

MpCCI
CouplingEnvironment

Part I

—

Overview

Version 4.5.0

MpCCI 4.5.0-1 Documentation

Part I Overview

PDF version

March 30, 2017

MpCCI is a registered trademark of Fraunhofer SCAI

www.mpcci.de



Fraunhofer Institute for Algorithms and Scientific Computing SCAI
Schloss Birlinghoven, 53754 Sankt Augustin, Germany

Abaqus and SIMULIA are trademarks or registered trademarks of Dassault Systèmes

ANSYS, FLUENT and ANSYS Icepak are trademarks or registered trademarks of Ansys, Inc.

Elmer is an open source software developed by CSC

FINE/Open and FINE/Turbo are trademarks of NUMECA International

Flowmaster is a registered trademark of Flowmaster Group NV

JMAG is a registered trademark of The JSOL Corporation

MATLAB is a registered trademark of The MathWorks, Inc.

MSC Adams, MSC Marc, MD NASTRAN and MSC NASTRAN are trademarks or registered trademarks of
MSC.Software Corporation

OpenFOAM is a registered trademark of OpenCFD Ltd.

PERMAS is a registered trademark of Intes GmbH

RadTherm, TAITherm is a registered trademark of ThermoAnalytics Inc.

SIMPACK is a registered trademark of SIMPACK AG.

STAR-CCM+ and STAR-CD are registered trademarks of CD adapco Group

ActivePerl has a Community License Copyright of Active State Corp.

FlexNet Publisher is a registered trademark of Flexera Software.

Java is a registered trademark of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds

Mac OS X is a registered trademark of Apple Inc.

OpenSSH has a copyright by Tatu Ylonen, Espoo, Finland

Perl has a copyright by Larry Wall and others

UNIX is a registered trademark of The Open Group

Windows, Windows XP, Windows Vista and Windows 7 are registered trademarks of Microsoft Corp.

I Overview – Contents

Preface	4
Typographical Conventions	6
Contents of all MpCCI Manuals	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 users 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	1
Preface		4
Typographical Conventions		6
Contents of all MpCCI Manuals		26
II	Release Notes	1
1	Introduction MpCCI 4.5	4
2	Changes and New Features in MpCCI 4.5.0-1	5
2.1	MpCCI Licensing	5
2.2	MpCCI Platforms	5
2.3	New Features	5
2.3.1	MpCCI License Manager	5
2.3.2	MpCCI GUI	5
2.3.3	MpCCI CouplingEnvironment	5
2.3.4	MpCCI Client	6
2.3.5	MpCCI FSIMapper	6
2.4	Further Enhancements of Existing Features	6
2.4.1	MpCCI GUI	6
2.4.2	MpCCI CouplingEnvironment	7
2.4.3	MpCCI Client	7
2.4.4	MpCCI FSIMapper	7
2.4.5	MpCCI Grid Morpher	7
2.4.6	MpCCI Batch	7
2.4.7	MpCCI Visualizer and MpCCI Monitor	8
2.4.8	MpCCI Tutorial	8
2.4.9	Code Specific Changes	8
2.5	Known Bugs and Limitations	10
3	Changes and New Features in the Earlier Releases	11
3.1	MpCCI 4.4.2-1	11
3.1.1	Further Enhancements of Existing Features	11
3.2	MpCCI 4.4.1-1	13
3.2.1	Further Enhancements of Existing Features	13
3.3	MpCCI 4.4.0-1	15
3.3.1	New Features	15
3.3.2	Further Enhancements of Existing Features	16
4	Prerequisites for MpCCI Installation	19
5	Supported Platforms in MpCCI 4.5	20
5.1	Platforms Supported by the MpCCI 4.5 Server	20
5.2	Codes Supported by MpCCI 4.5 on Different Platforms	21

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	Configure a License Manager as Linux Service	19
5.2.3	Configure a License Manager as Windows Service	19
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
9.1	Linux	32
9.2	Windows	32
9.2.1	Strawberry Perl for Windows	32
9.2.2	ActivePerl for Windows	32
9.2.3	File Name Associations for Perl under Windows	33

