. The periodic boundary conditions were defined at the section’s boundaries using a cyclic symmetry TIE contact. The total simulation time has been set to 0.05 s.

12.3 Setting Up the Coupled Simulation with MpCCI GUI

See ▶ IV-2 Setting up a Coupled Simulation ◀ and ▶ V-4 Graphical User Interface ◀ for a detailed description on how to use the MpCCI GUI. Following are the substantive rules to be set in the MpCCI GUI in order to run this tutorial.

At first start the MpCCI GUI by running the command `mpcci gui`.

12.3.1 Models Step

In the Models step select and configure the codes to be coupled (cf. ▶ IV-2.4 Models Step – Choosing Codes and Model Files ◀). Further information on the offered code parameters in the Models step can be looked up in the respective code section of the Codes Manual.

1. Choose your fluid mechanics code as first code to couple:

   **FLUENT**

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select FLUENT.</td>
</tr>
<tr>
<td>Version</td>
<td>The model is two-dimensional, thus select the FLUENT version 2ddp.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see ▶ VI-8 FLUENT ◀).</td>
</tr>
<tr>
<td>Case file</td>
<td>Select the case file &quot;FLUENT/fan_period1Arm.cas&quot;. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Steady state is displayed in this newly added option as a result of the scan process.</td>
</tr>
</tbody>
</table>

2. Choose your solid mechanics code as second code to couple:

   **Abaqus**

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code to couple</td>
<td>Select Abaqus.</td>
</tr>
<tr>
<td>Release</td>
<td>Should be set to latest or a version supported by MpCCI (see ▶ VI-2 Abaqus ◀).</td>
</tr>
<tr>
<td>Input file</td>
<td>Select input file &quot;ABAQUS/fan_period2halfArms.inp&quot;. The file will be scanned immediately and a green check mark should appear which means everything is set up correctly.</td>
</tr>
<tr>
<td>Solution type</td>
<td>Steady state is displayed in this newly added option as a result of the scan process.</td>
</tr>
<tr>
<td>Unit system</td>
<td>Select the SI unit system (which is the standard).</td>
</tr>
</tbody>
</table>

Proceed to the Algorithm step by pressing *Algorithm*.
12.3.2 Algorithm Step

In the Algorithm step define the coupling algorithm (cf. IV-2.5 Algorithm Step – Defining the Coupling Algorithm). The settings should be as follows. If no action is mentioned, keep the default option.

1. For the Common Basics panel:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Coupling analysis</td>
<td>Define the coupling scheme This option is automatically set to Explicit.</td>
</tr>
<tr>
<td>▼ Coupling algorithm</td>
<td>Algorithm type Set to Serial.</td>
</tr>
<tr>
<td>Leading code</td>
<td>Set to solid mechanics code.</td>
</tr>
</tbody>
</table>

2. For the fluid mechanics code:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Coupling steps</td>
<td>Type of coupling step size Set to Constant.</td>
</tr>
<tr>
<td></td>
<td>Constant coupling step size Set to 100 for number of steps without coupling.</td>
</tr>
</tbody>
</table>

3. For your solid mechanics code:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Coupling steps</td>
<td>Type of coupling step size Set to Constant.</td>
</tr>
<tr>
<td></td>
<td>Constant coupling step size Set to 0.001 for one increment size for the steady state analysis.</td>
</tr>
</tbody>
</table>

Press the Regions button at the bottom of the window to get to the Regions step.

12.3.3 Regions Step

In the Regions step edit the components to define the periodic information, build the coupling regions, define quantities to be exchanged and assign the defined quantities to the built regions (cf. IV-2.6 Regions Step – Defining Coupling Regions and Quantities). The settings should be as follows. If no action is mentioned, keep the default option.

1. Select the Mesh tab to get access to the mesh based element components.

2. Select the Edit Components tab in order to define the periodicity of the components (cf. V-4.6.2.3 Periodic Components). A default periodicity object “Periodicity,1” is present on the right hand
side. Rename it as “quarter” by right clicking on it. Configure the periodicity as follows:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of periodicity</td>
<td>Set to 4 which is the number of periodic passages which build the virtual full model.</td>
</tr>
<tr>
<td>Define the cyclic</td>
<td>symmetry axis by the two following specifications.</td>
</tr>
<tr>
<td>Rotation-axis origin</td>
<td>Set to X=0., Y=0., Z=0.</td>
</tr>
<tr>
<td>Rotation-axis direction</td>
<td>Set to X=0., Y=0., Z=1.</td>
</tr>
</tbody>
</table>

Now the periodicity “quarter” can be assigned to the components on the left. In this example the coupling region corresponds to the surface of the flexible 2D structure. MpCCI treats this 1D region as a “Face”. Therefore switch to the Face tabs (▲) for each components list, select the periodic components and apply the periodicity.

For fluid mechanics code

<table>
<thead>
<tr>
<th>Select component</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLUENT</td>
</tr>
<tr>
<td>▲ turbo</td>
</tr>
</tbody>
</table>

For solid mechanics code

<table>
<thead>
<tr>
<th>Select component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abaqus</td>
</tr>
<tr>
<td>▲ BLADE-WALL</td>
</tr>
</tbody>
</table>

For fluid mechanics code

<table>
<thead>
<tr>
<th>Select coupling component</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLUENT</td>
</tr>
<tr>
<td>☢ turbo</td>
</tr>
</tbody>
</table>

For solid mechanics code

<table>
<thead>
<tr>
<th>Select coupling component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abaqus</td>
</tr>
<tr>
<td>☢ BLADE-WALL</td>
</tr>
</tbody>
</table>

Note the change of the symbol in front of the selected component names to ☢.

3. Now select Build Regions tab and here the Setup tab. Choose the components to be coupled from the Face tabs (▲) in the components list and select the coupling components by dragging them down into the Coupled box.

Lookup your codes in the following table and select the components to be coupled:

<table>
<thead>
<tr>
<th>For region Region_1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>For fluid mechanics code</td>
</tr>
<tr>
<td>Select coupling component</td>
</tr>
<tr>
<td>FLUENT</td>
</tr>
<tr>
<td>☢ turbo</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>For fluid mechanics code</td>
</tr>
<tr>
<td>Select coupling component</td>
</tr>
<tr>
<td>Abaqus</td>
</tr>
<tr>
<td>☢ BLADE-WALL</td>
</tr>
</tbody>
</table>

4. Select Define Quantity Sets tab. As currently no quantities are assigned to “EmptyQuantitySet” (created by default), a warning symbol is displayed in front of it. Now choose the quantities to be exchanged.

<table>
<thead>
<tr>
<th>For quantity set</th>
<th>Select quantities</th>
<th>Set sender</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmptyQuantitySet</td>
<td>NPosition</td>
<td>Solid mechanics code</td>
</tr>
<tr>
<td></td>
<td>RelWallForce</td>
<td>Fluid mechanics code</td>
</tr>
</tbody>
</table>

The resulting name for the quantity set will be NPosition←Solid mechanics code | RelWallForce← Fluid mechanics code.

5. Add the Ramping operator to the quantity NPosition (cf. ▷ V-4.6.7.4 Applying Operators to Quantities of Mesh Based Components) to get an accelerated convergence.
Thus mark \texttt{NPosition} by clicking on its name and configuring the listed operator as follows:

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Operator</th>
<th>Operator option</th>
<th>Operator action</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{NPosition}</td>
<td>Select Relaxation.</td>
<td>Relaxation method</td>
<td>Select Ramping.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Initial ramp value</td>
<td>Set to 0.1 (default).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ramp delta</td>
<td>Set to 0.1 (default).</td>
</tr>
</tbody>
</table>

The relaxation operator is a post processor which will be applied to the receiver of the quantity.

6. Select Assign Quantity Sets tab and assign the defined quantity set to the built region.

<table>
<thead>
<tr>
<th>For selected region</th>
<th>Check quantity set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Region_1</td>
<td>\texttt{NPosition&lt;~\textit{Solid mechanics code}</td>
</tr>
</tbody>
</table>

No changes are required in the Monitors step. You may proceed to the Settings step by pressing \texttt{Settings}.

### 12.3.4 Settings Step

The monitor is configured to replicate the periodic coupled components such that a full model can be visualized with its quantities. Moreover the full model is saved in the ccvx file to be reviewed in MpCCI Visualizer.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor.Periodic</td>
<td>Set selected to display the replicated periodic parts and quantities.</td>
</tr>
<tr>
<td>Output.Tracefile.Writers.CCVX.Periodic</td>
<td>Set selected to include the replicated periodic parts and quantities.</td>
</tr>
</tbody>
</table>

Proceed to the Go step by pressing \texttt{Go}.

### 12.3.5 Go Step

No changes are required in the Go step. Now you may run the computation.

### 12.4 Running the Computation

Save the MpCCI project file with name "\textit{periodic.csp}" over the MpCCI GUI menu \texttt{File$\rightarrow$Save Project As}. Press the \texttt{Start} button in the Go step and the simulation codes should start.

Some codes require additional actions:

**FLUENT**

1. During the simulation, FLUENT will display the residual, the pressure and the velocity distribution in the fluid domain. You can also see the deformation of the flexible structure. Here you need to change in the window layout to three windows (as shown in Figure 7).

2. The computation needs to be initialized by selecting \texttt{Solve$\rightarrow$Initialization} resp. \texttt{Solution$\rightarrow$Solution Initialization} and

3. Select \texttt{Solve$\rightarrow$MpCCI Run FSI} resp. \texttt{MpCCI$\rightarrow$MpCCI Run FSI}. Set the \texttt{Number of Iterations} to 5000 and press \texttt{Run} to start the simulation. With 100 subiterations there will be 50 couplings.

...
12.5 Discussion of Results

Since the parameter Periodic was enabled for the monitor in Settings step, the periodic parts are shown with their virtual replicates, cf. Figure 5.

In the following a comparison of the results of this coupling is presented and is compared to the corresponding coupling of the full models. The full meshes were created out of the periodic meshes, i.e. the number of nodes and elements is four times higher.

The coupling of the periodic models took, as expected, four times less computation time. The relative difference of the computed displacement of a node at the outer corner of the cross compared to the mean displacement of the corresponding nodes of the full model is 0.61%. The displacement results of both periodic and full models are shown in Figure 6.

Figure 7 and Figure 8 show the pressure and velocity magnitude in the fluid domain, computed on the periodic and the full coupled model. The differences in the results of the two approaches is founded in the not completely periodic behaviour of the solver on the full computational domain.

Use the following files for the evaluation of the results:

MpCCI

The MpCCI tracefile "mpccirun-0000.ccvx" from the tracefile folder "mpccirun_<TIMESTAMP>_ccvx" can be opened by the command mpcci visualize which one can trace the exchange process of the coupled surface.

Abaqus

The Abaqus result file name is equal to the defined Abaqus job name parameter followed by the suffix ".odb", e.g. "abaqus_run.odb".

FLUENT

The FLUENT result file is "fan_period1Arm.cas".
Figure 6: Comparison of coupled deformations on periodic and full *Abaqus* models

Figure 7: Pressure and velocity as coupling results on periodic *FLUENT* model with convergence history
Figure 8: Pressure and velocity as coupling results on full FLUENT model with convergence history
## VIII Programmers Guide – Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td><strong>MpCCI API</strong></td>
<td>6</td>
</tr>
<tr>
<td>2.1</td>
<td>Code Integration and Simulation Code Requirements</td>
<td>7</td>
</tr>
<tr>
<td>2.1.1</td>
<td>Data Exchange and Data Access</td>
<td>7</td>
</tr>
<tr>
<td>2.1.2</td>
<td>MpCCI Interface for Code Integration</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>Code Integration with the MpCCI API Kit</td>
<td>9</td>
</tr>
<tr>
<td>2.2.1</td>
<td>A Simple Example</td>
<td>9</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Step-by-Step Procedure for Code Integration</td>
<td>12</td>
</tr>
<tr>
<td>2.2.3</td>
<td>Code Coupling with the Example</td>
<td>20</td>
</tr>
<tr>
<td>2.3</td>
<td>Code Configuration Directory</td>
<td>22</td>
</tr>
<tr>
<td>2.4</td>
<td>MpCCI GUI Configuration File gui.xcf</td>
<td>23</td>
</tr>
<tr>
<td>2.4.1</td>
<td>Code Information: <code>&lt;CodeInfo&gt;</code></td>
<td>23</td>
</tr>
<tr>
<td>2.4.2</td>
<td>Codes Menu: <code>&lt;CodesMenuEntries&gt;</code></td>
<td>23</td>
</tr>
<tr>
<td>2.4.3</td>
<td>Models Step: <code>&lt;ModelsMenuEntries&gt;</code></td>
<td>24</td>
</tr>
<tr>
<td>2.4.4</td>
<td>Component Types: <code>&lt;ComponentTypeDimensions&gt;</code></td>
<td>25</td>
</tr>
<tr>
<td>2.4.5</td>
<td>List of Quantities: <code>&lt;SupportedQuantities&gt;</code></td>
<td>25</td>
</tr>
<tr>
<td>2.4.6</td>
<td>Algorithm Step: <code>&lt;CouplingMenuEntries&gt;</code></td>
<td>26</td>
</tr>
<tr>
<td>2.4.7</td>
<td>Go Step: <code>&lt;GoMenuEntries&gt;</code></td>
<td>27</td>
</tr>
<tr>
<td>2.4.8</td>
<td>Environments for Scanner, Checker, Starter, Stopper and Killer</td>
<td>29</td>
</tr>
<tr>
<td>2.4.9</td>
<td>General MpCCI GUI Elements</td>
<td>31</td>
</tr>
<tr>
<td>2.4.10</td>
<td>Testing gui.xcf</td>
<td>39</td>
</tr>
<tr>
<td>2.5</td>
<td><strong>Perl Scripts</strong></td>
<td>40</td>
</tr>
<tr>
<td>2.5.1</td>
<td>Using Information from gui.xcf in Scripts</td>
<td>40</td>
</tr>
<tr>
<td>2.5.2</td>
<td>Scanner.pm</td>
<td>40</td>
</tr>
<tr>
<td>2.5.3</td>
<td>Checker.pm</td>
<td>42</td>
</tr>
<tr>
<td>2.5.4</td>
<td>Starter.pm</td>
<td>42</td>
</tr>
<tr>
<td>2.5.5</td>
<td>Stopper.pm</td>
<td>43</td>
</tr>
<tr>
<td>2.5.6</td>
<td>Killer.pm</td>
<td>43</td>
</tr>
<tr>
<td>2.5.7</td>
<td>Info.pm</td>
<td>43</td>
</tr>
<tr>
<td>2.5.8</td>
<td>Subcmd.pm</td>
<td>44</td>
</tr>
<tr>
<td>2.5.9</td>
<td>Testing the Perl Scripts</td>
<td>44</td>
</tr>
<tr>
<td>2.6</td>
<td><strong>MpCCI Adapter Implementation</strong></td>
<td>45</td>
</tr>
<tr>
<td>2.6.1</td>
<td>How to Initialize the Code?</td>
<td>45</td>
</tr>
<tr>
<td>2.6.2</td>
<td>How to Define the Mesh?</td>
<td>46</td>
</tr>
<tr>
<td>2.6.3</td>
<td>How to Deal with Angular Coordinates?</td>
<td>47</td>
</tr>
<tr>
<td>2.6.4</td>
<td>How to Transfer Data?</td>
<td>48</td>
</tr>
<tr>
<td>2.6.5</td>
<td>How to Terminate the Coupling?</td>
<td>48</td>
</tr>
</tbody>
</table>
2.6.6 How to Notify a Remeshing? ........................................................................ 48
2.7 MpCCI Coupling Manager Functions ................................................................. 49
  2.7.1 Definition of Output Functions: umpcci_msg(functs) ............................... 50
  2.7.2 Definition of Output Prefix: umpcci_msg(prefix) ......................... 51
  2.7.3 Get Transfer Information: ampcce_tinfo_init ................................ 52
  2.7.4 Initialize the Code Information Structure .............................................. 52
  2.7.5 Connect and Initialize an MpCCI Server: mpcci_init .................... 53
  2.7.6 Configure the Code Adapter: ampcce_config ...................................... 54
  2.7.7 Definition of Part: smpcci_defp .............................................................. 55
  2.7.8 Delete a Part: smpcci_delp ................................................................... 57
  2.7.9 Definition of Nodes: smpcci_pmod ......................................................... 58
  2.7.10 Definition of Elements: smpcci_pels ..................................................... 59
  2.7.11 Definition of the Moving Reference Frame: smpcci_pmot .................. 61
  2.7.12 Definition of the Baffle Thickness: smpcci_pshf ................................ 62
  2.7.13 Data Exchange: ampcce_transfer ....................................................... 63
  2.7.14 Notifying the Remeshing: ampcce_remesh ......................................... 65
  2.7.15 End of Coupled Simulation: mpcci_quit and mpcci_stop ............... 66
2.8 MpCCI Driver Functions .................................................................................. 67
  2.8.1 Description Values .................................................................................. 70
  2.8.2 Driver Methods Called Before/After Some Action ........................... 71
  2.8.3 Driver Mesh Definition Methods ........................................................... 73
  2.8.4 Driver Data Exchange Methods ........................................................... 75
2.9 Data Structures and Predefined Macros ......................................................... 76
  2.9.1 Supported Element Types ..................................................................... 76
  2.9.2 Coordinates System Definition ............................................................... 85
  2.9.3 Mesh Dimension Definition ................................................................... 85
  2.9.4 Moving Reference Frame Definition ..................................................... 85
  2.9.5 Remesh Flag Information .................................................................... 86
  2.9.6 Transfer Information: MPCCI_TINFO .................................................. 86
  2.9.7 Code Specific Information: MPCCI_CINFO ......................................... 92
  2.9.8 Coupling Components ......................................................................... 94
  2.9.9 Quantities ............................................................................................. 95
  2.9.10 Loop Functions ................................................................................... 96
  2.9.11 Memory Management ...................................................................... 98
1 Introduction

This “Programmers Guide” addresses to engineers who have developed their own simulation code and plan to use it within the MpCCI coupling environment. The current MpCCI version allows one level of interfacing own codes with the coupling environment:

- the MpCCI Adapter level used since MpCCI 3 for commercial codes.

The MpCCI SDK interface level provides various basic functions to define coupling regions, control communication between the coupled codes and to handle other MpCCI parameters. There is no fixed protocol by using the MpCCI SDK when to communicate which data with which other components. There is no guarantee that the MpCCI SDK integration of a code A has a compatible communication protocol as that of any other code B.

To avoid such protocol incompatibilities between different code integrations MpCCI 3.0 has introduced the concept of code adapters. These adapters have internal mechanisms to control the current coupling state and actions of each integrated code. The Code Coupling Manager guarantees a consistent behavior and communication protocol for all adapted codes. Chapter 2 MpCCI API describes the integration of new simulation codes using a code adapter.

The current MpCCI 4 supports the code Adapter level integration. In midterm the codes using MpCCI SDK interface of MpCCI 3 should migrate their coupling interface with that standard coupling interface.
2 MpCCI API

MpCCI API represents the Adapter Programming Interface. This section describes how to establish MpCCI support for your code, i.e. the code then can be coupled with all other codes, which are already supported by MpCCI.

This section is organized as follows:

- General description of code integration. Read this to get an idea how code integration works.
  - 2.1 Code Integration and Simulation Code Requirements

- Description of the MpCCI API Kit, which contains template files and an example. A step-by-step procedure for code integration is given here.
  - 2.2 Code Integration with the MpCCI API Kit

- Reference for MpCCI GUI integration.
  - 2.3 Code Configuration Directory
  - 2.4 MpCCI GUI Configuration File gui.xcf
  - 2.5 Perl Scripts

- Reference for the code adapter.
  - 2.6 MpCCI Adapter Implementation
  - 2.7 MpCCI Coupling Manager Functions
  - 2.8 MpCCI Driver Functions
  - 2.9 Data Structures and Predefined Macros

Before starting the integration of a new code, you should contact the MpCCI support, mpcci-support@scai.fraunhofer.de, to obtain:

- The MpCCI API Kit, a set of template files and an example.
- A license for an adapter to your code.
2.1 Code Integration and Simulation Code Requirements

In order to understand code integration, you should be aware of the general procedure of a coupled simulation with MpCCI, which is described in IV-1.3 Code Coupling with MpCCI.

2.1.1 Data Exchange and Data Access

For the coupling process, the analysis code must be able to send and receive data based on the mesh of a coupling component. This requires access to the following data of a simulation code:

Mesh information  MpCCI handles exchanges of data between non-matching grids, thus it needs to know the grids on both sides. Therefore, the mesh data of the coupling component must be transferred to MpCCI. This includes
- Number of nodes, number of elements, type of floating point values
- Nodes: ID, coordinates,
- Elements: ID, type, nodes.

Physical values to be exchanged  During the coupling process, data is exchanged, which must be transferred to MpCCI or received from MpCCI. This requires local
- Reading access to values of data to be transferred,
- Allocation of memory for receiving data,
- Method to put received data on local mesh.

Time/Iteration information  During the coupling process, data is exchanged, which must be identified by the simulation time or iteration. This allows the MpCCI server to manage the data exchange and to provide the corresponding data requested by a code.

The data is sent or received by “driver functions”, which must be implemented as part of the code adapter. In addition coupling manager functions of the adapter library must be called by the interface functions. The simulation calls these functions at certain stages as depicted in Figure 1.

MpCCI provides a C-Interface, which can be used from C, C++ and FORTRAN codes.

In addition to this actual adapter, a number of files can be provided to allow MpCCI to manage the coupled simulation and integrate the code into the MpCCI GUI.
2.1.2 MpCCI Interface for Code Integration

Two basic things are necessary for coupling a simulation code with MpCCI, see also Figure 2.

**GUI integration.** The integration into the MpCCI GUI is realized with a set of configuration files which describe the properties and capabilities of a code, how to scan input files and start the code. An overview of these files is given in > 2.3 Code Configuration Directory.

**Code Adapter.** The code adapter is needed to handle the data exchange. The adapter is a plug-in into the simulation code, which can be realized in different ways: It can be included directly into the code or be based on user functions which are provided by the code. MpCCI provides a basic C-interface for the development of code adapters. The structure of the code adapter is sketched in Figure 3. Its main parts are:

- **Driver functions** which perform the basic data exchange with the code. It is the passive part of the adapter, as the driver functions are called by the adapter library.
- **The Code Coupling Manager** is a set of functions for the communication between the code adapter and the MpCCI server. It is part of the MpCCI software package. The functions are defined in the header file "mpcci.h", the object files in "libmpcci*.a" must be linked with the simulation code. The coupling manager functions call the provided driver functions. The exact procedure is sketched exemplarily for `mpcci_init` in Figure 22.
2.2 Code Integration with the MpCCI API Kit

The MpCCI API Kit contains templates as well as a simple example to demonstrate the integration of a simulation code into MpCCI. The MpCCI API Kit contains the following files:

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;adapter&quot;</td>
<td>Templates for code adapter</td>
</tr>
<tr>
<td>&quot;configuration&quot;</td>
<td>Templates for configuration directory</td>
</tr>
<tr>
<td>&quot;example/configuration&quot;</td>
<td>Configuration directory files for the example</td>
</tr>
<tr>
<td>&quot;example/C/src&quot;</td>
<td>Source code of the example in C</td>
</tr>
<tr>
<td>&quot;example/C/src-nompcci&quot;</td>
<td>Source code of the example without code adapter in C</td>
</tr>
<tr>
<td>&quot;example/FORTTRAN/src&quot;</td>
<td>Source code of the example in FORTRAN</td>
</tr>
<tr>
<td>&quot;example/FORTTRAN/src-nompcci&quot;</td>
<td>Source code of the example without code adapter in FORTRAN</td>
</tr>
<tr>
<td>&quot;example/test&quot;</td>
<td>Files for testing the example</td>
</tr>
<tr>
<td>&quot;example/test-nompcci&quot;</td>
<td>Files for testing the example without code adapter</td>
</tr>
</tbody>
</table>

### 2.2.1 A Simple Example

As a simple example, the – simplified – simulation of an elastic foundation is included in the MpCCI API Kit. The original source code is included in the directory "example/C/src-nompcci", a FORTRAN version can be found in "example/FORTTRAN/src-nompcci". The code can be used for two- and three-dimensional computations. The functionality is implemented in "main.c" ("main.f") , data structures are defined in "data.h" and "data.c" ("data.f") together with a simple file input routine. The code is kept simple and thus does not contain any input file checks etc.

Only two element types are supported: Line elements with two nodes and quadrilateral elements with four...
nodes. For each element an example input file is included in "example/test-nompcci".

![Diagram](image)

Figure 4: Examples for the simple foundation code.

The structure modeled in "2dexample.fnd" is depicted in Figure 4 on the left, its content is:

```plaintext
EF bed1 2 1 -0.5
NODES 3
0 0 0
1 1 0
2 2 0
ELEMENTS 2
0 0 1
1 1 2
```

The acronym **EF** in the first line starts the definition of an elastic foundation (one file could contain more than one!). Its name is “bed1”, with space dimension “2” and springs in direction “1”, i.e. the y-direction. The spring constant of all springs is “-0.5”. The minus sign is added because it is loaded from top, i.e. a positive pressure should result in a motion in negative y-direction.

Running “foundation” with this input file yields:

```plaintext
> ../bin/foundation-nompcci 2dexample.fnd
Foundation - computation
Reading file >>2dexample.fnd<<...
foundation >bed1< dim=2 direction=1 stiffness=-0.5
nodes...
   0 : 0 0
   1 : 1 0
   2 : 2 0
elements...
   0 : 0 1
   1 : 1 2
   allocating memory...
read 1 foundations.

Step 0 of 3
Please enter pressure: 1.0

results for foundation 'bed1':
   node 0: force= 0.5 displacement= -1
```

```
### Results for Foundation 'bed1':

<table>
<thead>
<tr>
<th>Node</th>
<th>Force</th>
<th>Displacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>-2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>-4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>-2</td>
</tr>
</tbody>
</table>

In each step, the user is asked for a pressure value which is then applied and the resulting forces and displacements are printed to `stdout`. The number of “steps” can also be given as a second command line argument after the input file name, three steps are made if nothing is specified.

Please have a look at the files of the original code in "src-nompcci". The header file "data.h" contains the definition of the data structure `FOUNDATION` and of a list of such structures `fndlist` which is filled by the values from the input file.

In "main.c" ("main.f") all actual computations are contained, a number of functions is called from `main`, which controls what is done.

The example was created on a Linux 64 Bit machine but should also run on other Linux systems. The C version was compiled using the GNU compiler `gcc`, the FORTRAN version was tested with GNU, Absoft 9.0, Intel 9.1 and PGI 6.1 compilers, see the "Makefile".

For combining C and FORTRAN sources, as it is done in the FORTRAN example, the naming conventions depend on the compiler. In the sources of the example an underscore is added to the C function names and lowercase names are used. Please consult the documentation of your compiler to find how to combine C and FORTRAN sources.
2.2.2 Step-by-Step Procedure for Code Integration

The MpCCI API Kit provides general files for a new code adapter and an example called foundation code. The code adapter files are:

**General**
- "configuration/*"
- "adapter/adapter.h"
- "adapter/adapter.c"

**Example**
- "example/configuration/*"
- "example/C/src/adapter.h"
- "example/C/src/adapter.c"

The example files for the code foundation are provided in an original standalone version and a new version with the MpCCI code adapter:

<table>
<thead>
<tr>
<th>Standalone</th>
<th>Adapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;example/C/src-nompcci/data.h&quot;</td>
<td>&quot;example/C/src/data.h&quot;</td>
</tr>
<tr>
<td>&quot;example/C/src-nompcci/data.c&quot;</td>
<td>&quot;example/C/src/data.c&quot;</td>
</tr>
<tr>
<td>&quot;example/C/src-nompcci/main.c&quot;</td>
<td>&quot;example/C/src/main.c&quot;</td>
</tr>
<tr>
<td>&quot;example/C/src-nompcci/Makefile&quot;</td>
<td>&quot;example/C/src/Makefile&quot;</td>
</tr>
</tbody>
</table>

Each "Makefile" for the foundation example generates the executable "example/bin/foundation".

In order to switch between the standalone and the adapter version you have to rebuild the executable.

To understand the changes you can compare ("diff") the general code adapter files with the foundation example and the standalone foundation files with the adapter foundation files.

All changes are commented and marked with \(\text{CHANGED}\); (For FORTRAN see the corresponding files in "example/FORTRAN".)

The creation of a code adapter can be performed on a step-by-step basis:

1. **Step 1**: Preparations.
2. **Step 2**: Create the configuration directory.
3. **Step 3**: Test the configuration.
4. **Step 4**: Create the code adapter.
5. **Step 5**: Test the code adapter.

For each step, the corresponding changes in the example are mentioned as well.
STEP 1: Preparations.

Understand the basic concepts of MpCCI. If you have no experience with MpCCI, it is recommended to run some of the Tutorial examples first. Try to understand the way MpCCI works by reading V-3 Code Coupling ↓.

Decide which quantities shall be exchanged. Just select a basic set of quantities, more can be added later. It is sufficient to select one quantity for receiving and one for sending.

A complete list of quantities is given in the Appendix.

Identify possible coupling components. Decide whether data is exchanged on surfaces or volume and how they can be found in the model files.

The example “foundation” code computes displacements due to an external pressure. Thus the following quantities will be exchanged:

foundation code (elements) ← OverPressure ← partner code
foundation code (nodes) → NPosition → partner code
STEP 2: Create the configuration directory.

**Copy files from MpCCI API Kit**  The configuration directories for simulation codes are all located in the subdirectory "$MpCCI_home/codes" of the MpCCI home directory, which you can find with the command `mpcci home`.

In the "codes" directory, there is one configuration directory for each code. You should already find some of commercial simulation codes there.

Create a directory with the name of your code there and copy all template files from the MpCCI API Kit, subdirectory "configuration", to your new code configuration directory.

"gui.xcf"  Change options to your needs, a detailed description is given in § 2.4 MpCCI GUI Configuration File gui.xcf.

For the "foundation" example, the codename is set to "foundation". The foundation code does not use any units, so as default we select "SI". The code type is set to [SolidStructure], as deformations of solid structures are computed. A list of available types is given in § V-3.1.1 Physical Domains.

In the section <ModelsMenuEntries>, only "1.0" is added as version, the file extension is set to .fnd and the unit selection menus are kept as no fixed set of units is used in the code.

The <ComponentTypeDimensions> are a list of names for the coupling components. We actually only need Face, but add names [Global], [Line], [Face] and [Volume].

From the supported quantities, [OverPressure] and [NPosition] are kept, the locations are set appropriately and send and receive options are set to [Direct] as all data will be directly written to and read from the code's data structures. Both quantities can be exchanged on faces. In the 2D case, a line also represents a face.

As an additional Go-Menu entry, a selection of the number of steps is added. This is later passed to the code as a command line argument. It should also be checked by the Checker.

The Scanner only needs to know the model file, the Checker and Starter also get the number of steps, the stopper gets the model file name.

**Perl scripts:** "Checker.pm", "Info.pm", "Scanner.pm", "Starter.pm", "Stopper.pm", "Subcmd.pm"

The Perl scripts are needed to get information at run time and interact with your code. See § 2.5 Perl Scripts for a detailed description, each template file is commented.

Most important are "Scanner.pm" to scan the input file and "Starter.pm" to start the code. It is recommended to also use "Info.pm" to gather basic information and "Stopper.pm" to trigger a graceful exit of your code. The "Checker.pm" is optional and is used to validate the code configuration before the code is executed.

The MpCCI API Kit contains an additional "external Scanner.pm" if you do not want to use Perl to scan the model file, but e.g. the file reader of your simulation code.

In the information module "Info.pm" for foundation, the only thing which is really done is to find the executable in the path and return the requested information.

The "Scanner.pm" scans the model file "*.fnd" for coupling component definitions, in our case these are all elastic foundations which are defined. These start with the keyword [EF] which is followed by the name. The input file is simply searched for lines containing this information, which is then returned to the calling MpCCI function.

The "Checker.pm" checks whether the number of steps is sufficient and adds them to the summary information of the Checker output.

The "Starter.pm" obtains the model file name and the number of steps from the MpCCI GUI, creates an argument list [@argv] with the corresponding command line arguments and starts "foundation".
STEP 3: Test the configuration.

**Testing the Perl scripts.** Before testing the GUI integration, you should ensure that the Perl scripts work.

- **Start with the scanner.**
  
  The code has a list of available command line option, see *mpcci foundation help*. To run a test, you must provide the model file to scan. The MpCCI API Kit provides a model file in the subdirectory "example/test/foundation".

  ```bash
  mpcci foundation scan elasticwall.fnd
  ```

  The results of the scanning process are saved in a scan file "mpcci_<original filestem>.scan", i.e. for the above example the file would be named "mpcci_elasticwall.scan".

  All information your scanner found should be contained in this file. The model info is listed first, followed by the list of components. Please check if all information is present.

- **Test the starter using "test_Starter.pl".**
  
  You should edit the "test_Starter.pl" and modify the variable `$ENV{'MPCCI_MODEL_FILE'}` to point to the model file.

  If all necessary variables are set, you can simply run "test_Starter.pl". The starter should not do more than start your code with the appropriate command line options.

- **You can also test the stopper with "test_Stopper.pl".**

  For “foundation” the testing scripts "test_Stopper.pl" and "test_Starter.pl" were changed: The stopper only needs the model file, the starter model file and number of steps for testing.

**Testing "gui.xcf"** For testing "gui.xcf" you need to have a license for a code adapter, which you usually receive together with the MpCCI API Kit.

For testing, you must prepare a simple input file and one for the partner code and proceed as follows.

1. Go to the testing directory (for the foundation example this is "example/test") and start the MpCCI GUI with *mpcci gui*.
   
   → If you get an error message right after starting the MpCCI GUI, there is a syntax error in your "gui.xcf". Make sure, that the file is a proper XML-file.

2. Your newly added code should now appear in the list of codes on the left side.
   
   → If you cannot find your new code, ensure that you have created a new subdirectory in "<MpCCI_home>/codes" with the correct name.

3. Click on your code, and you should see what you defined as elements of the Models step.
   
   → If some elements are missing or wrong, check the `<ModelsMenuEntries>` section.

4. Fill in the required values in the form.
   
   → With selection of the model file the scanner is started. When the scanner finished, click on the box which appears next to the code selection and check the return values of the scanner.

5. Select and scan the partner code and proceed with the Regions step.
   
   → Here, you should be able to select the components scanned from the model files and select the quantities as defined in "gui.xcf". If you cannot select any quantity there are no quantities which are supported on the same geometric entity by both codes. If some quantities are missing check the `<SupportedQuantities>` block. Remember that only quantities will be shown, which are also supported by the partner code!

6. Finally, proceed to the Go step and check the entries in the panel of your code.
   
   → If entries are wrong check the `<GoMenuEntries>` section of "gui.xcf".

7. You can also start your code – it should simply run as usually. However, no coupling will occur as no code adapter is present. For the foundation example you may use the standalone version, which runs as usually without coupling, or you use the adapter version for a coupled simulation.
STEP 4: Create the code adapter.

- The adapter template files are located in the subdirectory "adapter". Add "adapter.h" and "adapter.c" to your source code.
- Change the definition of the list of driver functions `MpCCIDriverFunctions` in "adapter.c" to fit your needs: Add name of your code and select which data exchange functions you need. Usually the block with functions called before/after some actions can be left empty.
- Look if the interface functions are also useful for your code and make appropriate changes. In most cases they should already be OK.
- Adjust the driver functions. All driver functions which are declared in `MpCCIDriverFunctions` must be defined. See \ref{2.8 MpCCI Driver Functions} for a description.
- Change "adapter.h" to fit "adapter.c"
- Insert the calls of the interface functions at appropriate places in your code as described in \ref{2.1 Code Integration and Simulation Code Requirements}.
- Add "adapter.c", the MpCCI include directory "<MpCCI_home>/include" and the MpCCI client library for your platform to your "Makefile".

For “foundation” the most important changes are (please see source code files in "example/[C|FORTRAN]/src" for details):

**Driver Functions** Only a small set of driver functions,

- `MpCCI_Driver_partUpdate`
- `MpCCI_Driver_definePart`
- `MpCCI_Driver_getFaceNodeValues`
- `MpCCI_Driver_putFaceElemValues`

are used, all other functions were set to `NULL` in the `MpCCIDriverFunctions` structure.

**Additional helper functions** Additional helper functions are required: `getSurfaceID` to identify coupling components by their names and for `adapterOutput` and `error` for data output. In the C version they are included in "adapter.c" in the FORTRAN version they are located in "helpers.f".

**Data exchange** In the C version the data access is directly written into the driver functions `MpCCI_Driver_getFaceNodeValues` and `MpCCI_Driver_putFaceElemValues`.

The FORTRAN version uses additional subroutines in "helpers.f": `getInfo` to obtain basic information, `getMesh` to obtain mesh information and `getNPosition` and `putPressure` for data exchange. The FORTRAN helper functions are called from the C routines in "adapter.c".

**Makefiles** The Makefile must be changed to include "mpcci.h" and the MpCCI library. For FORTRAN, additionally a C compiler is required to compile "adapter.c" which can be linked directly with the FORTRAN objects.
STEP 5: Test the code adapter.

Please ensure first that the files in the configuration directory are correct (Step 3). So we can assume now, that you already have set up a sample problem.

- Install the license for your code - either on a license server or on your local machine, see III-5 Licensing for details. You can check the license status with `mpcci license mpcci` or from the MpCCI GUI in License → Check the MpCCI license status.
- Open your project ("*.csp") in the MpCCI GUI.
- In the Settings step, you may set the output level to 3 to obtain maximum output.
- In the Go step, select Run server processes inside xterm.
- Start the MpCCI server and both simulation codes by pressing the Start buttons in the Go step. One window for each code server (and the control process if enabled) and for both codes should pop up.
- Check the output of your code and the corresponding MpCCI server. The output of the servers should help you to find any errors in the code adapter functions.

For the foundation code, the code’s minimal output (in the yellow window) should look as follows:

```
Starting: foundation elasticwall.fnd 2> mpcci_elasticwall.log 2>&1
Waiting for logfile "mpcci_elasticwall.log"....

This means MpCCI is waiting for the code’s output. This message should be followed by the usual output of the code, here:

Foundation - computation
Reading file >>elasticwall.fnd<<...
foundation >elasticwall< dim=3 direction=2 stiffness=-5 nodes...

and so on. Further below you should find the call of `ampcci_tinfo_init`, with

```
Initializing the coupling process!
MpCCI: Coupling configuration setting:
MpCCI: Scheme: Explicit
MpCCI: Algorithm: Serial
MpCCI: Code behavior: Leading

followed by the list of coupled regions after the `ampcci_config` call,

```
MpCCI: Coupling grid definition for component "elasticwall" ...
MpCCI: Server: 47010@berber
MpCCI: Part : "elasticwall"
MpCCI: MeshDim(2), MeshId(1), PartId(0), Nodes(9), Elements(4)
MpCCI: 2 Quantities:
MpCCI: Direction(RECV), Dimension(1), Location(ELEM), Storage(Direct/0), Name(OverPressure)
MpCCI: Direction(SEND), Dimension(3), Location(NODE), Storage(Direct/0), Name(NPosition)

For each exchange you should see an output like

```
MpCCI: entered put_values...
MpCCI: finished put_values...

Remember that the first exchange depends on the used coupling algorithm (parallel with or without initialization or serial), thus – depending on your choice – the output `get_values` or `put_values` may
not appear.

The output in the server window provides basic information (because the output level is set to 1). This is followed by the actual server output, starting with

```
[MpCCI License] mpcci_adapter_foundation: feature test...
[MpCCI License] mpcci_clients: feature checkout...
```

which is followed by:

Loading code specific mesh and quantity settings for code "foundation":

```
Code specific quantity properties:

"NPosition": Location(NODE), Default(0)
  Send   : Method(0), Index(0)
  Receive: Method(0), Index(-1)

"OverPressure": Location(ELEM), Default(0)
  Send   : Method(0), Index(-1)
  Receive: Method(0), Index(0)

Code specific parts properties:

"elasticwall": MeshId(1), PartId(1)
  Send   : "NPosition"
  Receive: "OverPressure"

Code specific mesh scale: 1

Code specific mesh transformation:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Det: 1
```

which reflects the mesh and quantity settings. This is followed by a block starting with

```
Code: "foundation", ID(0), nice(64), clients(1), type(Finite Element).
  Mesh: "foundation/MESH-1", mid(1)
    Coord system: 3D
    Mesh type : FACE
    Distances : [0.5 .. 0.707107]
    Bounding box: [0 .. 1] [0 .. 1] [0 .. 0]
    Domain size : 1

  Send: "NPosition"
    Dimension : 3
    Direction : SEND
    Location : node
    Default : 0
    Buffers : 1
```
Recv: "OverPressure"
  Dimension : 1
  Direction : RECV
  Location : element
  Default : 0
  Buffers : 1
  Source : "FLUENT/MESH-1" -> rel_ml -> map_ml

Part: "foundation/MESH-1/elasticwall", pid(0)
  Coord system : 3D
  Mesh type : FACE
  NNodes : 9
  NElements : 4
  Type1 : QUAD4
  Ntypes: 1
  Total nodeids : 16
  Total vertices: 16
  Distances : [0.5 .. 0.707107]
  Bounding box : [0 .. 1] [0 .. 1] [0 .. 0]
  Domain size : 1

with the element definitions. Next follows the neighborhood computation, which lists the neighborhood information (see also ▷ V-3 Code Coupling◁), starting with

| Neighborhood relationship: "foundation/MESH-1" -> "FLUENT/MESH-1" |
| rel_ml::create_mapmesh(MESH="foundation/MESH-1"): new mesh representation. |
| rel_ml::create_mapmesh(MESH="FLUENT/MESH-1"): new mesh representation. |
| rel_ml::execute(MAPLIB_REL=0x8071158): Configuration: |
| nodetolerance=0.0292845 |
| normaldistance=0.146422 |
| tangentialdistance=0.146422 |
| distance=0.146422 |
| precision=1e-06 |
| multiplicity=4.82923 |
| orphaninfo=true |
| rel_ml::execute(MAPLIB_REL=0x8071158) |
| -> 0 seconds CPU. |
2.2.3 Code Coupling with the Example

For testing the code adapter for the example code, a set of sample problems is provided in "example/test" for coupling "foundation" with some commercial codes.

The sample problem is sketched in Figure 5. Both input files are given, "foundation/elasticwall.fnd" for "foundation" and for FLUENT "FLUENT/cube.cas".

To run the sample problem do the following:

- Go to the "example/test" directory and start theMpCCI GUI.
- Select “foundation” as the first code (you should first finish creating the configuration and code adapter for the foundation code, see 2.2.2 Step-by-Step Procedure for Code Integration) and "foundation/elasticwall.fnd" as model file. Keep the unit system set to SI.
- Start the Scanner for “foundation”, if you click on the green check mark, you should get

```
# MpCCI relevant information:
# Model dimensions : 3D
# Coordinate system : Cartesian
# Solution type : Static
# Load cases : ?
# Unit system : ?
# Precision : Double precision (64 bit)
```

- Select FLUENT as the second code, the FLUENT version 3ddp, and the latest FLUENT release. Finally choose "FLUENT/cube.cas" as model file.
- Proceed to the Regions step – the Face (2D)-card should be selected. Select elasticwall and bottom as coupling components and NPosition and OverPressure as quantities.

Figure 5: FLUENT– foundation sample problem. A cube filled with an ideal gas is coupled with an elastic foundation which deforms due to the gas pressure.
• Proceed to the Go step. Select Run server processes inside xterm for the server. For “foundation” select 3 as number of steps. For FLUENT do not change further options.

• Save the project and start the processes. One server window should pop up. A yellow window with the output of the foundation code should pop up and the FLUENT graphical interface.

• In the FLUENT window select Solution→Initialization and press the [Initialize] button to initialize the FLUENT solution.

• Select Solution→Run Calculation and set the Number of Time Steps to 3 and press Calculate.

• The MpCCI Monitor will automatically start.

• Now, FLUENT should perform 20 iterations while “foundation” finishes its computation. The FLUENT result is depicted in Figure 6. You should clearly recognize the deformation of the bottom where the elastic foundation is coupled.

• After the process terminated, the windows closed automatically.

Figure 6: FLUENT mesh and result of the cube example.
2.3 Code Configuration Directory

The files which are required to integrate the code into MpCCI must be located in a specific directory, as already described in ▶ 2.2 Code Integration with the MpCCI API Kit ◀. The directory "$MpCCI_home$codes" has one subdirectory for each code which is supported by MpCCI. At least six files, which always have the same file names, should be inside any code subdirectory:

"gui.xcf" This file contains all definitions which are required to fit the code into the MpCCI GUI, which includes
- code name and version information,
- extension of input files,
- additional code options for the Models and Go steps,
- and a list of supported quantities.
▶ 2.4 MpCCI GUI Configuration File gui.xcf ◀

"Scanner.pm" This Perl script is started to scan the input file of the simulation code for information which is needed for the coupling process, mainly to identify possible coupling components.
▶ 2.5.2 Scanner.pm ◀

"Starter.pm" The starter script starts the simulation code with appropriate command line options, which can be selected in the MpCCI GUI.
▶ 2.5.4 Starter.pm ◀

"Stopper.pm" The stopper script is called if the code shall be stopped, i.e. if the Stop button in the MpCCI GUI is clicked.
▶ 2.5.5 Stopper.pm ◀

"Info.pm" should collect application specific information like code release.
▶ 2.5.7 Info.pm ◀

"Checker.pm" should validate the code configuration before the code starts, i.e. the "Starter.pm" is executed. The Check button in the MpCCI GUI executes this script.
▶ 2.5.3 Checker.pm ◀

"Subcmd.pm" can be used to define MpCCI code specific subcommands. These subcommands define the command line interface.
▶ 2.5.8 Subcmd.pm ◀
2.4 MpCCI GUI Configuration File "gui.xcf"

The file "gui.xcf" is an XML-file, which contains definitions for the MpCCI GUI. Entries for the Models and Go step and for the Codes menu in the menu bar can be defined, and the selected options can be passed on to the scanner, starter, stopper and killer scripts.

It consists of several sections, which are discussed in the following. A general description of the entries is given in 2.4.9 General MpCCI GUI Elements.<code>

2.4.1 Code Information: <CodeInfo>

```xml
<CodeInfo>
    <Units type="string" default="SI" />  
    <Type type="string" default="CFD Fluid FluidThermal FluidPlasma" />
    <SelfCoupler type="bool" default="true" />
    <CopyComponents type="bool" default="true" />
    <Feature type="string" default="Negotiation iterativeCoupling" />
</CodeInfo>
```

The code information block contains the basic information of a code:

- **Units**: Unit system used by the code
- **Type**: Type of code
- **SelfCoupler**: Indicator if a code can be coupled with itself (default is false)
- **CopyComponents**: Indicator whether a code allows making copies of its components (default is false)
- **Feature**: Type of feature

The type of the code can be one or several of the code types CFD, ElectroMagnetism, FluidPlasma, FluidThermal, InjectionMoulding, Radiation, SolidStructure, SolidThermal. Usually one code can support different analysis types. These can be given separated by spaces. See V-3.1.1 Physical Domains for more information on their meaning.

The type of feature may be needed to determine possible coupling types. Current accepted values are:

- **Negotiation**: Describes the code capability to negotiate the time step size prior to the next time step.
- **IterativeCoupling**: Describes the support of Implicit coupling scheme.

2.4.2 Codes Menu: <CodesMenuEntries>

For each code commands can be defined which are provided in the menu bar beneath the codename item. If more than three codes are offered the code menus are collected under the **Codes** item in the menu bar.

Each provided command is defined as a menu element

```xml
<Release type="menu"
    text="Releases"
    tooltip="Display all installed releases located by MpCCI."
    command="mpcci Codename releases" />
```

- **Release** is the unique identifier of the selected value.
- **the type** is "menu".
- **the provided text** is displayed as the menu entry.
- **the tooltip** is shown as tooltip for the menu entry.
- **the command** will be passed to the local system to be executed.

The result of the executed command will be shown in a dialog box.
2.4.3 Models Step: <ModelsMenuEntries>

For each code additional options can be added to the Models step in the MpCCI GUI. The information given here is only partly evaluated by the MpCCI GUI itself. Most options are for use in the scanner or starter to hand it over to the simulation code. The definitions in the "gui.xcf" actually completely determine the appearance in the MpCCI GUI.

The template file from the MpCCI API Kit contains already two examples. An overview of all possible elements is given in > 2.4.9 General MpCCI GUI Elements.<

The first element in the example is an enumeration element

```xml
<Release type="enum" default="latest" sort="false"
          description="Please select the release:">
  <enum value="latest" />
  <enum value="1.1" />
  <enum value="1.0" />
</Release>
```

Version is the unique identifier of the selected value.

The second element is a file selector, which is usually part of every Models Step entry:

```xml
<ModelFile type="filename" required="true" default=""
           description="Please select the model file:">
  <filename suffix=".mod" />
</ModelFile>
```

Further <filename> lines can be added to allow a selection of different suffixes.

An example of a configuration is shown in Figure 8.
Figure 8: Example of a `<ModelsMenuEntries>` definition in "gui.xcf" and the resulting Models step.

### 2.4.4 Component Types: `<ComponentTypeDimensions>`

Here, you specify the dimension for component types. The components and their types are returned by the scanner in the scanner output file. Because the types for the components differ from code to code each type name used in the scanner output file has to be associated with a dimension. These dimensions correspond with the labels of the element collections in the MpCCI GUI Coupling Step.

- **Global**: means the component is a data structure and its elements are global values.
- **Ipoint**: means the component comprises 0-dimensional integration point elements.
- **Point**: means the component comprises 0-dimensional point elements.
- **Line**: means the component comprises 1-dimensional line elements.
- **Face**: means the component comprises 2-dimensional face elements.
- **Volume**: means the component comprises 3-dimensional volume elements.

Add each component type to the appropriate `<ComponentTypeDimensions>` block.