IV	Getting Started	1
1	Multiphysics Computation with MpCCI	4
1.1	Multiphysics	4
1.2	Solution of Coupled Problems	4
1.3	Code Coupling with MpCCI	5
2	Setting up a Coupled Simulation	7
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
2.4	Models Step – Choosing Codes and Model Files	9
2.5	Coupling Step – Defining Coupling Regions and Quantities	10
2.5.1	Build Coupling Regions	11
2.5.2	Define Quantities to be Exchanged	13
2.5.3	Assign Defined Quantities to Built Coupling Regions	15
2.6	Monitors Step – Defining Quantities for Monitoring	17
2.7	Edit Step – Specifying Further Coupling Options	17
2.8	Go Step – Configuring the Application Start-Up and Starting Server and Codes	18
2.8.1	Configuring the Initial Exchange Mode	18
2.8.2	Setting Option Parameters	19
2.8.3	Checking the Application Configuration	19
2.8.4	Starting the Coupled Simulation	20
2.8.5	Interrupting the Computation	25
3	Checking the Results	26
3.1	The MpCCI Visualizer	26
3.2	Post-Processing	27
V	User Manual	1
1	Introduction	7
1.1	Basic Structure of MpCCI	7
2	The MpCCI Software Package	9
2.1	Introduction	9
2.2	The MpCCI Home Directory	10
2.3	Environment and Environment Variables	11
2.3.1	MPCCI_ARCH - Architecture Tokens	12
2.3.2	MPCCI_DEBUG - for Debugging	13
2.3.3	MPCCI_TINFO	13
2.4	MpCCI Project and Output Files	15
2.4.1	MpCCI Project Files	15
2.4.2	MpCCI Server Input Files	15
2.4.3	Log Files	15
2.4.4	Tracefile	15
2.5	The MpCCI Resource Directory	16
2.6	Temporary Files	17
2.7	Third Party Software Used by MpCCI	19
2.7.1	Perl	19

2.7.2	Java	19
2.7.3	Remote Shell and Remote Copy	19
3	Code Coupling	20
3.1	Multiphysics	20
3.1.1	Physical Domains	20
3.1.2	Coupling Types	21
3.2	Mesh Checks	22
3.2.1	Mesh Motion Checks	23
3.2.2	Bounding Box Checks	23
3.2.3	Domain Check	23
3.2.4	Slave Node Mark	25
3.3	Data Exchange	26
3.3.1	Association	27
3.3.2	Interpolation	29
3.3.3	Handling of Coupled Quantities	32
3.4	Coupling Algorithms	43
3.4.1	Course of the Coupling Process	43
3.4.2	Stationary Problems	44
3.4.3	Transient Problems	44
3.4.4	Iterative Coupling	50
3.4.5	Exchange of Time Step Size	54
3.4.6	Subcycling	55
3.4.7	Non-matching Time Steps	57
3.4.8	Restarting a Coupled Simulation	59
3.5	Running MpCCI in a Network	60
3.5.1	Client-Server Structure of MpCCI	60
3.5.2	Hostlist File	61
3.5.3	Remote Shell and Remote Copy	62
3.6	Coupled Analysis in Batch Mode	62
3.6.1	General Approach	62
3.6.2	Job Scheduler Environment	63
4	Graphical User Interface	76
4.1	Starting and Exiting MpCCI GUI	76
4.1.1	Starting MpCCI GUI	76
4.1.2	Exiting MpCCI GUI	77
4.2	MpCCI GUI Properties	77
4.3	MpCCI GUI Menus	77
4.3.1	File Menu	78
4.3.2	Batch Menu	78
4.3.3	License Menu	79
4.3.4	Tools Menu	80
4.3.5	Codes Menu	80
4.3.6	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	Coupling Step	83
4.5.1	Global Variables	83
4.5.2	Editing Components	84

4.5.3	Coupling Components with Different Dimensions	88
4.5.4	Simplified Selection	88
4.5.5	Automatic Generation of Regions by Rules	89
4.5.6	Applying Region Properties	93
4.5.7	Quantity Specifications	94
4.5.8	Assigning Preconfigured Quantity Settings	97
4.5.9	Overview of the Coupled Regions	100
4.5.10	Requirements	101
4.6	Monitors Step	101
4.7	Edit Step	102
4.7.1	Monitor	102
4.7.2	Job	104
4.7.3	Relation Search	106
4.7.4	Output	107
4.8	Go Step	109
4.8.1	Configuring the MpCCI Coupling Server	109
4.8.2	Coupling Configuration Parameters	110
4.8.3	Checking the configuration	111
4.9	Remote File Browser	111
4.9.1	File Browser Handling	111
4.9.2	How to mount a new file system	113
5	Command Line Interface	115
5.1	Using the Command Line Interface	115
5.2	Overview of All Subcommands	116
5.3	Starting MpCCI	118
5.3.1	<code>mpcci fsimapper</code>	119
5.3.2	<code>mpcci gui</code>	120
5.3.3	<code>mpcci logviewer</code>	121
5.3.4	<code>mpcci monitor</code>	122
5.3.5	<code>mpcci visualize</code>	123
5.4	MpCCI Tools	125
5.4.1	<code>mpcci ccvxcat</code>	126
5.4.2	<code>mpcci cosimpre</code>	127
5.4.3	<code>mpcci morpher</code>	128
5.4.4	<code>mpcci observe</code>	131
5.4.5	<code>mpcci xterm</code>	132
5.5	Information and Environment	133
5.5.1	<code>mpcci arch</code>	134
5.5.2	<code>mpcci doc</code>	135
5.5.3	<code>mpcci info</code>	136
5.5.4	<code>mpcci env</code>	138
5.5.5	<code>mpcci home</code>	139
5.5.6	<code>mpcci where</code>	140
5.6	Installation and Licensing	141
5.6.1	<code>mpcci license</code>	142
5.6.2	<code>mpcci list</code>	143
5.6.3	<code>mpcci loutil</code>	144
5.6.4	<code>mpcci ssh</code>	145
5.6.5	<code>mpcci test</code>	146
5.7	Job Control	149
5.7.1	<code>mpcci backup</code>	150