```xml
<ComponentTypeDimensions>
  <Global type="string" default="globalType ..." />
  <Ipoint type="string" default="ipointType ..." />
  <Point type="string" default="pointType ..." />
  <Line type="string" default="lineType ..." />
  <Face type="string" default="faceType ..." />
  <Volume type="string" default="volumeType ..." />
</ComponentTypeDimensions>
```

The tags `Global`, `Ipoint`, `Point`, `Line`, `Face` and `Volume` indicate the associated dimension of the MpCCI GUI Coupling Step. The type is "string" and default should be set to the component type names used in the scanner output file. Several component types are separated by a blank. If no type name exists for a dimension, the block for that dimension may be omitted.

### 2.4.5 List of Quantities: `<SupportedQuantities>`

Before the actual list of quantities, the storage options are defined in the element `<StorageOptions>`. You need not change the definition given in the template. More definitions are only useful, if your code supports different storage locations of the quantities and the user should select them in the MpCCI GUI.
The tag `<SupportedQuantities>` contains a list of all quantities which are supported by the simulation code. The template which you unpacked in your code subdirectory contains a complete list of all quantities, which can be handled by MpCCI. For a description of the quantities see also the quantities list in the Appendix.

So please remove or comment out all quantities your code does not support.

The remaining lines must be fitted to the simulation code. Each line has following attributes, e.g.:

- The type is "quantity".
- The attribute `loc` which defines the location of the quantity can be "code", "node", "elem" or "global" or a combination of these e.g. "node elem", which means it can be of either location.
  - "code"  The location of the quantity is defined by the code itself.
  - "node"  Nodal quantity, i.e. the values are defined for each node.
  - "elem"  Element quantity, i.e. the values are defined per element, also for quantities defined at special points of the element, e.g. at integration points.
  - "global"  Global quantity, which is not related to nodes or elements, e.g. a time step size.
- The attributes `so` and `ro` which stand for "send option" and "receive option" can be set to any of the storage methods defined in `<StorageOptions>` or combinations like "Direct Usrmem". Usually `so` and `ro` are either set to the empty value "", which means no receiving or sending of this quantity is possible, or to "Direct", which means the values are read and written directly to the nodes or elements in the simulation code. The predefined storage methods are also listed in the description of the code API macros in V-2.9.9 Quantities.
- The attribute `valid` assigns the quantity to the element types `Global`, `Point`, `Line`, `Face` and `Volume` which are the labels of the element collections in the MpCCI GUI Coupling Step. Quantities which are valid for the element type `Point` are also valid for integration point elements. There exists no extra `Ipoint` type for the quantity `valid` attribute. If the `valid` attribute is omitted the quantity is valid for all of its defined coupling dimensions (see Quantity Reference in Appendix) which is often used for global values. On the other hand if it's set to the empty value "", the quantity isn't valid for any dimension and won't be offered in the MpCCI GUI Coupling Step.

Finally a quantity definition block should look like

```
<Quantities>
  <DeltaTime type="quantity" loc="node elem" so="Direct" ro="Direct" valid="Point Line Face Volume"/>
  <WallForce type="quantity" loc="elem " so="Direct" ro="" valid="Face"/>
  <NPosition type="quantity" loc="node" so="" ro="Direct" valid="Face"/>
</Quantities>
```

which means that the simulation code can exchange three different quantities. The time step size is a global quantity and can be sent or received, wall forces are defined at elements and can only be sent, while the node positions can only be received. Time step size has no `valid` attribute and therefore can be exchanged on its defined coupling dimension which is `Global`. The wall forces and node positions can only be exchanged on 2D surfaces indicated by `valid="Face"`. For the element types `Point`, `Line`, and `Volume` no quantities are supported.

### 2.4.6 Algorithm Step: `<CouplingMenuEntries>`

This section defines the code specific coupling algorithm for the simulation code. You find following default major blocks to be set up (cf. V-4.5 Algorithm Step):

- Analysis type
This Analysis section is common to all codes. The type of the analysis results from the "Scanner.pm" script by setting the solution type field.

```
<Analysis type="panel" description="Analysis">
  <AnalysisType type="enum" default="Transient" description="Analysis type" infoID="Info.analysisType.code">
    <enum value="Transient"/>
    <enum value="Steady state"/>
  </AnalysisType>
</Analysis>
```

- **Solver settings**
  This section defines how the solver step size is managed for the different analysis type and coupling scheme combinations (Steady state, Transient implicit, Transient explicit), which is mostly Defined by model.

```
<Solver type="panel" description="Solver settings">
  <StepSize description="Solver step size" default="Defined by model"/>
  ...
</Solver>
```

- **Coupling steps**
  This section defines the frequency of the data exchange, i.e. subcycling. The step size can be constant or variable. According to the coupling scheme and the code capabilities you may define some field to feed the number of inner iterations between the data exchanges for Transient implicit analysis or activate the Delay boundary updates.

```
<CouplingSteps type="panel" description="Coupling steps">
  <StepSizeType description="Coupling step size"/>
  <StepSize description="Constant coupling step size (solver steps)"/>
  <InnerIters description="Number of inner iterations"/>
  <DelayValue description="Delay boundary updates"/>
</CouplingSteps>
```

- **Runtime control**
  This section defines the options to manage the coupling runtime: start, number of iteration after the coupling finished.

```
<Runtime type="panel" description="Runtime control">
  <cbegType description="Define coupling start">
    <enum value="Take latest time"/>
    <enum value="Specify time"/>
  </cbegType>
  <time_cbeg description="Time of coupling start (s)"/>
  <cbegType description="Define coupling start">
    <enum value="Take latest iteration"/>
    <enum value="Specify iteration"/>
  </cbegType>
  <iter_cbeg description="Iteration number for coupling start"/>
  <iter_pend description="Number of iterations after the coupling"/>
</Runtime>
```

### 2.4.7 Go Step: 

Similar to the definitions for the Models step, the appearance of the Go step can be defined in "gui.xcf".
Additional Configuration for Parallel Code

A minimum set of definitions have to be provided in order to describe the parallel configuration of the code. This is the set of definitions to insert in `<GoMenuEntries>`:

```xml
<ParallelRun type="panel" default="false" description="Run parallel">
  <NumProcs type="int" default="2" min="2" max="512" description="No. of parallel processes" />
  <HostList type="hostlist" default="" description="Optional 'host host ...' to be used" />
  <HostFile type="filename" default="" description="Optional hostlist file">
    <filename suffix=".hosts" />
    <filename suffix=".hostlist" />
    <filename suffix=".hostfile" />
  </HostFile>
  <DefaultHosts type="bool" default="false" description="Use default hostfile" />
</ParallelRun>
```

The parallel configuration is encapsulated in a `panel` element. The following information may be configured:

- The number of parallel processes.
- The hosts to be used.

Additionally for the parallel configuration you have to add the following entries in the `<Starter>` environment which is stated more precisely in the next section.

```xml
<_MPCCI_<code name>_PARA_RUN type="string" default="%{GoMenuEntries.ParallelRun}" />
<_MPCCI_<code name>_PARA_NPROCS type="string" default="%{GoMenuEntries.ParallelRun.NumProcs}" />
<_MPCCI_<code name>_PARA_HOSTLIST type="string" default="%{GoMenuEntries.ParallelRun.HostList}" />
<_MPCCI_<code name>_PARA_HOSTFILE type="string" default="%{GoMenuEntries.ParallelRun.HostFile}" />
<_MPCCI_<code name>_PARA_DEFHOSTS type="string" default="%{GoMenuEntries.ParallelRun.DefaultHosts}" />
```

This environment information will be evaluated by the Perl function `code_start_prepare`.

```perl
my ($numProcs,$sharedFS,@hostList) = code_start_prepare($codeName, # required
  $compressHostList, # optional
  $printHostList, # optional
  $checkHostList, # optional
  $checkFS # optional);
```

This is a helper function to assist the preparation of the parallel run. It has to be called in your "Starter.pm" script. The parameters are:

- `$codeName` The name of your code.
- `$compressHostList` Indicator (1 or 0) whether the returned hostlist shall be compressed or not. When the check is set all multiple defined hosts will be removed from the list.
- `$printHostList` Indicator (1 or 0) whether the hostlist shall be printed to standard output or not.
- `$checkHostList` Indicator (1 or 0) whether the hosts shall be checked for being alive. When the check is set and at least one listed host isn’t alive an error message will be shown and the start of your code will fail.
$checkFS Indicator (1 or 0) whether it shall be checked if the hosts share one file system. The result of the check will be returned in $sharedFS.

The function will create a list of hosts where we fire up the processes. It returns the number of processes ($numProcs), the hostlist (@hostlist) and whether the hosts share one file system ($sharedFS = 1 or 0) if this was indicated to be checked. You may post process the returned values to parametrize the start of your code.

### 2.4.8 Environments for Scanner, Checker, Starter, Stopper and Killer

The last sections of "gui.xcf" define which information is passed to the Perl scripts which are called by the MpCCI GUI (see ∆ 2.5 Perl Scripts). Values are transferred in form of environment variables, which are defined in "gui.xcf" as follows:

Each of the elements `<Scanner>`, `<Checker>`, `<Starter>`, `<Stopper>` and `<Killer>` contains one sub-element `<Environment>` in which the variables are set to values which are defined in other sections of "gui.xcf" (e.g. ∆ 2.4.7 Go Step: `<GoMenuEntries>`).

There are two classes of values:

- **Required values** must be set. These are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Variable name</th>
<th>Required for script</th>
</tr>
</thead>
<tbody>
<tr>
<td>The name of the model file</td>
<td>_MPCCI_MODEL_FILE</td>
<td>Scanner, Checker, Starter, Stopper, Killer</td>
</tr>
<tr>
<td>Coupling configuration settings</td>
<td>MPCCI_TINFO_SCHEME  MPCCI_TINFO_DURATMPCCI_TINFO_CSTEP MPCCI_TINFO_DELAY</td>
<td>Checker, Starter</td>
</tr>
</tbody>
</table>

- **Optional values** can be added as necessary. All optional variables should follow the name convention and start with _MPCCI_<code name> to avoid conflicts with other variables.

  All variables must be of type 
  "string" like text field elements described in ∆ 2.4.9 General MpCCI GUI Elements. The default option is used to reference the value and the description option is unused.

A short definition of the environment of the starter script for the code “example” could look as follows:

```xml
<Starter>
  <Environment>
    <MPCCI_TINFO_SCHEME type="string" default="%(CouplingMenuEntries..." />
    <MPCCI_TINFO_DURAT type="string" default="%(CouplingMenuEntries..." />
    <MPCCI_TINFO_CSTEP type="string" default="%(mpcciserver:CouplingMenuEntr..." />
    <MPCCI_TINFO_DELAY type="string" default="%(mpcciserver:CouplingMenuEntr..." dependsOn="%(mpcciserver:Couplin...DelayedUpdate.DelayCode)" dependingValue="%(#codename)" />
    <MPCCI_MODEL_FILE type="string" default="%(ModelsMenuEntries.ModelFile)"/>
    <MPCCI_EXAMPLE_VERSION type="string" default="%(ModelsMenuEntries.Version)"/>
    <MPCCI_EXAMPLE_MODE type="string" default="%(GoMenuEntries.batchMode)"/>
  </Environment>
</Starter>
```

ℹ️ MPCCI_TINFO_DELAY is only required if delayed coupling is supported.

### Optional values

Optional values can be added as necessary. All optional variables should follow the name convention and start with _MPCCI_<code name> to avoid conflicts with other variables.

All variables must be of type "string" like text field elements described in ∆ 2.4.9 General MpCCI GUI Elements. The default option is used to reference the value and the description option is unused.
2.4.8.1 MPCCI_TINFO_* Variables

The MPCCI_TINFO_* variables are required to set the transfer information MPCCI_TINFO used by the adapter for the application (see ▶ 2.9.6 Transfer Information: MPCCI_TINFO\#). It is set automatically within the MpCCI GUI before starting an application. This variable is composed of several pieces of information for the coupling configuration.

The MPCCI_TINFO_* variables in the "gui.xcf" take the required information as references from the settings in the CouplingMenuEntries group, either of the code itself or of the server which is then marked by the mpcciserver: prefix.

The syntax of each variable is as follows:

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPCCI_TINFO_SCHEME</td>
<td>Analysis type : Coupling scheme : Algorithm : Algorithm code :</td>
</tr>
<tr>
<td>MPCCI_TINFO_DURATION</td>
<td>Coupling start time : Total coupling time : Coupling start iteration : Maximum number of coupling steps : Number of iterations after coupling :</td>
</tr>
<tr>
<td>MPCCI_TINFO_CSTEP</td>
<td>Type of coupling step size : Coupling step size : Maximum number of coupling step iterations : Number of solver’s inner iterations :</td>
</tr>
</tbody>
</table>

with these references and values:

- **Analysis type** = %(CouplingMenuEntries.Analysis.AnalysisType)
  - value = transient or steady state.

- **Coupling scheme** = %(mpcciserver:CouplingMenuEntries.Analysis.CouplingScheme)
  - value = explicit or implicit.

- **Algorithm** = %(mpcciserver:CouplingMenuEntries.AlgorithmDef.Algorithm)
  - value = serial for a Gauss Seidel algorithm or parallel for a Jacobi algorithm.

- **Algorithm code** = %(mpcciserver:CouplingMenuEntries.AlgorithmDef.Code)
  - value = name of leading code for serial algorithm or initializing code for parallel algorithm with initialization. This field is empty for parallel algorithm without initialization.

- **Coupling start time** = %(CouplingMenuEntries.Runtime.time cbeg)
  - value = time in seconds or empty if the latest time of the model should be taken (only set for transient analysis).

- **Total coupling time** = %(mpcciserver:CouplingMenuEntries.CouplingDuration.TotalTime)
  - value = time in seconds or empty if no time is specified in the MpCCI GUI (only set for transient analysis).

- **Coupling start iteration** = %(CouplingMenuEntries.Runtime.iter cbeg)
  - value = iteration number for coupling start or empty if the latest iteration of the model should be taken (only set for steady state analysis).

- **Maximum number of coupling steps** = %(mpcciserver:CouplingMenuEntries.CouplingDuration.MaxSteps)
  - value = a number >= 1 (only set for steady state analysis).

- **Number of iterations after coupling** = %(CouplingMenuEntries.Runtime.iter pend)
  - value = a number >= 0 (only set for steady state analysis).

- **Type of coupling step size** = %(mpcciserver:CouplingMenuEntries.CouplingSteps.StepSizeType)
  - value = Defined by each code, Sent by code, Constant or Negotiation.
Coupling step size = %($\text{CouplingMenuEntries.CouplingSteps.StepSize}$)
value = a number $\geq 1$ or a string list with variable step sizes (see \(\text{V-4.5.2 Code Specific Algorithm Settings} \)).

Maximum number of coupling step iterations = %($\text{mpcciserver:CouplingMenuEntries.CouplingDuration.MaxIterSteps}$)
value = a number $\geq 1$ (only set for implicit coupling scheme).

Number of solver's inner iterations = %($\text{CouplingMenuEntries.CouplingSteps.InnerIters}$)
value = a number $\geq 1$ (only set for implicit coupling scheme).

$$\text{MPCCI_TINFO_DELAY} = \text{DelayUnit : DelayValue : }$$

with these references and values:

DelayUnit = %($\text{mpcciserver:CouplingMenuEntries.CouplingSteps.IdleTimeReduction.DelayedUpdate.DelayUnit}$)
value = Number of substeps or Number of update tries.

value = a number $\geq 1$ or a string list with variable step sizes (see \(\text{V-4.5.1 Common Basic Algorithm Settings} \)).

### 2.4.9 General MpCCI GUI Elements

There is a choice of MpCCI GUI elements, which can be used in "gui.xcf" within <ModelsMenuEntries> or <GoMenuEntries>. For each element applies:

- **Identifier** is a unique identifier for that element. It can be arbitrary but should not begin with a number and it should not contain special characters.
- **type** defines the type for the special element.
- **description** is always used as title for the respective element.

#### Text Field

```xml
<Identifier type="string" default="mpccirun"
         description="Job name prefix for job files"/>
```

default is the default value of the text. It may contain references.

![Figure 9: Text field representation](image)

#### File Selector

```xml
<Identifier type="filename" default="" dironly="false"
           description="Data file (optional)">
   <filename value=".suffix1"/>
   <filename value=".suffix2"/>
</Identifier>
```

default should be left empty.
dironly optional attribute which can be set to true when the selection of a directory is desired.
value is one accepted and selectable file suffix in addition to the predefined selectable "All Files" option.
Figure 10: File selector representation

**Enumeration**

```
<Identifier type="enum" default="value" description="Select value" sort="true"
expandable="false"/>
```

default is the default value and will be added to the begin of the enumeration value list if it doesn’t exist.
sort optional attribute which can be set to false if the enumeration value list shall be presented in the given order. If omitted or set to true the values of the list will be sorted in an ascending alphabetical order.
expandable optional attribute which can be set to true if the enumeration value list shall be expanded when attempting to set a value which is not in the list (e.g. opening a project with an actually not supported release). If set to true the value will be added to the list. If this attribute is omitted or set to false setting a value which is not in the list will not be done (the default value is set instead) and leads to an error message.

There are three ways for getting the enumeration values:

1. List them in a value list.
2. Specify a local file as source which holds the values line by line.
3. Specify a command which will be executed at runtime when selecting this value for the first time. The command will provide the values line by line to standard output.
1. Value list:

```xml
<Identifier type="enum" default="SI" description="Select unit system">
  <enum value="British"/>
  <enum value="cgs"/>
  <enum value="mm-t-s"/>
  <enum value="US-ft-lbf-s"/>
  <enum value="US-in-lbf-s"/>
  <enum value="variable"/>
</Identifier>
```

e_num_value_ is one value of the value list.

2. Source file:

```xml
<Identifier type="enum" default="typeA" description="Title" source="fileWithEnumValues"/>
```

source is the name of the file which holds the value list. The file will be looked for in the current user directory and in the user home directory. Each line in the source file corresponds to an enum value in the value list.

3. Command:

```xml
<Identifier type="enum" default="latest" description="Title" command="command with arguments and %(reference)" hostref="%(ModelsMenuEntries.ModelFile)"/>
```

command is the command which will be executed to get the value list. References as described below are allowed to be used.

hostref is an optional specification of the host on which the command shall be executed. Therefore the referenced object has to be a file. The host of this file will be used to execute the command. If no hostref is specified the command is executed on the local host.

**Range Value**

There are two types of range values. You may create a range value of

- integer values, in that case you have to use the type int.
- floating values, in that case you have to use the type float.

```xml
<Identifier type="int" default="1" min="1" max="512" description="No. of parallel processes"/>
<Identifier type="float" default="0.01"
```
**Checkbox**

```xml
<Identifier type="bool" default="true"
    description="Start additional control process"/>
```

**default** must be initialized with true or false.

![Run in batch mode](image)

**Command**

A command is an element which executes a defined command. It is represented by a button which has to be click to execute the command.

```xml
<Identifier type="command" default="cmd arg1 -option %(ModelsMenuEntries.Release)"
    description="Decompose case"
    hostref="%(ModelsMenuEntries.ModelFile)"
    addComponentEnvironment="true"/>
```

**default** is the command to be executed with its arguments. References to other elements are allowed and will be resolved.

**hostref** is an optional specification of the host on which the command shall be executed. Therefore the referenced object has to be a file. The host of this file will be used to execute the command. If no **hostref** is specified the command is executed on the local host.

**addComponentEnvironment** is an optional specification whether the environment with the components and quantities (as used for the starter) shall be set or not. If no **addComponentEnvironment** is specified the environment won’t be set.

![Decompose case](image)
Hostlist

A hostlist is an element which checks the list of host names. It verifies the resolvability of the host and if it is alive. Whitespace, comma or semicolon may be used as delimiters to define the list of host names. A host name may be given as “[user@]host”.

```xml
<Identifier type="hostlist" default="" description="Optional 'host host ...' to be used"/>
```

default is the initial default value and may be left empty.

Figure 15: Hostlist representation

Panel

A panel is used to group elements by surrounding them with a border. There exist two types of panels:

- Enabling or disabling a special feature with its subelements.
- Grouping related elements.

**Enabling or disabling a special feature with its subelements**

The panel consists of a checkbox - named by the description of the feature - which indicates whether this special feature is used or not. If the checkbox is set, the panel expands and shows its subelements. Now these subelements can be set and will be evaluated. If the checkbox is unset, the panel will disappear and hide its subelements.

```xml
<Identifier type="panel" default="false" description="Run parallel">
   ...
   ...
</Identifier>
```

default must be initialized with true or false.

Figure 16: Panel representation for unused and used feature

**Grouping related elements**

The panel consists of several subelements and has no default value. Its description will be shown at the top of the panel. The panel can be opened and closed by clicking on its description. The subelements will be evaluated even when the panel is closed.

```xml
<Identifier type="panel" description="Setting for remote server start">
   ...
   ...
</Identifier>
```
description is used as heading of this panel.

Figure 17: Panel representation for a closed and opened element group

Common Feature: References

Sometimes it is necessary for a value to reference another value. Examples are:

- Setting environment variables for a code (e.g. Go Step: <GoMenuEntries>, Environments for Scanner, Checker, Starter, Stopper and Killer)
- Referencing a host when executing a command like in Enumeration or Command.
- Using the Dependency feature.
- Referencing just another value which e.g. acts as a parameter for a command like in Enumeration or Command.

The following references are possible:

- \%(<element>.<subelement>)\ value of an element specified within this configuration file.
- \%(mpcciserver:<element>.<subelement>)\ the same as before but the element will be taken from the MpCCI server configuration file.
- \%(properties:<element>.<subelement>)\ value of a control parameter which can be edited in the Settings step of the MpCCI GUI.
- \%(#codename)\ references the instance name of this code which is also used in the MpCCI GUI to specify this code.
- $(<environment variable name>)\ value of the specified environment variable on the local machine.

Common Feature: Dependency

For each of the previous elements a dependency may be added. This means that an element is only shown in the MpCCI GUI if the set dependency is complied.
dependingCompare="<startsWith | exact | exactIgnoreCase>"
dependingCondition="<true | false>"/>

dependsOn

dependingValue

dependingCompare

is a reference to the element this element depends on.

is the value the referenced value will be compared to. If this element depends on more than one value of the referenced element this depending value may be a list of values separated by a "|" (i.e.: dependingValue="value 1 | value 5")

provides the compare method used for the comparison between the depending value and the referenced value. Possible values are:

- **startsWith** Checks whether the referenced value starts with the depending value.
- **exact** Checks whether the referenced value matches the depending value case-sensitive.
- **exactIgnoreCase** Checks whether the referenced value matches the depending value case-insensitive. This is default used if no compare method is specified.

dependingCondition

provides the condition for the comparison between the depending value and the referenced value. If the condition is **true** this depending element is only shown if the compared values (resp. one of the compared values if the depending value is a list of values) are equal. On **false** this element is shown if the compared values (resp. all of the compared values if the depending value is a list of values) are not equal. The default condition **true** is taken when the dependingCondition is omitted.

If the dependency is used in a panel the panel cannot be set or unset by the user anymore because this will be done automatically in reliance on its dependency then.

An example can be viewed in the template which you unpacked in your code subdirectory. There the element **GridLengthUnit** in the **<ModelsMenuEntries>** block depends on the **Units** element of the same block. The **GridLengthUnit** element is only shown if the value of **Units** is set to **"variable"**.

**Common Option: Required**

Another option which is common to all previously described MpCCI GUI elements is the required option.

```xml
<Identifier type="<element type>" default="" description="Title"
required="true | false" />
```

required states that this element must be set before going on with the next step in the MpCCI GUI.

Normally the default value for the element is left empty so that the user must choose a value explicitly e.g. as with the **<ModelFile>**. In MpCCI GUI required elements are marked with a (*) at the end of the title. A dialog box is shown if required elements are not set while going on with the next step. The default for the required option is **false**.

![Figure 18: Required representation for a file selector element](image-url)
Common Options: Info and InfoId

Another option which is common to all previously described MpCCI GUI elements is the info option.

```xml
<Identifier type="<element type>" default="" description="Title"
    info="Some information about the current element." |
    infoId="Info.ipolOrder" />
```

info contains a short description of the element. Line breaks can be given by "\n" whereas other formatting instructions are not supported.

infoId contains a link to a short description of the element. The content of the link is looked up in the MpCCI GUI configuration properties file "$MPCCI_HOME/gui/mpcciBundle_en.properties". Here some HTML formatting can be added e.g. for generating lists, paragraphs or formatted text. The information tags in this file are primary for multiple used tags, so that they don’t need to be written down several times. You may use these predefined tags in your own configuration file.

You may either use info or infoId. If both options are used in an element the text for the info is appended to the text of infoId.

In MpCCI GUI info elements are marked with an 📩 symbol at the end of the title. A dialog box with the information message is shown by clicking on the information symbol. By default no information is assigned.

![Figure 19: Info representation with its dialog](image)

Common Option: Visible

Another option which is common to all previously described MpCCI GUI elements is the visible option.

```xml
<Identifier type="<element type>" default="" description="Title"
    visible="true | false" />
```

visible states whether this element is visible or hidden in the MpCCI GUI. This option is very rarely used and only makes sense for special needs. The default for the visible option is true.
2.4.10 Testing "gui.xcf"

Testing of the files in the configuration directory is described in 2.2.2 Step-by-Step Procedure for Code Integration. 
2.5 Perl Scripts

You need not really learn Perl to write the scripts to run with your code. Basic documentation on the language (Linux/UNIX: man perl) is included in the Perl distributions, Schwartz et al. [2005] is recommended as a book for beginners.

So just edit the provided templates.

2.5.1 Using Information from "gui.xcf" in Scripts

As mentioned in the description of "gui.xcf" you need to pass information from the MpCCI GUI to the scanner, checker, starter, stopper and killer scripts. This information is exported from the MpCCI GUI into the Perl world via environment variables. These variables need to be defined in the "gui.xcf" as described in 2.4.8 Environments for Scanner, Checker, Starter, Stopper and Killer  and are then evaluated within the Perl scripts. As a Perl programmer you can access the environment variables via the global Perl hash %ENV

$value = $ENV{'VARIABLE_NAME'};

During testing it may be helpful to read the variables with an automatic error check. For this purpose some Perl routines are provided in a Perl module MpCCI::Env which needs to be included (use or require) in your Perl scripts.

- env_optional_value(<variable name>) gets value of a variable, returns an empty string if not defined.
- env_required_value(<variable name>) same as above, but exits with an error if variable is not defined.
- env_boolean_value(<variable name>) interprets value of the variable as Boolean value, the variable must be one of t|true or 1 to yield “true” and f|false or 0 to yield “false”.
- env_optional_int(<variable name>, <default>, [<min>, [<max>]]) gets value of a variable as an integer value. If the variable is not defined, the given default value is returned. If <min> and <max> are given, it is checked if the value lies within the given limits.
- env_required_int(<variable name>, [<min>, [<max>]]) Same as above, but the variable must be defined, i.e. a default value is not needed.

2.5.2 "Scanner.pm"

This file contains one subroutine which is called by MpCCI to scan the model file, which is named sub code_scanner($$$). It has three arguments:

- $codename The name of your code.
- $modelFile The name of the model file which shall be scanned.
- $tmpName The name of a temporary file, which can be used during the scanning process if necessary.

Sometimes the scanner does not only retrieve information but also applies some changes to the model file.

It returns up to four hashes (i.e. associative arrays):

- $regionList contains all possible coupling components. The hash with the $regionName as key has the following structure:

  $regionList{$regionName} = [$regionName, $regionType, $regionId];
  $regionName = string with the name of the component for later use in adapter
$regionType = type of the region (references to Component Types in gui.xcf)
$regionID = number with ID of the component for later use in the adapter

See §2.4.4 Component Types: `<ComponentTypeDimensions>` for definition of the dimension of
$regionType (Typename) in "gui.xcf".

`%modelInfo` contains some general information on the model file and has the following structure:

| MDIM => '3', # Model dimension: 1,2,3 |
| CSYS => 'C', # Coordinate system: C=Cartesian, S=Spherical, A=axis symmetric |
| SOLU => 'S', # Solution type: S=Static, T=Transient, ?=undefined |
| LCAS => '?', # Load cases: Number of loaded cases |
| UNIT => '?', # Unit system |
| PREC => 'D' # Precision: S=Single precision (32 bit), D=Double precision (64 bit), L=Long double precision (128 bit) |

`%userInfo` is optional and contains some additional scanner information on the model file. This additional scanner information will be printed as comment into the scanner output. For example:

| Global variables => 'Autogenerated by the scanner' |
| Solver type => 'Explicit' |

Sometimes the scanner has to add some components generated from the given ones (e.g. in case of shell elements with front and backside). Therefor the new components get a name composed of the old one and an extra prefix or suffix, sometimes in combination with a special region type or component id. When MpCCI GUI looks for components matching by their name it makes sense to compare to the original name and not to the newly generated which would also consider the prefix and suffix. The key `Generated Components` provides a list of specifications for those automatically generated coupling components which enables the MpCCI GUI to reconstruct the name of the original component. Following attributes may be used by the scanner to specify the automatically generated components:

- **prefix**: Specifies the pattern for the prefix added.
- **suffix**: Specifies the pattern for the suffix added.
- **type**: Specifies a list of region types to be worth considering.
- **id**: Specifies the special ids by either a list of possible ids separated by blank or the keyword `negative` meaning any id < 0.

The attributes may be used in combination or alone where at least one of `prefix` or `suffix` must be present. They will be analyzed top down so the more general specifications should appear at the end of the list. For example:

| Generated Components => 'prefix="CopyOf-" type="Wall Surface" id="101 102 103",' |
| 'prefix="V-1_" suffix="-backside" id="negative",' |
| 'suffix="-backside" id="negative"' |

`%varHash` is optional and contains values of variables to be set into the MpCCI GUI configuration file "gui.xcf”. They will be printed as comment into the scanner output file which will be interpreted by the MpCCI GUI. The name of the variable is built as a dot separated xml tag path. For example:

ScannerInfo.SolverType => 'Explicit', # set used solver type.

# set default for maximum number of iterations:
GoMenuEntries.MpCCITinfo.IterationControl.iter_max => '300'

The "Scanner.pm" consists of three steps:
1. Parsing the model file. The file is opened, and a loop is run over all lines of the file. In the loop the file is scanned for coupling components. The search process must be adapted to your model file format to get the name, type and id of the component. A numerical id can also be omitted (specify \( 0 \)) if your file does not use ids.

2. Adding further variables, which are not contained in the model file, but always present. Usually this is only the case for global variables.

3. Defining the basic model information, which can be retrieved during the scan or given as fixed values. The scanner may be tested by using the command `mpcci <codename> scan modelfile`.

2.5.3 "Checker.pm"

"Checker.pm" is called by MpCCI to check the code configuration. It is only required if a checker environment in the code configuration file "gui.xcf" exists (see \( \geq 2.4.8 \) Environments for Scanner, Checker, Starter, Stopper and Killer). This Perl module contains one subroutine named `sub code_checker($$@)` It has two required arguments and one optional:

- `$codename` | The name of your code.
- `$modelFile` | The name of the model file assigned to your code.
- `@optional` | A place marker for possibly coming up arguments in the future.

The script may e.g. check that the end time or iteration of the coupling lies after the start or that the number of coupling steps or subiterations are plausible. Remember that every information needed to do the checking must be transported via environment variables.

Each result of the configuration check shall be printed out depending on its type. Use

- `checkInfo` for writing it to `stdout` if it is an information message,
- `checkWarn` for writing it to `stderr` if it is a warning message,
- `checkErr` for writing it to `stderr` if it is an error message and use

`checkSummaryEntry($text,$value)` for adding information to the table of coupling configuration summary entries.

The subroutine should return an array with the number of errors and warnings.

These messages will be gathered by MpCCI GUI and shown in one single dialog including all results from the Check action. The type of the result depends on the number of errors and warnings returned.

This script should return true if the check could be done, false if something went wrong while executing this checker script.

2.5.4 "Starter.pm"

"Starter.pm" contains a subroutine which does not actually start the code, but provides the necessary information. The starter function is named `sub code_starter($$@)` and obtains the arguments,

- `$codename` | The name of your code.
- `$modelFile` | The name of the model file assigned to your code.
- `@optional` | A place marker for possibly coming up arguments in the future.

As return values it must provide the command line arguments for starting the code and put it into the variable `@arg`, which is the first return value.
Additionally, \%infoHash is required to be returned to determine how the output of the simulation code is handled. The following options can be chosen:

<table>
<thead>
<tr>
<th>option</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STDOUT</td>
<td>null</td>
<td>What to do with data written to stdout, can be any of xterm (i.e. write to an xterm), mpcci (i.e. output handled by MpCCI), null (ignore output), combined with the keyword file if it shall also be written to a log file.</td>
</tr>
<tr>
<td>STDERR</td>
<td>null</td>
<td>Same as STDOUT, but for stderr.</td>
</tr>
<tr>
<td>STDLOG</td>
<td>&lt;empty string&gt;</td>
<td>Name of logfile</td>
</tr>
<tr>
<td>BACKGR</td>
<td>white</td>
<td>Background color of xterm.</td>
</tr>
<tr>
<td>FOREGR</td>
<td>black</td>
<td>Foreground color of xterm</td>
</tr>
<tr>
<td>ATSTART</td>
<td>&lt;not set&gt;</td>
<td>Reference to a Perl subroutine which is called before the code starts. You can define this routine in &quot;Starter.pm&quot;.</td>
</tr>
<tr>
<td>ATEXIT</td>
<td>&lt;not set&gt;</td>
<td>Reference to a Perl subroutine which is called as soon as the code exits. You can define this routine in &quot;Starter.pm&quot;.</td>
</tr>
</tbody>
</table>

A test for the starter is provided in "test_Starter.pl" of the MpCCI API Kit.

### 2.5.5 "Stopper.pm"

The stopper script should cause the code to perform a regular exit, i.e. not simply kill the code, but save data and quit. This is often achieved with a stop file, which is read by the code.

"Stopper.pm" contains a subroutine named \sub code_stopper($$$)\ and obtains the three arguments:

- \$codename\ The name of your code.
- \$modelFile\ The name of the model file assigned to your code.
- \$mpcciProcessId\ The process identification of the code.

As for the starter, a testing script is provided for "Stopper.pm", which is named "test_Stopper.pl".

### 2.5.6 "Killer.pm"

The killer script should kill the code with all its sub processes and clean up all files not needed anymore. "Killer.pm" contains a subroutine named \sub code_killer($$$)\ and obtains the three arguments:

- \$codename\ The name of your code.
- \$modelFile\ The name of the model file assigned to your code.
- \$mpcciProcessId\ The process identification of the code.

### 2.5.7 "Info.pm"

In the Perl module "Info.pm" the subroutine \sub code_information($)\ is defined which collects application specific information. It has one argument:

- \$codename\ The name of your code.

This subroutine will be called by MpCCI, e.g. when \sub code_print_releases\ or \sub code_print_info\ is called in "Subcmd.pm".

It finds out the installed releases for the code and returns a hash table \%infoHash\ (i.e. an associative array) with one entry for each release token. The array structure for each entry is:
1. Name
   The official name of the code.
2. Home
   The home path of the code installation.
3. Executable
   The full pathname of the executable to start the application.
4. Release
   The code internal release token.
5. Architecture
   The architecture token of the application.
6. Adapter release
   The adapter release token (should be equal to the code internal release token or 'STATIC').
7. Adapter architecture
   The adapter architecture token (should be equal to the architecture token of the application).

Example:

```perl
$infoHash{$release} = [  $codeName='solverxy',  $codeDirectory='/opt/code',  $pathToExecutable='$codeDirectory/bin/code.exe',  $release='1.2.4',  $architecture='linux_x86',  $adapterRelease=$release,  $arch=$architecture ];
```

2.5.8 "Subcmd.pm"

In this Perl module exactly one public subroutine `sub code_subcommand($)` is defined. This subroutine is called from the MpCCI command as a code specific subcommand like `mpcci <codename> releases`.

This file is optional, but it is recommended to use it! The subroutine has one argument:

```perl
$codename
```

The name of your code.

You need to inspect the global Perl array @ARGV and then make decisions based on its contents. The simplest way of implementing a command line parser is to use the parsing tools which come with MpCCI. You just define a Perl hash `%hash3` with the command line options as the key. The value for each command line option is a reference to an array containing three entries:

- The MpCCI environment setup level: 0=no, 1=arch 2=core, 3=gui
- A short help message
- The name of an environment variable or a code reference

The Perl subroutine `sub code_subcommand($)` then has to call `hash3ref_run('option',%hash3)` to process the hash. `hash3ref_run` then either prints the help text, displays the value of the environment variable or calls the subroutine specified via the code reference.

The returned value of `sub code_subcommand($)` is irrelevant. `sub code_subcommand($)` may either call exit(0) or exit(1) in case of a failure or simply return.

2.5.9 Testing the Perl Scripts

For some scripts, a "test.*.pl" script is included in the MpCCI API Kit, which can be used to test the scripts separately. Testing of the scripts in the configuration directory is described in § 2.2.2 Step-by-Step Procedure for Code Integration. The other scripts like "Info.pm" or "Subcmd.pm" may be tested with the `mpcci <code name>` subcommands as described in § VI-1.1 Common MpCCI Subcommands for Simulation Codes.
2.6 MpCCI Adapter Implementation

2.6.1 How to Initialize the Code?

The code has an `initcoupling()` method available to implement some stuff to initialize the code. At this level the information about the mesh should be available. The following call sequence must be implemented in the `initcoupling()`:

1. `umpcci_msgfuncts (▷ 2.7.1 Definition of Output Functions: umpcci_msgfuncts ◁)`
2. `umpcci_msgprefix (▷ 2.7.2 Definition of Output Prefix: umpcci_msgprefix ◁)`
3. `ampcci_tinfo_init (▷ 2.7.3 Get Transfer Information: ampcci_tinfo_init ◁). After calling this function the code must update the current time and / or iteration information from the `MPCCI_TINFO` structure.
4. A `MPCCI_CINFO` data structure (▷ 2.9.7 Code Specific Information: MPCCI_CINFO ◁) must be defined. The current time and iteration of the code should be set.
   - For a steady state iterative code it is recommended to set the time information to the value -1 and just update the iteration field. The same rule must be applied to the `MPCCI_TINFO` data structure.
   - For a transient code it is recommended to set the time information and optionally the iteration field could be used. If this iteration information is set to the value -1, this designs the code to be an explicit transient code. Otherwise this notifies an implicit transient code and enables the code to request data inside the same time step for different iteration steps.
5. `mpcci_cinfo_init (▷ 2.7.4 Initialize the Code Information Structure ◁) initializes the job settings according to the `MPCCI_TINFO`.
6. `mpcci_init (▷ 2.7.5 Connect and Initialize an MpCCI Server: mpcci_init ◁).`
7. `ampcci_config (▷ 2.7.6 Configure the Code Adapter: ampcci_config ◁) The list of driver functions must be passed to ampcci_config at this point.
2.6.2 How to Define the Mesh?

There are two ways to define the mesh to MpCCI.

1. First method: you can implement the `MpCCI.Driver_definePart` driver function,
2. Second method: you implement the `MpCCI.Driver_partInfo`, `MpCCI.Driver_getNodes` and `MpCCI.Driver_getElems` driver functions.

One of this method will be used by the coupling manager. The driver functions are described in $\triangledown$ 2.8.3 Driver Mesh Definition Methods $\triangledown$.

**First method**

The code manages the memory for allocating the array for storing the coordinates, node ids, element ids, etc.

These functions have to be used for implementing the `MpCCI.Driver_definePart`:

- `smpcci_defp` ($\triangledown$ 2.7.7 Definition of Part: `smpcci_defp` $\triangledown$)
- `smpcci_delp` ($\triangledown$ 2.7.8 Delete a Part: `smpcci_delp` $\triangledown$)
- `smpcci_pnod` ($\triangledown$ 2.7.9 Definition of Nodes: `smpcci_pnod` $\triangledown$)
- `smpcci_pels` ($\triangledown$ 2.7.10 Definition of Elements: `smpcci_pels` $\triangledown$)
- `smpcci_pmot` ($\triangledown$ 2.7.11 Definition of the Moving Reference Frame: `smpcci_pmot` $\triangledown$)
- `smpcci_pshf` ($\triangledown$ 2.7.12 Definition of the Baffle Thickness: `smpcci_pshf` $\triangledown$)

In this driver function, you may check the number of elements for the coupled part and delete it from the MpCCI server if no elements are available. This may be necessary in case of a parallel code. If elements are available, define the part, optionally define the moving reference frame (MRF) and then define the list of nodes and elements (see Figure 21).
Second method

The memory allocation to store the node coordinates, node ids, element ids, etc. is done by the coupling manager.

The code needs to implement this minimal set of driver functions:

- **MpCCI_Driver_partInfo** should provide the basic information about the couple part like the number of nodes, elements, coordinates system etc.

- **MpCCI_Driver_getNodes** should provide the node coordinates.

- **MpCCI_Driver_getElems** should define the element topology for the part.

### 2.6.3 How to Deal with Angular Coordinates?

There are many different formalisms to express rotations. If you intent to couple AngularCoordinates with a partner code, MpCCI must know which formalism is used to correctly interpret the received values. You can provide this information to MpCCI in the function **MpCCI_Driver_partUpdate** by adding a flag to your quantity with the line of C-code

```
quant->flags |= CONVENTION;
```

where **CONVENTION** is one of the following.

- **MPCCI_QFLAG_AC_QUATERNION**: MpCCI expects four values that represent a unimodular quaternion. The last value is assumed to be the real part.

Figure 21: Call sequence for mesh definition using **MpCCI_Driver_definePart**
- **MPCCI_QFLAG_AC_AXIS**: MpCCI expects three values that represent an axis of rotation. The magnitude of this axis corresponds to the angle in radians.
- **MPCCI_QFLAG_AC_AXISANGLE**: MpCCI expects four values, of which the first three represent a normalized axis of rotation. The forth value is the angle of rotation in radians.
- **MPCCI_QFLAG_AC_EULER321**: MpCCI expects three values and interprets them as intrinsic Tait-Bryan-angles in z,y,x convention, i.e. as yaw, pitch and roll.

### 2.6.4 How to Transfer Data?

The code has an `dotransfer()` method available to implement a call to the MpCCI data transfer function `ampcci_transfer` (§2.7.13 Data Exchange: `ampcci_transfer`). This basic sequence must be done before calling the transfer function:

1. First the code must update the `MPCCI_TINFO` data structure (See §2.9.6 Transfer Information: `MPCCI_TINFO`). The current time, iteration must be set.
2. then the call of `ampcci_transfer` could be performed.

### 2.6.5 How to Terminate the Coupling?

The code has an `exitcoupling()` method available to implement a call to the MpCCI exit function `mpcci_quit` or `mpcci_stop` (§2.7.15 End of Coupled Simulation: `mpcci_quit` and `mpcci_stop`).

### 2.6.6 How to Notify a Remeshing?

If the code has the capability to remesh the grid, it could use the `ampcci_remesh` (§2.7.14 Notifying the Remeshing: `ampcci_remesh`) routine to notify a remesh event by setting the remesh flag information (see §2.9.5 Remesh Flag Information). Otherwise the coupling manager could check if a remeshing action occurred by calling the driver function `MpCCI_Driver_getPartRemeshState` (see §2.8.2 Driver Methods Called Before/After Some Action). The code should implement this function and return the corresponding remesh state. In case of grid morphing the code should implement the driver function `MpCCI_Driver_moveNodes` (see §2.8.3 Driver Mesh Definition Methods).
2.7 MpCCI Coupling Manager Functions

The coupling manager functions are called by the calling code. They are defined in some MpCCI header files and you should add `#include "mpcci.h"` to the source files, from which you call the coupling manager functions.

Example calls are given in "adapter.c" in the MpCCI API Kit.

The coupling manager functions are:

- 2.7.1 Definition of Output Functions: `umpcci_msg_funcs` on page 50
- 2.7.2 Definition of Output Prefix: `umpcci_msg_prefix` on page 51
- 2.7.3 Get Transfer Information: `ampcci_tinfo_init` on page 52
- 2.7.4 Initialize the Code Information Structure: `mpcci_init` on page 53
- 2.7.5 Connect and Initialize an MpCCI Server: `mpcci_init` on page 54
- 2.7.6 Configure the Code Adapter: `ampcci_config` on page 54
- 2.7.7 Definition of Part: `smpcci_defp` on page 55
- 2.7.8 Delete a Part: `smpcci_delp` on page 57
- 2.7.9 Definition of Nodes: `smpcci_pnod` on page 58
- 2.7.10 Definition of Elements: `smpcci_pels` on page 59
- 2.7.11 Definition of the Moving Reference Frame: `smpcci_pmot` on page 61
- 2.7.12 Definition of the Baffle Thickness: `smpcci_pshf` on page 62
- 2.7.13 Data Exchange: `ampcci_transfer` on page 63
- 2.7.14 Notifying the Remeshing: `ampcci_remesh` on page 65
- 2.7.15 End of Coupled Simulation: `mpcci_quit` and `mpcci_stop` on page 66

Meaning of the function naming convention:

- `umpcci_*` represent MpCCI utility functions.
- `smpcci_*` functions are low level single server communication functions.
- `mpcci_*` functions are designed for multi server communication.
- `ampcci_*` are auxiliary adapter level functions.
2.7.1 Definition of Output Functions: `umpcci_msg_functs`

```c
void umpcci_msg_functs( void (*printFullMessage)(const char *str, int len),
                        void (*printFullWarning)(const char *str, int len),
                        void (*printFullFatal)(const char *str, int len),
                        void (*printLineMessage)(const char *str, int len),
                        void (*printLineWarning)(const char *str, int len),
                        void (*printLineFatal)(const char *str, int len),
                        void (*atExitHandler)(void));
```

Before calling any other coupling manager function, the functions for input and output should be defined. There are seven different functions, the three first functions get multi-line strings for ordinary output messages, warnings and fatal messages together with the string length.

The second set of functions only receives one line at a time, i.e. the strings contain no newline characters (line-printer mode), which is useful for calling from FORTRAN.

It is not necessary to define all functions (give NULL pointers instead), but you should define either the first set of three functions or the second set of three functions.

⚠️ Do not use `printf` for output as the strings are already formatted and may contain percent (%) signs!

### 2.7.1.1 Description Values

- **void printFullMessage(const char *str, int len)**
  For printing a message, typically use a pop-up window or print to log file.

- **void printFullWarning(const char *str, int len)**
  Called for printing of warnings.

- **void printFullFatal(const char *str, int len)**
  Called for printing of fatal errors, i.e. the program will be terminated by the code adapter after this call. Please do not include any cleaning up into this routine, only the printing. See also `atExitHandler` below.

- **void printLineMessage(const char *str, int len)**
  Line print for ordinary messages.

- **void printLineWarning(const char *str, int len)**
  Line print for warnings.

- **void printLineFatal(const char *str, int len)**
  Line print for fatal errors.

- **void atExitHandler(void)**
  This method is called before the code adapter terminates the code in case of fatal errors. You can use this method for cleaning up.
2.7.2 Definition of Output Prefix: `umpcci_msg_prefix`

This function is used to set a prefix for the coupled simulation. You may use this function for identifying the output of the running parallel processes for example.

```c
void umpcci_msg_prefix(const char *prefix);
```

The arguments is:
- `const char *prefix` Character string to use for output as prefix.
2.7.3 Get Transfer Information: `ampcci_tinfo_init`

This function is used to determine the transfer information selected in the MpCCI GUI. In the MpCCI GUI for each code the option Coupling configuration is available and provides the access to different options such as:

- the option Initial quantities transfer set to exchange, skip, receive or send.
- the option Send mode set to always, onewait or allwait.
- the option Receive mode set to all, available, any or complete.
- the option Check mode set to none or forward. Whereas forward provides further options to configure the coupling such as:
  - the option Time for starting the coupling,
  - the option Time for ending the coupling,
  - the option Time step size without coupling,
  - the option Iteration No. for starting the coupling,
  - the option Iteration No. for ending the coupling,
  - the option No. of iterations without coupling.

This information is given to the executed code via the environment variable MPCCI_TINFO. Each of the possible values corresponds to a bit mask, which can be retrieved with `ampcci_tinfo_init`.

```c
int ampcci_tinfo_init (MPCCI_TINFO *mpcciTinfo, const char* config);
```

The arguments are:
- `MPCCI_TINFO *mpcciTinfo` Pointer to the MPCCI_TINFO structure for the code. This structure will be initialized.
- `const char* config` Specify another environment variable to read the transfer information definition other than the standard variable MPCCI_TINFO (See \(\text{2.9.6 Transfer Information: MPCCI_TINFO}\)). Provide `NULL` to use the standard environment.

The return value is 1 if the transfer information was successfully set up.

2.7.4 Initialize the Code Information Structure

This function is used to determine the job information selected in the MpCCI GUI: the coupling scheme explicit or implicit, the coupling algorithm serial or parallel.