5.7.2	<code>mpcci batch</code>	151
5.7.3	<code>mpcci batch LSF</code>	153
5.7.4	<code>mpcci batch PBS</code>	154
5.7.5	<code>mpcci batch N1GE</code>	155
5.7.6	<code>mpcci batch LoadLeveler</code>	156
5.7.7	<code>mpcci batch GLOBUS</code>	157
5.7.8	<code>mpcci clean</code>	158
5.7.9	<code>mpcci kill</code>	159
5.7.10	<code>mpcci ps</code>	161
5.7.11	<code>mpcci ptoj</code>	162
5.7.12	<code>mpcci server</code>	163
5.7.13	<code>mpcci top</code>	167
6	MpCCI Visualizer	168
6.1	Using the MpCCI Visualizer	168
6.1.1	Data Flow	168
6.1.2	Supported Platforms	168
6.1.3	Starting the Monitor	169
6.1.4	Starting the Visualizer	169
6.2	MpCCI Visualizer for .ccvx and online monitoring	169
6.2.1	Introduction	169
6.2.2	Main Window	170
6.2.3	Menus and Toolbars	170
6.2.4	Panels	176
6.2.5	Viewport Area	182
6.2.6	Preferences Viewer Dialog	185
6.2.7	Preferences Server Dialog	187
6.2.8	Store Animated Files Dialog	187
6.2.9	Error Dialog	189
6.2.10	Command Line Parameters	190
6.3	Frequently Asked Questions	191
7	MpCCI Grid Morpher	192
7.1	Using the MpCCI Grid Morpher	192
7.1.1	Options description	192
VI	Codes Manual	1
1	Overview	11
1.1	Common MpCCI Subcommands for Simulation Codes	12
1.2	Unit Systems	14
2	Abaqus	15
2.1	Quick Information	15
2.1.1	Supported Coupling Schemes	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	17
2.2.1	Model Preparation	17

2.2.2	Restart	17
2.2.3	Models Step	17
2.2.4	Coupling Step	18
2.2.5	Go Step	19
2.2.6	Running the Computation	22
2.2.7	Post-Processing	23
2.3	Code-Specific MpCCI Commands	24
2.4	Code Adapter Reference	25
2.4.1	Patched Input File	25
2.4.2	SIMULIA's Co-Simulation Engine (CSE)	25
2.4.3	Update Code Adapter Installation	26
2.5	Co-Simulation Restart	28
2.6	Known limitations	29
3	ANSYS	30
3.1	Quick Information	30
3.1.1	Supported Coupling Schemes	30
3.1.2	Supported Platforms and Versions	30
3.1.3	References	30
3.1.4	Adapter Description	30
3.1.5	Prerequisites for a Coupled Simulation	30
3.1.6	Supported ANSYS product variable	31
3.2	Coupling Process	33
3.2.1	Model Preparation	33
3.2.2	APDL Script	36
3.2.3	Models Step	40
3.2.4	Coupling Step	40
3.2.5	Go Step	43
3.2.6	Running the Computation	44
3.3	Code-Specific MpCCI Commands	45
3.4	Code Adapter Reference	47
3.5	Frequently Asked Questions	48
4	ANSYS Icepak	50
4.1	Quick Information	50
4.1.1	Supported Platforms and Versions	50
4.1.2	Supported Quantities	50
4.2	Code-Specific MpCCI Commands	51
5	FINE/Open	53
5.1	Quick Information	53
5.1.1	Supported Coupling Schemes	53
5.1.2	Supported Platforms and Versions	53
5.1.3	References	53
5.1.4	Adapter Description	53
5.1.5	Prerequisites for a Coupled Simulation	53
5.2	Coupling Process	53
5.2.1	Model Preparation	53
5.2.2	Models Step	54
5.2.3	Coupling Step	54
5.2.4	Go Step	55
5.2.5	Running the Computation	56

5.2.6	Post-Processing	58
5.3	Code-Specific MpCCI Commands	58
5.4	Code Adapter Reference	59
5.5	Limitations	59
6	FINE/Turbo	60
6.1	Quick Information	60
6.1.1	Supported Coupling Schemes	60
6.1.2	Supported Platforms and Versions	60
6.1.3	References	60
6.1.4	Adapter Description	60
6.1.5	Prerequisites for a Coupled Simulation	60
6.2	Coupling Process	60
6.2.1	Model Preparation	60
6.2.2	Models Step	62
6.2.3	Coupling Step	62
6.2.4	Go Step	63
6.2.5	Running the Computation	64
6.2.6	Post-Processing	65
6.3	Code-Specific MpCCI Commands	65
6.4	Trouble shooting, open issues and known bugs	67
7	Flowmaster	68
7.1	Quick Information	68
7.1.1	Supported Coupling Schemes	68
7.1.2	Supported Platforms and Versions	68
7.1.3	References	68
7.1.4	Adapter Description	68
7.1.5	Prerequisites for a Coupled Simulation	68
7.2	Coupling Process	69
7.2.1	Model Preparation	69
7.2.2	Models Step	71
7.2.3	Coupling Step	71
7.2.4	Go Step	72
7.3	Code-Specific MpCCI Commands	73
7.4	Code Adapter Description	74
7.5	Trouble Shooting, Open Issues and Known Bugs	75
8	FLUENT	76
8.1	Quick Information	76
8.1.1	Supported Coupling Schemes	76
8.1.2	Supported Platforms and Versions	76
8.1.3	References	76
8.1.4	Adapter Description	76
8.1.5	Prerequisites for a Coupled Simulation	77
8.2	Coupling Process	77
8.2.1	Model Preparation	77
8.2.2	Models Step	79
8.2.3	Coupling Step	79
8.2.4	Go Step	81
8.2.5	Running the Computation	82
8.3	Code-Specific MpCCI Commands	88

8.4	Code Adapter Reference	90
8.4.1	The MpCCI UDF Library	90
8.4.2	UDF-Hooks	90
8.5	Trouble shooting, open issues and known bugs	93
8.6	Frequently Asked Questions	94
9	JMAG	95
9.1	Quick Information	95
9.1.1	Supported Coupling Schemes	95
9.1.2	Supported Platforms and Versions	95
9.1.3	References	95
9.1.4	Adapter Description	95
9.1.5	Prerequisites for a Coupled Simulation	95
9.1.6	Supported JMAG Modules	95
9.2	Coupling Process	96
9.2.1	Model Preparation	96
9.2.2	Models Step	99
9.2.3	Coupling Step	99
9.2.4	Go Step	100
9.3	Code-Specific MpCCI Commands	100
9.4	Code Adapter Description	101
10	MATLAB	102
10.1	Quick Information	102
10.1.1	Supported Coupling Schemes	102
10.1.2	Supported Platforms and Versions	102
10.1.3	References	102
10.1.4	Adapter Description	102
10.1.5	Prerequisites for a Coupled Simulation	102
10.1.6	Supported MATLAB Modules	102
10.2	Coupling Process	103
10.2.1	Model Preparation	103
10.2.2	MpCCI MEX Function	105
10.2.3	Models Step	110
10.2.4	Coupling Step	110
10.2.5	Go Step	113
10.3	Code-Specific MpCCI Commands	113
10.4	Code Adapter Description	114
11	MSC Adams	115
11.1	Quick Information	115
11.1.1	Supported Coupling Schemes	115
11.1.2	Supported Platforms and Versions	115
11.1.3	References	115
11.1.4	Adapter Description	115
11.1.5	Prerequisites for a Coupled Simulation	115
11.2	Coupling Process	116
11.2.1	Model Preparation	116
11.2.2	Dynamic or Kinematic Simulation	116
11.2.3	Static Simulation	117
11.2.4	Multiple Statements	117
11.2.5	Iterative Coupling	117

11.2.6	MSC Adams Template Products	117
11.2.7	Models Step	118
11.2.8	Coupling Step	119
11.2.9	Go Step	121
11.2.10	Running the Computation	123
11.2.11	Post-Processing	124
11.3	Code-Specific MpCCI Commands	125
11.4	Code Adapter Reference	126
11.4.1	Patched Input File	126
12	MSC Marc	128
12.1	Quick Information	128
12.1.1	Supported Coupling Schemes	128
12.1.2	Supported Platforms and Versions	128
12.1.3	References	128
12.1.4	Adapter Description	128
12.1.5	Prerequisites for a Coupled Simulation	128
12.2	Coupling Process	130
12.2.1	Model Preparation	130
12.2.2	Models Step	130
12.2.3	Coupling Step	131
12.2.4	Go Step	131
12.2.5	Running the Computation	132
12.2.6	Post-Processing	133
12.3	Code-Specific MpCCI Commands	134
12.4	Trouble Shooting, Open Issues and Known Bugs	135
13	MSC NASTRAN	136
13.1	Quick Information	136
13.1.1	Supported Coupling Schemes	136
13.1.2	Supported Platforms and Versions	136
13.1.3	Adapter Description	136
13.1.4	Prerequisites for a Coupled Simulation	136
13.2	Coupling Process	137
13.2.1	Model Preparation	137
13.2.2	Models Step	138
13.2.3	Coupling Step	139
13.2.4	Go Step	139
13.2.5	Running the Computation	140
13.2.6	Post-Processing	141
13.3	Code-Specific MpCCI Commands	142
13.4	Code Adapter Reference	143
13.5	Trouble Shooting, Open Issues and Known Bugs	144
14	OpenFOAM	145
14.1	Quick Information	145
14.1.1	Supported Coupling Schemes	145
14.1.2	Supported Platforms and Versions	145
14.1.3	References	145
14.1.4	Adapter Description	145
14.1.5	Prerequisites for a Coupled Simulation	145