```c
void mpcci_cinfo_init(MPCCI_CINFO *cinfo, const MPCCI_TINFO *mpcciTinfo);
```
2.7.5 Connect and Initialize an MpCCI Server: \texttt{mpcci_init}

The initialization function must be called from your code before the beginning of the iteration or time loop. This call should establish a connection to the MpCCI server.

\begin{verbatim}
MPCCI_JOB mpcci_init(const char *porthost, const MPCCI_CINFO *cinfo);
\end{verbatim}

The arguments are:
- \texttt{const char *porthost} The broker location \texttt{porthost}, only needed for coupling a code with a multi server configuration, should be \texttt{NULL} in other cases.
- \texttt{MPCCI_CINFO *cinfo} Code information structure. This structure keeps code specific information. (See \ref{2.9.7 Code Specific Information: MPCCI_CINFO}.)

The return value is an \texttt{MPCCI_JOB} value and not null if a connection to the server could be established. This can be used to detect whether a coupled or uncoupled simulation is run.

The initialization routine performs the following tasks:

- Check the code name.
- Check the job id by using the environment \texttt{MPCCI_JOBID} if it is not statically defined by the adapter.
- Check the code id by using the environment \texttt{MPCCI_CODEID} if it is not statically defined by the adapter.
- Define the server host by using the environment \texttt{MPCCI_SERVER} if no broker is defined.
- Connect to the server.
- Get the type of grid to use for the grid definition. See \ref{2.9.7 Code Specific Information: MPCCI_CINFO} for the types.
2.7.6 Configure the Code Adapter: ampcci_config

The code adapter configuration function must be called from your code before the beginning of the iteration or time loop and after having called the initialization function mpcci_init with a valid MPCCI_JOB. However, the geometry of the mesh should already be known, as it is passed to MpCCI at this point.

```c
int ampcci_config(MPCCI_JOB **job, const MPCCI_DRIVER *driver);
```

The arguments are:

- **job** The MPCCI_JOB value returned after calling the mpcci_init containing the connection information to the MpCCI server. This should be passed to this function.
- **driver** Pointer to list of driver functions. See □ 2.8 MpCCI Driver Functions ▷.

The return value is:

- 1 if the code successfully defined the mesh and quantities to the server.
- 0 if the co-simulation runs in pre-check mode. In this case only the mesh and quantities definition are sent to the server for a mesh check with a neighborhood calculation.
- -1 if an error occurs during the mesh and quantities definition to the server.

The adapter configuration routine performs the following tasks:

- Check if the supplied driver functions match the MpCCI version.
- Register driver functions, i.e. the pointers to the function pointers are saved to internal data structures.
- Get the coupled parts and their quantities and all globals from the server, i.e. all information defined in the MpCCI GUI, including the coupling components.
- Call the driver function [MpCCI_Driver_appendParts] if it is defined.
- Loop over all coupling components:
  - Call [MpCCI_Driver_partUpdate] in which the quantity could be checked and the part updated.
  - Call [MpCCI_Driver_partInfo] to get the mesh information, e.g. coordinate system, number of nodes and elements, part id, etc.
- Check the quantity setting.
- Print the list of partner codes.
- Loop over all coupling components:
  - Call [MpCCI_Driver_partSelect].
  - Define the grid:
    * Call [MpCCI_Driver_definePart] if defined (see Figure 22), otherwise
    * Call [MpCCI_Driver_getNodes] and [MpCCI_Driver_getElems], in which the mesh is automatically defined by calling [smpcci_defp], [smpcci_pnod], [smpcci_pels] and calls [MpCCI_Driver_getFaceNormals] and [MpCCI_Driver_getElemSize] if defined (see Figure 23).
  - Define the mesh based and global quantities.
  - List the information about coupled components and quantities.
  - Call the driver function [MpCCI_Driver_afterCloseSetup].
- Return to the calling code.
2.7.7 Definition of Part: `smpcci_defp`

The part definition should be called from the driver function `MpCCI_Driver_definePart` to define the coupling component, see also "adapter.c" of the MpCCI API Kit.

The syntax is:
int smpcci_defp(MPCCI_SERVER *server,
    int       mid,
    int       pid,
    unsigned  csys,
    int       nnodes,
    int       nelems,
    const char *name)

The parameters are:

**MPCCI_SERVER *server**  The MpCCI server pointer passed by the MpCCI_Driver_definePart.

**int mid**  mesh ID, should be set to MPCCI_PART_MSHID(part). The part is provided by MpCCI_Driver_definePart.

**int pid**  part ID, should be set to MPCCI_PART_PARTID(part).

**unsigned csys**  coordinates system of your mesh (see 2.9.2 Coordinates System Definition), should be set to MPCCI_PART_CSYS(part) if this has been defined in MpCCI_Driver_partInfo.

**int nnodes**  number of nodes of the component, can be simply set to MPCCI_PART_NNODES(part).

**int nelems**  number of elements of the component, can be simply set to MPCCI_PART_NELEMS(part).

**const char *name**  name of the component, can be simply set to MPCCI_PART_NAME(part).

It returns 0 for a successful operation.
2.7.8 Delete a Part: smpcci_delp

The part deletion should be called from the driver function `MpCCI_Driver_definePart` to delete the coupling component if the current process does not have any elements to couple.

The syntax is:

```c
int smpcci_delp(MPCCI_SERVER *server,
    int mid,
    int pid);
```

The parameters are:
- `MPCCI_SERVER *server` The MpCCI server pointer passed by the `MpCCI_Driver_definePart`.
- `int mid` mesh ID, should be set to `MPCCI_PART_MESHID(part)`. The part is provided by `MpCCI_Driver_definePart`.
- `int pid` part ID, should be set to `MPCCI_PART_PARTID(part)`.

It returns 0 for a successful operation.
2.7.9 Definition of Nodes: smpcci_pnod

The node definition should be called from the driver function [MpCCI_Driver_definePart] to define the nodes of a coupling component, see also "adapter.c" of the MpCCI API Kit. Only those nodes which send or receive data during the coupling process should be given.

The syntax is:

```c
int smpcci_pnod(MPCCI_SERVER *server,
                int mid,
                int pid,
                unsigned csys,
                int nnodes,
                const void *coords,
                unsigned realsize,
                const int *nodeids,
                const void *angles)
```

The parameters are:

- **MPCCI_SERVER *server** The MpCCI server pointer passed by the [MpCCI_Driver_definePart].
- **int mid** mesh ID, should be set to MPCCI_PART_MESHID(part). The part is provided by [MpCCI_Driver_definePart].
- **int pid** part ID, should be set to MPCCI_PART_PARTID(part).
- **unsigned csys** coordinates system of your mesh (see 2.9.2 Coordinates System Definition), should be set to MPCCI_PART_CSYS(part) if this has been defined in MpCCI_Driver_partInfo.
- **int nnodes** number of nodes of the component, can be simply set to MPCCI_PART_NNODES(part).
- **const void *coords** Array of node coordinates: (x,y) list or (x,y,z) list. This array should be allocated by the code.
- **unsigned realsize** The size of the floating point value written for the node coordinates.
- **const int *nodeids** Array of the node IDs. Optional value. This could be set to NULL.
- **const void *angles** Array of the node angles: (x,y) list or (x,y,z) list. Optional value. This could be set to NULL.

It returns 0 for a successful operation.
2.7.10 Definition of Elements: \texttt{smpcci\_pels}

The element definition should be called from the driver function \texttt{MpCCI\_Driver\_definePart} after the node definition to define the elements of a coupling component, see also “adapter.c” of the \textit{MpCCI API Kit}.

All nodes used in \texttt{smpcci\_pels} must be already defined with \texttt{smpcci\_pnod}.

The syntax is:

```c
int smpcci_pels(MPCCI_SERVER *server,
                   int mid,
                   int pid,
                   int nelems,
                   unsigned eltype1,
                   const unsigned *eltypes,
                   const int *elnodes,
                   const int *elemids)
```

The parameters are:

- \texttt{MPCCI\_SERVER *server} is the \textit{MpCCI} server pointer passed by the \texttt{MpCCI\_Driver\_definePart}.
- \texttt{int mid} is the mesh ID, should be set to \texttt{MPCCI\_PART\_MESHID(part)}.
- \texttt{int pid} is the part ID, should be set to \texttt{MPCCI\_PART\_PARTID(part)}.
- \texttt{int nelems} is the number of elements, can be set to \texttt{MPCCI\_PART\_NELEMS(part)}.
- \texttt{unsigned eltype1} is one element type. Define this element type if all elements use the same element type, otherwise the eltypes array should be defined.
- \texttt{const unsigned *eltypes} is the element types. This can be one of the predefined element types, which are given in \ref{2.9.1 Supported Element Types}. The eltypes array only needs a single element type if all elements use the same type. Otherwise, eltypes should be defined.
- \texttt{const int *elnodes} is an array of the node numbers (your node numbers) of the element nodes. See description below.
- \texttt{const int *elemids} is an array of element IDs. Size should be \texttt{nelems}. Optional value. This could be set to \texttt{NULL}.

It returns 0 for a successful operation.

2.7.10.1 Definition of Element Nodes

The nodes of each element are given in two arrays. The first, \texttt{eltypes}, contains one unsigned integer value for each element, which corresponds to the number of nodes for the element. This array is optional if all the elements have the same type. The nodes for all elements are simply listed in \texttt{elnodes}, the size of this array is therefore difficult to give, but it does not matter if it is too big.

So, if you have a 4-node quad element with nodes 1, 2, 3, 4 and two 3-node triangle elements with nodes 11, 12, 13 and 21, 22, 23, you should define:

```c
int eltypes[] = { MPCCI\_ETYP\_QUAD4, MPCCI\_ETYP\_TRIA3, MPCCI\_ETYP\_TRIA3};
int elemNodes[] = { 1, 2, 3, 4, 11, 12, 13, 21, 22, 23 };
```

It is important to specify the nodes in the correct order! The order which you must use is given in \ref{2.9.1 Supported Element Types}.

2.7.10.2 Definition of Element Types

The element types are listed in \ref{2.9.1 Supported Element Types} and defined in the file "mpcci\_elements.h".

The element types can be given in two ways: Either you have a mesh, which consists of elements of the same type. In this case you can set \texttt{eltype1} and \texttt{eltypes[] = NULL}, i.e. the list of element types only
consists of one type, which is valid for all elements.
If you have a mesh which consists of different types you can give the type for each element. Then size of
\texttt{eltypes} should have the same value as \texttt{nelems} and the argument \texttt{eltype1} should be set to 0.
2.7.11 Definition of the Moving Reference Frame: smpcci_pmot

The moving reference frame should be defined from the driver function `MpCCI_Driver_definePart` before the node definition (§ 2.7.9 Definition of Nodes: smpcci_pnod) or if the driver function `MpCCI_Driver_definePart` is not implemented, the `MpCCI_Driver_closeSetup` should be implemented and call the definition function for the moving reference frame.

You should then use the `MPCCI_SERVER *server` to loop over the coupled part and set the `motion` structure.

The syntax is:

```c
int smpcci_pmot(MPCCI_SERVER *server,
                 int mid,
                 int pid,
                 const MPCCI_MOTION *motion);
```

The parameters are:

- **MPCCI_SERVER *server**
  - The MpCCI server pointer passed by the `MpCCI_Driver_definePart`.

- **int mid**
  - mesh ID, should be set to `MPCCI_PART_MESHID(part)`. The part is provided by `MpCCI_Driver_definePart`.

- **int pid**
  - part ID, should be set to `MPCCI_PART_PARTID(part)`.

- **const MPCCI_MOTION *motion**
  - moving reference frame information (See § 2.9.4 Moving Reference Frame Definition).

It returns 0 for a successful operation, otherwise a number less than 0.
2.7.12 Definition of the Baffle Thickness: `smpcci_pshf`

The baffle thickness definition is an optional function.

This function can be called from the driver function [MpCCI_Driver_definePart] and it should be called after the element definition (>2.7.10 Definition of Elements: `smpcci_pels`).

The syntax is:

```c
int smpcci_pshf(MPCCI_SERVER *server, int mid, int pid, double sval);
```

The parameters are:

- **MPCCI_SERVER *server** The MpCCI server pointer passed by the MpCCI_Driver_definePart.
- **int mid** mesh ID, should be set to MPCCI_PART_MESHID(part). The part is provided by MpCCI_Driver_definePart.
- **int pid** part ID, should be set to MPCCI_PART_PARTID(part)
- **double sval** the thickness of the baffle. The thickness defines how much the current part ID should be shifted on the MpCCI server side.

Usually a baffle will be defined twice for the front and rear side. The call of this function should be done of the side you would like to shift.

It returns 0 for a successful operation.
2.7.13 Data Exchange: `ampcci_transfer`

The transfer function must be called from your code before or after the solver.

```c
int ampcci_transfer(MPCCI_JOB *job, MPCCI_TINFO *tinfo)
```

The arguments are:
- `MPCCI_JOB *job`: The current `MPCCI_JOB` returned during the initialization.
- `MPCCI_TINFO *tinfo`: The transfer information containing the current time, iteration for the quantities transferred and the transfer type to execute.

The return value is:
- `-1`: Error.
- `1`: Transfer done.
- `0`: No transfer occurred.

The transfer information `tinfo` is updated by the call. The code can check if all quantities have been received for comparing for example `tinfo.nquant_req` and `tinfo.nquant_r`, etc.

The transfer routine performs the following tasks:

- Check if the `MPCCI_JOB` is valid and that it is a coupled process.
- Get the first initial transfer action and check if the transfer is valid.
- If the check mode is set to `forward`, check if the transfer should be executed.
- Get the remesh status of each individual part if `MpCCI_Driver_getPartRemeshState` is defined. If remeshing:
  - Call the driver function `MpCCI_Driver_moveNodes` if it does not concern all the parts.
  - Redefine the topology.
- Check if any partner is waiting for a quantity.
- Check if any quantity is available
- Set the transfer information on each server.
- If the code should send:
  - Call the driver function `MpCCI_Driver_beforeGetAndSend`.
  - Loop over all coupling components:
    * Check if the quantity selection mode is matching
      (\$2.9.6 Transfer Information: `MPCCI_TINFO`).
    * Call the driver function `MpCCI_Driver_partSelect` if the component was not selected.
    * Call the corresponding get data driver function according to the coupled dimension.
    * Send the data to the server.
  - Call the driver function `MpCCI_Driver_getGlobValues`
  - Send the global values to the server.
  - Call the driver function `MpCCI_Driver_afterGetAndSend`
- Update the convergence flag `tinfo.conv_code` and check if no error occurred.
- If the code should receive:
  - Call the driver function `MpCCI_Driver_beforeRecvAndPut`.
  - Loop over all coupling components:
    * Check if the quantity selection mode is matching
      (\$2.9.6 Transfer Information: `MPCCI_TINFO`).
    * Call the driver function `MpCCI_Driver_partSelect` if the component was not selected.
    * Receive the data from the server.
    * Call the corresponding put data driver function according to the coupled dimension.
- Receive the global values from the server.
- Call the driver function `MpCCI_Driver_putGlobValues`.
- Call the driver function `MpCCI_Driver_afterRecvAndPut`.

- Save the current job.
- Return to calling code.
2.7.14 Notifying the Remeshing: ampcci_remesh

To notify a remeshing to MpCCI the following function must be called from the code. MpCCI will automatically get the new definition of the mesh.

The syntax is:

```c
int ampcci_remesh(MPCCI_JOB *job, unsigned flags);
```

The parameters are:

- `MPCCI_JOB *job` The MpCCI job pointer created by the `mpcci_init`.
- `unsigned flags` Specify the type of remeshing (See 2.9.5 Remesh Flag Information).

It returns 0 for a successful operation.
2.7.15 End of Coupled Simulation: \texttt{mpcci\_quit} and \texttt{mpcci\_stop}

The code quits and the MpCCI server continues

\begin{verbatim}
int mpcci_quit(MPCCI_JOB **job)
\end{verbatim}

The arguments are:
- MPCCI\_JOB **job: The MpCCI job.

The code stops and the other partners are informed and the MpCCI server stops, too

\begin{verbatim}
int mpcci_stop(MPCCI_JOB **job, const char *message);
\end{verbatim}

The arguments are:
- MPCCI\_JOB **job: the MpCCI job.
- const char *message: error message to print and send before disconnecting the MpCCI server.

These functions always return 0.
2.8 MpCCI Driver Functions

The driver functions are responsible for the exchange of data between the code adapter and the application. To enable a coupled simulation a set of functions must be provided. Exemplary implementations of the necessary driver functions are given in "adapter.c" of the MpCCI API Kit. A description of all functions is given on the following pages.

The driver functions are called by the coupling manager functions as described in 2.7 MpCCI Coupling Manager Functions, therefore they can be regarded as the passive part of the adapter.

A structure of pointers to these functions is handed over to MpCCI via the ampcci_config routine, see also "adapter.c" of the MpCCI API Kit and 2.7.6 Configure the Code Adapter: ampcci_config. The names of the functions can be chosen arbitrarily. Only two functions are required, which must be provided, all other functions need not to be defined, a NULL-pointer can be given instead. The complete structure without NULL pointers is:

```c
struct _MPCCI_DRIVER
{
    /**********************************************************************************
    * REQUIRED: information for compatibility checks
    **********************************************************************************
    unsigned this_size;            /* (unsigned)sizeof(MPCCI_DRIVER) */
    unsigned this_version;         /* value of MPCCI_CCM_VERSION defined in mpcci.h */

    /**********************************************************************************
    * REQUIRED: bitmask for or'ing MPPCCI_TINFO.tact
    **********************************************************************************
    unsigned tact_required;        /* bitmask: MPCCI_TACT_* */

    /**********************************************************************************
    * REQUIRED: define symbolic names for the types of parts in code terminology
    * example:
    *     [0] = "Point(s) with coordinates, single point or point cloud",
    *     [1] = "Line",
    *     [2] = "Surface",
    *     [3] = "Volume domain",
    *     [5] = "Global variable",
    *     [6] = NULL, not yet used
    *     [7] = NULL, not yet used
    **********************************************************************************
    const char *part_description[8];

    /**********************************************************************************
    * REQUIRED: (unsigned)sizeof(real) for all received quantities
    **********************************************************************************
    unsigned realsize;

    /**********************************************************************************
    * OPTIONAL: methods called before/after some actions
    **********************************************************************************
    void (*afterCloseSetup) (void);  /* method called after def_close */
    void (*beforeGetAndSend) (void); /* method called before send operation */
    void (*afterGetAndSend) (void);  /* method called after send operation */
}
```
void (*beforeRecvAndPut)(void); /* method called before receive operation */
void (*afterRecvAndPut)(void); /* method called after receive operation */

void (*partSelect)(const MPCCI_PART *part);
int (*partUpdate)(MPCCI_PART *part, MPCCI_QUANT *quant);
void (*appendParts)(const MPCCI_JOB *job, MPCCI_SERVER *server);
int (*getPartRemeshState)(const MPCCI_PART *part, unsigned flags);

void (*definePart)(MPCCI_SERVER *server, MPCCI_PART *part);

int (*partInfo)(int *rDim,
                unsigned *csys,
                int *nNodes,
                int *nElems,
                int *maxNodesPerElem,
                int *regId,
                const char *regName,
                int len);

int (*getNodes)(const MPCCI_PART *part, unsigned *csys, void *coords, int *nodeIds);
int (*getElems)(const MPCCI_PART *part, unsigned *elemType1, unsigned *elemTypes,
                int *elemNodes, int *elemIds);

int (*moveNodes)(MPCCI_SERVER *server, MPCCI_PART *part);

int (*definePart)(MPCCI_SERVER *server, MPCCI_PART *part);

void (*putPointValues)(const MPCCI_PART *part, const MPCCI_QUANT *quant, void *values);
void (*putLineNodeValues)(const MPCCI_PART *part, const MPCCI_QUANT *quant, void *values);
void (*putLineElemValues)(const MPCCI_PART *part, const MPCCI_QUANT *quant, void *values);
void (*putFaceNodeValues)(const MPCCI_PART *part, const MPCCI_QUANT *quant, void *values);
void (*putFaceElemValues)(const MPCCI_PART *part, const MPCCI_QUANT *quant, void *values);
void (*putVoluNodeValues)(const MPCCI_PART *part, const MPCCI_QUANT *quant, void *values);
void (*putVoluElemValues)(const MPCCI_PART *part, const MPCCI_QUANT *quant, void *values);
void (*putGlobValues)(const MPCCI_GLOB *glob, void *values);

void (*putPointValues)(const MPCCI_PART *part, const MPCCI_QUANT *quant, void *values);
void (*putLineNodeValues)(const MPCCI_PART *part, const MPCCI_QUANT *quant, void *values);
void (*putLineElemValues)(const MPCCI_PART *part, const MPCCI_QUANT *quant, void *values);
void (*putFaceNodeValues)(const MPCCI_PART *part, const MPCCI_QUANT *quant, void *values);
void (*putFaceElemValues)(const MPCCI_PART *part, const MPCCI_QUANT *quant, void *values);
void (*putVoluNodeValues)(const MPCCI_PART *part, const MPCCI_QUANT *quant, void *values);
void (*putVoluElemValues)(const MPCCI_PART *part, const MPCCI_QUANT *quant, void *values);
void (*putGlobValues)(const MPCCI_GLOB *glob, void *values);
2.8.1 Description Values

`size_t this_size`

The size of the structure, must be set to `sizeof(MPCCI_DRIVER)`.

`unsigned this_version`

The version of the code adapter, i.e. the MpCCI version, without dots. Should currently be set to `400` to match this description.

`unsigned tact_required;`

This defines the actions of the code coupling manager. In most cases it should be set to `0`. Other values `{MPCCI_TACT_GETQ|MPCCI_TACT_PUTQ}` should only be used in special case, e.g. codes with several processes, where the driver functions should be always called to avoid locks.

`const char *part_description[8]`

An array of descriptions to improve output. You should give a set of names that match the terminology of your code, e.g.

```
{
"point",    /* name for point */
"edge",    /* name for lines */
"surface", /* name for faces */
"body",    /* name for volumes */
"ipoint",  /* name for integration point */
"global variable" /* name for global quantities */
}
```

`unsigned realsize`

The size of the floating point values given to the put functions, usually set to one of `(unsigned)sizeof(float)` or `(unsigned)sizeof(double)`. 
2.8.2 Driver Methods Called Before/After Some Action

void (*afterCloseSetup)(void)
This method is called at the end of the setup of coupling components.
It is usually not necessary to define this function.

void (*beforeGetAndSend)(void)
This method is called before any get functions are called and data is sent. Define this method if you need
to perform any actions before the get functions can read data.

void (*afterGetAndSend)(void)
This method is called after all required get functions were called and the data was transferred to the MpCCI
server.

void (*beforeRecvAndPut)(void)
This method is called before data is received and before appropriate put functions are called.

void (*afterRecvAndPut)(void)
This method is called after the put functions were called.

void (*partSelect)(const MpCCI PART *part)
You can use this driver function to remember which components were selected in the MpCCI GUI for
coupling. It is called once for each component, which is given in part.
In most codes this function is not needed.

int (*partUpdate)(MpCCI PART *part, MpCCI QUANT *quant)
This function is called during ampcci_config. You should update the components in this function,
namely the number of nodes and elements of each component, the coordinates system. For details please
see "adapter.c" of the MpCCI API Kit. You can use this driver function to allocate memory for the
quantity quant on the component part or check the quantity setting.

Return 0 if this is successful.

If your code supports coupling of AngularCoordinates, you should add a flag to the quantity that
determines the rotational formalism used by your code, see 2.6.3 How to Deal with Angular
Coordinates?<.  

void (*appendParts)(const MpCCI JOB *job, MpCCI_SERVER *server)
This method is intended that coupling components MpCCI_PART can be added automatically and need not
be selected in the MpCCI GUI.

In most codes this function is not needed. All components can be chosen in the MpCCI GUI.

int (*getPartRemeshState)(const MpCCI PART *part, unsigned flags)
This function is called during ampcci_transfer to check the state of the mesh. It returns one of these
values:

- MpCCI REMESH_STATE_NONE : Nothing was done.
- MpCCI REMESH_STATE_REMESHED : The mesh has been remeshed.
- **MPCCI_REMESH_STATE_SMOOTHED**: The mesh has been smoothed.
- **MPCCI_REMESH_STATE_MIGRATED**: Some nodes have migrated to another process.

The argument `flags` is not used in this function.
2.8.3 Driver Mesh Definition Methods

The mesh definition functions are called if information on the meshes is needed. There are two ways to provide information on meshes:

1. The implementation of `definePart`, or
2. The implementation of `partInfo`, `getNodes`, `getElems`

One of them is required.

```c
int (*definePart)(MPCCI_SERVER *server, MPCCI_PART *part)
```

You should define nodes and elements of the coupling component given by `part`:

- Define the moving reference frame with `smpcci_pmot` (see 2.7.11 Definition of the Moving Reference Frame: `smpcci_pmot`).
- Set number of nodes with `smpcci_pnod` (see 2.7.9 Definition of Nodes: `smpcci_pnod`).
- Set number of elements with `smpcci_pels` (see 2.7.10 Definition of Elements: `smpcci_pels`).

In this case the code is responsible to allocate the memory for the nodes and elements.

It returns 0 for successful operation.

```c
void (*partInfo)(int *rDim, unsigned *csys, int *nNodes, int *nElems, int *maxNodesPerElem, int *regId, const char *regName, int len)
```

You should define information on the coupling component given by `regName` or `regId`:

- Set the dimension with `rDim` (see 2.9.3 Mesh Dimension Definition).
- Set the coordinates system with `csys` (see 2.9.2 Coordinates System Definition).
- Set number of nodes with `nNodes`.
- Set number of elements with `nElems`.
- Set the maximum number of nodes per element with `maxNodesPerElem`.

This provides information for the coupling manager for allocating the memory for the mesh definition operation.

```c
int (*getNodes)(const MPCCI_PART *part, unsigned *csys, void *coords, int *nodeIds)
```

You should provide information on the nodes on the coupling component given by `part`:

- `coords`: the array of coordinates of the nodes.
- `nodeIds`: an optional array of node ids. It could be set to NULL.

This function returns the size of the floating point values written in the coordinates, e.g., `sizeof(double)`.

```c
int (*getElems)(const MPCCI_PART *part, unsigned *elemType1, unsigned *elemTypes, int *elemNodes, int *elemIds)
```

You should provide information on the elements on the coupling component given by `part`:

- `elemType1`: element type if all the elements are equal (see 2.9.1 Supported Element Types).
- `elemTypes`: the array of element types if the mesh is built of different element types.
- `elemNodes`: array of the node numbers (your node numbers) of the element nodes.
- `elemIds`: optional array of element IDs. It could be set to NULL.

This function returns the following bit pattern value:
- 0x0000: the \texttt{elemIds} array is not used.
- 0x0001: the \texttt{elemIds} array is valid.

\begin{verbatim}
int (*moveNodes) (MPCCI_SERVER *server, MPCCI_PART *part)
\end{verbatim}

This function is required for moving nodes instead of full remeshing. It is called when the remeshing state is set to \texttt{MPCCI.REMESH.STATE.SMOOTHED}. In that case the code should define the moving nodes by using the function \texttt{smpcci.pnod} (see \texttt{\dagger 2.7.9 Definition of Nodes: smpcci.pnod}).
2.8.4 Driver Data Exchange Methods

There are two kinds of exchange functions. The get functions read data from the code and are called before sending data to the partner code via the MpCCI servers. The put functions are called after data was received and must write the data to the code.

The get functions are all similar and have the same syntax:

```c
int get...Values(const MPCCI_Part *part, const MPCCI_Quant *quant, void *values)
```

The parameters are:
- `const MPCCI_Part *part` pointer to coupling component structure.
- `const MPCCI_Quant *quant` pointer to quantity structure.
- `void *values` pointer to data array.

Return value:
- `int floatSize` size of the floating point values which are written to `values`. Typically this is `sizeof(float)` or `sizeof(double)`.

In the get functions, the required values must be written to the compact data array, which is already allocated. The pointer is given as parameter. You can give the data in any floating point format. All information on the coupling component and the requested quantity are given in `part` and `quant`.

The get functions are distinguished by the component dimension, `Point`, `Line`, `Face` or `Volu`, and by the location, i.e. `Node` or `Elem`. You only need to define those functions, which are supported by your code, i.e. if your code only supports coupling on faces with nodal values, only `MPCCI_DRIVER_getFaceNodeValues` must be defined.

In each function, loop over nodes or elements and collect the values. These values must be written to the array, which `values` points to. The size of the array is given by `MPCCI_Part_NNodes(part) * MPCCI_Part_MDim(part)` for nodal values and `MPCCI_Part_NElems(part) * MPCCI_Quant_DIM(quant)` for element values. `MPCCI_Quant_DIM(quant)` is the dimension of each value, e.g. 3 for a vector.

The order of nodes and elements is given by the order in which they are defined in `[smpcci_pnod]` and `[smpcci_pels]` or in `[getNodes]` and `[getElems]` in `[definePart]` or during the initialization stage.

```c
int getGlobValues(const MPCCI_Glob *glob, void *values)
```

The parameters are:
- `const MPCCI_Glob *glob` pointer to global variable structure.
- `void *values` pointer to data array.

Return value:
- `int floatSize` size of the floating point values which are written to `values`. Typically this is `sizeof(float)` or `sizeof(double)`.

```c
void put...Values(const MPCCI_Part *part, const MPCCI_Quant *quant, void *values)
```

The parameters are the same as for the get functions. The data array is now filled with values which can be read and transferred to the mesh. The values are of the type given as `realsize` in the driver function structure, see → 2.8 MpCCI Driver Functions 4.

```c
int putGlobValues(const MPCCI_Glob *glob, void *values)
```

The parameters are the same as for the `getGlobValues` functions.
2.9 Data Structures and Predefined Macros

MpCCI uses some predefined C preprocessor macros, which can be used in function calls and are described here briefly. Most of the macros described below are defined in the files "<MpCCI_home>/include/mpcci_defs.h" and "<MpCCI_home>/include/mpcci_types.h", which include short descriptions as well.

2.9.1 Supported Element Types

Supported beam elements

| MPCCI_ETYP_LINE2 | MPCCI_ETYP_LINE3 | MPCCI_ETYP_LINE4 |

Table 1: Beam element types in the MpCCI code API
Supported triangle elements

Table 2: Triangle element types in the MpCCI code API
Supported quadrilateral elements

![Diagram of quadrilateral elements](image)

Table 3: Quadrilateral element types in the MpCCI code API
Supported tetrahedral elements

Table 4: Tetrahedral element types in the MpCCI code API
Supported pyramid elements

Table 5: Pyramid element types in the MpCCI code API
Supported wedge elements

Table 6: Wedge element types in the MpCCI code API
Supported hexahedral elements

Table 7: Hexahedral element types in the MpCCI code API
**Supported polygonal face elements**

To define a polygon face element type,

```c
unsigned elementType = MPCCI_ESHP_POLY | N;
```

where `MPCCI_ESHP_POLY` is a bitpattern which defines the shape of the polygonal element type and `N` is the number of vertices resp. vertex ids of the polygon. A polygon has at least three vertices. Currently within MpCCI the max. no. of vertices of a polygonal face is limited to 256. Element types `MPCCI_ETYP_POLY2` or `MPCCI_ETYP_POLY300` are invalid and the MpCCI server would complain. You may also use predefined polygonal element types, for example:

```c
eltype[3] = MPCCI_ETYP_POLY6;
eltype[4] = MPCCI_ETYP_POLY20;
```

![Diagram of a polygon with vertices numbered 0 to 5, labeled MPCCI_ETYP_POLY5 in the center.](image)

Table 8: Polygonal element types in the MpCCI code API
Supported polyhedral elements

To define a general polyhedron element use this syntax for the element:

```
unsigned elementType = MPCCI_ESHP_PCELL | N
```

where [MPCCI_ESHP_PCELL] is a bitpattern which defines the shape of the element and \( N \) corresponds to
the number of faces+1 plus the number of vertices of all faces. In fact \( N \) is the number of integer values
required to describe the general polyhedron. A tetrahedron for example may also be expressed as a general
polyhedron. A tetrahedron is made from four faces with each three vertices:

```
eltype[3] = MPCCI_ESHP_PCELL | (4+1 + 4*3);
```

The elnodes array of a PCELL is organized in two parts:

- **elnodes[0...elnodes[0]-1]** lookup table for faces node ids
  
  Example:
  
  ```
elnodes[0] start index of first vertex of face[0] in elnodes.
...  
elnodes[nface-1] start index of first vertex of last face in elnodes.
elnodes[nface] start index of last vertex +1, just points behind the last vertex of the
....  
```

- **elnodes[elnodes[0]...nelems]**
  
  Example:
  
  ```
elnodes[elnodes[0]] vertex id of first vertex of face 0.
elnodes[elnodes[0]+1] vertex id of second vertex of face 0
elnodes[elnodes[0]+2] vertex id of third vertex of face 0.
...  
elnodes[elnodes[1]] vertex id of first vertex of face 1.
...  
elnodes[0]-1 number of faces in PCELL.
```

Table 9: Polyhedral element types in the MpCCI code API
2.9.2 Coordinates System Definition

The mesh coordinates system should specify one of these values:
- MPCCI_CSYS_C1D: One dimensional Cartesian system.
- MPCCI_CSYS_C2D: Two dimensional Cartesian system.
- MPCCI_CSYS_C3D: Three dimensional Cartesian system.
- MPCCI_CSYS_ARZ: Axis-symmetric with radial and z-axis order definition.
- MPCCI_CSYS_AZR: Axis-symmetric with z-axis and radial order definition.
- MPCCI_CSYS_CYL: Cylindrical coordinated system.
- MPCCI_CSYS_SPH: Spherical coordinated system.

2.9.3 Mesh Dimension Definition

The mesh dimension should specify one of these values:
- MPCCI_MDIM_NONE: No dimension assigned to this component.
- MPCCI_MDIM_NETP: 0 for a network point.
- MPCCI_MDIM_LINE: 1 for line.
- MPCCI_MDIM_FACE: 2 for face.
- MPCCI_MDIM_VOLU: 3 for volume.

2.9.4 Moving Reference Frame Definition

The structure used to transport mesh specific moving reference frame information specifies the following data:

```c
typedef struct _MPCCI_MOTION MPCCI_MOTION;
struct _MPCCI_MOTION
{
    double omeg;
    double axis[3];
    double orig[3];
    double velo[3];
    unsigned type;
};
```

2.9.4.1 Description Values

- **double omeg**
  Define the moving reference frame angular velocity.

- **double axis[3]**
  Define the moving reference frame rotational axis value components.

- **double orig[3]**
  Define the moving reference frame axis origin value components.

- **double velo[3]**
  Define the moving reference frame velocity value components.
unsigned type

Define the type resp. order of rotation resp. translation.

Valid values for moving reference frame which is neither moving grid nor moving quantity are:
- MPCCI_MOTION_FRAME_T: Translate only.
- MPCCI_MOTION_FRAME_R: Rotate only.
- MPCCI_MOTION_FRAME_RT: Rotate, then translate order.
- MPCCI_MOTION_FRAME_TR: Translate, then rotate order.
- MPCCI_MOTION_FRAME_CO: Simultaneous continuous transformations.

Valid values for moving grid which means that grid and quantities are moved, but the grid is not necessarily sent to the server are:
- MPCCI_MOTION_MGRID_T: Translate only.
- MPCCI_MOTION_MGRID_R: Rotate only.
- MPCCI_MOTION_MGRID_RT: Rotate, then translate order.
- MPCCI_MOTION_MGRID_TR: Translate, then rotate order.
- MPCCI_MOTION_MGRID_CO: Simultaneous continuous transformations.

2.9.5 Remesh Flag Information

Remesh flag values for calling 2.7.14 Notifying the Remeshing: mpcci_remesh:
- MPCCI_REMESH_FLAG_CHECK: Default value.
- MPCCI_REMESH_FLAG_NODES: Specify a remeshing on nodes level. The topology was not changed.
- MPCCI_REMESH_FLAG_ELEMS: Specify a remeshing on elements level. The mesh topology was changed.
- MPCCI_REMESH_FLAG_ALLPARTS: All parts have been remeshed.
- MPCCI_REMESH_FLAG_FULL: All parts and elements have been remeshed (equal to the bit pattern: MPCCI_REMESH_FLAG_ELEMS | MPCCI_REMESH_FLAG_ALLPARTS).

2.9.6 Transfer Information: MPCCI_TINFO

The C structure MPCCI_TINFO defines the transfer information for the code. This transfer information provides a time stamp for the quantities and allows to request quantities for a specific time stamp, too.

The MPCCI_TINFO contains the following attributes (See "mpcci_types.h"):

```c
typedef struct _MPCCI_TINFO MPCCI_TINFO;
struct _MPCCI_TINFO
{
    unsigned this_size;
    unsigned this_version;

    int     mpcci_state;
    int     mpcci_used;

    unsigned tact;
    unsigned tact_init;
    unsigned tact_last;

    int     mode_t;
    int     mode_a;
    unsigned mode_c;
    int     mode_s;
    int     mode_r;
};
```
int mode_q;

unsigned flag_r;

/*
 * in case of a time >= 0.0 we are transient and might use
 * the time and iteration for testing
 */
double dt;
double dt_last;
double rdt_min;
double rdt_max;

double time;
double time_cbeg;
double time_cend;
double time_init;
double time_last;

double implt_step;
double implt_next;
int impli_step;
int impli_max;

/*
 * in case of a time < 0.0 we are steady/iterative
 */
int iter;

int iter_cbeg;
int iter_cend;
int iter_init;
int iter_last;

/* subcycle */
int sub_nstep;
int sub_step;
int sub_next;

/*
 * information updated per call of ampcci_transfer()
 */
int count_s;
int count_r;

int nquant_s;
int nquant_r;
int nquant_req;
int nglob_s;
```c
int nglob_r;
int nglob_req;

/*
 * optional convergence status
 */
int conv_code;
int conv_job;
int conv_last;
}
```

### 2.9.6.1 Description Values

**unsigned this_size**

The size of the structure: must be set to (unsigned)sizeof(MPCCI_TINFO).

**unsigned this_version**

The version: 3.1.0=310, 4.0.0=400

**int mpcci_state**

This MpCCI state:
- -1 : Disconnected from the server.
- 0 : Nothing done before.
- 1 : Initially connected to the server.
- > 1 : Initialization done and at least one data transfer.

**int mpcci_used**

True if we use MpCCI.

**Information about the exchange communication:**

**unsigned tact**

Bit pattern: MPCCI_TACT_*
- MPCCI_TACT_SEND : Send quantities to the server.
- MPCCI_TACT_RECV : Receive quantities from the server.
- MPCCI_TACT_XCHG : Exchange quantities.
- 0 : Skip the transfer action.

**unsigned tact_init**

Bit pattern for the initial transfer.

**unsigned tact_last**

Bit pattern of the last transfer done.
int mode_t

Define the coupling scheme type:
MPCCI_TMODE_implicit : Implicit scheme.
MPCCI_TMODE_explicit : Explicit scheme.
MPCCI_TSOLU_TRANSIENT : The solution is transient.
MPCCI_TSOLU_STEADY : The solution is steady state.

You can use the macros to check the coupling scheme type:
- MPCCI_TINFO_WANT_IMPL(*MPCCI_TINFO): return true if implicit scheme is set.
- MPCCI_TINFO_WANT_EXPL(*MPCCI_TINFO): return true if explicit scheme is set.
- MPCCI_TSOLU_WANT_TRANS(*MPCCI_TINFO): return true if transient.
- MPCCI_TSOLU_WANT_STEADY(*MPCCI_TINFO): return true if steady state.

int mode_a

Define the coupling algorithm type:
MPCCI_TMODE_JACOBI : Jacobi algorithm.
MPCCI_TMODE_GAUSSSEIDEL : Gauss-Seidel algorithm.

You can use the macros to check the algorithm type:
- MPCCI_TINFO_WANT_JACOBI(*MPCCI_TINFO)
- MPCCI_TINFO_WANT_GAUSSSEIDEL(*MPCCI_TINFO)

int mode_c

The check mode types:
MPCCI_TMODE_CHECK_NONE : No check is done by the code coupling manager: option none.
MPCCI_TMODE_CHECK_FORWARD : The code can only go forward in time and iteration: option forward.

int mode_s

Levels of send modes:
MPCCI_TMODE_SEND_ALWAYS : The code always sends the data: option always.
MPCCI_TMODE_SEND_ONEWAIT : option onewait.
- Bi-directional coupling: The code sends if at least one partner is waiting, otherwise send is skipped.
- Uni-directional coupling: The code waits for at least one partner before sending the data.
MPCCI_TMODE_SEND_ALLWAIT : option allwait.
- Bi-directional coupling: The code sends if all partners are waiting, otherwise send is skipped.
- Uni-directional coupling: The code waits for all partners before sending the data.

int mode_r

Levels of receive modes:
MPCCI_TMODE_RECV_ANY : The code receives all if at least one quantity is available, otherwise the receive action is skipped: option any.
MPCCI_TMODE_RECV_COMPLETE : The code receives only if all quantities are available, otherwise the receive action is skipped: option complete.
MPCCI_TMODE_RECV_ALL : The code always waits for all requested quantities: option all.
```plaintext
int mode_q
Quantity selection mode for the data transfer:
MPCCI_TMODE_QUANT_ALL  : Exchange all quantities.
MPCCI_TMODE_QUANT_COORD : Exchange only coordinate type quantities.
MPCCI_TMODE_QUANT_OTHER : Exchange all non coordinate type quantities.

unsigned flag_r
Bit pattern for remeshing (see  2.9.5 Remesh Flag Information).  

Information about transient simulation:

double dt
Current physical time step size updated by the code before the call of \texttt{ampcci\_transfer()}  2.7.13 Data Exchange: \texttt{ampcci\_transfer}.  

double dt\_last
Physical time step size during the last call.  

double rdt\_min
Minimum relative time distance to interpolated values.  

double rdt\_max
Maximum relative time distance to interpolated values.  

double time
Current physical time updated by the code before the call of \texttt{ampcci\_transfer()}.  

double time\_cbeg
Time when the coupling starts in terms of the code times.  

double time\_cend
Time when the coupling ends in terms of the code times.  

double time\_init
Code time when the adapter was initialized.  

double time\_last
Code time of the last exchange.  