14.2	Coupling Process	146
14.2.1	Model Preparation	146
14.2.2	Models Step	148
14.2.3	Coupling Step	149
14.2.4	Go Step	150
14.2.5	Running the Computation	151
14.2.6	Post-Processing	153
14.3	Grid Morphing	153
14.3.1	MpCCI Grid Morpher	153
14.3.2	OpenFOAM Grid Morpher	153
14.4	Code-Specific MpCCI Commands	156
14.5	Code Adapter Reference	158
15	RadTherm/TAITherm	159
15.1	Quick Information	159
15.1.1	Supported Coupling Schemes	159
15.1.2	Supported Platforms and Versions	159
15.1.3	References	159
15.1.4	Adapter Description	159
15.1.5	Prerequisites for a Coupled Simulation	159
15.2	Coupling Process	159
15.2.1	Model Preparation	160
15.2.2	Models Step	160
15.2.3	Coupling Step	161
15.2.4	Go Step	161
15.2.5	Checking the Computation	167
15.2.6	Running the Computation	167
15.2.7	Post-Processing	170
15.3	Code-Specific MpCCI Commands	170
15.4	Code Adapter Reference	171
15.4.1	Quantity handling	171
16	SIMPACK	172
16.1	Quick Information	172
16.1.1	Supported Coupling Schemes	172
16.1.2	Supported Platforms and Versions	172
16.1.3	References	172
16.1.4	Adapter Description	172
16.1.5	Prerequisites for a Coupled Simulation	172
16.2	Coupling Process	173
16.2.1	Model Preparation	173
16.2.2	Simulation	173
16.2.3	Models Step	173
16.2.4	Coupling Step	175
16.2.5	Go Step	176
16.2.6	Running the Computation	177
16.2.7	Post-Processing	177
16.3	Code-Specific MpCCI Commands	178
17	STAR-CCM+	179
17.1	Quick Information	179
17.1.1	Supported Coupling Schemes	179

17.1.2	Supported Platforms and Versions	179
17.1.3	References	179
17.1.4	Adapter Description	179
17.1.5	Prerequisites for a Coupled Simulation	180
17.2	Coupling Process	180
17.2.1	Model Preparation	180
17.2.2	Models Step	181
17.2.3	Coupling Step	181
17.2.4	Go Step	182
17.2.5	Running the Computation	184
17.2.6	Post-Processing	187
17.3	Code-Specific MpCCI Commands	187
17.4	Grid Morphing	188
17.5	Code Adapter Reference	188
17.5.1	Java Macro Script	188
17.6	Trouble Shooting, Open Issues and Known Bugs	202
18	STAR-CD	203
18.1	Quick Information	203
18.1.1	Supported Coupling Schemes	203
18.1.2	Supported Platforms and Versions	203
18.1.3	References	204
18.1.4	Adapter Description	204
18.1.5	Prerequisites for a Coupled Simulation	204
18.2	Coupling Process	204
18.2.1	Model Preparation	204
18.2.2	Models Step	206
18.2.3	Coupling Step	207
18.2.4	Go Step	209
18.2.5	Running the Computation	210
18.2.6	Post-Processing	211
18.3	Code-Specific MpCCI Commands	212
18.4	Grid Morphing	213
18.4.1	MpCCI Grid Morpher	214
18.4.2	Restart with MpCCI Grid Morpher	215
18.4.3	pro-STAR Grid Morpher	216
18.5	Code Adapter Reference	219
18.5.1	STAR-CD 4.x	219
18.5.2	Automatic Model Preparation STAR-CD 4.x	219
18.6	Trouble Shooting, Open Issues and Known Bugs	221
VII	Tutorial	1
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	Coupling Step	17
2.3.3	Monitors Step	17
2.3.4	Edit Step	18
2.3.5	Go Step	18
2.4	Running the Computation	20
2.5	Discussion of Results	20
3	Driven Cavity	22
3.1	Problem Description	22
3.2	Model Preparation	22
3.2.1	Fluid Model	22
3.2.2	Solid Model	23
3.3	Setting Up the Coupled Simulation with MpCCI GUI	23
3.3.1	Models Step	23
3.3.2	Coupling Step	24
3.3.3	Monitors Step	25
3.3.4	Edit Step	26
3.3.5	Go Step	26
3.4	Running the Computation	27
3.5	Discussion of Results	27
4	Elastic Flap in a Duct	30
4.1	Problem Description	30
4.2	Model Preparation	30
4.2.1	Solid Model	30
4.2.2	Fluid Model	32
4.3	Setting Up the Coupled Simulation with MpCCI GUI	33
4.3.1	Models Step	33
4.3.2	Coupling Step	36
4.3.3	Monitors Step	38
4.3.4	Edit Step	38
4.3.5	Go Step	38
4.4	Running the Computation	41
4.4.1	Starting the Simulation	41
4.4.2	End of the Simulation	42
4.5	Discussion of Results	42
5	Elastic Flap in Water	44
5.1	Problem Description	44
5.2	Model Preparation	44
5.2.1	Solid Model	44
5.2.2	Fluid Model	45
5.3	Setting Up the Coupled Simulation with MpCCI GUI	46
5.3.1	Models Step	46
5.3.2	Coupling Step	48
5.3.3	Monitors Step	49
5.3.4	Edit Step	49
5.3.5	Go Step	49
5.4	Running the Computation	51
5.5	Discussion of Results	52