Information about implicit coupling parameters:

double implt\_step
Time step size for implicit coupling.  

double implt\_next
Next implicit synchronization time.  
```
**double impli_step**
No. of iterations between two exchanges.

**double impli_max**
Maximum no. of coupled iterations.

**Information about steady/iterative simulation:**

**int iter**
Current iteration counter: updated by the code before the call of `ampcci_transfer()`.

**int iter_cbeg**
First iteration where we exchange.

**int iter_cend**
Last iteration where we exchange.

**int iter_init**
Code iteration when the adapter was initialized.

**int iter_last**
Iteration of the last exchange.

**Information about subcycling:**

**int sub_nstep**
Current coupling step counter, incremented at each call of `ampcci_transfer()` function.

**double sub_step**
No. of steps between two exchanges. This contains the current step size for the subcycling. This value can be constant or variable if a table definition has been defined.

**int sub_next**
Step number for the next coupling step, updated after a coupled step number is reached by `ampcci_transfer()` function.

**Information updated per call of `ampcci_transfer()`:**

**int count_s**
Number of `ampcci_transfer()` send calls (independent of quantities and meshes).

**int count_r**
Number of `ampcci_transfer()` receive calls (independent of quantities and meshes).

**int nquant_s**
Number of sent quantities (sum( mesh * nquant )) within the last `ampcci_transfer()`.
2.9.7 Code Specific Information: **MPCCI_CINFO**

The MPCCI_CINFO defines some code specific information (see "mpcci_types.h"). It contains the following attributes:

```c
typedef struct _MPCCI_CINFO MPCCI_CINFO;
struct _MPCCI_CINFO
{
    const char *jobid;
    const char *codename;
    const char *codeid;
    const char *clientid;
    double time;
    unsigned flags;
    unsigned jflags;
    int iter;
    int nclients;
    int nprocs;
};
```
2.9.7.1 Description Values

`const char *jobid`
The job id identifying the coupling job.

Do not set this field, it will be automatically set.

`const char *codename` *(Required)*
The name of the code.

`const char *codeid`
This is the instance name of the code. This may be defined by the environment variable `MPCCI_CODEID` by MpCCI GUI before the start.

Do not set this field, it will be automatically set.

`const char *clientid`
This client id helps to identify parallel processes.

Do not set this field, it will be automatically set.

`double time` *(Required)*
The initial physical time of the code.

`unsigned flags` *(Required)*
Feature flags of the code. This may be defined by using these attributes (defined in "mpcci_defs.h"):

- `MPCCI_CFLAG_TYPE_FV`: The code is a finite volume code.
- `MPCCI_CFLAG_TYPE_FEA`: The code is a finite element analysis code.
- `MPCCI_CFLAG_TYPE_NET`: The code is a network code.
- `MPCCI_CFLAG_TYPE_RAD`: The code is a radiation code.
- `MPCCI_CFLAG_GRID_ORIG`: The code has the original grid available.
- `MPCCI_CFLAG_GRID_CURR`: The code has the current grid available.

`unsigned jflags`
Coupling feature flags of the code and job. This field is initialized by `mpcci_cinfo.init`.

`int iter` *(Required)*
The initial iteration counter of the code.

`int nclients` *(Required)*
The exact number of expected server clients.

`int nprocs` *(Optional)*
The number of parallel processes of the current code. This is not necessarily the number of clients. This field is optional if the number of expected server clients and number of parallel processes are equal.

These information may be used to dynamically reconfigure the MpCCI server at runtime when a code wants to disconnect and reconnect to the coupling server.
2.9.7.2 Code Information Macro

<table>
<thead>
<tr>
<th>Return value</th>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>MPCCI_CODE_HAS_CURRG(code)</td>
<td>check whether the code has the current grid.</td>
</tr>
<tr>
<td>bool</td>
<td>MPCCI_CODE_HAS_ORIGG(code)</td>
<td>check whether the code has the original grid.</td>
</tr>
<tr>
<td>bool</td>
<td>MPCCI_CODE_IS_FV(code)</td>
<td>check whether the code is a finite volume code.</td>
</tr>
<tr>
<td>bool</td>
<td>MPCCI_CODE_IS_FEA(code)</td>
<td>check whether the code is finite element analysis code.</td>
</tr>
<tr>
<td>bool</td>
<td>MPCCI_CODE_IS_NET(code)</td>
<td>check whether the code is a network code.</td>
</tr>
<tr>
<td>bool</td>
<td>MPCCI_CODE_IS_RAD(code)</td>
<td>check whether the code is a radiation code.</td>
</tr>
</tbody>
</table>

2.9.8 Coupling Components

The coupling component descriptor `MPCCI_PART` represents a coupling component for the code. Many driver functions obtain an argument `part` which is a pointer to the coupling component (coupling component pointer).

The structure contains the basic properties of a component and a list of the quantities, which shall be exchanged. The most important macros for `MPCCI_PART` are listed below, all macros take an argument of type `MPCCI_PART *`, thus they can be applied directly to the pointer `part`. E.g. `MPCCI_PART_NNODES(part)` should be used to set or determine the number of nodes of a component.

<table>
<thead>
<tr>
<th>Return value</th>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>char *</td>
<td>MPCCI_PART_NAME(part)</td>
<td>name of coupling component</td>
</tr>
<tr>
<td>int</td>
<td>MPCCI_PART_MDIM(part)</td>
<td>dimension of component. (See &gt; 2.9.3 Mesh Dimension Definition)</td>
</tr>
<tr>
<td>MPCCI_QUANT *</td>
<td>MPCCI_PART_QUANTS(part)</td>
<td>list of quantities to transfer</td>
</tr>
<tr>
<td>int</td>
<td>MPCCI_PART_NNODES(part)</td>
<td>number of nodes/points in component</td>
</tr>
<tr>
<td>int</td>
<td>MPCCI_PART_NELEMS(part)</td>
<td>number of faces/elements/cells in component</td>
</tr>
<tr>
<td>int</td>
<td>MPCCI_PART_NODMAX(part)</td>
<td>max. number of nodes per element in this component</td>
</tr>
<tr>
<td>int</td>
<td>MPCCI_PART_MESHD(part)</td>
<td>MpCCI mesh id</td>
</tr>
<tr>
<td>int</td>
<td>MPCCI_PART_PARTID(part)</td>
<td>MpCCI part id</td>
</tr>
<tr>
<td>unsigned</td>
<td>MPCCI_PART_CSYS(part)</td>
<td>coordinates system</td>
</tr>
<tr>
<td>void *</td>
<td>MPCCI_PART_USRPTR0(part)</td>
<td>pointer no. 0 to auxiliary optional information defined and used by the application</td>
</tr>
<tr>
<td>void *</td>
<td>MPCCI_PART_USRPTR1(part)</td>
<td>pointer no. 1 to auxiliary optional information defined and used by the application</td>
</tr>
<tr>
<td>void *</td>
<td>MPCCI_PART_USRPTR2(part)</td>
<td>pointer no. 2 to auxiliary optional information defined and used by the application</td>
</tr>
<tr>
<td>void *</td>
<td>MPCCI_PART_USRPTR3(part)</td>
<td>pointer no. 3 to auxiliary optional information defined and used by the application</td>
</tr>
<tr>
<td>void *</td>
<td>MPCCI_PART_USRPTR4(part)</td>
<td>pointer no. 4 to auxiliary optional information defined and used by the application</td>
</tr>
<tr>
<td>void *</td>
<td>MPCCI_PART_USRPTR5(part)</td>
<td>pointer no. 5 to auxiliary optional information defined and used by the application</td>
</tr>
<tr>
<td>void *</td>
<td>MPCCI_PART_USRPTR6(part)</td>
<td>pointer no. 6 to auxiliary optional information defined and used by the application</td>
</tr>
<tr>
<td>void *</td>
<td>MPCCI_PART_USRPTR7(part)</td>
<td>pointer no. 7 to auxiliary optional information defined and used by the application</td>
</tr>
<tr>
<td>void *</td>
<td>MPCCI_PART_USRPTR8(part)</td>
<td>pointer no. 8 to auxiliary optional information defined and used by the application</td>
</tr>
</tbody>
</table>
Return value | Macro | Description |
---|---|---|
void * | MPCCI_PART_USR_PTR9(part) | pointer no. 9 to auxiliary optional information defined and used by the application |
unsigned | MPCCI_PART_USR_FLAGS(part) | optional flags information field for the current mesh defined and used by the application |
int | MPCCI_PART_USR_STATE(part) | optional state information for the current mesh defined and used by the application |

Return value macro description |

bool | MPCCI_PART_IS_EMPTY(part) | check whether component represents an empty part with zero nodes and zero elements |
bool | MPCCI_PART_IS_NONE(part) | check whether component is nothing |
bool | MPCCI_PART_IS_NETP(part) | check whether component represents a network point |
bool | MPCCI_PART_IS_LINE(part) | check whether component is a line |
bool | MPCCI_PART_IS_FACE(part) | check whether component is a face |
bool | MPCCI_PART_IS_VOLU(part) | check whether component is a volume |

See also ▷2.9.10 Loop Functions for loops over components.

### 2.9.9 Quantities

The quantity descriptor MPCCI_QUANT represents one quantity. It contains the name and properties of the quantity.

Return value | Macro | Description |
---|---|---|
const char * | MPCCI_QUANT_NAME(quant) | symbolic name e.g. “temp”, “JHeat” ... |
int | MPCCI_QUANT_QID(quant) | ID code of quantity (see also the Quantity Reference in the Appendix) |
int | MPCCI_QUANT_STATE(quant) | state of transfer MPCCI_QUANT_TSTATE_xxx (used by the manager): |
| | MPCCI_QUANT_TSTATE_NONE | no transfer was done before |
| | MPCCI_QUANT_TSTATE_SEND | quantity was sent |
| | MPCCI_QUANT_TSTATE_RECV | quantity was received |
int | MPCCI_QUANT_FLAGS(quant) | contains the direction send/receive bits, the location bits etc. ... |
int | MPCCI_QUANT_DIM(quant) | dimension of quantity: 1=scalar, 3=vector etc. |
int | MPCCI_QUANT_LOC(quant) | location of value, one of: |
| | MPCCI_QFLAG_LOC_CODE | |
| | MPCCI_QFLAG_LOC_NODE | or MPCCI_QFLAG_LOC_VERT |
| | MPCCI_QFLAG_LOC_ELEM | or MPCCI_QFLAG_LOC_CELL |
| | MPCCI_QFLAG_LOC_POINT | |
int | MPCCI_QUANT_PHY(quant) | physical meaning of this quantity, one of |
<p>| | MPCCI_QPHY_ANY | quantity is not further specified |
| | MPCCI_QPHY_MASS | a mass sink/source |
| | MPCCI_QPHY_MOM | a momentum sink/source |
| | MPCCI_QPHY_ENTH | an energy sink/source |
| | MPCCI_QPHY_PROP | a property (material) |</p>
<table>
<thead>
<tr>
<th>Return value</th>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MPCCI_QPHY_BCVAL</td>
<td>a boundary condition value</td>
</tr>
<tr>
<td></td>
<td>MPCCI_QPHY_BCGRAD</td>
<td>a boundary condition wall normal gradient</td>
</tr>
<tr>
<td></td>
<td>MPCCI_QPHY_COORD</td>
<td>a grid coordinate/displacement</td>
</tr>
<tr>
<td></td>
<td>MPCCI_QPHY_MASSFR</td>
<td>a chemical component concentration</td>
</tr>
</tbody>
</table>

```c
int MPCCI_QUANT_DIR(quant)
```

Transfer direction, one of
- MPCCI_QFLAG_DIR_SEND: quantity is sent
- MPCCI_QFLAG_DIR_RECV: quantity is received

```c
int MPCCI_QUANT_SMETHOD(quant)
```

How to store sent/received quantities, one of
- MPCCI_QSM_UNDEF: invalid send method
- MPCCI_QSM_DIRECT: directly written into buffer
- MPCCI_QSM_BUFFER: quantity is buffered locally and copied later
- MPCCI_QSM_USRMEM: quantity to store in user's indexed memory
- MPCCI_QSM_SCALAR: quantity to store in user's index scalars
- MPCCI_QSM_SPECIES: quantity to store the chemical species

```c
int MPCCI_QUANT_SINDEX(quant)
```

Index to user supplied memory or user scalars

```c
void * MPCCI_QUANT_SBUFFER(quant)
```

Quantities local transfer buffer

```c
int MPCCI_QUANT_SBFSIZE(quant)
```

Size of allocated local buffer

<table>
<thead>
<tr>
<th>Return value</th>
<th>macro</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>MPCCI_QUANT_IS_SEND(quant)</td>
<td>check whether the quantity is sent</td>
</tr>
<tr>
<td>bool</td>
<td>MPCCI_QUANT_IS_RECV(quant)</td>
<td>check whether the quantity is received</td>
</tr>
<tr>
<td>int</td>
<td>MPCCI_QUANT_IS_NODE(quant)</td>
<td>check whether quantity is a nodal quantity</td>
</tr>
<tr>
<td>int</td>
<td>MPCCI_QUANT_IS_ELEM(quant)</td>
<td>check whether quantity is an element quantity</td>
</tr>
<tr>
<td>int</td>
<td>MPCCI_QUANT_IS_GLOB(quant)</td>
<td>check whether the quantity is a global quantity</td>
</tr>
<tr>
<td>int</td>
<td>MPCCI_QUANT_IS_MESH(quant)</td>
<td>check whether the quantity is a mesh based quantity</td>
</tr>
<tr>
<td>int</td>
<td>MPCCI_QUANT_IS_COORD(quant)</td>
<td>check whether the quantity is a coordinate/displacement</td>
</tr>
<tr>
<td>int</td>
<td>MPCCI_QUANT_IS_FIELD(quant)</td>
<td>check whether the quantity is a field value</td>
</tr>
<tr>
<td>int</td>
<td>MPCCI_QUANT_IS_FLUXI(quant)</td>
<td>check whether the quantity is a flux integral value</td>
</tr>
<tr>
<td>int</td>
<td>MPCCI_QUANT_IS_FLUXD(quant)</td>
<td>check whether the quantity is a flux density value</td>
</tr>
<tr>
<td>bool</td>
<td>MPCCI_QUANT_IS_SCALAR(part)</td>
<td>check whether the quantity is a scalar</td>
</tr>
<tr>
<td>bool</td>
<td>MPCCI_QUANT_IS_VECTOR(part)</td>
<td>check whether the quantity is a vector</td>
</tr>
<tr>
<td>bool</td>
<td>MPCCI_QUANT_IS_TENSOR(part)</td>
<td>check whether the quantity is a tensor</td>
</tr>
</tbody>
</table>

### 2.9.10 Loop Functions

In addition to the data access functions, a number of predefined loops is available:
FOREACH(obj, head) { ... }
Loop over all objects provided by the head which is a list of objects.
Example:

```c
MPCCI_SERVER *server;
FOREACH(server, job->servers)
   ...;
```

FOREACH_COND(obj, head, cond) { ... }
Loop over all objects provided by the head which is a list of objects. This loop stops if the obj verifies the condition cond.
Example:

```c
FOREACH_COND(part, server->parts, MPCCI_PART_IS_FACE(part))
   ...;
```

FOREACH_QRECV(quant, head) { ... }
Like FOREACH, but the loop is restricted to received-only quantities.
Example:

```c
MPCCI_QUANT *quant;
FOREACH_QRECV(quant, MPCCI_PART_QUANTS(part))
   ...;
```

FOREACH_QSEND(quant, head) { ... }
Like FOREACH_QRECV, but the loop is restricted to sent-only quantities.
Example:

```c
MPCCI_QUANT *quant;
FOREACH_QSEND(quant, MPCCI_PART_QUANTS(part))
```
2.9.11 Memory Management

For some actions, the code adapter needs to allocate memory. If your code has its own memory management system, it is preferable that the code adapter also uses this system. If no methods are specified, the standard C library routines are used.

You need to call the following function to point to the code memory management functions:

```c
void umpcci_mem_functs
(
    void *(*mallocp) (size_t size),
    void *(*reallocp) (void *ptr, size_t ssize),
    void (*freep) (void *ptr)
)
```

- **void *(*mallocp)(size_t size)**
  This routine is called for allocating memory, should return a pointer for the memory, a NULL pointer if it fails.

- **void *(*reallocp)(void *ptr, size_t size)**
  This routine is used for reallocation, i.e. changes of size of an array.

- **void (*freep)(void *ptr)**
  This routine is called for freeing memory.
Part IX

How To

Version 4.6.1
IX How To – Contents

1 Automotive Thermal Management 4
  1.1 Quick Information ................................................. 4
  1.2 Problem Description and Motivation .............................. 4
  1.3 Simulation Procedure ............................................ 5
    1.3.1 Model Preparation ............................................. 5
    1.3.2 MpCCI Setup .................................................. 6
    1.3.3 Running the Simulation ....................................... 9
    1.3.4 Results ........................................................ 10

2 Simulation of Fluid-Structure Interaction for a New Hydraulic Axial Pump 13
  2.1 Quick Information .................................................. 13
  2.2 Problem Description and Motivation .............................. 13
  2.3 Simulation Procedure ............................................ 14
    2.3.1 Axial Hydraulic Pumps with Compensation Chambers ........ 14
    2.3.2 Model Preparation ............................................. 16
    2.3.3 MpCCI Setup .................................................. 17
    2.3.4 Running the Simulation ....................................... 18
    2.3.5 Results ........................................................ 19
1 Automotive Thermal Management

1.1 Quick Information

<table>
<thead>
<tr>
<th>Keywords</th>
<th>Full vehicle thermal management, coupled simulation with MpCCI convection, conduction and radiation, steady state and transient</th>
</tr>
</thead>
</table>
| Simulation Codes | STAR-CCM+  
FLUENT  
TAITherm |
| MpCCI Version | MpCCI 4.2 and higher |
| Related Tutorials | \>| VII-9 Cube in a Duct Heater <|

Figure 1: Thermal management

1.2 Problem Description and Motivation

Concerning the thermal behavior of automotive vehicles it is pursued to accomplish simulations for the full complexity of a vehicle’s geometry and transport phenomena of heat including convection, radiation and conduction in fluids and solid bodies. There are several simulation codes meeting these requirements. On each transport mechanism the simulation codes have more or less effectiveness and accuracy, so that for a simulation procedure a combination of different codes is preferable. This procedure can be realized with MpCCI, which offers the following advantages compared to common file-based approaches:
• Fully automatic and fast data transfer over socket connections.
• Easy implementation for repeating data transfer as used for iterative steady or transient simulations.
• Fast and robust data mapping between non-matching meshes.
• Data inter- and extrapolation on geometrical irregularities like partially non-matching geometries.
• Flexible steering of simulation procedure for instance with subcycling, time-step sharing and further features depending on the applied simulation codes.

In this code of practice it is intended to demonstrate the procedure of coupling CFD codes STAR-CCM+ or FLUENT with TAItherm for steady state and transient thermal simulations.

1.3 Simulation Procedure

1.3.1 Model Preparation

Usually in CFD models thin solids like heat shields have two sides wetted, whereas the distance between the two sides, the thickness of the heat shield, is in the majority of cases small. If in TAItherm parts are modeled as shell elements a certain thickness must be assigned to the shell elements to properly account for heat conduction and heat capacity. When surfaces, boundaries and shells are coupled it has to be assured that front and back side of the coupled surfaces are correctly assigned. When setting up a model, the boundaries of thin parts of the CFD model have to be separated in a front and back side as depicted in Figure 2. Later in the MpCCI Coupling Step the front and back side can be assigned to different coupling regions, so that the mesh correlations are calculated properly.

Concerning setting the heat sources in the simulation one has to account for the quantities later exchanged in MpCCI. When sending the wall temperature from TAItherm it is not advisable to set wall temperatures for coupled boundaries in the CFD code. For an exhaust system for instance one might set a convection condition, internal heat rate and/or a fluid (stream) part in TAItherm inside the exhaust or model the exhaust flow in the CFD model.
1.3.2 MpCCI Setup

The model set-up in MpCCI corresponds to a typical thermal coupling procedure, where it is common and stable to exchange the heat coefficient and the film temperature to TAITherm and the wall temperature to the CFD code. As already depicted in 1.3.1 Model Preparation it is necessary for thin parts to separate the boundaries in CFD and define different coupling regions in MpCCI for front and back sides. A full vehicle features a lot of different boundaries which do not have to be coupled in separate coupling regions each. It is advisable to create groups of coupling regions which hold different part groups of the vehicle (Figure 3). It is also a good procedure to reflect the group name in the boundary name to use the MpCCI pattern matching (V-4.6.5 Automatic Generation of Regions by Rules). In STAR-CCM+ it is quite common to combine boundaries in certain groups and regions like engine, exhaust etc. In this case the boundary names available in MpCCI will already contain the group/region name used in STAR-CCM+.

Figure 3: Possible coupling meshes for a full vehicle reflecting part groups

It should be noted that not insulated back sides of shell element parts in TAITherm will appear in MpCCI with the string "-backside" added to the coupling part name VI-17.2.4 Regions Step.<

Although it is not desired in large full vehicle models, there are quite often geometrical discrepancies between the coupled models, which usually leads to orphaned regions (Figure 4). There are three procedures for handling these orphaned regions. It is possible to modify the search parameter in order to catch geometry deviations. Either the multiplicity factor for a dimensionless search or the normal dimensionful search distance is raised. If the granularity of the coupling regions is coarse, e.g. all components in one coupling region, it might result in wrong neighborhood associations especially when parts come in close contact. The second procedure is to accept a default value for orphaned regions. As this might be reasonable for the heat coefficient which is usually set to zero (adiabatic), for wall temperatures it is kind of random and therefore restrictive. To overcome this problem MpCCI offers extrapolation to orphaned regions, where matched parts around the orphans define the inter- and extrapolated values. In Figure 5 the procedures are compared for a hot spot in an underhood simulation. It is visible that in smaller orphaned regions the extrapolation is a pretty good approximation compared to the sender source distribution, but in large orphaned regions the extrapolated temperature distribution is getting more vague.
Figure 4: Orphaned regions on bottom of BMW vehicle visualized in the MpCCI visualizer
Orphaned regions on sender mesh

Wall temperature distribution on sender mesh

Wall temperature distribution on receiver mesh with default values for orphaned regions

Wall temperature distribution on receiver mesh with extrapolation for orphaned regions

Figure 5: Treatment of orphaned regions inMpCCI
1.3.3 Running the Simulation

The complete simulation process might be started from the MpCCI GUI, simply as batch job e.g. by `mpcci batch case.csp` or via a batch queuing system like PBS, SGE, etc. If the simulation is started from MpCCI GUI, for TAITherm, FLUENT and STAR-CCM+ it is possible to run the coupled simulation with their own GUI. This procedure offers the possibility to display results during simulation besides the MpCCI Visualizer or interact with the simulation model.

**TAITherm**

TAITherm cannot be steered by a script/macro file. When the necessary MpCCI function hooks are placed, TAITherm will exchange data with MpCCI through these functions. For a steady state simulation (duration=0s) after each iteration and for a transient simulation after each time step the exchange of data will be accomplished.

**STAR-CCM+**

STAR-CCM+ must be steered by a STAR-CCM+ java macro file, which can be provided in MpCCI or in the STAR-CCM+ GUI. The macro may contain all STAR-CCM+ java structures and additionally will contain MpCCI functions like "mpcci.init()", "mpcci.exchange()" and "mpcci.exit()". Usually the macro will contain a loop, where after or before each or several iterations or time-steps an exchange is committed. The information of the initial exchange defined in the MpCCI GUI will be handed over to the first call of "mpcci.exchange()".

**FLUENT**

In FLUENT the necessary routines (user-defined-functions) will be provided by MpCCI and it is possible to let MpCCI hook the necessary functions automatically. In addition all MpCCI functions might be carried out by the MpCCI Control Panel, which is available in the FLUENT GUI. When FLUENT is started in batch mode a journal file has to be supplied to run the simulation. The journal file might contain all FLUENT structures, in case the user did not allow MpCCI to set the necessary hooks it should additionally contain MpCCI function calls.

Smoothly interrupting a coupled simulation and then restarting the same simulation is a challenging task. If you run the job with the MpCCI GUI you can hit the [Stop] in order to provide a STOP/ABORT file or a signal to save the solution and quit the simulation codes gracefully. Unfortunately if one code is waiting for data at this moment it might not be successful. When running on a batch system there is actually no way to smoothly stop the codes with a proper file save. In this case it is advisable to turn on periodic autosave of files in the simulation codes and then restart with the last saved states.

For further information of the procedures for each code see the Codes Manual part of the MpCCI manual.
1.3.4 Results

This application for full vehicle thermal simulation is presented by courtesy of BMW Germany.

<table>
<thead>
<tr>
<th>STAR-CCM+</th>
<th>approx. 45 mio. cells: 15 regions, 525 boundaries</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MRF for fans and wheels</td>
</tr>
<tr>
<td></td>
<td>Porous regions for heat exchangers and cooling devices</td>
</tr>
<tr>
<td></td>
<td>Wall rotation of shafts and axles</td>
</tr>
<tr>
<td>TAItherm</td>
<td>approx 900,000 cells,</td>
</tr>
<tr>
<td></td>
<td>323 shell parts</td>
</tr>
<tr>
<td></td>
<td>13 fluid parts for exhaust system</td>
</tr>
<tr>
<td>MpCCI</td>
<td>coupled quantities:</td>
</tr>
<tr>
<td></td>
<td>STAR-CCM+ → TAItherm: film temperature, heat coefficient</td>
</tr>
<tr>
<td></td>
<td>TAItherm → STAR-CCM+: wall temperature</td>
</tr>
<tr>
<td></td>
<td>⇒ approx. 2 mio. faces coupled</td>
</tr>
</tbody>
</table>

Table 1: Model data

Figure 6: STAR-CCM+ full vehicle model of a BMW top and bottom view
Figure 7: TAITherm underbody model of a BMW top and bottom view

Figure 8: Wall temperature in STAR-CCM+ of BMW vehicle top and bottom view
Figure 9: Wall temperature in STAR-CCM+ of BMW vehicle front axle
2 Simulation of Fluid-Structure Interaction for a New Hydraulic Axial Pump

2.1 Quick Information

<table>
<thead>
<tr>
<th>Keywords</th>
<th>Fluid-structure interaction for compensation chamber in a new pump design coupled simulation with MpCCI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation Codes</td>
<td>STAR-CCM+</td>
</tr>
<tr>
<td></td>
<td>FLUENT</td>
</tr>
<tr>
<td></td>
<td>ANSYS</td>
</tr>
<tr>
<td></td>
<td>Abaqus</td>
</tr>
<tr>
<td>MpCCI Version</td>
<td>MpCCI 4 and higher</td>
</tr>
<tr>
<td>Related Tutorials</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: Hydraulic Axial Pump

2.2 Problem Description and Motivation

Numerical simulation plays an important role in the product development and design process in all engineering areas. Depending on the type of machinery developed structural or fluid dynamics simulation codes are used to find optimal designs.

Many real world applications – most prominently fluid-structure interactions – cannot be modeled sufficiently using only one specialized simulation code. The fluid and the structural problem influence each other resulting in the need to transfer information between the two simulation codes.

Therefore, a coupled numerical simulation might be necessary to find optimal designs for complex machinery and tools. MpCCI facilitates the transfer of relevant information between different simulation codes.
codes.
To achieve a robust and easy-to-use coupling of two or more simulation codes MpCCI automatically transfers the necessary quantity values via fast socket connections. Mapping of quantity values between non-matching meshes is realized with fast and robust interpolation methods. For partially non-matching geometries MpCCI provides extrapolation methods.
This document shows how to use MpCCI for numerical simulation of fluid-structure interaction during the design process for a new axial hydraulic pump.
The mechanism of the pump and the need for a FSI simulation will be described in section 2.3.1. Afterwards, a short overview of the used models for the CFD and CSM simulation will be given, followed by the necessary MpCCI settings and the presentation of some results.

2.3 Simulation Procedure

2.3.1 Axial Hydraulic Pumps with Compensation Chambers

Figure 2: Picture of PWK pumps on the left, main elements of the PWK pump on the right

Axial pumps with cam-driven commutation units – so-called PWK pumps – emerged as a result of a research project conducted in the Department of Hydraulics and Pneumatics at the Gdansk University of Technology. Several – in most cases seven – cylinder chambers are positioned around a rotating shaft with attached swashplates. The movement of the swashplates leads to an alternating increase and decrease of the volume of the cylinder chambers. These chambers connect to the low pressure in- and high pressure outtake channels via a moving bridge. The main elements of the pump can be seen in Figure 2.
Particularly interesting is the development of the variable displacement version of these new pumps. They can be controlled by a low-energy actuator while these pumps usually require a complicated hydraulic servomechanism. This reduces the pump’s cost and dimension drastically and is the key feature of the new development.
First prototypes of the new pump have been built and tested, e.g. in lifting devices on ships in extreme temperature environments, and showed good results.
Unfortunately, harmful pressure peaks, that might lead to pump damage, were observed. These peaks occur when the cylinder chamber is disconnected from the intake and outtake channels. For a short period of time the pressure reaches very high values - more than 20 MPa above the average pressure in the chamber (cf. Figure 3).
These peaks are influenced by many different factors ranging from fluid properties to pump speed or leakage...
Figure 3: Pressure variation in pump’s chamber (rotational speed 500 rpm, oil temperature 35°C, displacement adjustment 22%).

Figure 4: Pressure peak reduction with a compensation chamber with an elastic wall. a) shows high pressure in the cylinder chamber which causes the fluid to flow into the compensation chamber. In b) the pistons are moving outwards and the hydraulic oil flows back into the cylinder chamber.

and have been investigated quite thoroughly. As the very high pressure values might lead to pump damage and produce a lot of noise a compensation chamber was introduced into the pump design.

The compensation chamber significantly shortens the period of disconnection. When the connection bridge
moves from the intake to the outtake channels it connects the cylinder chamber to the compensation chamber. Additionally it gives the fluid more room. All cylinder chambers of one pump can use the same compensation chamber, which is usually located around the rotating shaft.

The elastic wall of the compensation chamber deforms under the load of the hydraulic oil which in turn changes the velocity and pressure field of the fluid leading to the classic case of a fluid-structure interaction.

2.3.2 Model Preparation

As the CAD model of a full pump with seven cylinder chambers is quite big and complicated, for most simulations the simplified symmetric half model on the left of Figure 5 was used.

The CFD model was developed in FLUENT as well as in STAR-CCM+. For both codes a slightly compressible fluid (with user defined functions) and the Spalart-Allmaras turbulence model are used. The motion of the bridge – connecting the cylinder chamber to the in- and outtake channels as well as the compensation chamber – is also realized with user defined functions.

In FLUENT hexahedral elements and the layering method to model the motion of the pistons are used. This is an easy and comfortable method for mesh motion.

In STAR-CCM+ this motion is realized with the help of a remeshing module that has been developed using the Java API of STAR-CCM+. The large motion of the pistons cannot be modeled with morphing and STAR-CCM+ does not provide automatic remeshing. The remeshing module checks with the help of mesh parameters whether remeshing is necessary, generates a new mesh and interpolates the values from the old mesh onto the new one.

The CSM model is a lot simpler than the CFD model since it only consists of the compensation chamber. Abaqus was used to model the deformation of the membrane under the hydraulic load. Two different models – a shell and a volume model – were developed to compare the results.

Isotropic elastic material with a Young's modulus of \( E = 2.110^9 \), a Poisson ration \( \nu = 0.35 \) and the density \( \rho = 7800\text{kg/m}^3 \) was used. The applied loads were ramped linearly over the time step.
2.3.3 MpCCI Setup

The setup of the coupling is quite straightforward for this case.

There is only one coupling surface – the membrane of the compensation chamber – that has to be selected in the MpCCI Coupling Step. The quantities that are exchanged between Abaqus and the CFD code are NPosition and WallForce.

For both quantities under-relaxation or ramping factors have to be defined in the MpCCI Coupling Step.

The movement of the wall and the almost incompressible fluid leads to convergence problems of the coupled simulation.

Using under-relaxation for the displacements as well as the wall forces and ramping the fluid loads over the Abaqus time step is necessary to achieve a convergent solution of the problem.

Problems with moving walls and incompressible fluids often require a strong or implicit coupling to get to a stable solution. In this example a stable solution could be obtained using weak coupling and under-relaxation.

The results of the coupled simulation are very sensitive to the time step size. This leads to very long simulation times with a fixed time increment in Abaqus and in the CFD code.
2.3.4 Running the Simulation

The complete simulation process might be started from the MpCCI GUI, simply as batch job e.g. by `mpcci batch case.csp` or via a batch queuing system like PBS, SGE, etc. If the simulation is started from MpCCI GUI, for Abaqus, FLUENT and STAR-CCM+ it is possible to run the coupled simulation with their own GUI. This procedure offers the possibility to display results during simulation besides the MpCCI Visualizer or interact with the simulation model.

Abaqus

All necessary settings can be made in the MpCCI Go Step. Additional command line options can be entered. The coupling time step and the options to allow subcycling or enforce exact target times can be set. Abaqus simulations always run in batch.

STAR-CCM+

STAR-CCM+ must be steered by a STAR-CCM+ java macro file, which can be provided in MpCCI or in the STAR-CCM+ GUI. The macro may contain all STAR-CCM+ java structures and additionally will contain MpCCI functions like `mpcci.init()`, `mpcci.exchange()` and `mpcci.exit()`. Usually the macro will contain a loop, where after or before each or several iterations or time-steps an exchange is committed. The information of the initial exchange defined in the MpCCI GUI will be handed over to the first call of `mpcci.exchange()`.

FLUENT

In FLUENT the necessary routines (user-defined-functions) will be provided by MpCCI and it is possible to let MpCCI hook the necessary functions automatically. In addition all MpCCI functions might be carried out by the MpCCI Control Panel, which is available in the FLUENT GUI. When FLUENT is started in batch mode a journal file has to be supplied to run the simulation. The journal file might contain all FLUENT structures, in case the user did not allow MpCCI to set the necessary hooks it should additionally contain MpCCI function calls.

Figure 6: Solid Abaqus model. U displacement contours with a maximum value of 0.0001.

Smoothly interrupting a coupled simulation and then restarting the same simulation is a challenging task. If you run the job with the MpCCI GUI you can hit the Stop button in order to provide a STOP/ABORT file or a signal to save the solution and quit the simulation codes gracefully. Unfortunately if one code is waiting for data at this moment it might not be successful. When running on a batch system there is actually no way to smoothly stop the codes with a proper file save. In this case it is advisable to turn on periodic autosave of files in the simulation codes and then restart with the last saved states.

For further information of the procedures for each code see the Codes Manual.
2.3.5 Results

The simulation results show the necessity of a coupled simulation for this application. The simulation including the fluid-structure interaction at the elastic wall agrees significantly better with the experimental findings than the stand-alone CFD simulation.

Results of the Abaqus simulation can be seen in Figure 6 and Figure 7. The deformation of the elastic wall is quite small and can only be seen when it is scaled.

The pressure contours of the fluid on the moving geometry can be visualized to get a general idea of the working pump and the quality of the simulation (cf. Figure 8) – but the slight differences between the stand-alone CFD solution and the MpCCI solution cannot be perceived visually on these contour plot.

For an easy comparison of simulation results with experimental data the pressure is monitored at three discrete points of the pump during the simulation: one point located in the center of the upper chamber, one in the lower chamber and the last point is located in the center of the compensation chamber. These pressure values can be compared for different pump parameters or simulation variations.

Figure 9 shows two examples for these pressure plots: one for a stand-alone CFD simulation and one for a fluid-structure interaction. The negative pressure peaks are much less pronounced in the FSI solution. Furthermore the pressure in the compensation chamber shows a different behavior in the two cases. The values during the "high pressure phase" are higher for the FSI solution. Also, a qualitative difference can be observed when investigating how the pressure rises or falls in the compensation chamber.

All in all the FSI simulations showed a very good agreement with the experiments that were conducted. The CFD model on its own is not capable of catching the pressure behavior – especially in the compensation chamber – in a satisfactory way.
Figure 8: Pressure contours for the PWK pump at different time values: on the left $t = 0.0002s$, in the middle $t = 0.01s$ and on the right $t = 0.02s$.

Figure 9: Pressure plots for three discrete points in the pump. CFD stand-alone (FLUENT) results are on the left, FSI results (FLUENT-Abaqus-MpCCI) are on the right. The differences in the pressure peaks and in the pressure value in the compensation chamber can be seen.
Part X

MpCCI FSIMapper

Version 4.6.1
X MpCCI FSIMapper – Contents

1 MpCCI FSIMapper Overview

2 MpCCI FSIMapper Installation
   2.1 Part of MpCCI Installation
   2.2 Standalone Version
      2.2.1 Local Microsoft Windows Installation
      2.2.2 Linux or Multi-Platform Installation
      2.2.3 Perl
      2.2.4 License
      2.2.5 Configure a License Manager as UNIX Service
      2.2.6 Configure a License Manager as Windows Service

3 MpCCI FSIMapper Command

4 MpCCI FSIMapper GUI
   4.1 Starting the MpCCI FSIMapper
   4.2 The “What to map” Panel
      4.2.1 Selection of Cases, Parts and Quantities
      4.2.2 Geometry Compare
      4.2.3 Quantity Identification for CSM Solver Output
      4.2.4 Execute a Mapping Process
   4.3 The “Transformation” Panel
      4.3.1 The “Geometry” Subpanel
      4.3.2 The “Quantity” Subpanel
   4.4 The “How to map” Panel
      4.4.1 Mapping Algorithms and Neighborhood Parameters
      4.4.2 Orphan Filling
      4.4.3 Quantity Location
      4.4.4 Saving Mapping Configurations
   4.5 The “Result” Panel
      4.5.1 Average over Rotation Axis
      4.5.2 Apply Fourier Transformation
      4.5.3 MSC NASTRAN Export Options
   4.6 The “Preferences” Panel
   4.7 The “Harmonic Wizard” Panel
   4.8 The “Log” Panel

Contents
5 Codes and Formats Information

5.1 EnSight Gold ........................................ 32
  5.1.1 Supported Quantities ............................... 32

5.2 FLUENT .............................................. 32
  5.2.1 Supported Quantities ............................... 32
  5.2.2 Exporting wallfunchtC to UDM0 in Data File .... 33

5.3 FINE/Turbo .......................................... 33
  5.3.1 Supported Quantities ............................... 33

5.4 CFX .................................................. 34
  5.4.1 Supported Quantities ............................... 34

5.5 FloTHERM ........................................... 34
  5.5.1 Supported Quantities ............................... 34
  5.5.2 Combining Static Result Files ...................... 34

5.6 FloEFD .............................................. 34
  5.6.1 Supported Quantities ............................... 35

5.7 MagNet ............................................. 35
  5.7.1 Limitations ......................................... 35
  5.7.2 Elements ........................................... 35
  5.7.3 Supported Quantities ............................... 35

5.8 ANSYS Maxwell ..................................... 35
  5.8.1 Elements ........................................... 36
  5.8.2 Supported Quantities ............................... 36

5.9 Abaqus ............................................. 36
  5.9.1 Limitations ......................................... 36
  5.9.2 Model Preparation ................................ 36
  5.9.3 Include Boundary Conditions ....................... 37
  5.9.4 Elements ........................................... 38
  5.9.5 Supported Quantities ............................... 38

5.10 ANSYS ............................................ 39
  5.10.1 Requirements ..................................... 39
  5.10.2 Model Preparation ................................ 39
  5.10.3 Supported Quantities ............................... 41
  5.10.4 ANSYS Scanner and Converter ................. 41

5.11 MSC NASTRAN .................................... 41
  5.11.1 Limitations ....................................... 41
  5.11.2 Include Boundary Conditions ...................... 42
  5.11.3 Elements .......................................... 43
  5.11.4 Supported Quantities ............................... 43

5.12 LS-Dyna ........................................... 43
5.12.1 Limitations .................................................... 43
5.12.2 Elements .................................................... 43
5.12.3 Supported Quantities ...................................... 44
5.12.4 Include Boundary Conditions ............................. 44

6 Batch Usage of the MpCCI FSIMapper .................. 45
6.1 File Scanners ................................................... 45
  6.1.1 FLUENT ................................................ 45
  6.1.2 MentorGraphics ........................................ 47
  6.1.3 FloEFD .................................................. 47
  6.1.4 FloTHERM ............................................. 48
  6.1.5 ANSYS .................................................. 48
  6.1.6 Abaqus ................................................. 49
  6.1.7 EnSight Gold Case ..................................... 50
6.2 Comparing Geometries ...................................... 53
6.3 Mapping Quantities ....................................... 54
6.4 Files Written by the MpCCI FSIMapper ............... 54
6.5 Configuration File ........................................ 54
6.6 Example for a Configuration File ....................... 57

7 Theory Guide .................................................. 60
7.1 Numerical Methods ........................................ 60
  7.1.1 Mapping Algorithms .................................. 60
  7.1.2 Parameters for the Mapping ........................... 60
  7.1.3 Orphan Filling .......................................... 62
7.2 Geometry and Quantity Transformations ............... 63
  7.2.1 Fourier Transformation and Windowing ............. 63
  7.2.2 Cyclic Symmetry of Quantities ....................... 64

8 Tutorial ......................................................... 66
8.1 Mapping of Electromagnetic Forces for Transient and Harmonic Analyses ............... 66
  8.1.1 Problem Description .................................. 66
  8.1.2 Source Result File .................................... 67
  8.1.3 Target Mesh File ...................................... 70
  8.1.4 Mapping ............................................... 70
  8.1.5 Target Simulation ..................................... 76
8.2 Mapping of Harmonic Pressure Excitations in Turbomachinery ..................... 77
  8.2.1 Using the Harmonic Balance Method of STAR-CCM+ ...................... 77
  8.2.2 Using the Nonlinear Harmonic Method of FINE/Turbo ................ 89
8.3 Mapping of Temperature for Microelectronic Devices ............................. 98
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.3.1 Problem Description</td>
<td>98</td>
</tr>
<tr>
<td>8.3.2 Source Result File</td>
<td>98</td>
</tr>
<tr>
<td>8.3.3 Target Mesh Files</td>
<td>99</td>
</tr>
<tr>
<td>8.3.4 Mapping of Temperature</td>
<td>99</td>
</tr>
<tr>
<td>8.3.5 Results and Target Simulation</td>
<td>100</td>
</tr>
</tbody>
</table>
1 MpCCI FSIMapper Overview

Mapping
For many problems the influence of the CFD solution on the CSM solution is a lot more remarkable and significant than vice versa. If a one-way connection between the fluid and the solid solver shall be established, quantity values – namely temperature or pressure values that were calculated as a part of the CFD solution – have to be transferred from the CFD to the FEM mesh. Then the structural mechanics solver can use these values as a boundary condition. At the interface between the fluid and solid domain the two meshes usually do not match and interpolation of quantities becomes necessary. This transfer of quantity values from one mesh to another can be done easily with the MpCCI FSIMapper. While “one–way” fluid–structure interactions certainly are the main application of the MpCCI FSIMapper, the tool can be used generally to map values from one mesh to another.

Supported software
At the moment the MpCCI FSIMapper is able to read FLUENT (.cas and .dat), CFX (.csv), MentorGraphics FloTHERM (.flofea, .csv), MentorGraphics FloTHERM XT (.txt), MentorGraphics FloEFD (.efdfea), MagNet (.vtk), FINE/Turbo (.cgns), MSC NASTRAN Bulk (.bdf, .nas, .dat), Abaqus (.inp), ANSYS Maxwell (*.unv) and EnSight Gold Case as source result files. Since the EnSight Gold Case export is also supported by CFX, FLUENT, ANSYS Icepak, STAR-CD, STAR-CCM+, etc., so these simulation codes are supported indirectly.

Supported native CSM codes are Abaqus (.inp) input decks, ANSYS (.db and .cdb), MSC NASTRAN Bulk (.bdf) data input files, LS-Dyna keyword (.key, .k, .dynain) and also neutral EnSight Gold Case file format is supported.
**Supported meshes and quantities**

The two meshes between the interpolation shall take place have to be 2-dimensional surface or 3-dimensional volume meshes in the 3-dimensional space. For Abaqus and MSC NASTRAN also 1-dimensional surfaces of 2-dimensional meshes are supported.

The physical quantities that can be read and mapped are temperature, wall heat transfer coefficient, the wall heat flux, pressure and force density. Furthermore, it is possible to compare two geometries with the MpCCI FSIMapper.

**MpCCI FSIMapper output files**

For Abaqus, ANSYS, MSC NASTRAN and LS-Dyna mapped quantity values can be written in native solver format, e.g. Abaqus input, ANSYS APDL, MSC NASTRAN Bulk and LS-Dyna keyword format respectively, that can be easily included in the original CSM input deck. In addition to native format, the MpCCI FSIMapper allows export to neutral EnSight Gold Case format which is supported by a wide range of simulation software.

Additionally, the MpCCI FSIMapper saves the source and the target model in the native .ccvx format of MpCCI Visualizer. In this way the mesh and data used for and resulting of the mapping can be visualized afterwards.

For mapped harmonic data or Fourier transformed transient data the MpCCI FSIMapper exports a .ccvx file where the complex excitations are transferred back to the time domain (per frequency). Thus, the real and imaginary part of the excitations can be visualized as transient fluctuations (using pseudo time steps).

**Supported operating systems**

The tool is available for Windows and Linux for 64-bit machines in graphical and batch mode.

**MpCCI FSIMapper concept**

The MpCCI FSIMapper comes with a graphical user interface where all relevant parameters for the mapping can be set. The usage of the tool is quite simple: after selecting the models and parts some choices concerning algorithms and parameters have to be made. After the mapping the results and the original values are loaded into the MpCCI Visualizer.

The immediate visualization of the results makes it possible to detect model errors or bad mapping results due to parameter problems. The user can identify the critical areas and start a new mapping process.

Additionally, the MpCCI FSIMapper can be used as a batch tool. The parameters and settings that are normally made in the GUI can be defined in a simple ASCII configuration file.
2 MpCCI FSIMapper Installation

The installation of the MpCCI FSIMapper is quite easy. It may be either part of the MpCCI installation or a standalone version.

2.1 Part of MpCCI Installation

As part of the MpCCI installation the MpCCI FSIMapper is completely integrated into the MpCCI environment.

No further actions have to be done.

2.2 Standalone Version

The installation of the standalone version is based on the following steps:

- The MpCCI FSIMapper has to be installed.
- Perl has to be installed.
- A license for the MpCCI FSIMapper must be available.

There are two ways to install the MpCCI FSIMapper:

- via the Windows installer for a local Microsoft Windows only installation and
- via the multi-platform distribution file for Linux and file server installations.

A Microsoft Windows installation requires a free disc space of approx. 50 MB and a multi-platform installation of approx. 100 MB (without tutorials).

2.2.1 Local Microsoft Windows Installation

For a local Microsoft Windows installation you download an "MSI" file.

Execute this installer file with administrative rights and follow the instructions. The installer unpacks the MpCCI FSIMapper for all users, does registry entries and adds firewall rules which are necessary for the communication between the MpCCI FSIMapper and the MpCCI Visualizer. It also checks if Perl is installed.

If no Perl executable could be found in the path, the user is offered an Open button to download Strawberry Perl. This installation also takes place via a Microsoft Windows installer ("MSI" file).

The MpCCI FSIMapper software is now installed on your system.

For running the MpCCI FSIMapper a license must be available (see \(\triangleright\) 2.2.4 License \(<\)).

If you did not install Perl or if your Perl installation is too old see \(\triangleright\) 2.2.3 Perl \(<\).

2.2.2 Linux or Multi-Platform Installation

If you intend to install MpCCI FSIMapper under Linux or on a file server with access for Linux and Microsoft Windows users you should have downloaded a multi-platform distribution file.

The installation is just done by unpacking the downloaded archive to a folder of your choice. Then the PATH environment variable needs to be extended by the "bin" directory of the MpCCI FSIMapper. For Linux everything can be done without any administrator privileges if you use a local installation directory. For Windows administrative rights are necessary to add firewall rules.

The steps in detail:
1. Unpack the downloaded archive on Linux:
   - Change to your installation directory e.g. `cd <your home>/software`.
   - Move the downloaded file to your installation directory
   - Extract the MpCCI FSIMapper files from the downloaded file with the command
     ```bash
tar xzvf mpccifsimapper-<FSIMapper-version>.tar.gz
```
   or, if the z option is not available, with the commands
   ```bash
gunzip mpccifsimapper-<FSIMapper-version>.tar.gz
tar xvf mpccifsimapper-<FSIMapper-version>.tar
```

2. Update your user PATH environment with the MpCCI FSIMapper "bin" directory:
   - **Linux:**
     ```bash
     export PATH=<your install dir>/FSIMapper/bin:$PATH
     ```
     for korn shell (ksh) and bash resp.
    ```bash
     setenv PATH <your install dir>/FSIMapper/bin:$PATH
     ```
     for c-shell (csh) and tc-shell (tcsh) users.
   - **Windows:**
     In the Windows Control Panel go to where you can change own user environment variables.
     Extend the PATH environment variable by
     ```bash
     <your install dir>/FSIMapper/bin
     ```

   In the Control Panel → System and Security → Windows Firewall select Allow a program or feature through Windows Firewall. Now click on Allow another program... and add
   ```bash
   "<your install dir>/FSIMapper/bin/windows_x64/mpcci_fsimapper.exe" and also add
   "<your install dir>/FSIMapper/bin/windows_x64/mpcci.visualizer.exe" and finally add
   "<your install dir>/FSIMapper/bin/windows_x64/mpcci_ccvxcat.exe"
   ```

The MpCCI FSIMapper software is now installed on your system.

For running the MpCCI FSIMapper a license must be available (see ▶ 2.2.4 License ◄).

If you run into problems with Perl while executing the MpCCI FSIMapper have a look at ▶ 2.2.3 Perl ◄.

### 2.2.3 Perl

Perl is a platform independent script language which allows running identical scripts under Linux as well as Microsoft Windows. The MpCCI FSIMapper uses perl scripts for executing its commands. Therefor it requires a working Perl installation. If Perl is already installed on your system - this is true for nearly any Linux system - you may check the Perl version by typing
```
> perl -version
```

For MpCCI FSIMapper under Linux you need at least Perl 5.6. Under Microsoft Windows you can either use Strawberry Perl or ActivePerl from version 5.8 up.

If Perl is not installed or if the version is too old, please contact your system administrator for installation resp. upgrade. On Linux systems it’s part of the distribution but Perl is never part of your Microsoft Windows system. For Microsoft Windows it can be downloaded from the world wide web. We recommend to have either Strawberry Perl or ActivePerl from version 5.8 up installed.

Strawberry Perl is free of charge and can be downloaded from strawberrypi.com/. A 64 bit version of Strawberry Perl for Windows 64 bit is available.

ActivePerl is covered by the ActiveState Community License and can be downloaded from www.activestate.com/activeperl.

Perl is not part of the MpCCI FSIMapper installation. Only for the local Microsoft Windows installation (▶ 2.2.1 Local Microsoft Windows Installation ◄) the download of the third party software Strawberry Perl is integrated. The installation of Strawberry Perl should be done as the next step.
2.2.4 License

2.2.4.1 Installing the MpCCI License Server

MpCCI FSIMapper uses a FlexNet Publisher based floating license mechanism. FlexNet Publisher license server has to be started on the license server host defined in the MpCCI FSIMapper license file (see 2.2.4 Requesting a License). You can run MpCCI FSIMapper anywhere in your internal network. For more information about FlexNet Publisher, please refer to the FlexNet Publisher license administration guide.

You will find the software packages for the MpCCI license server installation (FlexNet Publisher tools and vendor daemons for SVD) in the SCAI download area (www.mpcci.de → MpCCI Download Area → enter your username and password and login → License Tools). Please select the suitable platform and execute the installer for Microsoft Windows resp. unpack the downloaded package for Linux.

In general FlexNet Publisher comes with some tools (lmutil, lmstat, lmhostid, lmgrd, lmdown etc.) for managing the licenses - please refer to the FlexNet Publisher documentation.

MpCCI FSIMapper only needs three FlexNet Publisher executables:

- the utility: lmutil
- the license server: lmgrd
- the vendor daemon: SVD

The MpCCI FSIMapper vendor daemon is named SVD. The FlexNet Publisher port number of the SVD vendor daemon is 47000. If there are other software packages installed using also FlexNet Publisher there will be several FlexNet Publisher utils available. Depending on your local installation and your own PATH environment it is not always defined which of these lmutils will be executed upon a command call.

2.2.4.2 Requesting a License

You need to acquire a license file for the MpCCI FSIMapper from Fraunhofer SCAI.

Therefor you need to know the hostname and the hostid of the license server:

Please login on the host where the FlexNet Publisher license server for MpCCI FSIMapper should run on. In the following example, “> ” is your prompt:

```
> hostname
myHostName
> lmutil lmhostid -n
12345abcd
```

If you have multiple network devices installed (ethernet card, wireless LAN, docking station on a notebook) you may see multiple hostids. For the MpCCI FSIMapper license daemon the integrated ethernet card address should be the correct one.

Please go to the download area at www.mpcci.de to get an MpCCI FSIMapper license file.

If you already have an account for the download area the MpCCI FSIMapper license file can be requested via the License Request Form Sheet in the SCAI download area (www.mpcci.de → MpCCI Download Area → enter your username and password and login → License Request).

Please fill in the form with all required data - including the hostname and hostid of the license server (myHostName and 12345abcd in the example above) - select FSIMapper in the Mapper section, leave selections empty for MpCCI Jobs, Parallel Clients, Code Adapter and AddOns and submit. You will receive the required license file soon after via e-mail.
Please copy your received license file into the file "<fsimapper_home>/license/mpcci_fsimapper.lic". Then, please start the FlexNet Publisher license server daemon and SVD vendor daemon on the local host with the following command:

```
> lmgrd -c <LICENSE_FILE>
```

In the case of problems you can get information about the license server in a logfile with the command:

```
> lmgrd -c <LICENSE_FILE> -l <LICENSE_LOG_FILE>
```

To stop the license server please type:

```
> lmutil lmdown -c <LICENSE_FILE>
```

You should set the generic FlexNet Publisher variable SVD_LICENSE_FILE. For ksh or bash users:

```
> export SVD_LICENSE_FILE=47000@<HOSTNAME>
```

For csh users:

```
> setenv SVD_LICENSE_FILE 47000@<HOSTNAME>
```

For Microsoft Windows users:

```
> set SVD_LICENSE_FILE=47000@<HOSTNAME>
```

This variable is also referred to by other software packages using FlexNet Publisher; SVD_LICENSE_FILE may comprise several entries <port>@<host> (one for each application). If there is more than one entry they must be separated by a "." on Linux and by a ";" on Microsoft Windows.

We suggest that you set the SVD_LICENSE_FILE environment variable in your login file (".cshrc" or ".profile" or ".bashrc") for Linux users resp. for Microsoft Windows users in the Environment Variables window of the system settings.

FlexNet Publisher stores data of a previous successful connection to a license server in the file "<HOME>/.flexlmrc". You may create or adjust this file.

After setting this environment variable you can check whether the license server is running and can be reached from the machine where you are currently working and from where you plan to start MpCCI FSimapper:

```
> lmutil lmstat -vendor SVD
```

For more details command

```
> lmutil lmstat -a -c <LICENSE_FILE>
```

### 2.2.5 Configure a License Manager as UNIX Service

On UNIX, edit the appropriate boot script, which may be "/etc/rc.boot", "/etc/rc.local", "/etc/rc2.d/Sxxx", "/sbin/rc2.d/Sxxx", etc. Include commands similar to the following.

```
<LICENSE_TOOL_INSTALLATION>/bin/lmgrd -c <LICENSE_FILE> -l <LICENSE_LOG_FILE>
```

The LICENSE_TOOL_INSTALLATION is the full path of the license software installation. This command will create a log file under "LICENSE_LOG_FILE" and you have to ensure that the process could write in
the directory.

2.2.6 Configure a License Manager as Windows Service

Execute the "lmtools.exe" application from the license manager installation directory: "<LICENSE_TOOL_INSTALLATION>/bin/lmtools.exe"

- Select in the Service/License File tab section the option Configuration using Services.
- Click the Config Services tab section.

- Enter a service name e.g. MpCCI license manager or MpCCI FLEXlm.
- Select the path of the program "lmgrd.exe" with the [Browse] button:
  "<LICENSE_TOOL_INSTALLATION>/bin/lmgrd.exe"
- Select the license file "mpcci.lic" with the [Browse] button:
  "<LICENSE_TOOL_INSTALLATION>/license/mpcci.lic"
- Activate the Start Server at Power Up option.
• Activate the Use Services option.
• You can optionally add a log file by providing a file name for the Path to the debug log file option.
• Click on the **Save Service** button.

![Image](image.png)

• Select in the Start/Stop/Reread tab section the license service.
• Click on the **Start Server** button.
• The license server is now running and configured to start at power up.
3 MpCCI FSIMapper Command

Usage:

    fsimapper [-]option

Synopsis:

    'fsimapper' is used to launch the MpCCI FSIMapper.

Options:

    -batch <configFile> <source> [source_quant] <target>
    Use this option to launch the MpCCI FSIMapper in batch mode.
    Provide a configuration file, source and target model files and
    an optional source quantity file.

    -help
    This screen.

    -scan <FORMAT> <model>
    Use this option to scan models with the MpCCI FSIMapper.
    Supported scanner formats are ABAQUS, ANSYS, LSDYNA, FLUENT,
    NASTRAN, CFXCSV, FLOHERMMAPLIB, FLOEDMAPLIB, FLOHERMXT,
    FINETURBO, MAXWELL and ENSIGHT.
    The scanner output will be written to stdout.

    -convert <FORMAT> <fileToConvert> [releaseNumber] [ansysProduct]
    Use this option to convert an input file of special format
    into own intermediate format '.ml'.
    At this time only ANSYS '.cmd' and '.db' files are supported
    to be converted so FORMAT must be ANSYS.
    Optional arguments are the release number for the used ANSYS
    executable and the ANSYS product to use for starting ANSYS.
    By default the latest ANSYS release will be taken.

    -visualize <ccvx-File-1> <ccvx-File-2> ...
    Opens MpCCI Visualizer and loads all passed CCVX files

The batch usage is described in §6 Batch Usage of the MpCCI FSIMapper § the scan usage in §6.1 File Scanners § and the convert usage in §5.10.4 ANSYS Scanner and Converter §.

The MpCCI FSIMapper itself is described in §4 MpCCI FSIMapper GUI §.
4 MpCCI FSIMapper GUI

This section describes the usage of the MpCCI FSIMapper. The different panels of the graphical user interface are described in the next sections. Details on the used mapping and orphan filling parameters can be found in \( \text{7.1 Numerical Methods} \).

4.1 Starting the MpCCI FSIMapper

The MpCCI FSIMapper GUI is embedded in the MpCCI Visualizer and can be started from command line by

```
> fsimapper
```

launching the MpCCI Visualizer with activated plugin for file based mapping. To open the MpCCI FSIMapper GUI press button \( \text{ after the MpCCI Visualizer has started.} \)

![Figure 2: The MpCCI FSIMapper plugin location in the MpCCI Visualizer](image)

The MpCCI FSIMapper writes a configuration setup file containing user model and part selection and launches a separate process which executes the defined tasks. When finished the models together with quantities are being sent to the MpCCI Visualizer where each model appears in a separate viewport.
4.2 The “What to map” Panel

![Figure 3: The “What to map” panel of the MpCCI FSIMapper](image)

4.2.1 Selection of Cases, Parts and Quantities

In the “What to map” panel of the mapping dialog, at first the type of analysis needs to be specified. The user can switch between “Static”, “Transient” and “Harmonic”. Using the “Static” panel means to map one state from a source model to a target model. For the “Transient” panel it is assumed that the data is present for several time steps. Additionally to the creation of transient loading, there is the possibility to perform a Fourier transformation on the transient data in order to do a frequency response analysis, cf. ▷4.5.2 Apply Fourier Transformation◁.

In the “Harmonic” panel, complex loading is assumed in order to map excitation loads for NVH analyses.

At second the source and target models need to be specified. The file type for the two models can be selected in a drop-down list. The files can be selected with the open file dialog.
Source File Specification

For a “Static” source model FLUENT, Mentor Graphic’s FloTHERM, FloTHERM XT and FloEFD, CFX, FINE/Turbo, Infolytica’s MagNet and EnSight Gold Case are supported. Moreover, the MSC NASTRAN bulk data format and the Abaqus input format is supported.

For a “Transient” source model MSC NASTRAN, MagNet, FloTHERM, ANSYS Maxwell and EnSight Gold are supported. The target model can be either in Abaqus, ANSYS, LS-Dyna or MSC NASTRAN format. Currently only the “Force”, “Force Density” and “Pressure” variables are supported for transient analyses.

For a “Harmonic” source model FINE/Turbo and EnSight Gold are supported for the “Pressure” variable. The target model can be either in Abaqus, ANSYS or MSC NASTRAN format.

As harmonic CFD simulations often result in a set of harmonic quantities, the “Harmonic Wizard” offers the possibility to map all or a subset of the available harmonics at once. Refer to 4.7 The “Harmonic Wizard” Panel.<

For transient FloTHERM .csv format as target code only Abaqus is currently supported.

Depending on the selected source code, the following files are required:

- FLUENT : the native geometry (*.cas) and quantity (*.dat) file
- FloTHERM : resp. FloEFD (*.floea, *.csv) resp. (*.edflea) file exported by the code
- FloTHERM XT: (*.txt) file exported by the code
- EnSight Gold : (*.case) file which references the binary (*.geom) file and the binary quantity file(s)
- MagNet : the Infolytica “MpCCI Exporter” format *.vtk, where multiple files can be selected (defining multiple parts)
- CFX : the “CFD-Post Generic Export” format (*.csv)
- FINE/Turbo : the native (*.cgns) file and optionally the (*.run) file
- Abaqus : ASCII file in the native input file format (*.inp)
- MSC NASTRAN : ASCII file in the native bulk data format
- ANSYS Maxwell: the Maxwell export of geometry (*.unv) and quantity (*.unv) file

Target File Specification

The target model can be either in Abaqus, ANSYS, MSC NASTRAN, LS-Dyna or EnSight Gold Case format. Besides ASCII files in Abaqus input format, MSC NASTRAN bulk data format or ANSYS format, the binary ANSYS format .db and the binary EnSight Gold Case is supported.

Parts Selection

After the selection of the models, their files are scanned automatically and the available model parts are listed in the selection area below. For better distinction of area types the MpCCI FSIMapper lists each “part” by leading ‘V’ for volumetric parts, ‘S’ for surfaces or ‘E’ for element sets (Abaqus only). By clicking on its name, relevant “parts” for the mapping can be selected. In the most common case this is the interface between the fluid and a solid in fluid-structure interaction.

Either homogen volumes, surfaces or element sets can be selected for mapping. A composition of different area types is not supported and has to be mapped separately.
The “parts” selection can be performed by using the string matching function. Provide the string to match in the input field and click on the button Select to activate the selection. By clicking on the button Clear it will clear the current selection of “parts”.

Unit Systems

The next step in the GUI is to define the corresponding unit systems for the source and the target meshes. Figure 4 illustrates the part in the “What to map” panel where length, mass, temperature and time units can be defined for both models independently. This is useful when the source and target model length units for the model geometry or the quantity units do not match.

All mapped and displayed quantities are converted and exported due to the defined target unit system.

Figure 4: Overview of unit system configuration in graphical user interface

4.2.2 Geometry Compare

For the source and the target models it is possible to tick a check box labeled “Geometry Compare”. If this option is selected for the source or target model, additional geometric information is computed and the MpCCI Visualizer receives the two additional quantities NODAL_DISTANCE and ASSOCIATION_DISTANCE. If only the geometry of the models shall be compared without mapping quantities, this option has to be selected for both models.

For each node of one mesh the NODAL_DISTANCE is closest geometric distance to a point of the other mesh. However the ASSOCIATION_DISTANCE denotes the distance between a point of a mesh and the closest (associated) element surface of the other mesh. If the point lies “in” an element, this would mean that the association distance is 0.

4.2.3 Quantity Identification for CSM Solver Output

As the notation of a physical quantity may differ among native solvers file formats, the user has to specify the physical quantity by its name so that it can be used for export to either Abaqus input, MSC NASTRAN Bulk files or ANSYS Mechanical APDL files. A selection of the relevant quantities can be done in the lower left part of the “What to map” panel, cf. Figure 6.
For each of the supported physical quantities “Temperature”, “HTC” (Heat Transfer Coefficient), “Heat Flux”, “Pressure”, “Force” and “Force Density” there exists a drop-down menu where one of the quantity ids of the source file can be selected.

In some cases of the harmonic mapping it is obligatory to specify both real and imaginary part of the excitation quantity, cf. 4.3 The “Transformation” Panel. Having in mind that a complex excitation amplitude can be reformulated as amplitude and phase lag of a periodic fluctuation (Euler’s formula), it is more realistic to specify both the real and imaginary part of the quantity. If only the real part or only the imaginary part is given, all excitations will vibrate exactly in-phase or out-of-phase (only 0° or 180° phase shift). In this case the vibration amplitudes are the absolute values of the given data.

### 4.2.4 Execute a Mapping Process

If the actual configuration inside the “What to map” panel is in a valid state, pressing the “Map” button - which is available on all panels - starts the mapping process. A valid mapping state is present if the following prerequisites are satisfied:

- A source model file is specified
- A source quantity file is specified (FLUENT, MSC NASTRAN, Abaqus and ANSYS Maxwell only)
- A target model file is specified
- At least one part of the source model is selected
- At least one part of the target model is selected
- Quantity selection dependent on the analysis type

#### Static and Transient panel

At least one physical quantity selection is made OR both geometry compare boxes are checked

#### Harmonic panel

Depending on cyclic symmetry:

- **none**
  
  At least one physical quantity selection is made OR both geometry compare boxes are checked

- **cyclic symmetric**
  
  Depending on data periodicity (cf. 4.3 The “Transformation” Panel):
  
  * **constant/alternating**
    
    At least one physical quantity selection is made OR both geometry compare boxes are checked

  * **paired**
    
    At least both real and imaginary parts of one physical quantity selection are made OR both geometry compare boxes are checked
If the current state is valid the “Map” button gets available otherwise it is disabled.
### 4.3 The “Transformation” Panel

#### 4.3.1 The “Geometry” Subpanel

A common problem in one-way one-step coupling is the use of different local coordinate systems for CFD and CSM simulation. Here the models might not have the same spacial location in a common coordinate system or even represent only a symmetric part of the target model. Hence, for a position-dependent neighborhood computation, both models need to be aligned in a preprocessing step. This alignment step can be done in the “Transformation” panel (Figure 7) where different options for mesh positioning can be selected in the “Geometry” subpanel.

**None**

No mesh positioning is needed.

**Align**

Select one of the four different automatic alignment options:

**Automatic**

The MpCCI FSIMapper first applies a “Coarse” transformation followed by a “Fine” transformation on the source mesh to move it close to the target mesh location.
Coarse

The MpCCI FSIMapper computes the center of gravity of both models and translates the source towards the target. Then the principal axis of the models will be aligned.

Fine

The MpCCI FSIMapper minimizes the global total nodal distance from the source to the target model.

Bounding Box

The MpCCI FSIMapper computes the global expansion of the models and then translates the source model bounding box to the target one.

User defined

As an automatic alignment might fail in case of symmetric geometry or partial overlap of models the MpCCI FSIMapper also offers an alignment by user defined transformation.

Translate and Rotate

Here for each model the user can define a 3x3 rotation matrix $R$ as well as a translation vector $T$ so that new model coordinates $x'$ are computed as

$$x' = Rx + T.$$  

Only a source mesh gets transformed into the target mesh coordinate system

$$x' = R_t^{-1}R_s x + R_t^{-1}(T_s - T_t)$$

where the lower index $s$ denotes source and $t$ target transformation.

Cyclic Symmetry

This transformation provides for cyclic symmetric source models the corresponding full mesh and quantity. In this way a mapping to a full target model or to a periodic target model with a different section shape is possible, as shown in Figure 8. The user needs to define the number of sections as Number of Periodicities and two points (corresponding to the source model position) which define the axis of cyclic symmetry (Cyclic Axis Start Point and Cyclic Axis End Point).

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure8.png}
\caption{Mapping of data between different periodic sections (black lines) which represent the same full model (grey lines)}
\end{figure}
Reflection Symmetry

This transformation provides for mirror symmetric source models the corresponding full mesh and quantity by reflection. In this way a mapping to a full target model is possible, as shown in Figure 9. The user needs to define the Plane’s Origin and the Plane’s Normal vector of the symmetry plane.

![Reflection Symmetry Diagram](image)

Figure 9: Mapping of data between a mirror symmetric source model and the corresponding full target model.

4.3.2 The “Quantity” Subpanel

In the “Quantity” subpanel (Figure 10), transformations of the quantity can be defined. Following options are available:

None
No quantity transformation is needed.

User defined

Cyclic Symmetry
Enter the cyclic symmetry mode i.e. the Nodal Diameter and the excitation direction (Mode) of the transient or harmonic quantity in order to define the quantity transformation for cyclic symmetric models. For further information refer to \(7.2.2\) Cyclic Symmetry of Quantities.

Filter Transient Id Range
The data import of transient result data, e.g. pressure and forces, can be truncated using the following options

Filter Transient Id Range
Read transient data beginning and ending at the time step integer IDs given by Start Id and End Id. With the option Step it is possible to read only every \(n\)-th time step id.

Filter Time Range
Read transient data beginning at the time value given by Start Time for a defined Period. With the option Step it is possible to read only every \(n\)-th time step id.
4.4 The “How to map” Panel

Additional settings concerning the type of algorithm used for the mapping, the necessary parameters, the used orphan fillers or the quantity location can be made in the “How to map” panel (Figure 11).

4.4.1 Mapping Algorithms and Neighborhood Parameters

The MpCCI FSIMapper comes along with three different mapping algorithm approaches that can be used for data interpolation between meshes: the “Shape function”, the “Nearest” and the “Weighted Element” algorithm. Fundamental difference between shape function / weighted element and nearest interpolation is the underlying neighborhood relation which is computed. For shape function and weighted element interpolation an element-node relation is required in contrast to nearest interpolation which uses node-node relation. Detailed descriptions of the algorithms can be found in ∇ 7.1 Numerical Methods ◄.

For each type of mapping algorithm there exist different parameters that control the mapping process. After the selection of the algorithm type the relevant parameters are enabled and can be changed below in the “Neighborhood parameters” section (Figure 18).

It is possible to use “Default” parameter selection which should produce good mapping results for many
applications where geometries do not differ too much but they may result in a quite time-consuming neighborhood computation. However, depending on the meshes it might be necessary to adjust the default parameters.

### 4.4.2 Orphan Filling

Depending on the type of mapping algorithm used, certain elements or nodes of the target geometry may not receive values from the source mesh, for example if the two parts - target and source - do not have the same extension in one direction. These nodes or elements are called “orphans”.

The MpCCI FSIMapper provides different methods to assign values to these “orphans”. These can be selected in the “Orphanfiller” section on the “How to map” panel. Just like the mapping algorithms the orphan filling methods need some parameters that can be edited after selecting the fill type.

Further information about the different fill methods and parameters can be found in $\Rightarrow$ 7.1 Numerical Methods $\alpha$, where the numerical methods are described.
4.4.3 Quantity Location

Finally – in the last part of the “How to map” panel – the type, i.e. the location of the target quantity can be chosen. By default, the mapped quantity on the target mesh will be located on the same entities (either element or node) as the source quantity.

Additionally the possibility exists to force the target quantity values to be located on either elements or nodes – independent of the source quantity location.

4.4.4 Saving Mapping Configurations

At the bottom of the “How to map” panel the button “Save” provides the capability to store the current configuration – the selected algorithms and parameters – in a configuration file. This offers the possibility of interactive adjustment and optimization of neighborhood parameters for future use in batch mode production runs.

4.5 The “Result” Panel

Figure 12: The “Result” panel of the MpCCI FSIMapper
4.5.1 Average over Rotation Axis

The first part of the “Result” panel is only relevant if the CFD simulation uses a so-called “Frozen Rotor” model. In this case the rotation of a geometry part is not modeled with a moving geometry but with the help of additional source terms. This makes the simulation of rotating parts a lot simpler.

The FSIMapper offers the possibility to average these simulation results of a frozen rotor computation by mapping the values on the rotating CSM mesh. Select “Compute Rotative Average” when required.

After ticking the box “Export Result Options” the rotation axis of the target mesh and rotation angle in angular degree need to be specified. The mapping process then is done for a 360° rotation of the target model split up into $N$ substeps where each step is defined by a rotation by the user defined rotation angle around the axis defined through axis start and end point.

4.5.2 Apply Fourier Transformation

MpCCI FSIMapper allows to perform Fourier transformations of transient quantities (currently pressure, force and force density supported). The resulting complex quantities can be used in frequency response or spectrogram analyses.

To export loads for this type of analyses select Apply Fourier Transformation when having selected the “Transient” panel in the “What to map” panel.

The following variants are available:

Single Frequency Response Analysis
The transient data is mapped to the target model and for each node/element the quantity is reformulated by the Fourier transformation as frequency dependent complex load.

Spectrogram
The time period is devided into several, equally long, potentially overlapping subwindows. In each subwindow a Fourier transformation is performed in order to create complex frequency dependent loads changing over time. Define the Number of Subwindows and the Overlap (between 0 and 0.99).

For a signal of a total length $L$, a number of subwindows $m$ and an overlap of $p$, the subwindows have a size of $L_S = \lfloor L / (m - (m - 1) \cdot p) \rfloor$ time steps. The overlap consists of $\lfloor L_S \cdot p \rfloor$ time steps.

Windowing is recommended here.

The created complex data is only exported up to the given Maximal Frequency, where the default $-1$ means no export truncation.

Windowing is available through the selection of a pre- or user defined Window Function, see ▶ 7.2.1 Fourier Transformation and Windowing ◄.

The time steps have to be constant and at least two time steps need to be present in the transient source model for the use of this option.

If the time steps are only equidistant within a certain tolerance, MpCCI FSIMapper will take the mean time step calculated by the difference of the last and the first time step divided by the number of time steps minus 1. A warning is displayed in the Log panel.

For further information about the Fourier transformation refer to ▶ 7.2.1 Fourier Transformation and Windowing ◄.

For visualization of the complex data a .ccvx file is written, where for each frequency content of the time signal, the complex quantity (here $\hat{q} = x + iy$) is inverse Fourier transformed into $n$ time steps by

$$q_j = A \cdot \cos(2\pi \cdot j/n + \phi), \quad j = 1, \ldots, n$$

with the amplitude $A = \sqrt{x^2 + y^2}$ and the phase shift $\phi = \text{atan2}(y, x)$. For vector quantities this is done
The ccvx-file can be read into MpCCI Visualizer by File → Open.... By the “Play” button the generated time steps of the selected frequency (“Scalars” and “Vectors” in “Result Panel”) can be visualized.

### 4.5.3 MSC NASTRAN Export Options

Mapped quantities can be included as initial or boundary condition in native CSM solver format, i.e. Abaqus input, ANSYS APDL and MSC NASTRAN Bulk. For definition of MSC NASTRAN temperature and pressure loads there exist several card formats and a unique load id (SID) needs to be specified. Therefore MpCCI FSIMapper supports the export of temperature either in SPC (MSC NASTRAN single point constraint) or TEMP format, pressure can be exported as PLOAD, PLOAD2 or PLOAD4. In “Result” panel (Figure 12) both temperature and pressure format can easily be selected in a combo box and unique SID can be assigned.

### 4.6 The “Preferences” Panel

![Image of the Preferences panel of the MpCCI FSIMapper](image)

Figure 13: The Preferences panel of the MpCCI FSIMapper
The “Preferences” panel allows just two minor changes: first the user can change the working directory of the MpCCI FSIMapper. The files that are written during the mapping – .vtk files for visualization and include files for a CSM simulation – can be found in the working directory. Additionally, a box can be ticked if all intermediate files shall be deleted after the mapping is finished.

4.7 The “Harmonic Wizard” Panel

![Figure 14: The “Harmonic Wizard” panel of the MpCCI FSIMapper](image)

The “Harmonic Wizard” is designed to map a set of harmonic turbomachinery CFD results at once. It prepares all the necessary information for a mapping between periodic models.

By default, the “Harmonic Wizard” panel is disabled. It is only available for the mapping of harmonic pressure fields simulated by FINE/Turbo, where additionally the FINE/Turbo .run file is given by the user. It can be activated by selecting the check box labeled “Use Harmonic Wizard” at the “What to map/Harmonic” panel.

The “Harmonic Wizard” panel lists all harmonic pressure quantities available in the FINE/Turbo .run file for the selected source parts (which have to belong to one single row). Also the information about the excitation shape (forward/backward nodal diameter) is given for each harmonic, which is used for the mapping between periodic models.
The mapping process (started by pressing the “Map” button) will map the selected harmonics one by one. The mapping results are saved in folders with the naming convention “Row-RowID_H-HarmonicID”.

4.8 The “Log” Panel

The MpCCI FSIMapper has a fixed workflow which can be viewed in the “Log” panel:

1. Evaluation of configuration file content
2. Reading source mesh, quantities and unit conversions
3. Reading target mesh
4. Model positioning if requested
5. Prepare mapping setup
6. Mapping of quantities
7. Export results to files
5 Codes and Formats Information

5.1 EnSight Gold

MpCCI FSIMapper supports reading and writing of binary EnSight Gold .case and .encas files for static, transient and harmonic simulation results. Only scalar quantities are supported. EnSight Gold only knows 3-dimensional meshes. If a mapping is performed between the surfaces of 2-dimensional models, “EnSight Case (2D)” must be selected; it converts internally the 3-dimensional model to a 2-dimensional model by removing the third degree of freedom (z).

5.1.1 Supported Quantities

Due to the open structure of the EnSight Gold Case format, MpCCI FSIMapper supports reading of all saved scalar and vector quantities.

5.2 FLUENT

MpCCI FSIMapper supports reading of FLUENT native .cas (geometry) and .dat (solution result) files for static simulation results. In general .cas and .dat files between FLUENT version 6.3.26 and version 20.1.x are supported and more recent versions should be compatible if native file format does not change. FLUENT CFF files (.h5) are supported from FLUENT 20.0.

Only zone types “wall” and “fluid” can be coupled with MpCCI FSIMapper.

5.2.1 Supported Quantities

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Dim.</th>
<th>Default Unit</th>
<th>Location</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEMPERATURE</td>
<td>Scalar</td>
<td>K</td>
<td>fluid</td>
<td></td>
</tr>
<tr>
<td>FILMTEMPERATURE</td>
<td>Scalar</td>
<td>K</td>
<td>wall</td>
<td></td>
</tr>
<tr>
<td>HTC (computed)</td>
<td>Scalar</td>
<td>W/(m² K)</td>
<td>wall</td>
<td>$\alpha = q/(T_\text{w} - T_\text{c})$</td>
</tr>
<tr>
<td>wallfuncHTC</td>
<td>Scalar</td>
<td>W/(m² K)</td>
<td>wall</td>
<td>stored in UDM0</td>
</tr>
<tr>
<td>WALLHEATFLUX</td>
<td>Scalar</td>
<td>W/m²</td>
<td>wall</td>
<td></td>
</tr>
<tr>
<td>STATICPRESSURE</td>
<td>Scalar</td>
<td>N/m²</td>
<td>wall</td>
<td>STATICPRESSURE + operating pressure</td>
</tr>
<tr>
<td>ABSPRESSURE</td>
<td>Scalar</td>
<td>N/m²</td>
<td>wall</td>
<td>STATICPRESSURE + operating pressure - reference pressure</td>
</tr>
<tr>
<td>OVERPRESSURE</td>
<td>Scalar</td>
<td>N/m²</td>
<td>wall</td>
<td>OVERPRESSURE + SHEARFORCE</td>
</tr>
<tr>
<td>WALLFORCE</td>
<td>VECTOR</td>
<td>N</td>
<td>wall</td>
<td>ABSPRESSURE + SHEARFORCE</td>
</tr>
<tr>
<td>RELATIVEWALLFORCE_STATIC</td>
<td>VECTOR</td>
<td>N</td>
<td>wall</td>
<td>STATICPRESSURE + SHEARFORCE</td>
</tr>
<tr>
<td>RELATIVEWALLFORCE_OVER</td>
<td>VECTOR</td>
<td>N</td>
<td>wall</td>
<td>OVERPRESSURE + SHEARFORCE</td>
</tr>
</tbody>
</table>
5.2.2 Exporting wallfuncHTC to UDM0 in Data File

For a surface heat transfer analysis “FILMTEMPERATURE” as well as the wall heat transfer coefficient need to be mapped onto a CSM model. The latter, defined as

\[ \alpha = \frac{Q}{A \Delta T} \]

where

- \( Q \) = heat flow, J/s = W
- \( \alpha \) = heat transfer coefficient, W/(m\(^2\)K)
- \( A \) = heat transfer surface area, m\(^2\)
- \( \Delta T \) = difference in temperature between the solid surface and surrounding fluid area, K.

It can be displayed in the FLUENT GUI Display/Graphics and Animations/Contours/Set Up/Wall Fluxes.../Wall Func. Heat Tran. Coef.. By default the wall heat transfer coefficient is not stored to a FLUENT data (.dat) file but can be added by following the steps:

1. Define/User-Defined/Memory:
   Set number to 1

2. Define/Custom-Field-Function:
   Select “wallfuncHTC” below “Wall Fluxes”.
   Accept by ‘Define’.

3. Solve/Initialization/Patch:
   Activate “Use Field Function” and select defined custom field function from previous step.
   Select all fluids and solids as well as “Variable User Memory 0” at bottom left side.
   Select “Patch”.

4. Case/Data:
   Save HTC to new .dat file.

5.3 FINE/Turbo

MpCCI FSIMapper supports reading of FINE/Turbo native .cgns files for static and harmonic simulation results. The corresponding .run file (model setup) can be imported in order to insert the surface and volume names originally defined in FINE/Turbo (instead of the internally defined names in the .cgns file).

Mapping is possible only if FINE/Turbo was executed using the expert parameter ICGP3D is set to 1

If the harmonic wizard is to be used, the import of the .run file is necessary

5.3.1 Supported Quantities

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Dim.</th>
<th>Default Unit</th>
<th>Location</th>
<th>comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEMPERATURE</td>
<td>Scalar</td>
<td>K</td>
<td>wall</td>
<td></td>
</tr>
<tr>
<td>HEATTRANSFERCOEF</td>
<td>Scalar</td>
<td>W/(m(^2)K)</td>
<td>wall</td>
<td></td>
</tr>
<tr>
<td>HEATFLUX</td>
<td>Scalar</td>
<td>W/m(^2)</td>
<td>wall</td>
<td></td>
</tr>
<tr>
<td>PRESSURE</td>
<td>Scalar</td>
<td>N/m(^2)</td>
<td>wall</td>
<td></td>
</tr>
<tr>
<td>REP(_x)</td>
<td>Scalar</td>
<td>N/m(^2)</td>
<td>volume</td>
<td>complex harmonic pressures</td>
</tr>
<tr>
<td>IMP(_x)</td>
<td>Scalar</td>
<td>N/m(^2)</td>
<td>volume</td>
<td>(NLH method)</td>
</tr>
</tbody>
</table>
FINE/Turbo only saves the results of the Nonlinear Harmonic method in the fluid volume and not on the wall.

### 5.4 CFX

MpCCI FSIMapper supports reading of CFX results exported in “CFD-Post Generic Export” format.

#### 5.4.1 Supported Quantities

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Dim.</th>
<th>Unit</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILMTEMPERATURE</td>
<td>Scalar</td>
<td>K</td>
<td>Face</td>
</tr>
<tr>
<td>HTC</td>
<td>Scalar</td>
<td>W/(m² K)</td>
<td>Face</td>
</tr>
<tr>
<td>PRESSURE</td>
<td>Scalar</td>
<td>N/m²</td>
<td>Face</td>
</tr>
</tbody>
</table>

### 5.5 FloTHERM

MpCCI FSIMapper supports reading of FloTHERM export format .flofea available in version 10 for static simulation results.

#### 5.5.1 Supported Quantities

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Dim.</th>
<th>Default Unit</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEMPERATURE</td>
<td>Scalar</td>
<td>C</td>
<td>solid</td>
</tr>
</tbody>
</table>

#### 5.5.2 Combining Static Result Files

MpCCI FSIMapper supports the combination of several static FloTHERM result files of the same geometry by creating a tabular master input file. To assign the individual time step information per *.flofea result file, a comma separated two-column format is used. The master *.csv file has to be located in the same folder to address the individual time step location. The following example shows the master file structure for a mapping setup with four different time step results.

```
#comment line
#time value, file name
0.0, result_at_0.0.flofea
5.0, result_at_5.0.flofea
10.0, result_at_10.0.flofea
20.0, result_at_20.0.flofea
```

### 5.6 FloEFD

MpCCI FSIMapper supports reading of FloEFD export format .efdfea available in version 13 for static simulation results.
5.6.1 Supported Quantities

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Dim.</th>
<th>Default Unit</th>
<th>Location</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEMPERATURE</td>
<td>Scalar</td>
<td>K</td>
<td>solid/wall</td>
<td></td>
</tr>
<tr>
<td>HTC</td>
<td>Scalar</td>
<td>W/(m² K)</td>
<td>wall</td>
<td></td>
</tr>
<tr>
<td>SURFACE HEAT FLUX</td>
<td>Scalar</td>
<td>W/m²</td>
<td>wall</td>
<td></td>
</tr>
<tr>
<td>PRESSURE</td>
<td>Scalar</td>
<td>N/m²</td>
<td>wall</td>
<td></td>
</tr>
<tr>
<td>RELATIVE PRESSURE</td>
<td>Scalar</td>
<td>N/m²</td>
<td>wall</td>
<td></td>
</tr>
</tbody>
</table>

5.7 MagNet

MpCCI FSIMapper supports reading of MagNet export format .vtk available from version 7.5.1 for static and transient simulation results. Multiple components are written to distinct .vtk files which can be inputted to MpCCI FSIMapper by multi selection. Mapping of static and transient data is possible.

5.7.1 Limitations

Currently only stationary components can be considered for mapping in both static and transient analyses.

5.7.2 Elements

<table>
<thead>
<tr>
<th>MpCCI FSIMapper</th>
<th>VTK Element Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D_SURF_TRIA3</td>
<td>5</td>
</tr>
<tr>
<td>3D_SURF_TRIA6</td>
<td>22</td>
</tr>
<tr>
<td>3D_SURF_QUAD4</td>
<td>9</td>
</tr>
<tr>
<td>3D_SURF_QUAD8</td>
<td>23</td>
</tr>
<tr>
<td>2D_SURF_LINE2</td>
<td>3</td>
</tr>
<tr>
<td>2D_SURF_LINE3</td>
<td>21</td>
</tr>
</tbody>
</table>

Table 1: Supported MagNet (.vtk) structural elements.

5.7.3 Supported Quantities

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Dim.</th>
<th>Default Unit</th>
<th>Location</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>SFD</td>
<td>Vector</td>
<td>N/m²</td>
<td>wall</td>
<td>Surface Force Density</td>
</tr>
</tbody>
</table>

5.8 ANSYS Maxwell

MpCCI FSIMapper supports reading of ANSYS Maxwell export to Universal file format .unv. Universal file node blocks (781 and 2411), element blocks (780 and 2412) and quantity block 2414 with node and element location are supported.
Table 2: Supported structural elements in Universal file.

5.8.1 Elements

5.8.2 Supported Quantities

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Dim.</th>
<th>Default Unit</th>
<th>Location</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORCE</td>
<td>Vector</td>
<td>N</td>
<td>wall</td>
<td>Surface Force</td>
</tr>
</tbody>
</table>

5.9 Abaqus

5.9.1 Limitations

Writing For an Abaqus CSM simulation the five quantities “Temperature”, “Heat Transfer Coefficient”, “Heat Flux”, “Pressure” and “Force” can be exported on surface definitions, “Temperature” can also be exported as solid initial condition.

Reading MpCCI FSIMapper supports reading of the Abaqus ASCII input format for static simulation results. Following quantities (with keywords) are supported: nodal temperature (**TEMPERATURE), thermal surface film coefficients (**SFILM) and nodal force (**CLOAD).

5.9.2 Model Preparation

The Abaqus model can be prepared with Abaqus/CAE or as an input file. Please consider the following advice:

- The model can be defined in either SI or mm-t-s unit systems to be specified in the 'What to map' panel of the MpCCI FSIMapper GUI.
- The Abaqus model must contain a definition of coupling components.
  - This can be either an element-based surface for surface coupling:
    - **Abaqus/CAE**: Create a surface using the Surfaces tool. See also “13.7.6 Using sets and surfaces in the Assembly module” in the Abaqus/CAE User’s Manual. You can also use surfaces defined in the Part module.
    - **Input file**: A surface is created with **SURFACE, NAME=<surface name>, TYPE=ELEMENT**, see section “2.3.3 Defining element-based surfaces” of the Abaqus Analysis User’s Manual.
  - The surface which serves as coupling component can be selected (marked with 'S' for surface) in the 'What to map' panel of the MpCCI FSIMapper GUI.
- This can be either an element set for volumetric temperature mapping:

3D_SURF_TRIA3
3D_SURF_TRIA6
3D_SURF_QUAD4
3D_VOL_TET4
3D_VOL_TET10
3D_VOL_HEX8
3D_VOL_HEX20
3D_VOL_WEDGE6
**Abaqus/CAE:** Create an element set using the *Sets* module. See also “13.7.6 Using sets and surfaces in the Assembly module” in the Abaqus/CAE User’s Manual.

**Input file:** An element set is created with *ELSET, NAME=⟨elset name⟩*, see section “2.2.1 Element definition” of the Abaqus Analysis User’s Manual.

The elset which serves as coupling component can be selected (marked with ‘E’ for elset) in the ‘What to map’ panel of the MpCCI FSIMapper GUI.

### 5.9.3 Include Boundary Conditions

To use the mapped quantity values as boundary conditions in an Abaqus CSM simulation a separate Abaqus input file is written for each boundary condition. This file, which is saved in the model directory, can be included in the original Abaqus input deck using the simple Abaqus include option in the input deck.

Include a mapped temperature by:

*INCLUDE, INPUT="abaqusFile-mapped_FilmTemp.inp"

Include a mapped temperature and heat transfer coefficient by:

*INCLUDE, INPUT="abaqusFile-mapped_FilmTempHTC.inp"

Include a mapped heat flux by:

*INCLUDE, INPUT="abaqusFile-mapped_HeatFlux.inp"

Include a mapped pressure field by:

*INCLUDE, INPUT="abaqusFile-mapped_Pressure.inc"

Include a mapped transient pressure field by:

*INCLUDE, INPUT="abaqusFile-mapped_TranstientPressure.inc"

Include a mapped transient pressure field which was Fourier transformed by:

*INCLUDE, INPUT="abaqusFile-mapped_FreqRespPressure.inc"

Include a mapped harmonic pressure field by:

*INCLUDE, INPUT="abaqusFile-mapped_HarmonicPressure_RE.inc" and *INCLUDE, INPUT="abaqusFile-mapped_HarmonicPressure_IM.inc"

Include a mapped force field by:

*INCLUDE, INPUT="abaqusFile-mapped_Force.inc"

Include a mapped transient force field by:

*INCLUDE, INPUT="abaqusFile-mapped_TransientForce.inc"

Include a mapped transient force field which was Fourier transformed by:

*INCLUDE, INPUT="abaqusFile-mapped_FreqRespForce.inc"

Include a mapped transient temperature using amplitude definition:

*INCLUDE, INPUT="abaqusFile-mapped_Amp.inc"

*INCLUDE, INPUT="abaqusFile-mapped_Temp_Amp.inc"

To make sure that the values can be used for an Abaqus analysis, the include has to be placed between the *STEP* and *END STEP* keywords. Furthermore, for all thermal quantities the step has to be defined as a heat transfer step, e.g. under *COUPLED TEMPERATURE-DISPLACEMENT* for a thermal stress analysis. Please refer to the Abaqus keyword manual for a detailed use.

⚠️ Definition of a transient temperature using amplitude definition normally requires definition of initial reference temperature, e.g. *Initial Conditions, type=TEMPERATURE*. Here also a statically mapped temperature can be used.
5.9.4 Elements

5.9.5 Supported Quantities

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Dim.</th>
<th>Unit (see Table 5)</th>
<th>Location</th>
<th>Abaqus Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEMPERATURE</td>
<td>Scalar</td>
<td>θ</td>
<td>*Elset</td>
<td>*TEMPERATURE</td>
</tr>
<tr>
<td>FILMTEMPERATURE</td>
<td>Scalar</td>
<td>θ</td>
<td>*Surface</td>
<td>*SFILM</td>
</tr>
<tr>
<td>HTC</td>
<td>Scalar</td>
<td>JT⁻¹ L⁻² θ⁻¹</td>
<td>*Surface</td>
<td>*DFLUX</td>
</tr>
<tr>
<td>WALLHEATFLUX</td>
<td>Scalar</td>
<td>JL⁻² T⁻¹</td>
<td>*Surface</td>
<td>*DLOAD</td>
</tr>
<tr>
<td>PRESSURE</td>
<td>Scalar</td>
<td>ML⁻¹ θ⁻²</td>
<td>*Surface</td>
<td></td>
</tr>
<tr>
<td>FORCE</td>
<td>Vector</td>
<td>MLT⁻²</td>
<td>*Surface</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Unit</th>
<th>Length</th>
<th>Mass</th>
<th>Temperature</th>
<th>Time</th>
<th>Energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol</td>
<td>L</td>
<td>M</td>
<td>θ</td>
<td>T</td>
<td>J</td>
</tr>
</tbody>
</table>

Table 5: Abaqus unit symbols.
5.10 ANSYS

5.10.1 Requirements

To use MpCCI FSIMapper for file based mapping to ANSYS you need the following:

- Ordinary ANSYS installation.
- It is required that ANSYS can run in standalone correctly and can get a license without any issue.

5.10.2 Model Preparation

There are some issues which you should consider while creating an ANSYS model for a co-simulation:

<table>
<thead>
<tr>
<th>ANSYS Element Types</th>
<th>MpCCI FSIMapper</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLANE13, PLANE25, PLANE42, PLANE55, PLANE67, PLANE181, PLANE182, SHELL28, SHELL41, SHELL43, SHELL57, SHELL63, SHELL131, SHELL143, SHELL157, HYPER56, VISCO106, CPT212, SURF252, SURF152, SURF154</td>
<td>3D_SHELL_TRIA3, 3D_SHELL_QUAD4</td>
</tr>
<tr>
<td>PLANE2, PLANE35</td>
<td>3D_SHELL_TRIA3, 3D_SHELL_TRIA6</td>
</tr>
<tr>
<td>SHELL91, SHELL93, SHELL99, SHELL132, SHELL150, SHELL281, PLANE53, PLANE145,PLANE223, PLANE77, PLANE78, PLANE82, PLANE83, PLANE121, PLANE183, PLANE146, PLANE230, SURF152, SURF154, CPT213, HYPER74, VISCO88, VISCO108</td>
<td>3D_SHELL_TRIA6, 3D_SHELL_QUAD4, 3D_SHELL_QUAD8</td>
</tr>
<tr>
<td>SOLID5, SOLID46, SOLID62, SOLID64, SOLID65, SOLID69, SOLID70, SOLID96, SOLID97, SOLID164, SOLID285, SOLID185, HYPER56, HYPER86, VISCO107, SOLSH190, CPT215</td>
<td>3D_VOL_TET4, 3D_VOL_WEDGE6, 3D_VOL_HEX8, 3D_VOL_PYRAM5</td>
</tr>
<tr>
<td>SOLID87, SOLID92, SOLID123, SOLID127, SOLID148, SOLID168, SOLID186, SOLID187, SOLID227, SOLID232, SOLID237, CPT217</td>
<td>3D_VOL_TET4, 3D_VOL_TET10</td>
</tr>
<tr>
<td>SOLID90, SOLID95, SOLID122, SOLID117, SOLID122, SOLID128, SOLID147, SOLID186, SOLID191, SOLID226, SOLID231, SOLID236, VISCO89, CPT216</td>
<td>3D_VOL_TET4, 3D_VOL_TET10, 3D_VOL_WEDGE6, 3D_VOL_WEDGE15, 3D_VOL_HEX8, 3D_VOL_HEX20, 3D_VOL_PYRAM5, 3D_VOL_PYRAM13</td>
</tr>
<tr>
<td>MESH200</td>
<td>MPCCI_ETYP_LINE2, MPCCI_ETYP_TRIA3, MPCCI_ETYP_QUAD4, MPCCI_ETYP_QUAD8</td>
</tr>
</tbody>
</table>

Table 6: Supported ANSYS element types.

**Supported element types.** Not all ANSYS element types are supported, the supported types are given in Table 6.

**Dummy surface elements for surface coupling.** For surface coupling, additional surface elements must be used for quantity exchange. For instance SHELL63 elements for a fluid structure interaction or SHELL57 elements for thermal coupling can be used as dummy elements to receive forces from the
partner code. The dummy elements must have the corresponding quantities you want to receive or send. Only these elements take part in the coupling process and must either be deselected when the solution is performed or defined so weak that they do not influence the solution in case of fluid structure interaction. They can be deselected before solution is executed if only nodal quantities are sent or received, because the data transfer will then put the values to the nodes and the nodes of dummy elements are shared by the “real” solid model elements. In case of thermal coupling the dummy elements cannot be deselected because the quantities wall heat transfer coefficient, wall temperature and wall heat flux are only supported as element quantities. Therefore the additional layer of shell elements (SHELL57) have to be a part of the solution. To reduce the influence on the solution you should give the shell elements the same material properties as the adjacent solid elements and a small thickness. Such dummy elements are shown in Figure 16.

**Put elements for coupling in a component.** The elements of the coupling region must be grouped into one or more element components using the `cm` command.

**Predefined loads on dummy surface elements.** If element based quantities HTC, TEMPERATURE, WALLHEATFLUX or PRESSURE are received by ANSYS using dummy surface elements, you have to predefine surface loads on these elements in order to enable the MpCCI ANSYS adapter to select the correct element sides to store the received values. Therefore select the coupled element set and the attached nodes and define a dummy load, depending on the received quantities:

- HTC and TEMPERATURE: `~SF, ALL, CONV, 1, 300`
- WALLHEATFLUX: `~SF, ALL, HFLUX, 1`
- PRESSURE: `~SF, ALL, PRES, 1`

**Possible quantities depend on degrees of freedom.** Normally the degree of freedom of the elements involved in the coupling process determines which quantities can be transferred. It is laborious to find out if all degrees of freedom are actually supported by the ANSYS API. As this API is used for the MpCCI ANSYS adapter, it is not guaranteed that all theoretically supported degrees of freedom are valid.

**Not carefully tested.** Only few of the element type mappings are already validated with certain quantities. The compatibility index in Table 7 shows the validated element-quantity pairs. It is constantly added. Please contact us if you have problems with other combinations or if you need additional capabilities.

Figure 16: ANSYS volume model and additional dummy SHELL63 elements for surface coupling
Table 7: Quantities supported by MpCCI FSIMapper for different ANSYS elements.

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Dim.</th>
<th>Unit</th>
<th>Location</th>
<th>ANSYS Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEMPERATURE</td>
<td>Scalar</td>
<td>$\theta$</td>
<td>Solid/Shell</td>
<td>BF,,,TEMP</td>
</tr>
<tr>
<td>HTC</td>
<td>Scalar</td>
<td>$J^{-1}T^{-1}L^{-1}\theta^{-1}$</td>
<td>Shell</td>
<td>SFE,,,CONV</td>
</tr>
<tr>
<td>WALLHEATFLUX</td>
<td>Scalar</td>
<td>$JL^{-2}T^{-1}$</td>
<td>Shell</td>
<td>SFE,,,HFLUX</td>
</tr>
<tr>
<td>PRESSURE</td>
<td>Scalar</td>
<td>$ML^{-1}T^{-2}$</td>
<td>Shell</td>
<td>SFE,,,PRES</td>
</tr>
</tbody>
</table>

5.10.4 ANSYS Scanner and Converter

ANSYS provides two native solver file formats. The .cmd format is ASCII based whereas the more common data base .db format is binary. The MpCCI FSIMapper makes use of the MpCCI code adapter to convert both .cmd and .db format into an own intermediate format.

When selecting code ANSYS as target model in “What to map” panel of the MpCCI FSIMapper GUI and specifying either a .cmd or .db file, the model automatically gets converted into intermediate format. This is done by executing the command

> fsimapper convert ANSYS modelFile.db

(see ▶ MpCCI FSIMapper Command for details)

creating a new .ml file in same system folder where original model file is located. This result file is automatically selected as target model in the “What to map” panel.

Then the file scanner for intermediate format .ml is launched by

> fsimapper scan ANSYS modelFile.ml

listing available components that have been converted for a mapping.

5.11 MSC NASTRAN

5.11.1 Limitations

**Writing** As boundary conditions for a CSM simulation with Nastran temperature, pressure or force values can be exported. For temperature values the keywords TEMP or SPC are used and the quantity values are mapped to and written on nodes. The keywords PLOAD, PLOAD2 and PLOAD4 are used for the export of element-based pressure values (see also ▶ 4.5.3 MSC NASTRAN Export Options). Forces from a static mapping are exported using the keywords FORCE, a transient mapping uses the keywords TSTEP, TLOAD1, DAREA and TABLED1. When a Fourier transformation has been performed on transient data, then the keywords FREQ, RLOAD1, RLOAD2, DAREA and TABLED1 are used.
The Nastran mesh has to consist of shell elements for exporting pressure in the **PLOAD** format.
The Nastran mesh has to consist of shell elements for exporting force.
The Nastran mesh has to consist of shell or solid elements for exporting temperatures in **TEMP** or **SPC** format.

When analysing solid **MSC NASTRAN** models with surface loads and **MpCCI FSIMapper** can handle only shell elements, create a shell mesh at the coupled surface using the same grids as the solid model. Use this model for the mapping process.

**Reading** Moreover, **MpCCI FSIMapper** supports reading of **MSC NASTRAN** files in bulk data format. For static simulation results the following quantities (with keywords) are supported: temperature (**TEMP** and **TEMP**\*), pressure (**PLOAD2** and **PLOAD2**\* (single element per line assumed), **PLOAD4** and **PLOAD4**\* (assuming a constant value per element)), force (**FORCE** and **FORCE**\* (ignoring the coordinate system identification number)) and heat source (**QVOL** and **QVOL**\*).

For transient results, only force is supported (defined by the keywords **TLOAD1** (ignoring the delay), **DAREA**, **TABLED1** or their corresponding long formats).

### 5.11.2 Include Boundary Conditions

To use the mapped quantity values as boundary conditions in an **MSC NASTRAN** CSM simulation a separate **MSC NASTRAN** input file is written for each boundary condition. This file, which is saved in the model directory, can be included in the original **MSC NASTRAN** input deck using the simple **MSC NASTRAN** include option in the input deck.

The file extension of the exported include files is the same as of the target mesh file. Here, the Bulk Data Entries are shown for .bdf. Include a mapped temperature using the **TEMP** keyword by:

```
INCLUDE 'nastranFile-mapped_Temp.bdf'
```

Include a mapped temperature using the **SPC** keyword by:

```
INCLUDE 'nastranFile-mapped_Temperature_SPC.bdf'
```

Include a mapped pressure field by:

```
INCLUDE 'nastranFile-mapped_Pressure.bdf'
```

Include a mapped transient pressure field by:

```
INCLUDE 'nastranFile-mapped_TransientPressure.bdf'
```

Include a mapped transient pressure field which was Fourier transformed by:

```
*INCLUDE, INPUT="nastranFile-mapped_FreqRespPressure.bdf"
```

Include a mapped harmonic pressure field by:

```
*INCLUDE, INPUT="abaqusFile-mapped_HarmonicPressure.bdf"
```

Include a mapped force field by:

```
INCLUDE 'nastranFile-mapped_Force.bdf'
```

Include a mapped transient force field by:

```
INCLUDE 'nastranFile-mapped_TransientForce.bdf'
```

Include a mapped transient force field which was Fourier transformed by:

```
INCLUDE 'nastranFile-mapped_FreqRespForce.bdf'
```

To make sure that the values can be used for an **MSC NASTRAN** analysis, the include has to be placed between the **BEGIN BULK** and **ENDDATA** keywords.

The set identification numbers (**MSC NASTRAN** **SID** of Bulk Data Entry) defined by **MpCCI FSIMapper** have to be referenced in the Case Control Commands of the **MSC NASTRAN** analysis deck. Please refer to the **MSC NASTRAN** keyword reference manual for a detailed use.
5.11.3 Elements

5.11.4 Supported Quantities

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Dim.</th>
<th>Default Unit</th>
<th>Location</th>
<th>MSC NASTRAN Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>(FILM)TEMPERATURE</td>
<td>Scalar</td>
<td>$\theta$</td>
<td>Shell/Volume</td>
<td>TEMP SPC</td>
</tr>
<tr>
<td>PRESSURE</td>
<td>Scalar</td>
<td>$ML^{-1}\theta^{-2}$</td>
<td>Shell</td>
<td>PLOAD PLOAD2 PLOAD4 RLOAD1 RLOAD2</td>
</tr>
<tr>
<td>FORCE</td>
<td>Vector</td>
<td>$MLT^{-2}$</td>
<td>Node</td>
<td>FORCE TLOAD1 RLOAD1 RLOAD2</td>
</tr>
</tbody>
</table>

5.12 LS-Dyna

MpcCCI FSImapper supports writing of LS-Dyna files in ASCII keyword data format.

5.12.1 Limitations

The structural target code LS-Dyna is only available when the “Transient” panel is used for the source model and the Fourier transformation is enabled.