6	Exhaust Manifold	54
6.1	Problem Description	54
6.2	Model Preparation	54
6.2.1	Solid Model	55
6.2.2	Fluid Model	55
6.2.3	Uncoupled Flow Simulation	57
6.2.4	Prepare Models for Coupled Simulation	57
6.3	Setting Up the Coupled Simulation with MpCCI GUI	58
6.3.1	Models Step	58
6.3.2	Coupling Step	60
6.3.3	Monitors Step	61
6.3.4	Edit Step	61
6.3.5	Go Step	61
6.4	Running the Computation	63
6.5	Post-processing	63
7	Busbar System	65
7.1	Problem Description	65
7.2	Model Preparation	66
7.2.1	Fluid Model	66
7.2.2	Electromagnetic Model	66
7.3	Setting Up the Coupled Simulation with MpCCI GUI	67
7.3.1	Models Step	67
7.3.2	Coupling Step	68
7.3.3	Monitors Step	70
7.3.4	Edit Step	70
7.3.5	Go Step	71
7.4	Running the Computation	72
7.5	Discussion of Results	72
8	Three Phase Transformer	75
8.1	Problem Description	75
8.2	Model Preparation	76
8.2.1	Fluid Model	76
8.2.2	Electromagnetic Model	76
8.3	Setting Up the Coupled Simulation with MpCCI GUI	78
8.3.1	Models Step	79
8.3.2	Coupling Step	79
8.3.3	Monitors Step	80
8.3.4	Edit Step	81
8.3.5	Go Step	81
8.4	Running the Computation	82
8.5	Discussion of Results	82
9	Pipe Nozzle	85
9.1	Problem Description	85
9.2	Model Preparation	85
9.2.1	Fluid Model	86
9.2.2	Solid Model	87
9.3	Setting Up the Coupled Simulation with MpCCI GUI	87
9.3.1	Models Step	88
9.3.2	Coupling Step	89

9.3.3	Monitors Step	90
9.3.4	Edit Step	90
9.3.5	Go Step	90
9.4	Running the Computation	91
9.5	Discussion of Results	92
10	Cube in a Duct Heater	93
10.1	Problem Description	93
10.2	Model Preparation	93
10.2.1	Radiation Model	93
10.2.2	Fluid Model	94
10.3	Setting Up the Coupled Simulation with MpCCI GUI	95
10.3.1	Models Step	96
10.3.2	Coupling Step	97
10.3.3	Monitors Step	98
10.3.4	Edit Step	98
10.3.5	Go Step	98
10.4	Running the Computation	100
10.4.1	Starting the Simulation	100
10.5	Discussion of Results	101
11	Y-Junction	105
11.1	Problem Description	105
11.2	Model Preparation	105
11.2.1	Network Model	106
11.2.2	Fluid Model	109
11.3	Setting Up the Coupled Simulation with MpCCI GUI	111
11.3.1	Models Step	111
11.3.2	Coupling Step	112
11.3.3	Monitors Step	115
11.3.4	Edit Step	115
11.3.5	Go Step	115
11.4	Running the Computation	117
11.4.1	Starting the Simulation	117
11.5	Discussion of Results	118
12	Spring Mass System	122
12.1	Problem Description	122
12.2	Model Preparation	122
12.2.1	Model Description	122
12.2.2	Model A	123
12.2.3	Model B	124
12.3	Setting Up the Coupled Simulation with MpCCI GUI	125
12.3.1	Models Step	125
12.3.2	Coupling Step	129
12.3.3	Monitors Step	132
12.3.4	Edit Step	132
12.3.5	Go Step	132
12.4	Running the Computation	138
12.4.1	Starting the Simulation	138
12.5	Discussion of Results	138
12.5.1	Iterative Coupling compared to Explicit Coupling	138