5.12.2 Elements

<table>
<thead>
<tr>
<th>LS-Dyna</th>
<th>MpcCCI FSImapper</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 node shell elements</td>
<td>3D_SHELL_TRIA3</td>
</tr>
<tr>
<td>4 node shell elements</td>
<td>3D_SHELL_QUAD4</td>
</tr>
</tbody>
</table>

Table 9: Supported LS-Dyna structural elements.
5.12.3 Supported Quantities

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Dim.</th>
<th>Default Unit</th>
<th>Location</th>
<th>LS-Dyna Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORCE Vector</td>
<td>$MLT^{-2}$</td>
<td>Node</td>
<td>*DATABASE_FREQUENCY_BINARY_D3SSD</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*FREQUENCY_DOMAIN_SSDFORCE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*DEFINE_CURVE</td>
</tr>
</tbody>
</table>

5.12.4 Include Boundary Conditions

To use the mapped quantity values as boundary conditions in an LS-Dyna CSM simulation a separate LS-Dyna input file is written for each boundary condition. This file, which is saved in the model directory, can be included in the original LS-Dyna input deck using the simple LS-Dyna include option in the input deck.

Include a mapped frequency response file using the statement

```
*include lsdynaFile-mapped_FreqRespForce.inc
```
6 Batch Usage of the MpCCI FSIMapper

The MpCCI FSIMapper can also be used as a batch tool. All functionalities of the graphical user interface are also available in the batch version.

The basic MpCCI FSIMapper is an executable that needs several arguments for execution: the file names of the source and the target models and a configuration file where all necessary mapping parameters are set (see 3 MpCCI FSIMapper Command). A detailed description of the configuration file format can be found in 6.5 Configuration File.

To find the relevant parts of the models for the interface file scanners for FLUENT, Abaqus, ANSYS, FloTHERM, FloEFD and EnSight Gold are part of the MpCCI FSIMapper. These perl modules list all found parts and the assigned ids.

6.1 File Scanners

The MpCCI FSIMapper uses a configuration file to decide which quantities to map between which geometry parts. All geometry parts of a model can be addressed by ids, which can be identified by the file scanners. The user has to enter the relevant ids for the CFD and the FEA mesh into the configuration file.

For a thermal mapping case usually only the ids for the interface between the fluid and solid domain will be entered in the configuration file. The rest of the geometry does not need to be read by the MpCCI FSIMapper. Depending on the type of application the user can decide which geometry parts are to be read by the MpCCI FSIMapper.

To get the ids of the geometry parts, solver files can be scanned by perl modules that are part of the MpCCI FSIMapper package. The only requirement is a running perl installation with its executables in the system PATH variable. All file scanners – for FLUENT, Abaqus, ANSYS, FloTHERM, FloEFD, EnSight Gold, CFX and FINE/Turbo – can be called via the command line:

> fsimapper scan FLUENT file_name.cas
> fsimapper scan ABAQUS file_name.inp
> fsimapper scan ANSYS file_name.ml
> fsimapper scan ENSIGHT file_name.case
> fsimapper scan FLOEFDMAPLIB file_name.efdfea
> fsimapper scan FLOTHERMMAPLIB file_name.flofea
> fsimapper scan FINETURBO file_name.cgns
> fsimapper scan CFXCSV file_name.csv

MpCCI FSIMapper does not allow scanning of native ANSYS solver formats directly but of converted .ml files generated by “fsimapper convert ANSYS” (5.10.4 ANSYS Scanner and Converter).

6.1.1 FLUENT

The FLUENT scanner lists all zones that are defined in the .cas file. The names of the zones are listed, as well as the zone type and the zone id. The type of the zones is important to find the parts of the model that form the interface between the fluid and the structural domain.

Usually, the zones of type “wall” are to be considered for the MpCCI FSIMapper. The zone id is used in the configuration file to identify the part of the model from which quantity values shall be mapped to the CSM model.

The format of the output is the following:
6.1.1.1 Example

As an example the output of the FLUENT scanner for a simple case file “exampleFluent.cas” is shown below.

```bash
> fsimapper scan FLUENT ./exampleFluent.cas
</CPLDATA>

# This file was automatically created for the MpCCI GUI.
#
# Model dimensions : 3D
# Coordinate system : Cartesian
# Solution type : Steady state
# Load cases : 1
# Unit system : SI
# Precision : Double precision (64 bit)
#
# Additional scanner information:
# Global variables : Autogenerated by the scanner
# Release : 13.0.0
# Saved by : fluent
# Used UDM : 0
# Used UDS : 0
# Version : 3ddp
#
fluid_1 fluid 7
wall_1 wall 64
interior_1 interior 78
interface_1 interface 79
</CPLDATA>
```

This FLUENT model consists of four different zones: one of each of the four types “fluid”, “wall”, “interior” and “interface”. The zone with the name `wall_1` of type “wall” has the id 64. If we now want to couple `wall_1` we have to add the following lines to the configuration file (▷6.5 Configuration File◁) defining FLUENT as source file format and part 64 to be read:

```plaintext
sourceFileType
FLUENT
sourceMeshPartIds
64
```

For the mapping of quantity data usually the zones of type “wall” are of interest, as the interface between
the CFD and FEA domains is normally a wall boundary in the CFD model.

For large FLUENT models and Linux machines it might be helpful to only list the wall zones of the model:

```
> fsimapper scan FLUENT ./exampleFluent.cas | grep wall
```

This gives a clearer overview of the possible interfaces without listing all interior or fluid zones.

### 6.1.2 MentorGraphics

In cooperation Fraunhofer SCAI and MentorGraphics have developed a neutral file format allowing export of FloEFD and FloTHERM simulation results for mapping to a CSM code.

### 6.1.3 FloEFD

The scanner for FloEFD export files can be selected by its name FLOEFDMAPLIB in the generic MpCCI FSIMapper scan command, e.g.

```
> fsimapper scan FLOEFDMAPLIB file_name.efdfea
```

The resulting output of the scanner has a simple tabular format with just the keywords MESH, PART or QUANT. MESH assigns the name of the model to be read in. PART specifies 'S' or 'V', if it is a surface or a volume, the part name as well as two integers specifying the total number of nodes and elements of the part. Parts are implicitly numbered by their order of appearance starting at index 1. Keyword QUANT announces a quantity followed by its name, location of the values, the number of values and the file position of the quantity.

#### 6.1.3.1 Example

```
> fsimapper scan FLOEFDMAPLIB ./Plate.efdfea
MESH EFD_MESH
PART S Surface-1 302 600
PART V Volume-1 390 216
QUANT RELATIVE_PRESSURE ELEM 816 54917
QUANT PRESSURE ELEM 816 58198
QUANT SURFACE_HEAT_FLUX ELEM 816 61470
QUANT TEMPERATURE ELEM 816 64751
QUANT HTC ELEM 816 68026
```

One surface (Surface-1) addressed by 'S' and id 1 and one volume (Volume-1) addressed by 'V' and with id 2 are defined in this FloEFD model. In addition FloEFD file does contain a list of element based quantities, e.g. RELATIVE_PRESSURE, PRESSURE, SURFACE_HEAT_FLUX, TEMPERATURE and HTC, which can be read from file.

Parts to be mapped can be selected by its ids separated by whitespace in the configuration file. In this example we now want to couple the volume part and therefore have to add the following lines to the configuration file:

```
sourceFileType
FLOEFDMAPLIB
sourceMeshPartIds
2
```

A mapping of the surface Surface-1 would require to add
sourceFileType  
FLOEFDMAPLIB  
sourceMeshPartIds  
1  
to configuration file.

### 6.1.4 FloTHERM

The scanner for FloTHERM export files can be selected by its name FLOTHERMMAPLIB in the generic MpCCI FSIMapper scan command, e.g.

```bash  
> fsimapper scan FLOTHERMMAPLIB file_name.flofea  
```

The resulting output of the scanner has a simple tabular format with just the keywords MESH, PART or QUANT. MESH assigns the name of the model to be read in. PART specifies ‘S’ or ‘V’, if it is a surface or a volume, the part name as well as two integers specifying the total number of nodes and elements of the part. Parts are implicitly numbered by their order of appearance starting at index 1. Keyword QUANT announces a quantity followed by its name, location of the values, the number of values and the file position of the quantity.

#### 6.1.4.1 Example

```bash  
> fsimapper scan FLOTHERMMAPLIB ./BGA_detailed.flofea  
MESH Root_Assembly  
PART V encapsulant  7168  5766  
PART V die  1200  722  
PART V solder_ball_1_1  20  4  
PART V solder_ball_1_2  20  4  
QUANT TEMPERATURE ELEM  39475  2966255  
```

In this example four volumes are addressed by 'V', they have the ids 1,2,3 and 4. In addition FloTHERM file does contain element based TEMPERATURE which can be read from file. One or more ids can be entered separated by whitespaces in the configuration file to obtain mapped quantity values. In this example we now want to couple all volume parts of the model and therefore have to add the following lines to the configuration file:

```
sourceFileType  
FLOTHERMMAPLIB  
sourceMeshPartIds  
1 2 3 4  
```

### 6.1.5 ANSYS

The MpCCI FSIMapper scanner for ANSYS does not work with native solver files so that original "*.cdb" and "*.db" files need to be converted with fsimapper convert ANSYS model.db (see 5.10.4 ANSYS Scanner and Converter for details) producing intermediate .ml format related to FloTHERM and FloEFD format described in 6.1.4 FloTHERM and 6.1.3 FloEFD.

⚠️ Please note ANSYS requirements and model preparation section of 5.10 ANSYS.
The scanner for ANSYS export files can be selected by its name `ANSYS` in the generic `MpCCI FSIMapper` scan command, e.g.

```
> fsimapper scan ANSYS file_name.ml
```

The resulting output of the scanner has a simple tabular format with just the keywords `MESH` or `PART`. `MESH` assigns the name of the model to be read in. `PART` specifies 'S' or 'V', if it is a surface or a volume, the part name as well as two integers specifying the total number of nodes and elements of the part. Parts are implicitly numbered by their order of appearance starting at index 1.

### 6.1.5.1 Example

```
> fsimapper scan ANSYS ./plate.ml
MESH ANSYSDB2MAPLIB
PART S SHELLS__ 5802 11600
PART V SOLIDS__ 15565 76037
```

One shell or surface part (`SHELLS__`) addressed by 'S' and id 1 and one solid part (`SOLIDS__`) addressed by 'V' and with id 2 are defined in this `ANSYS` model.

A mapping of the surface component `SHELLS__` would require to add

- `targetFileType` `ANSYS`
- `targetMeshPartIds` 1

of the solid component `SOLIDS__` would require to add

- `targetFileType` `ANSYS`
- `targetMeshPartIds` 2

to the configuration file.

### 6.1.6 Abaqus

The part of the Abaqus model on which the quantity values are going to be mapped, has to be defined as a surface (with the keyword `*SURFACE` in the input deck) or in case of solid temperature mapping as a separate element set (`*ELSET`) in the Abaqus model.

The Abaqus file scanner will list all defined elsets and surfaces found in the Abaqus input deck and assign unique positive ids for surfaces and negative ids for elsets.

For the `MpCCI FSIMapper` surfaces or elsets to be mapped need to be addressed in the configuration file. Please see the following example in the file `6.1.5.1 Example` on how to determine a correct surface or elset id for a given input deck.

⚠️ If the scanned `.inp` file contains `*INCLUDE` directives for other parts of the Abaqus model stored in separate `.inp` files, the scanner also lists all surfaces found in the included input decks due to their appearance in the file flow.
6.1.6.1 Example

As an example the output of the Abaqus scanner for a simple input file “exampleAbaqus.inp” is shown below.

```bash
> fsimapper scan ABAQUS exampleAbaqus.inp
[FSIMapper] SET-1 elset -1
[FSIMapper] SET-2 elset -2
[FSIMapper] FLAP_SLAVE surface 0
[FSIMapper] WALL_MASTER surface 1
[FSIMapper] COUPLING_SURFACE surface 2
```

Three surfaces (FLAP_SLAVE, WALL_MASTER and COUPLING_SURFACE) addressed by the global ids 0, 1 and 2 and two elsets (SET-1 and SET-2) with ids -1 and -2 are defined in this Abaqus model.

One or more ids can be entered separated by whitespace in the configuration file to obtain mapped quantity values on the surfaces or elsets.

In this example we now want to couple the last surface for an Abaqus target model and therefore have to add the following lines to the configuration file:

```bash
targetFileType
ABAQUS
targetMeshPartIds
2
```

A mapping of temperature on elset SET-1 would require to add

```bash
targetFileType
ABAQUS
targetMeshPartIds
-1
```

to the configuration file.

MpCCI FSIMapper does not support mixing elset and surface ids in targetMeshPartIds, e.g. a target model can either be defined by surface ids or elset ids.

6.1.7 EnSight Gold Case

The EnSight Gold Case scanner lists all scalar variables which are defined in the given .case file. At present the MpCCI FSIMapper reads all defined surfaces and volume parts together with user selected quantities. As in EnSight Gold Case format the quantity names may vary but denote the same physical meaning, the user has to select the quantities of interest by its name. The scanner process is split up into two steps. At first the given Case master file is scanned for available scalar quantities followed by geometric scan process which identifies and lists the part structures of containing geometries.

6.1.7.1 Example

As an example the output of the EnSight Gold Case scanner for a simple case file “exampleEnSight.case” is shown below.

```bash
> fsimapper scan ENSIGHT exampleEnSight.case
<CPLDATA>
# This file was automatically created for the MpCCI GUI.
```
The first “CPLDATA” block lists all defined scalar variables by name, the second block lists available volumetric ‘V’ and surface ‘S’ parts which can be used with MpCCI FSIMapper. To inform the MpCCI FSIMapper to handle variable “Total_Temperature” as physical temperature quantity, the user has to select it by name in the GUI or add

```
EXPORT_1002_TEMPERATURE
Total_Temperature
```

to the configuration file when using batch mode. Currently supported keywords in the MpCCI FSIMapper configuration file are:

```
EXPORT_1002_TEMPERATURE
Name_Of_Temperature
EXPORT_1003_WALLHTCOEFF
Name_Of_Heat_Transfer_Coefficient
EXPORT_1004_WALLHEATFLUX
Name_Of_Heat_Flux
EXPORT_1010_PRESSURE
Name_Of_Pressure
```
As the EnSight Gold Case reader can be used as source or target model we additionally have to add

```plaintext
sourceFileType
ENSIHGTLGOLD
```

to the MpCCI FSIMapper configuration file if the case EnSight is used as source format

```plaintext
targetFileType
ENSIHGTLGOLD
```

otherwise.
6.2 Comparing Geometries

Figure 17: The workflow of the MpCCI FSIMapper in batch usage when comparing two geometries

It is possible to use the MpCCI FSIMapper to compare two geometries defined in two model files source and target.

For the comparison of geometries the nodal distance as well as the association distance are computed. As can be seen in Figure 17 these distances are written to the .vtk file containing the target geometry.

For each node of the target mesh the nodal distance is the distance to the nearest point of the source mesh. The association distance is the distance between the point of the target mesh and the closest (associated) element of the source mesh. If the point lies “in” a source element, this would mean that the association distance is 0.

If the MpCCI FSIMapper is used to compare two geometries rather than map actual quantity values, this has to be set in the configuration file via the option sourceQuantityMode or targetQuantityMode (one of them is sufficient):

```
sourceQuantityMode
GEOMETRYCOMPARE
or
targetQuantityMode
GEOMETRYCOMPARE
```

Furthermore, the configuration file needs to contain all relevant information about the source and target file types, units and part ids. More information concerning these parameters is given in > 6.5 Configuration File, that gives an introduction to the MpCCI FSIMapper configuration file. The ids for the different mesh parts can be obtained by the respective mesh scanners.

With the appropriate configuration file the MpCCI FSIMapper can now be started by the command line:

```
> fsimapper batch configfile source target
```
(see > 3 MpCCI FSIMapper Command)
6.3 Mapping Quantities

The mapping of quantity values can be done with the same command line call as the comparison of geometries:

```bash
> fsimapper batch configFile source [source_quant] target
```

The configuration file contains all relevant parameters that are necessary for the mapping process. The different keywords and values are described in §6.5 Configuration File.<

The quantity file `source_quant` is an optional argument. It is necessary if the names of the model and quantity files are not the same, for example if the FLUENT.cas and .dat files do not have the same base name.

The MpCCI FSIMapper will print some output to the console while the mapping is in progress. After the mapping the result files can be found in the working directory.

6.4 Files Written by the MpCCI FSIMapper

As mentioned before MpCCI FSIMapper exports

- mapped quantities as input files for CSM codes and
- both source and target meshes as .vtk files.

The results with the mapped quantities can be used for an Abaqus (§5.9 Abaqus <), ANSYS or MSC NASTRAN (§5.11 MSC NASTRAN <) analysis.

Files in the open source format .vtk can be visualized by the open source viewer “Paraview” (www.paraview.org). Each of the two files (named for example “abaqus-mapped.vtk” and “fluent.vtk”) contains the mesh definition as well as the original or mapped quantity data respectively. For the most common case of thermal mapping from e.g. FLUENT to Abaqus this means, that the FLUENT file contains the temperature, film temperature and wall heat flux values while the Abaqus file holds the mapped values of the film temperature and the wall heat coefficient.

6.5 Configuration File

The configuration file contains all parameters about file formats, geometry parts, quantities and mapping or orphan filling algorithms that define the complete process. The file is in a simple ASCII format. Different keywords define certain parameters. In the line following the keyword the respective value for the keyword is given. Lines starting with a “#” sign will be ignored.

```
#Example of configuration format structure
keyword_1 value
keyword_2 value
keyword_3 value_1 value_2 ... value_n
```

Information about the geometry or quantities have to be given for the source and the target models respectively.
Several different units are supported. Length, time, mass and temperature units can be specified for the source and the target model. The unit of pressure is expected to fit to the given mass, length and time units: \( \text{Pa} = \frac{\text{kg}}{\text{m}^2\text{s}^2} \), for example, would be the matching pressure unit to kg, m and s. If the two models use different units all necessary conversions are done by the MpCCI FSIMapper.

Minimum required configuration file parameters:

```plaintext
#define source format type
sourceFileType

#define target format type
targetFileType

#define quantity mode for mapping
sourceQuantityMode

#define source parts
sourceMeshPartIds
  id_1 ... id_n
#define target parts
targetMeshPartIds
  id_1 ... id_m

#use default mapping algorithm and parameters
optionUseNeighborhoodDefaultParameters
true
```

All possible keywords and values are listed in the following table.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>sourceFileType</td>
<td>FLUENT ( \rightarrow ) Code FLUENT</td>
</tr>
<tr>
<td></td>
<td>FLOOTHERMMAPLIB ( \rightarrow ) Code FloTHERM</td>
</tr>
<tr>
<td></td>
<td>FLOEFDMAPLIB ( \rightarrow ) Code FloEFD</td>
</tr>
<tr>
<td></td>
<td>ENSIGHTGOLD ( \rightarrow ) EnSight Case Gold</td>
</tr>
<tr>
<td>targetFileType</td>
<td>ABAQUS ( \rightarrow ) Code Abaqus</td>
</tr>
<tr>
<td></td>
<td>ANSYS ( \rightarrow ) Code ANSYS</td>
</tr>
<tr>
<td></td>
<td>NASTRAN ( \rightarrow ) Code MSC NASTRAN</td>
</tr>
<tr>
<td></td>
<td>ENSIGHTGOLD ( \rightarrow ) EnSight Case Gold</td>
</tr>
<tr>
<td>sourceQuantityMode</td>
<td>ABSPRESSURE ( \Rightarrow ) Read and map pressure</td>
</tr>
<tr>
<td></td>
<td>STATICPRESSURE ( \Rightarrow ) Read and map static pressure</td>
</tr>
<tr>
<td></td>
<td>OVERPRESSURE ( \Rightarrow ) Read and map overpressure</td>
</tr>
<tr>
<td></td>
<td>PRESSURETRANSIENT ( \Rightarrow ) Read and map transient pressure</td>
</tr>
<tr>
<td></td>
<td>PRESSUREHARMONIC ( \Rightarrow ) Read and map harmonic pressure</td>
</tr>
<tr>
<td></td>
<td>FILMTEMPHTC ( \Rightarrow ) Read and map surface film temperature and htc</td>
</tr>
</tbody>
</table>
6 Batch Usage of the MpCCI FSIMapper

- **FILMTEMPUDM0**
  - Read and map surface film temperature and htc stored in UDM0 (only FLUENT)

- **FILMTEMP**
  - Read and map surface film temperature

- **TEMPERATURE**
  - Read and map volumetric temperature

- **TEMPERATURETRANSIENT**
  - Read and map transient temperature

- **WALLHEATFLUX**
  - Read and map surface heat flux

- **FORCE**
  - Read and map surface forces

- **FORCETRANSIENT**
  - Read and map transient surface forces

- **FORCEHARMONIC**
  - Read and map harmonic surface forces

- **FORCEDENSITY**
  - Read and map surface force density

- **FORCEDENSITYTRANSIENT**
  - Read and map transient surface force density

- **GEOMETRYCOMPARE**
  - Compare both geometries
  
  - **default**: NONE

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>targetQuantityMode</td>
<td><strong>GEOMETRYCOMPARE</strong></td>
</tr>
<tr>
<td></td>
<td><strong>default</strong>: NONE</td>
</tr>
<tr>
<td>sourceMeshPartIds</td>
<td>the zone ids of source and target mesh from the file scanner separated by whitespace</td>
</tr>
<tr>
<td>targetMeshPartIds</td>
<td></td>
</tr>
<tr>
<td>sourceMeshLengthUnit</td>
<td><strong>m</strong>, <strong>ft</strong>, <strong>in</strong>, <strong>cm</strong> or <strong>mm</strong></td>
</tr>
<tr>
<td>targetMeshLengthUnit</td>
<td><strong>default</strong>: <strong>m</strong></td>
</tr>
<tr>
<td>sourceTemperatureUnit</td>
<td><strong>K</strong> or <strong>C</strong></td>
</tr>
<tr>
<td>targetTemperatureUnit</td>
<td><strong>default</strong>: <strong>K</strong></td>
</tr>
<tr>
<td>sourceMeshMassUnit</td>
<td><strong>kg</strong>, <strong>t</strong> or <strong>g</strong></td>
</tr>
<tr>
<td>targetMeshMassUnit</td>
<td><strong>default</strong>: <strong>kg</strong></td>
</tr>
<tr>
<td>sourceMeshTimeUnit</td>
<td><strong>s</strong> or <strong>ms</strong></td>
</tr>
<tr>
<td>targetMeshTimeUnit</td>
<td><strong>default</strong>: <strong>s</strong></td>
</tr>
<tr>
<td>targetQuantityLocation</td>
<td><strong>NODE</strong>, <strong>ELEMENT</strong> or <strong>DEFAULT</strong></td>
</tr>
<tr>
<td></td>
<td><strong>default</strong>: <strong>DEFAULT</strong> = same location as source</td>
</tr>
<tr>
<td>optionMapAlgoType</td>
<td><strong>type=shapefct</strong></td>
</tr>
<tr>
<td></td>
<td><strong>type=weightedelement</strong></td>
</tr>
<tr>
<td></td>
<td><strong>type=nearest</strong></td>
</tr>
<tr>
<td></td>
<td><strong>type=conservative</strong></td>
</tr>
<tr>
<td></td>
<td><strong>default</strong>: <strong>type=shapefct</strong></td>
</tr>
<tr>
<td>optionUseNeighborhoodDefaultParameters</td>
<td>true or false</td>
</tr>
<tr>
<td></td>
<td><strong>default</strong>: false</td>
</tr>
<tr>
<td>optionNeighborhoodSearch</td>
<td>list of parameters [targetMeshLengthUnit unit]</td>
</tr>
<tr>
<td></td>
<td>necessary for the neighborhood search: normaldistance, numberofnodes and radius</td>
</tr>
<tr>
<td>optionMultiplicity</td>
<td>value that increases neighborhood searching range</td>
</tr>
<tr>
<td></td>
<td><strong>default</strong>: 1</td>
</tr>
</tbody>
</table>
### 6.6 Example for a Configuration File

Here you find an example for an **MpCCI FSIMapper** configuration file. Lines starting with the “#” sign will be ignored. This example is a mapping from a **FLUENT** mesh (with unit system “m”) to an **Abaqus** mesh (with unit system “mm”).

The **FLUENT** film temperature (in K) is read from the .dat file. The temperature is written for an **Abaqus** simulation in °C. The target quantity location is element–based and the shape function mapping algorithm is used with default parameters. The orphaned elements are going to be filled with 0.0. The `optionComputeRotativeAverage` keyword is set to false.

A detailed description of the mapping algorithms and the concept of orphan filling can be found in [7.1 Numerical Methods](#). The meaning of the parameters for the different mapping algorithms is also explained in this chapter in [7.1.2 Parameters for the Mapping](#). The easiest solution for many applications might be to use the default parameters.

---

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>orphanFillType</td>
<td>type=distance, type=average, type=nearest, type=none default: type=distance</td>
</tr>
<tr>
<td>fillOrphansValue</td>
<td>value that is assigned to orphaned nodes or elements. default: 0</td>
</tr>
<tr>
<td>optionComputeRotativeAverage</td>
<td>true or false default: false</td>
</tr>
<tr>
<td>rotationAngle</td>
<td>angle in angular degree</td>
</tr>
<tr>
<td>rotationAxisBeginPoint</td>
<td>x–, y– and z–coordinate of starting point separated by whitespace</td>
</tr>
<tr>
<td>rotationAxisEndPoint</td>
<td>x–, y– and z–coordinate of end point separated by whitespace</td>
</tr>
<tr>
<td>temperatureScaleValue</td>
<td>instead of several simulations with different temperatures it is also possible to scale temperature values by a certain factor e.g. having done a simulation with 500K and wanting to map a simulation with 600K the necessary scaling value would be 600/500 = 1.2</td>
</tr>
</tbody>
</table>

**export**

**export** 1002 TEMPERATURE Assign the quantity name to be regarded as temperature.

**export** 1003 WALLHTCOEFF Assign the quantity name to be regarded as wall heat transfer coefficient.

**export** 1004 WALLHEATFLUX Assign the quantity name to be regarded as wall heat flux.

**export** 1010 PRESSURE Assign the quantity name to be regarded as pressure.

**export** 1016 FORCE Assign the quantity name to be regarded as force.
FLUENT

targetFileType
ABAQUS

sourceQuantityMode
FLMTEMPHTC

targetQuantityMode
NONE

sourceMeshPartIds
136 146 114

targetMeshPartIds
1

sourceMeshLengthUnit
m

targetMeshLengthUnit
mm

sourceTemperatureUnit
K

targetTemperatureUnit
C

sourceMeshMassUnit
kg

targetMeshMassUnit
kg

sourceMeshTimeUnit
s

targetMeshTimeUnit
s

targetQuantityLocation # location of quantities on Abaqus mesh
ELEMENT

# mapping algorithm type option
optionMapAlgoType # mapping algorithm type option
type=shapefct

# neighborhood search options
optionUseNeighborhoodDefaultParameters
true
# fill orphans with a distance, average or nearest method
orphanFillType
type=distance

# fill-value for orphaned entities
fillOrphansValue
0.

optionComputeRotativeAverage
false

rotationAngle
30

rotationAxisBeginPoint
0 0 0

rotationAxisEndPoint
1 0 0

temperatureScaleValue
1.2
7 Theory Guide

7.1 Numerical Methods

7.1.1 Mapping Algorithms

The MpCCI FSIMapper currently offers three different algorithms to map quantity values between non-matching discretizations (see Figure 18). The distinctions and basic features of the mapping algorithms are:

Shape Function
Shape function mapping simply interpolates a field using the shape functions. It can map linear functions exactly if linear elements are used; respectively quadratic functions need quadratic elements (i.e. elements with mid-nodes) to be mapped exactly.

Weighted Element
Weighted element algorithm is designed to map element based values with quite small smoothing. It creates a relation between each target mesh node to a source mesh element and computes target element value by a distance weighted approach. Due to less smoothing it does recover higher gradients much better than the shape function approach.

Nearest
The nearest mapping is based on a very simple neighborhood search algorithm. Every node in the target mesh receives the average value of the k closest nodes in the source mesh. The number k of nodes to be searched is given to the searching algorithm as a parameter. Alternatively, the average values from nodes lying within a certain radius can be used. Since for every node the k closest nodes can be found – no matter how far away they are positioned – this mapping algorithm never produces any orphaned nodes.

Conservative
The conservative mapping is based on the nearest mapping algorithm and dedicated to integral quantities, e.g. forces. Every node in the target mesh receives a relative amount of the k closest nodes in the source mesh. Then, all unused nodes of source mesh distribute their information onto the k closest nodes of the target mesh.

Since for every node the k closest nodes can be found – no matter how far away they are positioned – this mapping algorithm never produces any orphaned nodes.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Shape Function</th>
<th>Nearest</th>
<th>Weighted Element</th>
<th>Conservative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neighborhood parameters</td>
<td>Default</td>
<td>Surface Mapping</td>
<td>Normal Distance</td>
<td>1.0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NHighest</td>
<td>Number of Nodes</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Radius</td>
<td>1.0000</td>
<td>Exponent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Multiplicity</td>
<td>1.0000</td>
<td></td>
</tr>
</tbody>
</table>

Figure 18: Mapping algorithms and parameters in MpCCI FSIMapper

7.1.2 Parameters for the Mapping

The mapping algorithms need different parameters to control the mapping process (see Figure 18). According to the chosen mapping algorithm the following parameters need to be set:
Default
No further parameter values are needed. The MpCCI FSIMapper uses default values for the chosen algorithm. These values depend on the source and target discretizations. For many applications, where the distances between the two geometries is quite small, this should yield good mapping results.

SurfaceMapping – Normal Distance
The maximal distance which is still accepted in normal direction. This is needed for “Shape Function” and “Weighted Element”.

\[ d_n \]

Figure 19: Normal distance \( d_n \) computation for element to node neighborhood used in shape function and weighted element mapping

A reasonable choice for the distance parameter might be

\[
\text{Normal Distance} \quad \text{max(SourceMaxEdgeLength, TargetMaxEdgeLength)}
\]

NNearest – Number of Nodes
This parameter is only needed for the neighborhood search used by the “Nearest” mapping algorithm. It gives the number of close nodes that are searched for every target point in the source mesh. A sensible choice for this parameter is e.g. 4.

Radial – Radius
This parameter is only needed for the neighborhood search used by the “Nearest” mapping algorithm. As an alternative to the search for a certain number of close nodes, the specification of this parameter leads to a search for all source nodes lying in a circle around the target point with the given radius.

Conservative – Maximal Distance
This parameter is only needed for the neighborhood search used by the “Conservative” mapping algorithm. If the target model is only a subset of the source model, a conservative transfer of non-matching areas to the subset will lead to unwanded commutation of loads. Thus, the maximal search distance of data transfer can be limited.

Multiplicity
This parameter is needed for “Shape Function” and “Weighted Element” mapping algorithms. There it is used for the iterative neighborhood search leading to a more robust algorithm. The default value is 1.0. If a bigger value is chosen by the user, the searching radius (distance up to which a point is still regarded as a neighbor) will be enlarged (doubled) step-by-step until the given multiplicity of the starting radius is reached.

These suggestions for parameter settings should work fine with well matching geometries.

The parameter Normal Distance is not needed for the “Nearest” mapping algorithm. If the quantity values are mapped using this algorithm only one of the two parameters NNearest – Number of Nodes or Radial – Radius needs to be given.
7.1.3 Orphan Filling

The MpCCI FSIMapper offers four different methods to fill the orphaned nodes or elements of a mesh. These are the nodes or elements of the target mesh, that did not receive any quantity values during the mapping process. This can happen if the two geometries do not match closely, e.g. if the target geometry contains model parts that are not present in the source geometry.

The easiest way to assign a value to the orphaned nodes is to define a constant “orphan fill value”. Alternatively one of four available methods for orphan filling can be used. The distance filler can take an exponent value as an optional parameter.

During the filling process the orphan filler builds a “hole list”. Most orphans will be located in one of these holes, but it is possible for orphans to not appear in the hole list. For example free floating orphaned regions of a mesh, that are not connected to any other part, do not form a hole because there are no border points from where values might be interpolated.

It is possible to hand one optional parameter to the distance type orphan filler. For all other fillers the specification of the type is all that is needed. The four different available orphan filling methods are:

**Distance**
This orphan filler computes the distance to every border point for each orphaned node in a hole. The border values are then scaled with the reciprocal distance and added up for every orphan. One optional parameter can be handed to this filler: The **Exponent Factor**. This parameter determines to which power the distance - in the denominator - is taken. The default value for the **Exponent Factor** is 2.0, achieving a scaling with the square distance.

**Nearest**
The **Nearest** orphan filler divides the target mesh by mapped and orphaned points or elements into two parts. It then uses a nearest search algorithm finding the by **Number of Nodes** specified number of closest mapped values for an orphan and computes the orphan value by a distances weighted averaging. The default value for **Number of Nodes** is 4.

**Average**
This simple algorithm assigns the average quantity value of the border points to every orphan. The quantity therefore is constant on every hole.

**Default Value**
All orphans are going to receive the specified orphan fill value.

![Figure 20: Methods and parameters for orphan filling in MpCCI FSIMapper](image-url)
7.2 Geometry and Quantity Transformations

7.2.1 Fourier Transformation and Windowing

The Fourier transformation of the discrete transient signal \( (s(t_k))_{k=0}^{L-1} \) with \( L \) equidistant time steps of size \( \tau \) results in frequency dependent complex data. The maximal frequency, which can be extracted from the signal is \( F = 1/2\tau \). The size of the discrete frequency steps is determined by the reversed time period of \( s \).

If the transient signal does not build a full period, Windowing is recommended. This feature multiplies the signal with a so-called window function \( w \). The Fourier transformation is then applied to the signal \( (s(t_k) \cdot w(k))_{k=0}^{L-1} \).

Several predefined window functions\(^1\) are offered:

- **Hann**
  \[
  w(k) = 0.5 - 0.5 \cdot \cos \left( \frac{2\pi k}{L-1} \right), \quad k = 0, \ldots, L-1
  \]

- **Hamming**
  \[
  w(k) = 0.54 - 0.46 \cdot \cos \left( \frac{2\pi k}{L-1} \right), \quad k = 0, \ldots, L-1
  \]

- **Blackman**
  \[
  w(k) = 0.42 - 0.5 \cdot \cos \left( \frac{2\pi k}{L-1} \right) + 0.08 \cdot \cos \left( \frac{4\pi k}{L-1} \right), \quad k = 0, \ldots, L-1
  \]

- **Blackman-Harris**
  \[
  w(k) = a_0 - a_1 \cos \left( \frac{2\pi k}{L-1} \right) + a_2 \cos \left( \frac{4\pi k}{L-1} \right) - a_3 \cos \left( \frac{6\pi k}{L-1} \right), \quad k = 0, \ldots, L-1
  \]
  with \( a_0 = 0.35875 \), \( a_1 = 0.48829 \), \( a_2 = 0.14128 \), \( a_3 = 0.01168 \).

- **Blackman-Nuttall**
  \[
  w(k) = a_0 - a_1 \cos \left( \frac{2\pi k}{L-1} \right) + a_2 \cos \left( \frac{4\pi k}{L-1} \right) - a_3 \cos \left( \frac{6\pi k}{L-1} \right), \quad k = 0, \ldots, L-1
  \]
  with \( a_0 = 0.3635819 \), \( a_1 = 0.4891775 \), \( a_2 = 0.1365995 \), \( a_3 = 0.0106411 \).

- **Barlett**
  \[
  w(k) = \frac{2}{L-1} \cdot \left( \frac{L-1}{2} - \left| k - \frac{L-1}{2} \right| \right), \quad k = 0, \ldots, L-1
  \]

- **Barlett-Hann**
  \[
  w(k) = 0.62 - 0.48 \cdot \left| \frac{k}{L-1} - 0.5 \right| - 0.38 \cdot \cos \left( \frac{2\pi k}{L-1} \right), \quad k = 0, \ldots, L-1
  \]

- **Cosinus**
  \[
  w(k) = \cos \left( \frac{\pi k}{L-1} - \frac{\pi}{2} \right), \quad k = 0, \ldots, L-1
  \]

\(^1\) taken from https://de.wikipedia.org/wiki/Fensterfunktion, July 5, 2016
• Tukey

\[
    w(k) = \begin{cases} 
    1, & \text{if } 0 \leq k \leq \frac{\alpha(L-1)}{2} \\
    \frac{1}{2} \left[ 1 + \cos \left( \frac{2\pi k}{\alpha(L-1)} - \pi \right) \right], & \text{if } \frac{\alpha(L-1)}{2} \leq k \leq (L-1)(1-\frac{\alpha}{2}) \\
    \frac{1}{2} \left[ 1 + \cos \left( \frac{2\pi k}{\alpha(L-1)} - \pi \left( \frac{L-1}{\alpha} - 1 \right) \right) \right], & \text{if } (L-1)(1-\frac{\alpha}{2}) \leq k \leq (L-1) 
\end{cases}
\]

with \( \alpha = 0.5 \).

• Lanczos

\[
    w(k) = \text{sinc} \left( \frac{2k}{L-1} - 1 \right), \quad k = 0, \ldots, L-1
\]

The user can define his own window function as a one-columned file, where the \( L \) window function values \( w(k) \), \( k = 0, \ldots, L-1 \) are defined.

### 7.2.2 Cyclic Symmetry of Quantities

The mapping of quantities present on a cyclic symmetric source model to a full target model requires to build previously the corresponding full source mesh with data.

The (complex) data on cyclic symmetric models can exhibit a special periodicity, defined by the so-called (forward/backward) nodal diameter \( ND \in \mathbb{N}, 0 \leq ND \leq \left\lfloor \frac{n}{2} \right\rfloor \) (or cyclic symmetry mode).

It defines the temporal phase lag between the data of two neighboring sections as \( \frac{2\pi ND}{n} \) (forward) or \( -\frac{2\pi ND}{n} \) (backward). A nodal diameter 0 means that the data are the same on each section (“constant”). For an even number of cyclic symmetric sections, the nodal diameter \( n/2 \) indicates, that the data changes sign from section to section (“alternating”). An illustration of the meaning of nodal diameters is given in Figure 21 for the example of a periodic model with \( n = 8 \).

![Figure 21: Phase lag between two neighboring sections for different nodal diameters on a 8-periodic model](image-url)
Quantities of cyclic symmetric structures can exhibit the following nodal diameters in MpCCI FSIMapper:

<table>
<thead>
<tr>
<th>Analysis Type</th>
<th>n even/odd</th>
<th>Available Nodal Diameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static</td>
<td>n even</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>n odd</td>
<td>0</td>
</tr>
<tr>
<td>Transient</td>
<td>n even</td>
<td>0, n/2</td>
</tr>
<tr>
<td></td>
<td>n odd</td>
<td>0</td>
</tr>
<tr>
<td>Harmonic</td>
<td>n even</td>
<td>0, 1, ..., n/2</td>
</tr>
<tr>
<td></td>
<td>n odd</td>
<td>0, 1, ..., (n - 1)/2</td>
</tr>
</tbody>
</table>

The nodal diameters $0 < ND < n/2$, which only occur for harmonic cyclic symmetric data, are referred to as “paired”. Those quantities are described with both real and imaginary part and both are used simultaneously in order to create the data of the full model.

The data’s cyclic symmetry mode, which is needed to calculate the quantity’s values on the full model, is determined by the periodicities in the considered system.

For instance, periodic (rotating) disturbances produced by the $k$-th harmonic of an $m$-periodic part lead to a forward excitation nodal diameter $ND \in \{0, 1, \ldots, \lfloor n/2 \rfloor \}$, if there exists a positive integer $a$ with $ND = -(k \cdot m - a \cdot n)$. If there is a positive integer $b$ with $ND = k \cdot m - b \cdot n$ the excitation shape is in backward mode. The cyclic symmetry axis needs to be chosen, such that the relative rotation speed is positive around this axis. Please refer to Wirth and Oeckerath [2015] for further information.

1. Only with a properly selected (forward/backward) nodal diameter, the data will be spatially continuous over the periodic boundaries in the corresponding full model. This is the prerequisite of a correct mapping to a full model or a model with different section shape, as shown in Figure 8.

2. The direction of the axis of cyclic symmetry and the forward/backward excitation mode are closely connected through the following: if the excitation shape is defined by a forward resp. backward nodal diameter $ND$ with given cyclic symmetry axis $\overrightarrow{a}$, then it is equivalent to define it as backward resp. forward nodal diameter $ND$ with axis $-\overrightarrow{a}$.

3. For harmonic analyses, please check how the structural solver defines the nodal diameter/cyclic symmetry mode of the excitation. If the solver only knows the forward excitation mode (e.g. Abaqus) use the previous note in order to switch between backward and forward by changing the sign of the cyclic axis in the structural solver.
8 Tutorial

The tutorials demonstrate the use and the capabilities of FSIMapper for some special applications.

The general process of mapping data from a result file (“source”) to a structural model (“target”) is visualized in Figure 1. First, the simulation of the source model is performed. The source result file (native or special export depending on simulation software) and a prepared mesh of the target model are read by MpCCI FSIMapper. With application dependent settings, which need to be defined in the GUI or for batch usage in the config file, the mapping is done. As a result MpCCI FSIMapper creates a file(s) which contains the mapped quantities for the target mesh in the target code format. The file(s) is included in the target simulation in order to define the loading applied by the mapped source data.

8.1 Mapping of Electromagnetic Forces for Transient and Harmonic Analyses

8.1.1 Problem Description

In this tutorial, a two pole generator, called TEAM-24, is studied with respect to its vibration behavior. A transient electromagnetic simulation is performed using Infolytica MagNet where the resulting forces acting on the stator are calculated. By a transient mapping and a subsequent Fourier transformation of the data, the loading for a frequency response analysis is created. Here, MSC NASTRAN is used as target simulation code.

Rotor and stator exhibit two poles each (cf. Figure 22), where the rotor turns 1° per ms, i.e. with a frequency of 2.7 Hz. Due to the symmetry of the problem the model is truncated at its symmetry plane.

![Figure 22: TEAM-24 model in MagNet](image)

The files concerning this tutorial are located in

- MpCCI FSIMapper as part of the MpCCI installation:
  
  
  
  "$<MpCCI_home>/tutorial/FSIMapper/Team24-Stator_ElectromagneticForces"

- MpCCI FSIMapper standalone installation:
  
  
  
  "$<MpCCI_FSIMapper_home>/tutorial/Team24-Stator_ElectromagneticForces"

Before mapping, copy the files to your working directory.
8.1.2 Source Result File

In MagNet the TEAM-24 generator is simulated by the Solve → Transient 3D with Motion... solver. The transient solver options (Solve → Set Transient Options...) are set such that a constant time step is used (cf. Figure 23) which is obligatory for the Fourier transformation in MpCCI FSIMapper. It is sufficient to simulate only one half turn (180° in 180 ms) because of the cyclic symmetry with two poles.

The magnitude of the force density at 23 ms calculated by MagNet is shown in Figure 24.

For the use in MpCCI FSIMapper the surface force density has to be exported in MagNet using a special export called “Exporter for MpCCI”. The procedure is the following (cf. Figure 25):

- Select in the object tree the component whose surface load is to be exported (here “Stator”)
- Open Extensions → Exporter for MpCCI
- Enter as “Period” 180
• Push the Start button

![MagNet Exporter for MpCCI](image)

Figure 25: MagNet’s “Exporter for MpCCI”

The exporter creates the file "SFD-Stator.vtk" including all time steps between 0 ms and 179 ms.

For a Fourier transformation without windowing (as it is implemented in MpCCI FSIMapper) it is necessary to use transient data points which can be stacked one behind the other in order to create continuously the data for the following periods, as shown in the first picture in Figure 26.

As the data at 180 ms is exactly the same as at 0 ms and the time steps 0 to 179 prescribe one period (in the sense of the first picture in Figure 26), the prerequisites for a correct Fourier transformation are fulfilled.

In the .vtk file the SI unit system is used.
Figure 26: How to define transient data used by the Fourier transformation (without windowing). The blue dots are the data points provided for the mapping (here, a discrete sine-signal of frequency 5 Hz and sampling time step 0.01 s). The red dots are generated by stacking the data behind the end of the provided time signal. Only the first definition is correct.
8.1.3 Target Mesh File

The MSC NASTRAN target mesh models the stator by first order hexa and penta elements, see Figure 27. The mapping surface is defined by shell elements (green) which use the same nodes as the volume elements (blue).

![Figure 27: Target structural mesh (blue) with shell elements (green) defining the mapping surface](image)

The unit system used is \( \text{mm}\cdot\text{t}\cdot\text{s} \) which leads to force unit \( \text{N} \).

8.1.4 Mapping

Open the MpCCI FSIMapper GUI and change the settings in the “How to map/Transient” panel as follows (shown in Figure 28):

1. Select MagNet in the source file type drop-down menu and choose the "SFD-Stator.vtk" by pressing the ... button
2. Select “SFD-Stator.vtk” from the parts list
3. Set the source unit system to SI
4. Select the force density “SFD” in the “Force” variable drop-down menu
5. For the target simulation code select MSC NASTRAN and the mesh file "Stator_Hexa_Surface.bdf"
6. Select the surface abbreviated by “S” with MSC NASTRAN [PSHELL] PID 2 in the parts box.
7. Set the unit system of the target model to \( \text{mm}\cdot\text{t}\cdot\text{s} \).

This configuration maps for each time step the electromagnetic surface force density to the stator. In this tutorial we want the mapped transient force density to get transformed using Fourier analysis. Therefore, check “Apply Fourier Transformation” in the “Result” panel in order to output a corresponding include file that defines the loading for linear vibration analyses, cf. Figure 29.

Pressing the Map button starts the mapping process.
8.1 Mapping of Electromagnetic Forces for Transient and Harmonic Analyses

Figure 28: The “What to map” panel

Figure 29: The “Apply Fourier Transformation” option in the “Result” panel
When the mapping finishes, the **MpCCI Visualizer** shows the mapping results for each time step (cf. **Figure 30** at 23 ms). Due to the coarser target mesh, the mapped surface force density differs at first glance from the source data. But integrating both densities will lead to the same total force.

![Figure 30: Transient mapping result shown in MpCCI Visualizer at 23 ms](image)

With a sampling time step of $\tau = 0.001$ s and a number of samples $L = 180$ the Fourier transformation results in the frequency range from

$$0 \text{ Hz to } \frac{L/2}{\tau \cdot L} = 500 \text{ Hz in } \frac{1}{\tau \cdot L} = 5.5 \text{ Hz steps}$$

**MpCCI FSI Mapper** creates a MSC NASTRAN include file "**Stator_Hexa_Surface-mapped_FreqRespForce.bdf**" where the Fourier transformation results are applied as complex forces to the nodes building the mapping surface. It also defines the frequencies for the frequency response analysis:

```plaintext
$DL\text{OAD }= 600$
$FRE\text{Q }= 601$
$-----------------2-3-4-5-6-7-8-9-0-----------------
FRE\text{Q* } 
   601 0.000000000E+00 5.555555556E+00 1.111111111E+01
* 1.666666667E+01 2.222222222E+01 ...$
...$
$-----------------2-3-4-5-6-7-8-9-0-----------------
$N\text{ode 1, DDF 1}$
$RL\text{OAD1 }= 600 1 1 2$
$DAREA 1 1 11.00E+00$
$\text{Real part for each frequency}$
$TABLE\text{D1* }$
* 0.000000000E+00 5.05031394E-03 5.555555556E+00-1.127201827E-03
* 1.111111111E+01-2.292065593E-03 1.666666667E+01 1.449636200E-03
```
Moreover, \textit{MpCCI FSIMapper} creates the file "\texttt{Stator.Hexa_Surface-mapped_FreqRespForce.ccvx}" in order to post-process the Fourier transformed forces. It is opened in \textit{MpCCI Visualizer} and shows the corresponding transient force fluctuations (at 10 pseudo time steps) for each frequency. Open \textbf{Panels}→\textbf{Results Panel} and select in the “Results” list e.g. “Frequency 005.555555555” as “Scalar” and “Vector” quantity, see Figure 31a and 31b. Switch off \textbf{View}→\textbf{Perspective Projection} and select in the \textbf{Panels}→\textbf{Settings Panel} the outline representation as shown in Figure 31c.

Pressing the \textbf{button} animates the 10 pseudo time steps. With the value given in \textbf{File}→\textbf{Preferences}→\textbf{Maximum animation FPS} the animation speed is set. In this way one can visualize the frequency content of the transient signal, see Figure 32. Frequency 5.5 Hz has the highest influence in the transient force progression which is twice the rotation speed (resulting from the two poles). The higher orders play a decreasing role and the excitation shapes get more and more complicated.
Figure 31: MpCCI Visualizer settings

(a) “Results Panel”, “Scalar” quantity

(b) “Results Panel”, “Vector” quantity

(c) “Settings Panel”
Figure 32: Corresponding transient force excitation at 5.5 Hz generated for 10 pseudo time steps
8.1.5 Target Simulation

MpCCI FSIMapper exported a MSC NASTRAN include file containing the nodal force excitation resulting from a Fourier transformation of the electromagnetic surface force density calculated by MagNet. It is located in the same folder as the MSC NASTRAN input deck. The load case definition in MSC NASTRAN uses the file in the following way:

```plaintext
SOL 111 $ Modal Frequency Response
CEND
TITLE = Team-24_Stator3D_FreqResp
$ $METHOD = 150
SPC = 1
FREQ = 601
DLOAD = 600
DISP(PLOT) = ALL
STRESS(PLOT) = ALL
RESVEC = NO
ECHO = NONE
$ $BEGIN BULK
$ $-------2-------3-------4-------5-------6-------7-------8-------9-------0------
EIGRL 150 -0.1 1250
$ INCLUDE 'Stator_Hexa.bdf'
INCLUDE 'Stator_Hexa_Surface-mapped_FreqRespForce.bdf'
$ $-------2-------3-------4-------5-------6-------7-------8-------9-------0------
PARAM, POST, -1
$ ENDDATA
```

For the simulation the MSC NASTRAN mesh "Stator_Hexa.bdf" is used, i.e. the volume mesh without the shells defining the mapping surface (keeping the same node ID’s). Please refer to the MSC NASTRAN Dynamic Analysis User’s Guide for more information about the frequency response analyses types SOL 108 and SOL 111.