12.5.2	Non-matching Time Step Sizes	139
13	Periodic Rotating Fan Model	141
13.1	Problem Description	141
13.2	Model Preparation	142
13.2.1	Fluid Model	142
13.2.2	Solid Model	143
13.3	Setting Up the Coupled Simulation with MpCCI GUI	144
13.3.1	Models Step	144
13.3.2	Coupling Step	145
13.3.3	Monitors Step	146
13.3.4	Edit Step	146
13.3.5	Go Step	146
13.4	Running the Computation	147
13.5	Discussion of Results	148
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	Go Step: <GoMenuEntries>	26
2.4.7	Environments for Scanner, Checker, Starter, Stopper and Killer	28
2.4.8	General MpCCI GUI Elements	28
2.4.9	Testing gui.xcf	34
2.5	Perl Scripts	35
2.5.1	Using Information from gui.xcf in Scripts	35
2.5.2	Scanner.pm	35
2.5.3	Checker.pm	36
2.5.4	Starter.pm	37
2.5.5	Stopper.pm	37
2.5.6	Killer.pm	38
2.5.7	Info.pm	38
2.5.8	Subcmd.pm	38
2.5.9	Testing the Perl Scripts	39
2.6	MpCCI Adapter Implementation	40
2.6.1	How to Initialize the Code?	40

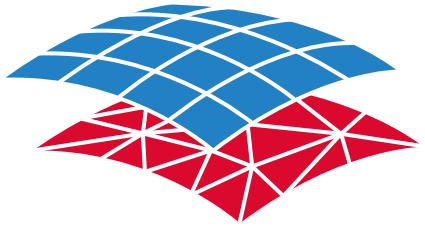
2.6.2	How to Define the Mesh?	41
2.6.3	How to Deal with Angular Coordinates?	42
2.6.4	How to Transfer Data?	43
2.6.5	How to Terminate the Coupling?	43
2.6.6	How to Notify a Remeshing?	43
2.7	MpCCI Coupling Manager Functions	44
2.7.1	Definition of Output Functions: <code>umpcci_msg_functs</code>	45
2.7.2	Definition of Output Prefix: <code>umpcci_msg_prefix</code>	46
2.7.3	Get Transfer Information: <code>ampcci_tinfo_init</code>	47
2.7.4	Connect and Initialize an MpCCI Server: <code>mpcci_init</code>	48
2.7.5	Configure the Code Adapter: <code>ampcci_config</code>	49
2.7.6	Definition of Part: <code>smpcci_defp</code>	50
2.7.7	Delete a Part: <code>smpcci_delp</code>	52
2.7.8	Definition of Nodes: <code>smpcci_pnod</code>	53
2.7.9	Definition of Elements: <code>smpcci_pels</code>	54
2.7.10	Definition of the Moving Reference Frame: <code>smpcci_pmot</code>	56
2.7.11	Definition of the Baffle thickness: <code>smpcci_pshf</code>	57
2.7.12	Data Exchange: <code>ampcci_transfer</code>	58
2.7.13	Notifying the Remeshing: <code>ampcci_remesh</code>	60
2.7.14	End of Coupled Simulation: <code>mpcci_quit</code> and <code>mpcci_stop</code>	61
2.8	MpCCI Driver Functions	62
2.8.1	Description Values	65
2.8.2	Driver Methods Called before/after some Action	66
2.8.3	Driver Mesh Definition Methods	68
2.8.4	Driver Data Exchange Methods	70
2.9	Data Structures and Predefined Macros	71
2.9.1	Supported Element Types	71
2.9.2	Coordinates System Definition	80
2.9.3	Mesh Dimension Definition	80
2.9.4	Moving Reference Frame Definition	80
2.9.5	Remesh Flag Information	81
2.9.6	Transfer Information: <code>MPCCI_TINFO</code>	81
2.9.7	Code Specific Information: <code>MPCCI_CINFO</code>	87
2.9.8	Coupling Components	89
2.9.9	Quantities	90
2.9.10	Loop Functions	91
2.9.11	Memory Management	93

IX	How To	1
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
X	MpCCI FSIMapper	1
1	MpCCI FSIMapper Overview	6
2	MpCCI FSIMapper Installation	8
2.1	Part of MpCCI Installation	8
2.2	Standalone Version	8
2.2.1	Local Microsoft Windows Installation	8
2.2.2	Linux or Multi-Platform Installation	8
2.2.3	Perl	9
2.2.4	License	10
2.2.5	Configure a License Manager as UNIX service	11
2.2.6	Configure a License Manager as Windows service	12
3	MpCCI FSIMapper Command	14
4	MpCCI FSIMapper GUI	15
4.1	Starting the MpCCI FSIMapper	15
4.2	The “What to map” Panel	16
4.2.1	Selection of Cases, Parts and Quantities	16
4.2.2	Geometry Compare	18
4.2.3	Quantity Identification for CSM Solver Output	18
4.2.4	Execute a Mapping Process	19
4.3	The “Transformation” Panel	21
4.3.1	Truncation of transient result data	25
4.4	The “How to map” Panel	25
4.4.1	Mapping Algorithms and Neighborhood Parameters	25
4.4.2	Orphan Filling	26
4.4.3	Quantity Location	26
4.4.4	Saving Mapping Configurations	27
4.5	The “Result” Panel	27
4.5.1	Average over Rotation Axis	27
4.5.2	Apply Fourier Transformation	28
4.5.3	MSC NASTRAN Export Options	29
4.6	The “Preferences” Panel	30
4.7	The “Harmonic Wizard” Panel	30
4.8	The “Log” Panel	31
5	Codes and Formats Information	33
5.1	EnSight Gold	33
5.1.1	Supported Quantities	33
5.2	FLUENT	33
5.2.1	Supported Quantities	33
5.2.2	Exporting wallfuncHTC to UDM0 in Data File	33