If the “Apply Fourier Transformation” option is not checked in the “Result” panel, then MpCCI FSIMapper creates the file "Stator_Hexa_Surface-mapped_TransientForce.bdf" which contains the mapped transient nodal forces. It can be used in SOL 109 and SOL 112 analyses.
8.2 Mapping of Harmonic Pressure Excitations in Turbomachinery

Periodic pressure fluctuations excite blade vibrations which can endanger the integrity of the whole system. The harmonic CFD methods offer the possibility to simulate those excitations in an efficient way without the initial attack time. They approximate the transient steady state flow behavior as a superposition of the time-averaged state variable and periodic fluctuations around this mean value. Due to the periodic nature of steady state turbomachinery flows the pressure oscillations can be split into frequency components (harmonics) which are multiples of the blade passing frequency. The solution approach of the harmonic CFD methods is to determine the complex amplitudes (magnitude and phase lag) for each of the superposed harmonics. Thus, the methods do not step successively in time but work in frequency domain.

Structural frequency response analyses can use directly the complex pressure data of the harmonic CFD simulation. They define the flow-induced excitations at every considered frequency.

In this tutorial the mapping process is shown for the Harmonic Balance Method of STAR-CCM+ and the Nonlinear Harmonic Method of FINE/Turbo. Here, the concept of cyclic symmetry and nodal diameters is exemplarily discussed.

8.2.1 Using the Harmonic Balance Method of STAR-CCM+

8.2.1.1 Problem Description

In this tutorial the flow-induced blade vibration of the NASA Rotor37 in operation is simulated. It is based on the STAR-CCM+ tutorial “Harmonic Balance: Single Stage Periodic Flow” where the transient pressure fluctuation is approximated by three harmonics and the time-averaged pressure. Here, the target simulation code is Abaqus.

The mapping process presented here creates the loading for a structural frequency response analysis in the following steps:

1. map 1\textsuperscript{st} harmonic pressure excitation
2. map 2\textsuperscript{nd} harmonic pressure excitation
3. map 3\textsuperscript{rd} harmonic pressure excitation
4. map time-averaged pressure

The tutorial also shows the mapping capability of MpCCI FSIMapper for periodic source and full target models.

Rotor37 has a geometrical periodicity of $n = 36$ blades. The upstream stator exhibits a periodicity of $m = 48$ blades; only the influence of this component is considered for this tutorial. The rotor rotates around the $z$-axis by $\omega = 17200 \text{ rpm} = 286.5 \text{ Hz}$.

The files concerning this tutorial are located in

- MpCCI FSIMapper as part of the MpCCI installation:
  "<MpCCI\_home>/tutorial/FSIMapper/Rotor37"

- MpCCI FSIMapper standalone installation:
  "<MpCCI\_FSIMapper\_home>/tutorial/Rotor37"

Before mapping, copy the files to your working directory.
8 Tutorial

8.2.1.2 Source Result File

In the STAR-CCM+ tutorial “Harmonic Balance: Single Stage Periodic Flow” the flow conditions in the Rotor37 are simulated using the Harmonic Balance method. The transient solution is approximated by 3 harmonics (in STAR-CCM+ called “modes”), where the 48-bladed stator is modelled as Gaussian wake (so not geometrically). Only one blade passage is modelled of the rotor.

The magnitude of the first ($k = 1$) pressure harmonic at $k \cdot m \cdot \omega = 13760$ Hz is shown in Figure 33. $m \cdot \omega$ refers to as blade passing frequency.

![Figure 33: Magnitude of the first pressure harmonic](image)

The harmonic results of the simulation are exported for the wetted surfaces into the EnSight Gold case format via File→Export... in STAR-CCM+, as shown in Figure 34.

The main resulting file is "singleRowHarmonic.case" which refers to the geometry file "singleRowHarmonic00000.geo", the exported time-averaged ("*.PressureFourierMode0") and harmonic pressures ("*.PressureFourierMode*Real", "*.PressureFourierMode*Imag").
Figure 34: Export the harmonic results as EnSight Gold case file
8.2.1.3 Target Mesh File

The structural target mesh file "rotor37\_fine.inp" comprises the whole rotor, as shown in Figure 35. Second order tetra elements were used.

\[\text{Figure 35: Target structural mesh of the full Rotor37}\]

The mapping surface, i.e., the surface where the pressure excitation acts, needs to be defined. Internally, Abaqus uses the keyword \*SURFACE, NAME=WettedSurface, TYPE=ELEMENT).

8.2.1.4 Mapping

First we map the first harmonic \((k = 1)\) to the structural Abaqus model. Open the MpCCI FSIMapper GUI and change the settings in the “How to map/Harmonic” panel as follows (shown in Figure 36):

1. Select “EnSight Case” in the source file type drop-down menu and choose the file "singleRowHarmonic.case" by pressing the . . . button.
2. Select the wetted surfaces in the source model: “Hub\_Surface\_Blade\_Row\_1” and “Blade\_Surface\_Blade\_Row\_1”
3. Set the source unit system to SI
4. Select for the real part of the mapped quantity “PressureFourierMode1Real” and for the imaginary part “PressureFourierMode1Imag”
5. For the target simulation code select “Abaqus” and the mesh file "rotor37\_fine.inp"
6. Select the surface “WettedSurface” in the parts box.
7. Set the unit system of the target model to m\(m\)-t-s.

Since the source mesh models only a periodic section of the rotor and the target mesh comprises the full rotor, a transformation is needed in order to generate matching geometries. For this purpose give the following information in the “Transformation/Geometry” subpanel (cf. Figure 37):

- Select the “User defined” transformation and check the option “Cyclic Symmetry”
- For the number of periodicities put in 36
- Define the cyclic symmetry axis via the two points \([0, 0, 0]\) and \([0, 0, 1]\), which is the positive z-axis
Due to the 48-bladed stator, the excitation of the first harmonic is of a nodal diameter 12 periodicity in backward mode, i.e. the excitation travels in the opposite direction as the rotation sense. This can be seen by the formula given in a note in 4.3 The “Transformation” Panel:

\[ ND = k \cdot m - b \cdot n = 1 \cdot 48 - b \cdot 36 = 12 \quad \text{with} \quad b = 1 \]

Since the rotation is positive around the z-axis, which was given as cyclic symmetry axis in the “Geometry” subpanel, select the paired backward nodal diameter 12 in the “Quantity” subpanel.

Pressing the [Map] button starts the mapping process.
Figure 37: The “Transformation/Geometry” subpanel
When the mapping finishes, the MpCCI Visualizer shows the mapping results, cf. Figure 38.

The periodic mesh and data were transformed by the given information in order to create the full source model and the corresponding data. The first harmonic complex pressure excitation has been revolved such that the data are spatially continuous over the periodic boundaries, see Figure 39. This is only the case for the correctly selected nodal diameter and excitation mode.
Figure 39: Cyclic Symmetry transformation creates continuous data over the periodic boundaries

MpCCI FSIMapper creates two Abaqus include files "rotor37_fine-mapped_HarmonicPressure_RE.inc" and "rotor37_fine-mapped_HarmonicPressure_IM.inc" where the real and imaginary part of the pressure is applied to the elements building the wetted surfaces:

```
** Exported real part of surface pressure
** *DLOAD, REAL
 205, P1, -0.00033113
 473, P1, -0.00028541
 10650, P1, 0.00043067
 10651, P1, 0.00057767
 10654, P1, -0.00037426
 10656, P1, -0.00048984
...
```

```
** Exported imaginary part of surface pressure
** *DLOAD, IMAGINARY
 205, P1, -0.00028541
 473, P1, -0.00028541
 10650, P1, -0.00037426
 10651, P1, -0.00037426
 10654, P1, 0.00042306
 10656, P1, 0.00062463
...
```

Moreover, MpCCI FSIMapper creates the file "rotor37_fine-mapped_HarmonicPressure.ccvx". It can be opened in MpCCI Visualizer and shows the corresponding transient pressure fluctuations for 18 pseudo time steps. Pressing the button animates them with a speed set in File→Preferences→Maximum animation FPS. In this way one can observe the backward travelling wave respective to the rotation sense.

Before mapping the two remaining harmonics, rename the exported files as "<filename>_H1."."
For the second harmonic \((k = 2)\) select the variables “PressureFourierMode2Real” and “PressureFourierMode2Imag”. The nodal diameter 12 in forward mode has to be selected in the “Quantity” subpanel. Also rename the newly created files as "<filename>_H2.<extension>". Repeat the procedure for the third harmonic with a nodal diameter 0.

Also, we map the time-averaged pressure which represents the pressure level around which the harmonics oscillate. Switch in the “What to map” panel to “Static” and select the same file and parts as in the “Harmonic” subpanel, see Figure 40. Select the quantity “PressureFourierMode0” and map by pressing the [Map] button. For static analyses only the nodal diameter 0 is available which is selected here by default.

Figure 40: The “What to map/Static” panel

The mapped static pressure is shown in Figure 41.
Figure 41: Mapping result of the time-averaged pressure
8.2.1.5 Target Simulation

MpCCI FSIMapper exported in summary for each of the tree harmonics two include files containing the real and the imaginary part of the pressure excitation. Moreover, an include file was exported which defines the mean/time-averaged loading. They are located in the same folder as the Abaqus input deck. The load case definition in Abaqus uses these files in the following way:

```plaintext
**
*STEP, NAME=MeanPressure, NLGEOM=YES, INC=100000
**
*STATIC
0.1,1,1e-4,1
**
*DLOAD, OP=MOD
*INCLUDE, INPUT=rotor37_fine-mapped_Pressure.inc
**
*END STEP
**
**
*STEP, NAME=ModalAnalysis
**
*FREQUENCY, EIGENSOLVER=LANCZOS
, 0, 100000, , ,
**
*END STEP
**
**
*STEP, NAME=SSD_1H
**
*STAYE D STATE DYNAMICS
13760, 0, , ,
**
*DLOAD, REAL
*INCLUDE, INPUT=rotor37_fine-mapped_HarmonicPressure_RE_H1.inc
**
*DLOAD, IMAGINARY
*INCLUDE, INPUT=rotor37_fine-mapped_HarmonicPressure_IM_H1.inc
**
*END STEP
**
**
*STEP, NAME=SSD_2H
**
*STAYE S TATE DYNAMICS
27520, 0, , ,
**
*DLOAD, REAL
*INCLUDE, INPUT=rotor37_fine-mapped_HarmonicPressure_RE_H2.inc
**
*DLOAD, IMAGINARY
```
*INCLUDE, INPUT=rotor37_fine-mapped_HarmonicPressure_IM_H2.inc
**
*END STEP
**
**
*STEP, NAME=SSD_3H
**
*STEADY STATE DYNAMICS
41280, 0, , ,
**
*DLOAD, REAL
*INCLUDE, INPUT=rotor37_fine-mapped_HarmonicPressure_RE_H3.inc
**
*DLOAD, IMAGINARY
*INCLUDE, INPUT=rotor37_fine-mapped_HarmonicPressure_IM_H3.inc
**
*END STEP
**

Please check the Abaqus User’s Keywords Reference Guide for the options of *STEADY STATE DYNAMICS concerning direct and modal solution methods.

⚠️ For the sake of completeness use in the first step also the centrifugal and the coriolis forces (by the keyword *DLOAD).

⚠️ The damping in the system has fundamental influence to the magnitude of blade vibration, so it is recommended to define it.

Here, the full Rotor37 is simulated. Another possibility would have been to model it in Abaqus by a periodic section using the keyword *CYCLIC SYMMETRY MODEL. The data periodicity (nodal diameter) is given in Abaqus by the option CYCLIC MODE=<nodal diameter> in the keywords *DLOAD or *CLOAD.

⚠️ Since Abaqus only knows forward excitation modes, for backward excitations the cyclic symmetry axis (which was determined such that the rotation is positive) has to be turned around (only) for the definition in the Abaqus keyword *CYCLIC SYMMETRY MODEL.
8.2.2 Using the Nonlinear Harmonic Method of FINE/Turbo

8.2.2.1 Problem Description

In this tutorial the blade vibration of the axial turbine “Aachen” in operation is simulated. It is based on the FINE/Turbo tutorial “2. Axial Turbine” and the transient pressure fluctuation is approximated by three harmonics and the time-averaged pressure. Here, the target simulation code is Abaqus.

In this tutorial, the 1st harmonic pressure excitation is mapped to a structural frequency response analysis. The remaining harmonics can be mapped in the same way using the information given in 8.2.2.4 Mapping a. The time-averaged pressure is mapped in the “What to map/Static” panel.

The tutorial also shows the mapping capability of MpCCI FSIMapper for periodic source and target models with different section shapes.

The considered rotor in the “Aachen” turbine has a geometrical periodicity of \( n = 41 \) blades. The upstream and downstream stators exhibit a periodicity of \( m = 36 \) blades both. The rotor rotates around the \( z \)-axis by \( \omega = -3500 \text{ rpm} = -58.3 \text{ Hz} \).

The files concerning this tutorial are located in

- MpCCI FSIMapper as part of the MpCCI installation:
  
  
  "<MpCCI_home>/tutorial/FSIMapper/AxialTurbineAachen"

- MpCCI FSIMapper standalone installation:
  
  "<MpCCI_FSIMapper_home>/tutorial/AxialTurbineAachen"

Before mapping, copy the files to your working directory.

8.2.2.2 Source Result File

The result of the static FINE/Turbo tutorial “2. Axial Turbine” is used as initial solution for a Nonlinear Harmonic simulation, where the transient solution is approximated by 3 harmonics in a rank-1 approach (equivalent to “Basic”). Only one blade passage is modelled for each of the three stages. For demonstration purposes the computation has been performed on grid level 1.

The magnitude of the first \( (k = 1) \) pressure harmonic at \( k \cdot m \cdot \omega = 2100 \text{ Hz} \) is shown in Figure 42. \( m \cdot \omega \) refers to as blade passing frequency. Since the stators have the same number of blades, the influence of both is included in this harmonic.

The results of the simulation are comprised in the FINE/Turbo .cgns result file "Tutorial2_NLH_Rank1_Har3_111.cgns".

The harmonic data are only available in the volume mesh.

8.2.2.3 Target Mesh File

The structural target mesh file "aachenBlade.inp" comprises one section of the rotor, as shown in Figure 43. The section shape is different to the section in FINE/Turbo. First order hexa and penta elements were used.

The mapping surface, i.e. the surface where the pressure excitation acts, needs to be defined. Internally, Abaqus uses the keyword *SURFACE, NAME=MappingSurface, TYPE=ELEMENT.
Figure 42: Magnitude of the first pressure harmonic at 2100 Hz

Figure 43: Target structural mesh (blue) with wetted surface definition (red) of one periodic section of the axial turbine “Aachen”
8.2.2.4 Mapping

Here, we map only the first harmonic \((k = 1)\) to the structural Abaqus model. The remaining two harmonics are handled equivalently \((k = 2: \text{forward nodal diameter } 10, k = 3: \text{forward nodal diameter } 15)\).

Open the MpCCI FSIMapper GUI and change the settings in the “How to map/Harmonic” panel as follows (shown in Figure 44):

1. Select “FINE/Turbo” in the source file type drop-down menu and choose the file "Tutorial2_NLH_Ranki_Har3_111.cgns" by pressing the button.
2. Select the volume blocks (abbreviated by “V”) which surround the wetted surface of the rotor in the source model: “domain8” to “domain16”.
3. Set the source unit system to SI
4. Select for the real part of the mapped first harmonic pressure “ReP\(_1\)” and for the imaginary part “ImP\(_1\)”
5. For the target simulation code select “Abaqus” and the mesh file "aachenBlade.inp"
6. Select the surface “MappingSurface” in the parts box.
7. Set the target unit system to \(\text{mm} - \text{t-s}\).

Since the source mesh models a periodic section, which does not match the target mesh section, a periodic transformation is needed in order to provide the data on the surfaces which are not covered by the source mesh. For this purpose give the following information in the “Transformation/Geometry” subpanel (cf. Figure 45):

- Select the “User defined” transformation and check the option “Cyclic Symmetry”
- For the number of periodicities put in 41
- Define the cyclic symmetry axis via the two points \([0, 0, 0]\) and \([0, 0, -1]\), which is the negative z-axis (which implies a positive rotation direction)

Due to the 36-bladed stators, the excitation of the first harmonic is of a nodal diameter 5 periodicity in forward mode, i.e. the excitation travels in the same direction as the rotation sense. This can be seen by the formula given in a note in ▷4.3 The “Transformation” Panel ▷:

\[
ND = -(k \cdot m - a \cdot n) = -(1 \cdot 36 - a \cdot 41) = 5 \quad \text{with} \quad a = 1
\]

Since the rotation is positive around the negative z-axis, which was given as cyclic symmetry axis in the “Geometry” subpanel, select the paired forward nodal diameter 5 in the “Quantity” subpanel, see Figure 46.

Pressing the Map button starts the mapping process.
Figure 44: The “What to map” panel
Figure 45: The “Transformation/Geometry” panel
Figure 46: The “Transformation/Quantity” panel
When the mapping finishes, the MpCCI Visualizer shows the mapping results, cf. Figure 47.

![Harmonic mapping result shown in MpCCI Visualizer](image)

**Figure 47:** Harmonic mapping result shown in MpCCI Visualizer

The periodic mesh and data were transformed by the given information in order to create the full source model and the corresponding data (cf. Figure 47a). The first harmonic complex pressure excitation has been revolved such that the data are spacially continuous over the periodic boundaries, which ensures a correct mapping to the target surfaces which are not covered by the periodic source mesh, see Figure 47b. The continuity over the periodic source boundaries is only provided for the correctly selected nodal diameter and excitation mode.

MpCCI FSIMapper creates two Abaqus include files "aachenBlade-mapped HarmonicPressure_RE.inc" and "aachenBlade-mapped HarmonicPressure_IM.inc" where the real and imaginary part of the pressure is applied to the nodes building the wetted surfaces:

```plaintext
** ** Exported real part of surface pressure
** ** *LOAD, REAL
37881, 8, 0.00040665
37882, 8, 0.00046127
37883, 8, 0.00038815
37884, 8, 0.00032466
37885, 8, 0.00037576
37886, 8, 0.00031139
...```

MpCCI 4.6.1-1 X 95
** Exported imaginary part of surface pressure
**  *CLOAD, IMAGINARY
37881, 8, -0.00018233
37882, 8, -0.00010771
37883, 8, 0.00001771
37884, 8, -0.00033982
37885, 8, -0.00023237
37886, 8, -0.00014165
...

For an element-based pressure definition, select “Element” in the “How to map” panel in “Quantity location (target)”. The keyword defining the pressure is then *DLOAD instead of *CLOAD.

Moreover, MpCCI FSIMapper creates the file "aachenBlade-mapped_HarmonicPressure.ccvx". It can be opened in MpCCI Visualizer and shows the corresponding transient pressure fluctuations for 18 pseudo time steps. Pressing the button animates them with a speed set in File→Preferences→Maximum animation FPS.

### 8.2.2.5 Target Simulation

MpCCI FSIMapper exported for the first harmonic two include files containing the real and the imaginary part of the pressure excitation. They are located in the same folder as the Abaqus input deck. The load case definition in Abaqus uses these files in the following way:

**
*CYCLIC SYMMETRY MODEL, N=41
0, 0, 0, 0, 0, -1
**
**
**
*STEP, NAME=ModalAnalysis
**
*FREQUENCY, EIGENSOLVER=LANCZOS
, 0, 5250, , , ,
*SELECT CYCLIC SYMMETRY MODES, NMIN=5, NMAX=5
**
*END STEP
**
**
**
*STEP, NAME=SSD_1H
**
*STEADY STATE DYNAMICS
2100, 0, , ,
**
*CLOAD, REAL, CYCLIC MODE=5
*INCLUDE, INPUT=aachenBlade-mapped_HarmonicPressure_RE.inc
**
*CLOAD, IMAGINARY, CYCLIC MODE=5
*INCLUDE, INPUT=aachenBlade-mapped_HarmonicPressure_IM.inc
**
The periodicity of the “Aachen” rotor blade is defined in Abaqus via the keyword *CYCLIC SYMMETRY MODEL. Since the excitation is in forward mode with respect to the negative z-axis, \(0, 0, 0, 0, 0, -1\) is given here to define the symmetry axis. The first harmonic pressure will excite only nodal diameter 5 mode shapes, which are determined in the *FREQUENCY step. The excitation shape needs to be defined in the keyword *CLOAD by the option CYCLIC_MODE=5.

Please check the Abaqus User’s Keywords Reference Guide for the options of *STEADY_STATE_DYNAMICS concerning direct and modal solution methods.

1. For the sake of completeness use also the mean pressure (by a “Static” mapping), the centrifugal and the coriolis forces (by the keyword *DLOAD).

2. The damping in the system has fundamental influence to the magnitude of blade vibration, so it is recommended to define it.

3. Since Abaqus only knows forward excitation modes, for backward excitations the cyclic symmetry axis (which was determined such that the rotation is positive) has to be turned around (only) for the definition in the Abaqus keyword *CYCLIC SYMMETRY MODEL.
8.3 Mapping of Temperature for Microelectronic Devices

8.3.1 Problem Description

In this tutorial, called “Board”, the stationary temperature field inside a microelectronic device was simulated using Mentor Graphics computational fluid dynamics software FloTHERM. By a volume mapping of the resulting temperature field to a structural mechanics model a thermal stress analysis can be performed. Here, the mapping of temperature field is demonstrated for the software packages Abaqus, ANSYS and MSC NASTRAN.

Figure 48: Temperature distribution and external flow field simulated by FloTHERM

The files concerning this tutorial are located in

- MpCCI FSIMapper as part of the MpCCI installation:
  "<MpCCI_home>/tutorial/FSIMapper/Board"

- MpCCI FSIMapper standalone installation:
  "<MpCCI_FSIMapper_home>/tutorial/Board"

Before mapping, copy the files to your working directory.

8.3.2 Source Result File

The stationary simulation of the temperature field under operating conditions was performed using FloTHERM. The native FloTHERM model is not part of the MpCCI FSIMapper tutorial but its result exported (as .flofea file) from FloTHERM project manager. To export a simulation result from FloTHERM V10 and later, select the assembly in the Model panel. Using right-click on selected assembly, choose format “FLOFEA” below “Export Assembly”.

The FloTHERM model consists of 14611 hexahedron elements, as length unit meter and temperature unit Celsius is used.
8.3.3 Target Mesh Files

The structural target mesh consists of 1279 8-node brick elements, as unit system mm-ton-second-celsius is used.

**Abaqus**

In Abaqus DC3D8 elements are used. The model is split up in three Abaqus parts. Each part has a single part instance in the Abaqus *ASSEMBLY section. To receive volumetric temperature distribution, one *ELSET definition per part is created.

**ANSYS**

In ANSYS SOLID185 elements are used. The elements of the model are grouped in three components using the ANSYS cm command.

**MSC NASTRAN**

In MSC NASTRAN CHEXA elements are used. The elements of the model are assigned to three part ids.

8.3.4 Mapping of Temperature

Open the MpCCI FSIMapper GUI and go to the “Static” tab of the “What to map” panel.

1. As source code select FloTHERM in the format selection dropdown menu.
2. Click on “Specify source geometry file” and select "Board/FloTHERM/board.flofea"
3. Confirm selection by pressing Open
4. The selected file is scanned in background and available parts for mapping are listed
5. Select all parts of the source model
6. In “Temperature” dropdown menu select “TEMPERATURE”
7. As target code select:

**Abaqus**

a) Abaqus in the format selection dropdown menu.
b) Click on “Specify target geometry file” and select "Board/abqaus/board.inp"
c) Confirm selection by pressing Open
d) The selected file is scanned in background and available parts for mapping are listed
e) Select Elset -1 "Chips_All_Elements", Elset -2 “Board_All_Elements” and Elset -3 “Cooling_All_Elements”
f) The “What to map” panel of the mapping process now should look as shown in figure 49

**ANSYS**

a) ANSYS in the format selection dropdown menu.
b) Click on “Specify target geometry file” and select "Board/ANSYS/board.db"
c) Confirm selection by pressing Open

⚠️ Scanning of ANSYS.db files requires a present ANSYS Mechanical APDL installation.

If no installation is available for this tutorial select "Board/ANSYS/board.ml" instead. See section 5.10.4 for details about scan process.
d) The selected file is scanned in background and available components for mapping are listed
e) Select Volume 1 “BOARD”, Volume 2 “CHIPS” and Volume 3 “COOLING”
f) The “What to map” panel of the mapping process now should look as shown in figure 49.

### MSC NASTRAN

- **a)** MSC NASTRAN in the format selection dropdown menu.
- **b)** Click on “Specify target geometry file” and select “Board/MSC.Nastran/board.bdf”.
- **c)** Confirm selection by pressing Open.
- **d)** The selected file is scanned in background and available parts for mapping are listed.
- **e)** Select Volume 1 “1”, Volume 2 “2” and Volume 3 “2”.
- **f)** The “What to map” panel of the mapping process now should look as shown in figure 49.

8. Go to the “How to map” panel and select Node in the “Quantity location” section.

9. Press Map at the bottom of the MpCCI FSIMapper GUI and perform mapping.

![Figure 49: The “What to map” panel after step 7 for target code Abaqus](image)

### 8.3.5 Results and Target Simulation

After pressing the “Map” button, using the predefined process setup, the MpCCI FSIMapper runs the mapping. In the background both source and target model as well as the temperature distribution are...
8.3 Mapping of Temperature for Microelectronic Devices

Figure 50: Temperature mapping result of the microelectronic device at the end of the mapping process.

read from file system and the data is interpolated to the new geometry. Then, all data is sent to MpCCI Visualizer for postprocessing (cf. figure 50). Then, MpCCI FSIMapper exports nodal based temperature to an ASCII-based input file in the native solver syntax. They are located in the same folder as the structural mechanics input deck.

** Abaqus **

For Abaqus the include file "board-mapped_Temperature.inp" is written, wherein the *TEMPERATURE keyword is used to define initial temperature values.

** Exported nodal temperature  
** *TEMPERATURE  
ChipsInstance.1, 95.07985578  
ChipsInstance.2, 87.69397911  
ChipsInstance.3, 83.75685351  
ChipsInstance.4, 91.68932411  
ChipsInstance.5, 91.48773958  
ChipsInstance.6, 91.26212328

The temperature mapping result file can be included in another Abaqus simulation using the following include statement at *STEP level of the loadcase definition:

*INCLUDE, INPUT=board-mapped_Temperature.inp

** ANSYS **

For ANSYS the include file "board-mapped_Temperature.inc" is written, wherein the BF keyword in combination with TEMP is used to define initial temperature values.
The temperature mapping result file can be included in another ANSYS simulation by “Read Input” inside ANSYS Mechanical APDL.

### MSC NASTRAN

For MSC NASTRAN the include file "board-mapped_Temperature_SPC.inc" is written, wherein the SPC keyword is used to define initial temperature values.

<table>
<thead>
<tr>
<th>$ Exported nodal temperature</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPC 1 1 195.07985</td>
</tr>
<tr>
<td>SPC 1 2 187.69398</td>
</tr>
<tr>
<td>SPC 1 3 183.75684</td>
</tr>
<tr>
<td>SPC 1 4 191.68932</td>
</tr>
<tr>
<td>SPC 1 5 191.48774</td>
</tr>
<tr>
<td>SPC 1 6 191.26212</td>
</tr>
</tbody>
</table>

The temperature mapping result file can be included in another MSC NASTRAN simulation using the following include statement:

```
INCLUDE 'board-mapped_Temperature_SPC.inc'
```
Part XI

Appendix

Version 4.6.1
XI Appendix – Contents

Quantity Reference 4

Literature 38

Glossary 39

Keyword Index 42
## Quantity Reference

<table>
<thead>
<tr>
<th>Quantity</th>
<th>ANSYS</th>
<th>Abaqus</th>
<th>MSC Adams</th>
<th>FINE/Open</th>
<th>FINE/Turbo</th>
<th>FLUENT</th>
<th>FloMASTER</th>
<th>ANSYS icepak</th>
<th>MATLAB</th>
<th>MSC Marc</th>
<th>MSC NASTRAN</th>
<th>OpenFOAM</th>
<th>SIMPACK</th>
<th>STAR-CCM+</th>
<th>T/AItherm</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbsPressure</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acceleration</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AcstPressure</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AngularAcceleration</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AngularCoordinate</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AngularVelocity</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BodyForce</td>
<td>s</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CGAngle</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CGOmega</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CGPosition</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CGVelocity</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ChargeDensity</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Current1</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Current2</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Current3</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Current4</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CurrentDensity</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeltaTime</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Density</td>
<td>s</td>
<td>s</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DynPressure</td>
<td>s</td>
<td>s</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ElectrCond1</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ElectrCond3</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ElectrCondX</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ElectrCondY</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ElectrCondZ</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ElectricField</td>
<td>s</td>
<td>s</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ElectricFlux</td>
<td>s</td>
<td>s</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ElectricPot</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ElectrRes1</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ElectrRes3</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ElectrResX</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ElectrResY</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ElectrResZ</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enthalpy</td>
<td>s</td>
<td>s</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FilmTemp</td>
<td>r</td>
<td>r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Force</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

gs00                      |       |        |           |           |             |        |            |               |        |          |             |          |         |           |          |
gs01                      |       |        |           |           |             |        |            |               |        |          |             |          |         |           |          |
gs02                      |       |        |           |           |             |        |            |               |        |          |             |          |         |           |          |
gs03                      |       |        |           |           |             |        |            |               |        |          |             |          |         |           |          |
gs04                      |       |        |           |           |             |        |            |               |        |          |             |          |         |           |          |

gs05                      |       |        |           |           |             |        |            |               |        |          |             |          |         |           |          |
<table>
<thead>
<tr>
<th>Quantity</th>
<th>Ansys</th>
<th>Abaqus</th>
<th>MSC Adams</th>
<th>FINE/Open</th>
<th>FINE/Turbo</th>
<th>FLUENT</th>
<th>FloMASTER</th>
<th>ANSYS Icepak</th>
<th>JMag</th>
<th>MATLAB</th>
<th>MSC Marc</th>
<th>MSC NASTRAN</th>
<th>OpenFOAM</th>
<th>SIMPACK</th>
<th>STAR-CCM+</th>
<th>TATherm</th>
</tr>
</thead>
<tbody>
<tr>
<td>gs06</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gs07</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gv00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gv01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gv02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gv03</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gv04</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gv05</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gv06</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gv07</td>
<td>HeatFlux</td>
<td>s</td>
<td>s/r</td>
<td>s/r</td>
<td>HeatRate</td>
<td>r</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>HeatSource</td>
<td>s</td>
<td>s/r</td>
<td>s/r</td>
<td>IntFlag</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IterationNo</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>JouleHeat</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>JouleHeatLin</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>LorentzForce</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MagneticField</td>
<td>s</td>
<td>s/r</td>
<td>s/r</td>
<td>MagneticFlux</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MassFlowRate</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>MassFlowRate</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MassFlowRate</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>MassFlowRate</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NPosition</td>
<td>s</td>
<td>s/r</td>
<td>s/r</td>
<td>OverPressure</td>
<td>s/r</td>
<td>r</td>
<td>s/r</td>
<td>r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>OverPressure</td>
<td>s/r</td>
<td>r</td>
<td>s/r</td>
<td>PhysicalTime</td>
<td>s/r</td>
<td>r</td>
<td>s/r</td>
<td>r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PhysicalTime</td>
<td>s/r</td>
<td>r</td>
<td>s/r</td>
<td>PointPosition</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PointPosition</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>PorePressure</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PorePressure</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>PorousFlow</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PorousFlow</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>RealFlag</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RealFlag</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>RelPressure</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RelPressure</td>
<td>r</td>
<td>r</td>
<td>s/r</td>
<td>RelWallForce</td>
<td>r</td>
<td>r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RelWallForce</td>
<td>r</td>
<td>r</td>
<td>s/r</td>
<td>Residual</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Residual</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>SpecificHeat</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SpecificHeat</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>StaticPressure</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>StaticPressure</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>Temperature</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Temperature</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>ThermCond1</td>
<td>s</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ThermCond1</td>
<td>s</td>
<td>s/r</td>
<td>s/r</td>
<td>ThermCond2</td>
<td>s</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ThermCond2</td>
<td>s</td>
<td>s/r</td>
<td>s/r</td>
<td>ThermCond3</td>
<td>s</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ThermCond3</td>
<td>s</td>
<td>s/r</td>
<td>s/r</td>
<td>ThermCondX</td>
<td>s</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ThermCondX</td>
<td>s</td>
<td>s/r</td>
<td>s/r</td>
<td>ThermCondY</td>
<td>s</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ThermCondY</td>
<td>s</td>
<td>s/r</td>
<td>s/r</td>
<td>ThermCondZ</td>
<td>s</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ThermCondZ</td>
<td>s</td>
<td>s/r</td>
<td>s/r</td>
<td>TimeStepNo</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TimeStepNo</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>Torque</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quantity</td>
<td>ANSYS</td>
<td>Abaqus</td>
<td>MSC Adams</td>
<td>FINE/Open</td>
<td>FINE/Turbo</td>
<td>FLUENT</td>
<td>FloMASTER</td>
<td>ANSYS Icepak</td>
<td>JMag</td>
<td>MATLAB</td>
<td>MSC Marc</td>
<td>MSC NASTRAN</td>
<td>OpenFOAM</td>
<td>SIMPACK</td>
<td>STAR-CCM+</td>
<td>TAItherm</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------</td>
<td>--------</td>
<td>-----------</td>
<td>-----------</td>
<td>------------</td>
<td>--------</td>
<td>-----------</td>
<td>---------------</td>
<td>------</td>
<td>--------</td>
<td>----------</td>
<td>-------------</td>
<td>----------</td>
<td>---------</td>
<td>-----------</td>
<td>---------</td>
</tr>
<tr>
<td>TotalPressure</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/ r</td>
<td>s/r</td>
<td>s/ r</td>
<td>s/r</td>
<td>s/r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td></td>
</tr>
<tr>
<td>TotalTemp</td>
<td></td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td></td>
</tr>
<tr>
<td>Velocity</td>
<td>r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td></td>
</tr>
<tr>
<td>VelocityMagnitude</td>
<td></td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td></td>
</tr>
<tr>
<td>Voltage1</td>
<td>s/ r</td>
<td></td>
<td>s/ r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Voltage2</td>
<td>s/ r</td>
<td></td>
<td>s/ r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Voltage3</td>
<td>s/ r</td>
<td></td>
<td>s/ r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Voltage4</td>
<td>s/ r</td>
<td></td>
<td>s/ r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VolumeFlow</td>
<td></td>
<td>s/ r</td>
<td>s/ r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VolumeFlowRate</td>
<td></td>
<td>s/ r</td>
<td>s/ r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WallForce</td>
<td>r</td>
<td>s/ r</td>
<td>s/ r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WallHeatFlux</td>
<td>s/ r</td>
<td>r</td>
<td>s/ r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WallHTCoeff</td>
<td>r</td>
<td>r</td>
<td>s/ r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WallTemp</td>
<td>s/ r</td>
<td>s/ r</td>
<td>s/ r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Codes and Quantities: s = “code can send quantity”, r = “code can receive quantity”

**AbsPressure**

- **Absolute pressure [N/m²]**
- **Code API symbol:** MPCCI_QID_ABSPRESSURE
- **Default value:** 0.0
- **Dimension:** Scalar
- **Physical meaning:** Boundary condition: value
- **Interpolation type:** flux density
- **Coupling Dimensions:** Point, Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct ETAB</td>
<td>Direct</td>
</tr>
<tr>
<td>Abaqus</td>
<td>Face</td>
<td>Code</td>
<td>Buffer</td>
<td></td>
</tr>
<tr>
<td>FINE/Open</td>
<td>Face</td>
<td>Node, Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>FINE/Turbo</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>FLUENT</td>
<td>Face, Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Face</td>
<td>Code, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>MSC Marc</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>OpenFOAM</td>
<td>Line, Face, Volume</td>
<td>Code</td>
<td>Dir</td>
<td></td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>
### Acceleration
Acceleration vector \([m/s^2]\)
- **Code API symbol:** `MPCCI_QID_ACCELERATION`
- **Default value:** 0.0
- **Dimension:** Vector
- **Physical meaning:** Boundary condition: value
- **Interpolation type:** field
- **Coupling Dimensions:** Point, Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abaqus</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Buffer</td>
</tr>
<tr>
<td>MSC Adams</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td>Usermemory</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>

### Acoustic Pressure
Acoustic pressure \([N/m^2]\)
- **Code API symbol:** `MPCCI_QID_ACSTPRESSURE`
- **Default value:** 0.0
- **Dimension:** Scalar
- **Physical meaning:** Boundary condition: value
- **Interpolation type:** flux density
- **Coupling Dimensions:** Point, Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLUENT</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
</tbody>
</table>

### Angular Acceleration
Angular acceleration \([rad/s^2]\)
- **Code API symbol:** `MPCCI_QID_ANGULARACCELERATION`
- **Default value:** 0.0
- **Dimension:** Vector
- **Physical meaning:** Boundary condition: value
- **Interpolation type:** field
- **Coupling Dimensions:** Point, Line

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abaqus</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Buffer</td>
</tr>
<tr>
<td>MSC Adams</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td>Usermemory</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>

### Angular Coordinate
Quaternion \((i,j,k,w)\) \([-\sim]\)
- **Code API symbol:** `MPCCI_QID_ANGULARCOORD`
- **Default value:** 0.0
- **Dimension:** Quaternion
- **Physical meaning:** Grid displacement/coordinate
- **Interpolation type:** mesh coordinate
- **Coupling Dimensions:** Point, Line

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abaqus</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Buffer</td>
</tr>
<tr>
<td>MSC Adams</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td>Usermemory</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>SIMPACK</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>
AngularVelocity
Angular velocity [rad/s]
Code API symbol: MPCCI_QID_ANGULARVELOCITY
Default value: 0.0
Dimension: Vector
Physical meaning: Boundary condition: value
Interpolation type: field
Coupling Dimensions: Point, Line

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abaqus</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Buffer</td>
</tr>
<tr>
<td>MSC Adams</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td>UseMemory</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Buff</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>SIMPACK</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>

BodyForce
General body force density vector [N/m³]
Code API symbol: MPCCI_QID_BODYFORCE
Default value: 0.0
Dimension: Vector
Physical meaning: Momentum source
Interpolation type: flux density
Coupling Dimensions: Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Volume</td>
<td>Node, Element</td>
<td>ETAB</td>
<td>Buffer</td>
</tr>
<tr>
<td>Abaqus</td>
<td>Volume</td>
<td>Code</td>
<td>Direct</td>
<td>Buffer</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>

CGAngle
Moving obstacle CG angle [rad]
Code API symbol: MPCCI_QID_MO_ANGLE
Default value: 0.0
Dimension: Vector
Physical meaning: Grid displacement/coordinate
Interpolation type: g-max
Coupling Dimensions: Global

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
</tbody>
</table>

CGOmega
Moving obstacle CG angular velocity [rad/s]
Code API symbol: MPCCI_QID_MO_OMEGA
Default value: 0.0
Dimension: Vector
Physical meaning: Boundary condition: value
Interpolation type: g-max
Coupling Dimensions: Global

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
</tbody>
</table>
**CGPosition**  Moving obstacle CG position [m]
- Code API symbol: `MPCCI_QID_MO_POSITION`
- Default value: 0.0
- Dimension: Vector
- Physical meaning: Grid displacement/coordinate
- Interpolation type: g-max
- Coupling Dimensions: Global

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
</tbody>
</table>

**CGVelocity**  Moving obstacle CG velocity [m/s]
- Code API symbol: `MPCCI_QID_MO_VELOCITY`
- Default value: 0.0
- Dimension: Vector
- Physical meaning: Boundary condition: value
- Interpolation type: g-max
- Coupling Dimensions: Global

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
</tbody>
</table>

**ChargeDensity**  Charge density [C/m³]
- Code API symbol: `MPCCI_QID_CHARGEDENSITY`
- Default value: 0.0
- Dimension: Scalar
- Physical meaning: General
- Interpolation type: field
- Coupling Dimensions: Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Line, Volume</td>
<td>Element</td>
<td>ETAB</td>
<td>Direct</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
</tbody>
</table>

**Current1**  Electric current - phase 1 [A]
- Code API symbol: `MPCCI_QID_CURRENT1`
- Default value: 0.0
- Dimension: Scalar
- Physical meaning: General
- Interpolation type: g-max
- Coupling Dimensions: Global

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
</tbody>
</table>
### Current2

Electric current - phase 2 [A]

- **Code API symbol:** `MPCCI_QID_CURRENT2`
- **Default value:** 0.0
- **Dimension:** Scalar
- **Physical meaning:** General
- **Interpolation type:** g-max
- **Coupling Dimensions:** Global

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
</tbody>
</table>

### Current3

Electric current - phase 3 [A]

- **Code API symbol:** `MPCCI_QID_CURRENT3`
- **Default value:** 0.0
- **Dimension:** Scalar
- **Physical meaning:** General
- **Interpolation type:** g-max
- **Coupling Dimensions:** Global

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
</tbody>
</table>

### Current4

Electric current - phase 4 [A]

- **Code API symbol:** `MPCCI_QID_CURRENT4`
- **Default value:** 0.0
- **Dimension:** Scalar
- **Physical meaning:** General
- **Interpolation type:** g-max
- **Coupling Dimensions:** Global

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
</tbody>
</table>

### CurrentDensity

Electric current density vector [A/m²]

- **Code API symbol:** `MPCCI_QID_CURRENTDENSITY`
- **Default value:** 0.0
- **Dimension:** Vector
- **Physical meaning:** Boundary condition: face normal gradient
- **Interpolation type:** flux density
- **Coupling Dimensions:** Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Face, Volume</td>
<td>Code</td>
<td>UDM UDS</td>
<td>UDM</td>
</tr>
<tr>
<td>JMAG</td>
<td>Volume</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>MATLAB</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>
**DeltaTime**  
Time step size [s]

- **Code API symbol:** MPCCI_QID_TIMESTEP_SIZE
- **Default value:** 1.0
- **Dimension:** Scalar
- **Physical meaning:** General
- **Interpolation type:** g-min
- **Coupling Dimensions:** Global

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>Abaqus</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>MSC Adams</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td>Usermemory</td>
</tr>
<tr>
<td>FINE/Open</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>FINE/Turbo</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>FLUENT</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>FloMASTER</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ANSYS Icepak</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>JMAG</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>MCALE</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>MSA NASTRAN</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>OpenFOAM</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>SIMPACK</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>TAItherm</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>

**Density**  
Density [kg/m³]

- **Code API symbol:** MPCCI_QID_DENSITY
- **Default value:** 1.0
- **Dimension:** Scalar
- **Physical meaning:** Material property/general property
- **Interpolation type:** field
- **Coupling Dimensions:** Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>FINE/Open</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>FINE/Turbo</td>
<td>Volume</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>FLUENT</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
</tbody>
</table>

**DynPressure**  
Dynamic pressure [N/m²]

- **Code API symbol:** MPCCI_QID_DYNPRESSURE
- **Default value:** 0.0
- **Dimension:** Scalar
- **Physical meaning:** Boundary condition: value
- **Interpolation type:** flux density
- **Coupling Dimensions:** Point, Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLUENT</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td></td>
</tr>
</tbody>
</table>
**ElectrCond1**  
Electric conductivity - $xyz$ [S/m]  
Code API symbol: `MPCCI_QID_ELECTCOND1`  
Default value: 0.0  
Dimension: Scalar  
Physical meaning: Material property/general property  
Interpolation type: field  
Coupling Dimensions: Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Face, Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>JMAG</td>
<td>Volume</td>
<td>Node</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>MATLAB</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>

**ElectrCond3**  
Electric conductivity - $(x,y,z)$ [S/m]  
Code API symbol: `MPCCI_QID_ELECTCOND3`  
Default value: 0.0  
Dimension: Vector  
Physical meaning: Material property/general property  
Interpolation type: field  
Coupling Dimensions: Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Face, Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>JMAG</td>
<td>Volume</td>
<td>Node</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>MATLAB</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>

**ElectrCondX**  
Electric conductivity - $x$ [S/m]  
Code API symbol: `MPCCI_QID_ELECTCONDX`  
Default value: 0.0  
Dimension: Scalar  
Physical meaning: Material property/general property  
Interpolation type: field  
Coupling Dimensions: Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>

**ElectrCondY**  
Electric conductivity - $y$ [S/m]  
Code API symbol: `MPCCI_QID_ELECTCONDY`  
Default value: 0.0  
Dimension: Scalar  
Physical meaning: Material property/general property  
Interpolation type: field  
Coupling Dimensions: Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>
Electric Conductivity - $z$ [S/m]

- **Code API symbol**: MPCCI_QID_ELECTCONDZ
- **Default value**: 0.0
- **Dimension**: Scalar
- **Physical meaning**: Material property/general property
- **Interpolation type**: field
- **Coupling Dimensions**: Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>

Electric Field

- **Electric field vector** [V/m]

- **Code API symbol**: MPCCI_QID_ELECTRICFIELD
- **Default value**: 0.0
- **Dimension**: Vector
- **Physical meaning**: Boundary condition: face normal gradient
- **Interpolation type**: field

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Line, Volume</td>
<td>Element</td>
<td>ETAB</td>
<td></td>
</tr>
<tr>
<td>FLUENT</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDS UDM</td>
</tr>
</tbody>
</table>

Electric Flux

- **Electric flux vector** [C/m$^2$]

- **Code API symbol**: MPCCI_QID_ELECTRICFLUX
- **Default value**: 0.0
- **Dimension**: Vector
- **Physical meaning**: Boundary condition: face normal gradient
- **Interpolation type**: flux density

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Line, Volume</td>
<td>Element</td>
<td>ETAB</td>
<td></td>
</tr>
<tr>
<td>FLUENT</td>
<td>Face, Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDS UDM</td>
</tr>
</tbody>
</table>

Electric Potential

- **Electric Potential** [V]

- **Code API symbol**: MPCCI_QID_ELECTRICPOT
- **Default value**: 0.0
- **Dimension**: Scalar
- **Physical meaning**: Boundary condition: value
- **Interpolation type**: field

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Line, Face, Volume</td>
<td>Node</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>
**ElectrRes1**  
Electric resistivity - xyz [ohm m]  
Code API symbol: MPCCI_QID_ELECTRESV1  
Default value: 0.0  
Dimension: Scalar  
Physical meaning: Material property/general property  
Interpolation type: field  
Coupling Dimensions: Line, Face, Volume  

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Face, Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>

**ElectrRes3**  
Electric resistivity - (x,y,z) [ohm m]  
Code API symbol: MPCCI_QID_ELECTRESV3  
Default value: 0.0  
Dimension: Vector  
Physical meaning: Material property/general property  
Interpolation type: field  
Coupling Dimensions: Line, Face, Volume  

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Face, Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>

**ElectrResX**  
Electric resistivity - x [ohm m]  
Code API symbol: MPCCI_QID_ELECTRESVX  
Default value: 0.0  
Dimension: Scalar  
Physical meaning: Material property/general property  
Interpolation type: field  
Coupling Dimensions: Line, Face, Volume  

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>

**ElectrResY**  
Electric resistivity - y [ohm m]  
Code API symbol: MPCCI_QID_ELECTRESVY  
Default value: 0.0  
Dimension: Scalar  
Physical meaning: Material property/general property  
Interpolation type: field  
Coupling Dimensions: Line, Face, Volume  

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>
ElectrResZ  
Electric resistivity - \(\Omega\cdot m\)

- Code API symbol: MPCCI_QID_ELECTRESVZ
- Default value: 0.0
- Dimension: Scalar
- Physical meaning: Material property/general property
- Interpolation type: field
- Coupling Dimensions: Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>

Enthalpy  
Enthalpy density \([\text{W/m}^3]\)

- Code API symbol: MPCCI_QID_ENTHALPY
- Default value: 0.0
- Dimension: Scalar
- Physical meaning: General
- Interpolation type: flux density
- Coupling Dimensions: Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Volume</td>
<td>Node, Element</td>
<td>ETAB</td>
<td></td>
</tr>
<tr>
<td>FLUENT</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>ANSYS Icepak</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
</tbody>
</table>

FilmTemp  
Film temperature \([\text{K}]\)

- Code API symbol: MPCCI_QID_FILMTEMPERATURE
- Default value: 300.0
- Dimension: Scalar
- Physical meaning: Boundary condition: value
- Interpolation type: field
- Coupling Dimensions: Line, Face

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Face</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>Abaqus</td>
<td>Face</td>
<td>Code</td>
<td>Buffer Direct</td>
<td></td>
</tr>
<tr>
<td>FLUENT</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>ANSYS Icepak</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Face</td>
<td>Code, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>MSC Marc</td>
<td>Line, Face</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>OpenFOAM</td>
<td>Line, Face</td>
<td>Code</td>
<td>Dir</td>
<td></td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>TAITherm</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>
**Force**

Force [N]

- **Code API symbol:** MPCCI_QID_FORCE
- **Default value:** 0.0
- **Dimension:** Vector
- **Physical meaning:** Momentum source
- **Interpolation type:** flux integral
- **Coupling Dimensions:** Point, Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>Abaqus</td>
<td>Point, Face</td>
<td>Code</td>
<td>Direct</td>
<td>Buffer</td>
</tr>
<tr>
<td>MSC Adams</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td>Usermemory</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td></td>
</tr>
<tr>
<td>JMAG</td>
<td>Volume</td>
<td>Node</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>MATLAB</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>SIMPACK</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>

The quantity is currently not supported by any standard code.

---

**gs00**

Global scalar 00 [-]

- **Code API symbol:** MPCCI_QID_UGS_00
- **Default value:** 0.0
- **Dimension:** Scalar
- **Physical meaning:** General
- **Interpolation type:** g-max
- **Coupling Dimensions:** Global

The quantity is currently not supported by any standard code.

---

**gs01**

Global scalar 01 [-]

- **Code API symbol:** MPCCI_QID_UGS_01
- **Default value:** 0.0
- **Dimension:** Scalar
- **Physical meaning:** General
- **Interpolation type:** g-max
- **Coupling Dimensions:** Global

The quantity is currently not supported by any standard code.

---

**gs02**

Global scalar 02 [-]

- **Code API symbol:** MPCCI_QID_UGS_02
- **Default value:** 0.0
- **Dimension:** Scalar
- **Physical meaning:** General
- **Interpolation type:** g-max
- **Coupling Dimensions:** Global

The quantity is currently not supported by any standard code.
gs03  Global scalar 03 [-]
Code API symbol: MPCCI_QID_UGS_03
Default value: 0.0
Dimension: Scalar
Physical meaning: General
Interpolation type: g-max
Coupling Dimensions: Global

The quantity is currently not supported by any standard code.

gs04  Global scalar 04 [-]
Code API symbol: MPCCI_QID_UGS_04
Default value: 0.0
Dimension: Scalar
Physical meaning: General
Interpolation type: g-max
Coupling Dimensions: Global

The quantity is currently not supported by any standard code.

gs05  Global scalar 05 [-]
Code API symbol: MPCCI_QID_UGS_05
Default value: 0.0
Dimension: Scalar
Physical meaning: General
Interpolation type: g-max
Coupling Dimensions: Global

The quantity is currently not supported by any standard code.

gs06  Global scalar 06 [-]
Code API symbol: MPCCI_QID_UGS_06
Default value: 0.0
Dimension: Scalar
Physical meaning: General
Interpolation type: g-max
Coupling Dimensions: Global

The quantity is currently not supported by any standard code.

gs07  Global scalar 07 [-]
Code API symbol: MPCCI_QID_UGS_07
Default value: 0.0
Dimension: Scalar
Physical meaning: General
Interpolation type: g-max
Coupling Dimensions: Global

The quantity is currently not supported by any standard code.
gv00

Global vector 00 [-]

- Code API symbol: MPCCI_QID_UGV_00
- Default value: 0.0
- Dimension: Vector
- Physical meaning: General
- Interpolation type: g-max
- Coupling Dimensions: Global

The quantity is currently not supported by any standard code.

gv01

Global vector 01 [-]

- Code API symbol: MPCCI_QID_UGV_01
- Default value: 0.0
- Dimension: Vector
- Physical meaning: General
- Interpolation type: g-max
- Coupling Dimensions: Global

The quantity is currently not supported by any standard code.

gv02

Global vector 02 [-]

- Code API symbol: MPCCI_QID_UGV_02
- Default value: 0.0
- Dimension: Vector
- Physical meaning: General
- Interpolation type: g-max
- Coupling Dimensions: Global

The quantity is currently not supported by any standard code.

gv03

Global vector 03 [-]

- Code API symbol: MPCCI_QID_UGV_03
- Default value: 0.0
- Dimension: Vector
- Physical meaning: General
- Interpolation type: g-max
- Coupling Dimensions: Global

The quantity is currently not supported by any standard code.

gv04

Global vector 04 [-]

- Code API symbol: MPCCI_QID_UGV_04
- Default value: 0.0
- Dimension: Vector
- Physical meaning: General
- Interpolation type: g-max
- Coupling Dimensions: Global

The quantity is currently not supported by any standard code.
The quantity is currently not supported by any standard code.

The quantity is currently not supported by any standard code.

The quantity is currently not supported by any standard code.

The quantity is currently not supported by any standard code.

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Volume</td>
<td>Node, Element</td>
<td>ETAB</td>
<td></td>
</tr>
<tr>
<td>FLUENT</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>ANSYS Icepak</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
</tbody>
</table>
### HeatRate

Heat rate [W]

- **Code API symbol:** MPCCI_QID_HEATRATE
- **Default value:** 0.0
- **Dimension:** Scalar
- **Physical meaning:** Boundary condition: value
- **Interpolation type:** flux density
- **Coupling Dimensions:** Point, Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abaqus</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>Buffer</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td></td>
</tr>
<tr>
<td>OpenFOAM</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td></td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>

### HeatSource

General heat source density [W/m³]

- **Code API symbol:** MPCCI_QID_HEATSOURCE
- **Default value:** 0.0
- **Dimension:** Scalar
- **Physical meaning:** Energy source
- **Interpolation type:** flux density
- **Coupling Dimensions:** Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Volume</td>
<td>Node, Element</td>
<td>ETAB</td>
<td></td>
</tr>
<tr>
<td>FLUENT</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>ANSYS Icepak</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
</tbody>
</table>

### IntFlag

Control switch(Int) [-]

- **Code API symbol:** MPCCI_QID_INT_SWITCH
- **Default value:** 0
- **Dimension:** Scalar
- **Physical meaning:** General
- **Interpolation type:** g-max
- **Coupling Dimensions:** Global

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>ANSYS Icepak</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
</tbody>
</table>
### IterationNo

**Iteration number** [-]

- **Code API symbol:** `MPCCI_QID ITERATION_COUNT`
- **Default value:** 0
- **Dimension:** Scalar
- **Physical meaning:** General
- **Interpolation type:** g-max
- **Coupling Dimensions:** Global

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>FINE/Open</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>FLUENT</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>ANSYS Icepak</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>TAITherm</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>

### JouleHeat

**Joule heat density** [W/m³]

- **Code API symbol:** `MPCCI_QID JOULEHEAT`
- **Default value:** 0.0
- **Dimension:** Scalar
- **Physical meaning:** Energy source
- **Interpolation type:** flux density
- **Coupling Dimensions:** Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Volume</td>
<td>Element</td>
<td>Direct</td>
<td>ETAB</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>ANSYS Icepak</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>JMAG</td>
<td>Volume</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>MATLAB</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>

### JouleHeatLin

**Joule heat linearization** [W/m³K]

- **Code API symbol:** `MPCCI_QID JOULEHEATLIN`
- **Default value:** 0.0
- **Dimension:** Scalar
- **Physical meaning:** Energy source
- **Interpolation type:** flux density
- **Coupling Dimensions:** Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLUENT</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>ANSYS Icepak</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
</tbody>
</table>
LorentzForce  Lorentz force density vector [N/m³]

- Code API symbol: `MPCCI_QID_LORENTZFORCE`
- Default value: 0.0
- Dimension: Vector
- Physical meaning: Momentum source
- Interpolation type: flux density
- Coupling Dimensions: Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct ETAB</td>
<td>Direct</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Volume</td>
<td>Code</td>
<td>UDM</td>
<td>UDM</td>
</tr>
<tr>
<td>JMAG</td>
<td>Volume</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>MATLAB</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>

MagneticField  Magnetic field vector [A/m]

- Code API symbol: `MPCCI_QID_MAGNETICFIELD`
- Default value: 0.0
- Dimension: Vector
- Physical meaning: Boundary condition: face normal gradient
- Interpolation type: field
- Coupling Dimensions: Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Line, Volume</td>
<td>Element</td>
<td>ETAB</td>
<td></td>
</tr>
<tr>
<td>FLUENT</td>
<td>Volume</td>
<td>Code</td>
<td>UDM UDS</td>
<td>UDM</td>
</tr>
</tbody>
</table>

MagneticFlux  Magnetic flux density vector [T]

- Code API symbol: `MPCCI_QID_MAGNETICFLUX`
- Default value: 0.0
- Dimension: Vector
- Physical meaning: Boundary condition: face normal gradient
- Interpolation type: flux density
- Coupling Dimensions: Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>FLUENT</td>
<td>Face, Volume</td>
<td>Code</td>
<td>UDM UDS</td>
<td>UDM</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>

MassFlowRate  Mass flow rate [kg/s]

- Code API symbol: `MPCCI_QID_MASSFLOWRATE`
- Default value: 0.0
- Dimension: Scalar
- Physical meaning: Mass source
- Interpolation type: flux integral
- Coupling Dimensions: Point, Face

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLUENT</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>FloMASTER</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>OpenFOAM</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>
**MassFlowVect**

Mass flow vector [kg/s]

- Code API symbol: `MPCCI_QID_MASSFLOWVECT`
- Default value: 0.0
- Dimension: Vector
- Physical meaning: Mass source
- Interpolation type: flux integral
- Coupling Dimensions: Volume

The quantity is currently not supported by any standard code.

**MassFluxRate**

Mass flux rate [kg/m²s]

- Code API symbol: `MPCCI_QID_MASSFLUXRATE`
- Default value: 0.0
- Dimension: Scalar
- Physical meaning: Mass source
- Interpolation type: flux density
- Coupling Dimensions: Point, Face

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLUENT</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>FloMASTER</td>
<td>Point</td>
<td>Code</td>
<td>Direct Direct</td>
<td></td>
</tr>
<tr>
<td>MATLAB</td>
<td>Point</td>
<td>Code</td>
<td>Direct Direct</td>
<td></td>
</tr>
<tr>
<td>OpenFOAM</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td></td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td>Face</td>
<td>Code</td>
<td>Direct Direct</td>
<td></td>
</tr>
</tbody>
</table>

**MassFluxVect**

Mass flux vector [kg/m²s]

- Code API symbol: `MPCCI_QID_MASSFLUXVECT`
- Default value: 0.0
- Dimension: Vector
- Physical meaning: Mass source
- Interpolation type: flux density
- Coupling Dimensions: Volume

The quantity is currently not supported by any standard code.
### NPosition

Nodal position [m]

- **Code API symbol:** MPCCI_QID_NPOSITION
- **Default value:** 0.0
- **Dimension:** Vector
- **Physical meaning:** Grid displacement/coordinate
- **Interpolation type:** mesh coordinate
- **Coupling Dimensions:** Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Line, Face, Volume</td>
<td>Node</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>Abaqus</td>
<td>Face, Volume</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>FINE/Open</td>
<td>Face</td>
<td>Node</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>FINE/Turbo</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>FLUENT</td>
<td>Face, Volume</td>
<td>Code</td>
<td>Dir</td>
<td>Buf</td>
</tr>
<tr>
<td>JMAG</td>
<td>Volume</td>
<td>Node</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Face, Volume</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>MSC Marc</td>
<td>Line, Face, Volume</td>
<td>Node</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>MSC NASTRAN</td>
<td>Line, Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>OpenFOAM</td>
<td>Line, Face, Volume</td>
<td>Code</td>
<td>Dir</td>
<td></td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>

### OverPressure

Relative pressure [N/m²]

- **Code API symbol:** MPCCI_QID_OVERPRESSURE
- **Default value:** 0.0
- **Dimension:** Scalar
- **Physical meaning:** Boundary condition: value
- **Interpolation type:** flux density
- **Coupling Dimensions:** Point, Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct ETAB</td>
<td>Direct</td>
</tr>
<tr>
<td>Abaqus</td>
<td>Face</td>
<td>Code</td>
<td>Buffer</td>
<td></td>
</tr>
<tr>
<td>FINE/Open</td>
<td>Face</td>
<td>Node, Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>FINE/Turbo</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>FLUENT</td>
<td>Face, Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>JMAG</td>
<td>Volume</td>
<td>Node</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>MATLAB</td>
<td>Face, Volume</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>MSC Marc</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>OpenFOAM</td>
<td>Line, Face, Volume</td>
<td>Code</td>
<td>Dir</td>
<td></td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>
**PhysicalTime**

Physical time [s]

- **Code API symbol:** MPCCI_QID_PHYSICAL_TIME
- **Default value:** 0.0
- **Dimension:** Scalar
- **Physical meaning:** General
- **Interpolation type:** g-min
- **Coupling Dimensions:** Global

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>FINE/Open</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>FINE/Turbo</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>FLUENT</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>FloMASTER</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ANSYS Icepak</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>TAITherm</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>

**PointPosition**

Point position [m]

- **Code API symbol:** MPCCI_QID_POINTPOSITION
- **Default value:** 0.0
- **Dimension:** Vector
- **Physical meaning:** Grid displacement/coordinate
- **Interpolation type:** mesh coordinate
- **Coupling Dimensions:** Point

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>Abaqus</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Buffer</td>
</tr>
<tr>
<td>MSC Adams</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td>Usermemory</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>SIMPACK</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>

**PorePressure**

Pore pressure [N/m²]

- **Code API symbol:** MPCCI_QID_POREPRESSURE
- **Default value:** 0.0
- **Dimension:** Scalar
- **Physical meaning:** Boundary condition: value
- **Interpolation type:** flux density
- **Coupling Dimensions:** Point, Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abaqus</td>
<td>Face, Volume</td>
<td>Code</td>
<td>Direct</td>
<td>Buffer</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
</tbody>
</table>
**PorousFlow**  
Pore fluid flow [m/s]

- Code API symbol: MPCCI_QID_POREFLOW
- Default value: 0.0
- Dimension: Scalar
- Physical meaning: Boundary condition: face normal gradient
- Interpolation type: flux density
- Coupling Dimensions: Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abaqus</td>
<td>Face, Volume</td>
<td>Code</td>
<td>Direct</td>
<td>Buffer</td>
</tr>
</tbody>
</table>

**RealFlag**  
Control switch(Real) [-]

- Code API symbol: MPCCI_QID_REAL_SWITCH
- Default value: 0.0
- Dimension: Scalar
- Physical meaning: General
- Interpolation type: g-max
- Coupling Dimensions: Global

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>MSC Adams</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td>Usermemory</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>ANSYS Icepak</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>SIMPACK</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>

**RefPressure**  
Reference pressure [N/m²]

- Code API symbol: MPCCI_QID_REF_PRESSURE
- Default value: 1.12e5
- Dimension: Scalar
- Physical meaning: General
- Interpolation type: g-max
- Coupling Dimensions: Global

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>FINE/Open</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>FINE/Turbo</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>FLUENT</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>
### RelWallForce

Boundary relative force vector [N]

- **Code API symbol:** MPCCI_QID_RELWALLFORCE
- **Default value:** 0.0
- **Dimension:** Vector
- **Physical meaning:** Boundary condition: value
- **Interpolation type:** flux integral
- **Coupling Dimensions:** Line, Face

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Face</td>
<td>Node</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>Abaqus</td>
<td>Face</td>
<td>Code</td>
<td>Buffer</td>
<td></td>
</tr>
<tr>
<td>FINE/Open</td>
<td>Face</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>FINE/Turbo</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>FLUENT</td>
<td>Face, Line, Face</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>MSC Marc</td>
<td>Line, Face</td>
<td>Node</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>MSC NASTRAN</td>
<td>Line, Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>OpenFOAM</td>
<td>Line, Face</td>
<td>Code</td>
<td>Dir</td>
<td></td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>

### Residual

Global residual [-]

- **Code API symbol:** MPCCI_QID_GLOBAL_RESIDUAL
- **Default value:** 0.0
- **Dimension:** Scalar
- **Physical meaning:** General
- **Interpolation type:** g-max
- **Coupling Dimensions:** Global

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>ANSYS Icepak</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>TAITherm</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>

### SpecificHeat

Specific heat [J/kg K]

- **Code API symbol:** MPCCI_QID_SPECHEAT
- **Default value:** 1.0
- **Dimension:** Scalar
- **Physical meaning:** Material property/general property
- **Interpolation type:** field
- **Coupling Dimensions:** Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLUENT</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>ANSYS Icepak</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>
**Static Pressure**

Static pressure \[ \text{N/m}^2 \]

- **Code API symbol:** `MPCCI_QID_STATICPRESSURE`
- **Default value:** 0.0
- **Dimension:** Scalar
- **Physical meaning:** Boundary condition: value
- **Interpolation type:** flux density
- **Coupling Dimensions:** Point, Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAR-CCM+</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>

**Temperature**

Temperature \[ \text{K} \]

- **Code API symbol:** `MPCCI_QID_TEMPERATURE`
- **Default value:** 300.0
- **Dimension:** Scalar
- **Physical meaning:** Boundary condition: value
- **Interpolation type:** field
- **Coupling Dimensions:** Point, Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Line, Volume</td>
<td>Node, Element</td>
<td>Direct ETAB</td>
<td>Direct</td>
</tr>
<tr>
<td>Abaqus</td>
<td>Volume</td>
<td>Code</td>
<td>Direct</td>
<td>Buffer</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Face, Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>FloMASTER</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>ANSYS Icepak</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>JMAG</td>
<td>Volume</td>
<td>Node</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>MATLAB</td>
<td>Point, Volume</td>
<td>Code, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>OpenFOAM</td>
<td>Line, Face, Volume</td>
<td>Code</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>

**ThermCond1**

Thermal conductivity - xyz \[ \text{W/m} \cdot \text{K} \]

- **Code API symbol:** `MPCCI_QID_THERMCOND1`
- **Default value:** 0.0
- **Dimension:** Scalar
- **Physical meaning:** Material property/general property
- **Interpolation type:** field
- **Coupling Dimensions:** Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>FLUENT</td>
<td>Face, Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>ANSYS Icepak</td>
<td>Face, Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>
### ThermCond3

Thermal conductivity \(-(x,y,z)\) [W/m K]

- **Code API symbol:** `MPCI_QID_THERMCOND3`
- **Default value:** 0.0
- **Dimension:** Vector
- **Physical meaning:** Material property/general property
- **Interpolation type:** Field
- **Coupling Dimensions:** Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td>UDM</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Face, Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>ANSYS Icepak</td>
<td>Face, Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>

### ThermCondX

Thermal conductivity - x [W/m K]

- **Code API symbol:** `MPCI_QID_THERMCONDX`
- **Default value:** 0.0
- **Dimension:** Scalar
- **Physical meaning:** Material property/general property
- **Interpolation type:** Field
- **Coupling Dimensions:** Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td>UDM</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>ANSYS Icepak</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>

### ThermCondY

Thermal conductivity - y [W/m K]

- **Code API symbol:** `MPCI_QID_THERMCONDY`
- **Default value:** 0.0
- **Dimension:** Scalar
- **Physical meaning:** Material property/general property
- **Interpolation type:** Field
- **Coupling Dimensions:** Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td>UDM</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>ANSYS Icepak</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>
**ThermCondZ**  
Thermal conductivity - z [W/mK]

- **Code API symbol:** MPCCI_QID_THERMCONDZ
- **Default value:** 0.0
- **Dimension:** Scalar
- **Physical meaning:** Material property/general property
- **Interpolation type:** field
- **Coupling Dimensions:** Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Line, Face, Volume</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>FLUENT</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>ANSYS Icepak</td>
<td>Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Volume</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>

**TimeStepNo**  
Time step number [-]

- **Code API symbol:** MPCCI_QID_TIMESTEP_COUNT
- **Default value:** 0
- **Dimension:** Scalar
- **Physical meaning:** General
- **Interpolation type:** g-max
- **Coupling Dimensions:** Global

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>FINE/Open</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>FLUENT</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>ANSYS Icepak</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>TAITherm</td>
<td>Global</td>
<td>global</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>

**Torque**  
Torque [N.m]

- **Code API symbol:** MPCCI_QID_TORQUE
- **Default value:** 0.0
- **Dimension:** Vector
- **Physical meaning:** Momentum source
- **Interpolation type:** flux integral
- **Coupling Dimensions:** Point, Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>Abaqus</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Buffer</td>
</tr>
<tr>
<td>MSC Adams</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td>Usermemory</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>SIMPACK</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>
**TotalPressure**  
Total pressure [N/m²]  
Code API symbol: `MPCCI_QID_TOTALPRESSURE`  
Default value: 0.0  
Dimension: Scalar  
Physical meaning: Boundary condition: value  
Interpolation type: flux density  
Coupling Dimensions: Point, Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLUENT</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>FloMASTER</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>OpenFOAM</td>
<td>Line, Face, Volume</td>
<td>Code</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>

**TotalTemp**  
Total Temperature [K]  
Code API symbol: `MPCCI_QID_TOTALTEMP`  
Default value: 300.0  
Dimension: Scalar  
Physical meaning: Boundary condition: value  
Interpolation type: field  
Coupling Dimensions: Line, Face, Volume

The quantity is currently not supported by any standard code.

**Velocity**  
Velocity vector [m/s]  
Code API symbol: `MPCCI_QID_VELOCITY`  
Default value: 0.0  
Dimension: Vector  
Physical meaning: Boundary condition: value  
Interpolation type: field  
Coupling Dimensions: Point, Line, Face, Volume

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Volume</td>
<td>Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>Abaqus</td>
<td>Point, Face, Volume</td>
<td>Code</td>
<td>Direct</td>
<td>Buffer</td>
</tr>
<tr>
<td>MSC Adams</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td>Usermemory</td>
</tr>
<tr>
<td>FINE/Open</td>
<td>Face</td>
<td>Node, Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>FLUENT</td>
<td>Point, Face, Volume</td>
<td>Code</td>
<td>Dir</td>
<td>UDM Buf</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Point</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>MSC NASTRAN</td>
<td>Line, Face</td>
<td>Code</td>
<td>Direct</td>
<td>Dir</td>
</tr>
<tr>
<td>OpenFOAM</td>
<td>Line, Face, Volume</td>
<td>Code</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>SIMPACK</td>
<td>Point</td>
<td>Node</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td>Point, Face</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>
Velocity Magnitude

- **Velocity magnitude [m/s]**
  - Code API symbol: `MPCCI_QID_VELOCITYMAG`
  - Default value: 0.0
  - Dimension: Scalar
  - Physical meaning: Boundary condition: value
  - Interpolation type: field
  - Coupling Dimensions: Point, Line, Face, Volume

Voltage 1

- **Electric voltage - phase 1 [V]**
  - Code API symbol: `MPCCI_QID_VOLTAGE1`
  - Default value: 0.0
  - Dimension: Scalar
  - Physical meaning: General
  - Interpolation type: g-max
  - Coupling Dimensions: Global

Voltage 2

- **Electric voltage - phase 2 [V]**
  - Code API symbol: `MPCCI_QID_VOLTAGE2`
  - Default value: 0.0
  - Dimension: Scalar
  - Physical meaning: General
  - Interpolation type: g-max
  - Coupling Dimensions: Global

Voltage 3

- **Electric voltage - phase 3 [V]**
  - Code API symbol: `MPCCI_QID_VOLTAGE3`
  - Default value: 0.0
  - Dimension: Scalar
  - Physical meaning: General
  - Interpolation type: g-max
  - Coupling Dimensions: Global
### Voltage4

Electric voltage - phase 4 \( [V] \)

- **Code API symbol:** MPCCI_QID_VOLTAGE4
- **Default value:** 0.0
- **Dimension:** Scalar
- **Physical meaning:** General
- **Interpolation type:** g-max
- **Coupling Dimensions:** Global

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Global</td>
<td>global</td>
<td>APDL</td>
<td>APDL</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Global</td>
<td>global</td>
<td>Dir</td>
<td>Dir</td>
</tr>
</tbody>
</table>

### VolumeFlow

Volume flow vector \( [m^3/s] \)

- **Code API symbol:** MPCCI_QID_VOLFLOWVECT
- **Default value:** 0.0
- **Dimension:** Vector
- **Physical meaning:** Mass source
- **Interpolation type:** flux integral
- **Coupling Dimensions:** Volume

The quantity is currently not supported by any standard code.

### VolumeFlowRate

Volume flow rate \( [m^3/s] \)

- **Code API symbol:** MPCCI_QID_VOLFLOWRATE
- **Default value:** 0.0
- **Dimension:** Scalar
- **Physical meaning:** Mass source
- **Interpolation type:** flux integral
- **Coupling Dimensions:** Point, Face

The quantity is currently not supported by any standard code.

### WallForce

Boundary absolute force vector \( [N] \)

- **Code API symbol:** MPCCI_QID_WALLFORCE
- **Default value:** 0.0
- **Dimension:** Vector
- **Physical meaning:** Boundary condition: value
- **Interpolation type:** flux integral
- **Coupling Dimensions:** Line, Face

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Face</td>
<td>Node</td>
<td>Direct</td>
<td>Fixed</td>
</tr>
<tr>
<td>Abaqus</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td>Buffer</td>
</tr>
<tr>
<td>FINE/Turbo</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>FLUENT</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Face</td>
<td>Code, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>MSC Marc</td>
<td>Line, Face</td>
<td>Node</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>MSC NASTRAN</td>
<td>Line, Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>OpenFOAM</td>
<td>Line, Face</td>
<td>Code</td>
<td>Dir</td>
<td></td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
</tbody>
</table>
### WallHeatFlux

Boundary normal heat flux density [W/m²]

- **Code API symbol:** MPCCI_QID_WALLHEATFLUX
- **Default value:** 0.0
- **Dimension:** Scalar
- **Physical meaning:** Boundary condition: face normal gradient
- **Interpolation type:** flux density
- **Coupling Dimensions:** Line, Face

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Face</td>
<td>Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>Abaqus</td>
<td>Face</td>
<td>Code</td>
<td>Buffer</td>
<td></td>
</tr>
<tr>
<td>FINE/Open</td>
<td>Face</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>FINE/Turbo</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>FLUENT</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>ANSYS Icepak</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>MSC Marc</td>
<td>Line, Face</td>
<td>Element</td>
<td></td>
<td>Direct</td>
</tr>
<tr>
<td>OpenFOAM</td>
<td>Line, Face</td>
<td>Code</td>
<td>Dir</td>
<td></td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>TAITherm</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>

### WallHTCoeff

Boundary heat transfer coefficient [W/m² K]

- **Code API symbol:** MPCCI_QID_WALLHTCOEFF
- **Default value:** 0.0
- **Dimension:** Scalar
- **Physical meaning:** Boundary condition: value
- **Interpolation type:** field
- **Coupling Dimensions:** Line, Face

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Face</td>
<td>Element</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>Abaqus</td>
<td>Face</td>
<td>Code</td>
<td>Buffer</td>
<td></td>
</tr>
<tr>
<td>FLUENT</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>ANSYS Icepak</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Face</td>
<td>Code, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>MSC Marc</td>
<td>Line, Face</td>
<td>Element</td>
<td></td>
<td>Direct</td>
</tr>
<tr>
<td>OpenFOAM</td>
<td>Line, Face</td>
<td>Code</td>
<td>Dir</td>
<td></td>
</tr>
<tr>
<td>STAR-CCM+</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>TAITherm</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>
WallTemp  
Boundary temperature [K]

Code API symbol: MPCCI_QID_WALLTEMPERATURE
Default value: 300.0
Dimension: Scalar
Physical meaning: Boundary condition: value
Interpolation type: field
Coupling Dimensions: Line, Face

<table>
<thead>
<tr>
<th>Code</th>
<th>Coupling Dimensions</th>
<th>Location</th>
<th>Send option</th>
<th>Receive option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSYS</td>
<td>Face</td>
<td>Node, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>Abaqus</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td>Buffer</td>
</tr>
<tr>
<td>FINE/Open</td>
<td>Face</td>
<td>Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>FINE/Turbo</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>FLUENT</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>ANSYS Icepak</td>
<td>Face</td>
<td>Code</td>
<td>Dir</td>
<td>UDM</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Face</td>
<td>Code, Element</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>MSC Marc</td>
<td>Line, Face</td>
<td>Node</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>OpenFOAM</td>
<td>Line, Face</td>
<td>Code</td>
<td>Dir</td>
<td>Dir</td>
</tr>
<tr>
<td>STAR-CMM+</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td>Direct</td>
</tr>
<tr>
<td>TAITherm</td>
<td>Face</td>
<td>Code</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>

YI00  
Species mass fraction 00 [-]

Code API symbol: MPCCI_QID_YI_00
Default value: 0.0
Dimension: Scalar
Physical meaning: Chemical component
Interpolation type: field
Coupling Dimensions: Line, Face, Volume

The quantity is currently not supported by any standard code.

YI01  
Species mass fraction 01 [-]

Code API symbol: MPCCI_QID_YI_01
Default value: 0.0
Dimension: Scalar
Physical meaning: Chemical component
Interpolation type: field
Coupling Dimensions: Line, Face, Volume

The quantity is currently not supported by any standard code.

YI02  
Species mass fraction 02 [-]

Code API symbol: MPCCI_QID_YI_02
Default value: 0.0
Dimension: Scalar
Physical meaning: Chemical component
Interpolation type: field
Coupling Dimensions: Line, Face, Volume

The quantity is currently not supported by any standard code.
**YI03**  
Species mass fraction 03 [-]  
Code API symbol: MPCCI_QID_YI_03  
Default value: 0.0  
Dimension: Scalar  
Physical meaning: Chemical component  
Interpolation type: field  
Coupling Dimensions: Line, Face, Volume

The quantity is currently not supported by any standard code.

**YI04**  
Species mass fraction 04 [-]  
Code API symbol: MPCCI_QID_YI_04  
Default value: 0.0  
Dimension: Scalar  
Physical meaning: Chemical component  
Interpolation type: field  
Coupling Dimensions: Line, Face, Volume

The quantity is currently not supported by any standard code.

**YI05**  
Species mass fraction 05 [-]  
Code API symbol: MPCCI_QID_YI_05  
Default value: 0.0  
Dimension: Scalar  
Physical meaning: Chemical component  
Interpolation type: field  
Coupling Dimensions: Line, Face, Volume

The quantity is currently not supported by any standard code.

**YI06**  
Species mass fraction 06 [-]  
Code API symbol: MPCCI_QID_YI_06  
Default value: 0.0  
Dimension: Scalar  
Physical meaning: Chemical component  
Interpolation type: field  
Coupling Dimensions: Line, Face, Volume

The quantity is currently not supported by any standard code.

**YI07**  
Species mass fraction 07 [-]  
Code API symbol: MPCCI_QID_YI_07  
Default value: 0.0  
Dimension: Scalar  
Physical meaning: Chemical component  
Interpolation type: field  
Coupling Dimensions: Line, Face, Volume

The quantity is currently not supported by any standard code.
**Y108**  
Species mass fraction 08 [-]  
Code API symbol: MPCCI_QID_YI_08  
Default value: 0.0  
Dimension: Scalar  
Physical meaning: Chemical component  
Interpolation type: field  
Coupling Dimensions: Line, Face, Volume  

The quantity is currently not supported by any standard code.

**Y109**  
Species mass fraction 09 [-]  
Code API symbol: MPCCI_QID_YI_09  
Default value: 0.0  
Dimension: Scalar  
Physical meaning: Chemical component  
Interpolation type: field  
Coupling Dimensions: Line, Face, Volume  

The quantity is currently not supported by any standard code.
Literature


Glossary

The aim of this glossary is to give a short description of the most important technical terms which are used in this manual and the context of MpCCI. For further reference, links to the corresponding manual sections are given.

**Algorithm step**

The *Algorithm step* is a panel of the wizard-like MpCCI GUI, where the coupling algorithm is defined. It specifies as well basic as code specific settings for the analysis type, solver settings, coupling steps, algorithm type, simulation duration, and code runtime see V-4.5 Algorithm Step ▷.

**architecture**

In the MpCCI context, architecture refers to the computer system, i.e. the combination of operating system and system hardware. Each characteristic combination supported by MpCCI is denoted by an *architecture token*, see V-2.3.1 MPCCI_ARCH - Architecture Tokens ▷. A list of current architecture tokens is given in II-5 Supported Platforms in MpCCI 4.6 ▷.

**association**

Association is the search for nodes/elements which lie close to a node/element of the partner mesh, also referred to as *contact search* or *neighborhood search*. The association is carried through during the initialization phase of MpCCI as a preparation for data interpolation. See V-3.3.1 Association ▷.

**client**

A *simulation code* with a *code adapter* acts as a client of the MpCCI server, i.e. the term *client* refers to a code or adapter, see V-3.5.1 Client-Server Structure of MpCCI ▷.

**code adapter**

A *code adapter* is a plug-in into a *simulation code* which allows coupling of the code with other codes via MpCCI. Code adapters can be based on user-defined functions and are usually linked with the simulation code, preferably as a shared library, see V-3.5.1 Client-Server Structure of MpCCI ▷. Code adapters are provided for all codes which are officially supported by MpCCI, further code adapters can be added using the code API (see VIII-2 MpCCI API ▷).

**coupling algorithm**

A *coupling algorithm* is the procedure of a coupled simulation. It determines when data is transferred, which code starts computing a new time step, etc. An overview of possible coupling algorithms is given in V-3.4 Coupling Process ▷.

**coupling component**

A coupling component is a part of a coupling region, typically a set of elements, see description of coupling region.

**coupling region**

Coupling regions define a part of a mesh which is coupled. In one coupling region, certain quantities are exchanged with the partner code. Each coupling region has a counterpart in the mesh of the partner code, which should have roughly the same geometry. A coupling region is composed of one or several coupling components. Coupling regions are defined in the MpCCI GUI (see IV-2.6 Regions Step – Defining Coupling Regions and Quantities ▷ and V-4.6 Regions Step ▷).

**coupling type**

A set of typical *coupling types* can be selected in the *Regions* step of the MpCCI GUI. Each type corresponds to a predefined selection of quantities for a coupling dimension, see V-4.6.8 Assigning Preconfigured Quantity Settings ▷ for details.

**Settings step**

The *Settings step* is a panel in the wizard-like MpCCI GUI. The control parameters for MpCCI can be set in this panel as described in V-4.8 Settings Step ▷.

**exchange**

Other term for transfer.
field
One of the two interpolation principles. Field interpolation is applied for quantities which do not depend on the element area, e.g. density, electric resistivity or temperature. See \ref{V-3.3.2.2 Field Interpolation}.

grid
See mesh.

Go step
The Go step is the last panel in the wizard-like MpCCI GUI, where options for starting the simulation codes can be set and the coupled simulation is started (and stopped). See \ref{V-4.9 Go Step}.

initialization
The initialization is carried through at the beginning of a coupled simulation. Both codes send their mesh data to MpCCI and the meshes are associated. See \ref{V-3.4 Coupling Process} for details.

interpolation
In most coupled simulations, data is transferred between non-matching grids. I.e. the data is given on one mesh and a corresponding set of data for a different mesh is expected by the other code. The process of finding such a distribution is called interpolation. Different methods for this procedure are described in \ref{V-3.3 Data Exchange}.

iterative coupling
Explicit steady-state and implicit coupling schemes are referred to as iterative coupling. For a steady-state analysis the final solution is reached by iterating the problem until a converged solution is reached. For implicit transient analyses each coupling step is repeated until a converged solution for the current coupled time step is reached. (cf. \ref{V-3.4.2 Coupling Schemes}).

mesh
The mesh (also called grid) consists of a set of nodes which are given by their spatial coordinates and a set of elements each which is connected to a typical number of nodes. The connection of nodes and elements is known as mesh topology. MpCCI can transfer data between mesh-based simulation codes, e.g. Finite Element (FE) or Finite Volume (FV) codes.

Models step
The Models step is the first panel in the wizard-like MpCCI GUI, where the models which shall be coupled are selected. In addition some code-specific options can be selected, see \ref{V-4.4 Models Step}.

Monitors step
The Monitors step is a panel in the wizard-like MpCCI GUI, where components and quantities of a code can be selected for monitoring, see \ref{V-4.7 Monitors Step}.

multiphysics
The purpose of MpCCI is to couple codes for the solution of multiphysics problems. Such problems consist of partial problems each of which can be classified into a different physical domain. Typical domains are fluid mechanics, solid mechanics, heat transfer or electromagnetism. See \ref{V-3.1 Multiphysics} for details.

neighborhood search
Other term for association.

orphaned nodes
If mesh association fails, some nodes cannot be associated with nodes or elements of the other mesh. This means they cannot be considered during data transfer, which usually yields severe errors. See \ref{V-3.3.2.3 Orphaned Nodes and Elements}.

quantity
Physical quantities must be exchanged between meshes for a coupled simulation. For each coupling case a typical set must be selected in the Regions step of MpCCI. A quantity has a typical SI-unit.
Regions step

The Regions step is a panel in the wizard-like MpCCI GUI where the coupling regions and their quantities are defined, see ▶ V-4.6 Regions Step ◄.

remote file browser

MpCCI uses a remote file browser for model file selection, which allows to connect to a remote computer via a remote shell. By selecting model file on a remote computer, a user determines where to run a simulation code. See ▶ V-4.10 Remote File Browser ◄ for a description of the MpCCI remote file browser.

scanner

In the Models step of the MpCCI GUI the scanner must be started to scan the model file. The scanner searches the file for definition of possible coupling regions and further code-dependent information. See ▶ VIII-2.5.2 Scanner.pm ◄ for a detailed description of what the scanner does.

server

The communication of a simulation code with MpCCI is based on a client-server model, i.e. during a computation a set of MpCCI servers is started, which are connected to the simulation codes (clients). The servers perform all tasks required for data transfer between two codes. See ▶ IV-1.3 Code Coupling with MpCCI ◄ and ▶ V-3.5.1 Client-Server Structure of MpCCI ◄ for a description.

simulation code

All software programs which can be coupled by MpCCI are commonly referred to as simulation codes. A description of all codes is given in the Codes Manual.

staggered method

MpCCI uses the staggered approach to solve multiphysics problems: Each code computes one time step independently and data is only exchanged after such a partial solution is obtained. This is also known as weak coupling. The solution procedure is discussed in ▶ IV-1.2 Solution of Coupled Problems ◄.

tracefile

The tracefile can be best described as graphical log file. If a tracefile is written during a computation (by an additional control process), the mesh geometry and transferred data is stored. The contents of the tracefile can be viewed with the MpCCI Visualizer as described in ▶ V-7 MpCCI Visualizer ◄.
Keyword Index

All page numbers are preceded by the roman part numbers:

I Overview
II Release Notes
III Installation Guide
IV Getting Started
V User Manual
VI Codes Manual
VII Tutorial
VIII Programmers Guide
IX How To
X MpCCI FSI Mapper
XI Appendix

Important entries are bold, italic entries refer to the glossary.

**MPCCI_TINFO,** VIII-29, VIII-30

*Abaqus, VI-15*
acoustics, V-20
*ActivePerl, III-9*
Adams, see MSC Adams
adaptive relaxation, V-35
Aitken relaxation, V-35
Algorithm step, *XI-39*
align, VI-13
angular interpolation, V-108

**ANSYS, VI-26**
API Kit, VIII-9
applied-force coupling, VII-118
   supported by MpCCI, II-33
Automotive Thermal Management, IX-4
axisymmetry, VII-79

backup, V-155
baffle shift, V-108
baffle thickness definition, VIII-62, VIII-62
batch execution, V-60, V-76, V-156
beam element types, VIII-76
bounding box check, V-22

**CFD, IV-7, IV-8, I-9, V-19**
cgs units, VII-11
clean
   mesh, V-21
checker, VIII-29, VIII-42
clean
data flow
   visualizer, V-177
data structure, VIII-76
debugging, V-14, V-143
directory

**client, XI-39, V-58**
code adapter, IV-6, VI-11, *XI-39*
code configuration directory, VIII-22
code integration, VIII-6, VIII-7
   step-by-step, VIII-12
command line interface, V-119
complex start, V-57
Computational Fluid Dynamics, see CFD configuration, VIII-54
configuration directory, VIII-22
conformal mesh, V-108
connection, VIII-45
coupling, *V-19*
coupling algorithm, *XI-39*
coupling component, XI-39
coupling manager functions, VIII-49
coupling point, VII-120
coupling region, *XI-39*
coupling schemes, V-49
coupling type, V-20, XI-39, V-103
couplingProcess, V-48

**data flow**
   visualizer, V-177
data structure, VIII-76
debugging, V-14, V-143
directory
MpCCI resource, V-15
code configuration, VIII-22
domain
physical, V-19
domain check, V-22, V-108
domain check visualizer, V-23
download, III-7
driven cavity, VII-23
driver function, VIII-67
duc heater, VII-88
elastic flap, VII-34
 elastic foundation, VIII-9
electromagnetic, VII-60, VII-69
 electromagnetism, V-20
electrothermal analysis, V-21
element definition, VIII-59, VIII-59
 element-element relationship, see intersection
dendianess, V-59
environment variable, V-12, III-14, VIII-29, V-138, V-143
example
code integration, VIII-9
exchange, XI-39, VIII-63, VIII-75
 initial, VIII-52
exhaust, VII-48
FE, IV-7, IV-8, I-9
FEM, V-19
field, XI-40
file system, V-117
files, V-74
temporary, V-16
finalization, V-48
FINE/Open, VI-48
FINE/Turbo, VI-55
Finite Element, see FE
firewall, V-59
flap
 elastic, VII-34
FLEXlm, see license
FlexNet Publisher, III-17
FloMASTER, VI-63
FLUENT, VI-71
fluid domain, IV-8
Fluid Dynamics, see CFD
fluid-structure interaction, see FSI
Fluid-structure interaction pump simulation, IX-13
force/displacement coupling, VII-118
foundation example, VIII-9
FSI, VII-11, V-20, VII-23, VII-34, VII-79
MpCCI FSIMapper, X-16
 Batch, X-45
 command, V-123
 How to map, X-25
 Transformation, X-22
 What to map, X-17
glossary, XI-39
Go step, XI-40, see GUI
Graphical User Interface, see GUI
grid, XI-40
GUI, V-76, V-124
 Algorithm step, IV-11, VIII-26, V-83
 check, V-114
 Go step, IV-20, VIII-27, V-113
 menus, V-77
 Models step, IV-10, VIII-24, V-81
 Monitors step, IV-18, V-106
 properties, V-77
 Regions step, IV-12, V-88
 automatic region generation, V-94
 region generation, V-94
 rules, V-94
Settings step, IV-19, V-107
 start, IV-9, V-114
 status, V-115
title, V-77
heat transfer, V-20
hexahedral element types, VIII-82
history size, V-108
HOME, III-14
home
MpCCI, V-11
home directory, V-13
hostlist file, V-60
ANSYS Icepak, VI-45
 implicit coupling, V-51
 info, VI-12, VIII-43
 information, V-138
 initial exchange, VIII-52
 initialization, XI-40, VIII-45, V-48, VIII-53
 installation, III-5, V-146
 Perl, III-32
 Windows, III-12
 multi-platform, III-11
 prerequisites, III-7
 interpolation, IV-5, V-28, XI-40, V-53
 iteration, V-48
 iterative coupling, XI-40
Java, III-9
JMAG, VI-91
job, V-108
job control, V-154
job scheduler, V-62
killer, VIII-29, VIII-43
killing jobs, V-164
license, III-17, V-146, V-147, V-149
FSIMapper, III-18
Morpher, III-18
limiter, V-42
Logviewer, V-173
loop, VIII-96
LSF, V-69, V-159
macro, VIII-76
Magneto-Thermal, VII-60, VII-69
manifold, VII-48
Marc, see MSC Marc
MATLAB, VI-98
mesh, XI-40
mesh check, V-21
mesh definition, VIII-55, VIII-73
mesh deletion, VIII-57
mesh motion, V-22, V-44, V-108
Microsoft Windows, see Windows
mixed solution type coupling, V-55
model file, VI-13
model preparation, IV-7
Models step, XI-40, see GUI
monitor
options, V-107
Monitors step, XI-40
morpher
MpCCI, VII-34, V-132, V-200
OpenFOAM, VI-149
STAR-CCM+, VI-170
mount, V-117
moving reference frame, VIII-61, VIII-61
moving reference frame correction (MRF correct), V-108
MpCCI
Run Implicit FSI, VI-80
API, VIII-6
Control Panel (FLUENT), VI-79
FSIMapper, X-7
Grid Morpher, VII-34, V-132, VI-149, V-200
GUI, see GUI, V-76
Logviewer, V-125, V-173
Monitor, V-126
Observer, V-136
Tools, V-129
Visualization, see Visualization, V-177
MPCCI_ARCH, see architecture
MPCCI_DEBUG, V-14
MPCCI_HOME, V-11, V-144
MPCCI_HOSTLIST_FILE, V-60
MPCCI_LICENSE_FILE, III-22
MPCCI_NOREMESH, III-24
MPCCI_RSHTYPE, III-24, V-60
MPCCI_TMPDIR, V-16
MS HPC, V-67
MSC Adams, VI-111
MSC Marc, VI-124
MSC NASTRAN, VI-132
MSI, see windows installer
multi-platform installation, III-11
multiphysics, IV-4, V-19, XI-40
N1GE, V-71, V-161
Nastran, see MSC NASTRAN
neighborhood search, XI-40, see association
network, V-20, III-24, V-58, V-115
network device, V-108
node definition, VIII-58, VIII-58
non-matching time steps, V-53
nozzle, VII-79
online monitor, V-107
OpenFOAM, VI-140
Morpher, VI-149
OpenPBS, V-70, V-160
OpenSSH, III-9
orphaned nodes, V-29, XI-40
output, VIII-50, V-111
overlap check, V-108
parallel run, V-58
part definition, VIII-55
part deletion, VIII-57
PATH, III-11, V-11
PBS, V-70, V-160
PBSPro, V-70, V-160
periodic components, V-90
periodic model, V-47
periodic models, VII-135
Perl, III-9, III-14, VIII-40
installation, III-32
physical domain, V-19
pipe nozzle, VII-79
platform, see architecture
point-element relationship, see minimal distance
polygonal element types, VIII-83
polyhedral element types, VIII-84
port, V-59
post limiter, V-42
post-processing, IV-29
pre check, V-108
pre limiter, V-42
prefix, VIII-51
predecessors
installation, III-7
pressure
reference, V-20
ptoj, V-167
pyramid element types, VIII-80
quadrilateral element types, VIII-78
quantity, IV-4, VIII-25, XI-40
convergence check, V-41
descriptor, VIII-95
disable conservation, V-41
limiter, V-42
operator, V-40
reference, XI-1
scale, V-42
Quasi-Newton
method, V-36
parameters, V-108
tutorial, VII-23
queuing system, V-62
radiation, V-20
ramping, V-33
reference pressure, V-20
FLUENT, VI-72
OpenFOAM, VI-141
region, see coupling region
Regions step, XI-41, see GUI
relation search, V-111
relaxation, V-32
adaptive, V-35
Aitken, V-35
Quasi-Newton, V-36
ramping, V-33
relaxation corrector, V-108
releases, VI-12
remeshing, VIII-65
remote computing, V-58
remote file browser, XI-41, V-115
remote shell, V-18, III-24, V-60, V-117, V-142, V-150
    testing, III-25
requirements
    code integration, VIII-7
resource directory, V-15
restart, V-57
results, IV-28
MpCCI-RSH, III-10
Preparing the .rhosts file, III-29
rsh, see remote shell
scaling, V-42
scanner, VI-13, VIII-29, VIII-40, XI-41
Start, IV-11
server, XI-41, V-58, V-168
Settings step, XI-39, see GUI
SGE, V-71, V-161
SGEEE, V-71, V-161
shape function, V-28
SIMPACK, VI-156
simulation code, VI-11, III-15, XI-41
Slave node mark, V-24
SLURM, V-72, V-162
software
    MpCCI, V-10
    third party, V-18
solid mechanics, V-19
spring mass, VII-118
ssh, see remote shell
stability, VII-118
staggered method, IV-5, XI-41
STAR-CCM+, VI-162
    Morpher, VI-170
starter, VIII-29, VIII-42
starting MpCCI, V-122
step-by-step code integration, VIII-12
stopper, VIII-29, VIII-43
subcmd, VIII-44
subcommands
    list, V-120
subcycling, V-50
surface coupling, IV-4
symmetry
    axial, VII-79
TAITherm, VI-186
temporary files, V-16
terminal window, V-137
testing
    installation, III-14
    license server, III-23
    MpCCI installation, V-151
    remote shell, III-25
tetrahedral element types, VIII-79
thermal coupling, V-21, VII-60, VII-69
third party software, V-18

time step size
  exchange, V-52

time tolerance, V-108

Torque, V-70, V-160

tracefile, IV-28, XI-41, V-111, V-177

transfer, VIII-63, VIII-63

transfer information, VIII-52

transformation matrix, VIII-61

triangle element types, VIII-77

unit, VII-79

units, VII-11, VI-14

user-defined function, VII-79

version
  mpcci, V-141

vibration, VII-11

Visualizer, IV-28, V-127, V-177
  Menus and Toolbars, V-179
  Mouse Interaction, V-191
  Panels, V-184
  Preferences Server Dialog, V-195
  Preferences Viewer Dialog, V-193
  Viewport Area, V-190

volume coupling, IV-4

vortex street, VII-11

wedge element types, VIII-81

Windows
  rcp, III-28
  rsh, III-28
  rlogin, III-28

Windows, III-14, III-28, III-28

windows installer, III-12

writer, V-111

xterm, V-137

Y-Junction, VII-100