5.3	FINE/Turbo	34
5.3.1	Supported Quantities	34
5.4	CFX	34
5.4.1	Supported Quantities	35
5.5	FloTHERM	35
5.5.1	Supported Quantities	35
5.6	FloEFD	35
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	Abaqus	36
5.8.1	Limitations	36
5.8.2	Model Preparation	36
5.8.3	Include boundary conditions	37
5.8.4	Elements	37
5.8.5	Supported Quantities	37
5.9	ANSYS	38
5.9.1	Requirements	38
5.9.2	Model Preparation	38
5.9.3	Supported Quantities	40
5.9.4	ANSYS Scanner and Converter	41
5.10	MSC NASTRAN	41
5.10.1	Limitations	41
5.10.2	Include boundary conditions	41
5.10.3	Elements	42
5.10.4	Supported Quantities	43
5.11	LS-Dyna	43
5.11.1	Limitations	43
5.11.2	Elements	43
5.11.3	Supported Quantities	43
5.11.4	Include boundary conditions	43
6	Batch Usage of the MpCCI FSIMapper	44
6.1	File Scanners	44
6.1.1	FLUENT	44
6.1.2	MentorGraphics	46
6.1.3	FloEFD	46
6.1.4	FloTHERM	47
6.1.5	ANSYS	47
6.1.6	Abaqus	48
6.1.7	EnSight Gold Case	49
6.2	Comparing Geometries	52
6.3	Mapping Quantities	53
6.4	Files Written by the MpCCI FSIMapper	53
6.5	Configuration File	53
6.6	Example for a Configuration File	56
7	Numerical Methods	59
7.1	Mapping Algorithms	59
7.2	Parameters for the Mapping	59

7.3	Orphan Filling	60
8	Tutorial	62
8.1	Mapping of Electromagnetic Forces for Transient and Harmonic analyses	62
8.1.1	Problem Description	62
8.1.2	Source Result File	63
8.1.3	Target Mesh File	66
8.1.4	Mapping	66
8.1.5	Target Simulation	72
8.2	Mapping of Harmonic Pressure Excitations in Turbomachinery	73
8.2.1	Using the Harmonic Balance Method of STAR-CCM+	73
8.2.2	Using the Nonlinear Harmonic Method of FINE/Turbo	85
XI	Appendix	1
Quantity Reference		4
Literature		39
Glossary		40
Keyword Index		43



MpCCI
CouplingEnvironment

Part II

Release Notes

Version 4.5.0

MpCCI 4.5.0-1 Documentation
Part II Release Notes
PDF version
March 30, 2017

MpCCI is a registered trademark of Fraunhofer SCAI
www.mpcci.de



Fraunhofer Institute for Algorithms and Scientific Computing SCAI
Schloss Birlinghoven, 53754 Sankt Augustin, Germany

Abaqus and SIMULIA are trademarks or registered trademarks of Dassault Systèmes
ANSYS, FLUENT and ANSYS Icepak are trademarks or registered trademarks of Ansys, Inc.
Elmer is an open source software developed by CSC
FINE/Open and FINE/Turbo are trademarks of NUMECA International
Flowmaster is a registered trademark of Flowmaster Group NV
JMAG is a registered trademark of The JSOL Corporation
MATLAB is a registered trademark of The MathWorks, Inc.
MSC Adams, MSC Marc, MD NASTRAN and MSC NASTRAN are trademarks or registered trademarks of
MSC.Software Corporation
OpenFOAM is a registered trademark of OpenCFD Ltd.
PERMAS is a registered trademark of Intes GmbH
RadTherm, TAITherm is a registered trademark of ThermoAnalytics Inc.
SIMPACK is a registered trademark of SIMPACK AG.
STAR-CCM+ and STAR-CD are registered trademarks of CD adapco Group

ActivePerl has a Community License Copyright of Active State Corp.
FlexNet Publisher is a registered trademark of Flexera Software.
Java is a registered trademark of Oracle and/or its affiliates.
Linux is a registered trademark of Linus Torvalds
Mac OS X is a registered trademark of Apple Inc.
OpenSSH has a copyright by Tatu Ylonen, Espoo, Finland
Perl has a copyright by Larry Wall and others
UNIX is a registered trademark of The Open Group
Windows, Windows XP, Windows Vista and Windows 7 are registered trademarks of Microsoft Corp.

II Release Notes – Contents

1	Introduction MpCCI 4.5	4
2	Changes and New Features in MpCCI 4.5.0-1	5
2.1	MpCCI Licensing	5
2.2	MpCCI Platforms	5
2.3	New Features	5
2.3.1	MpCCI License Manager	5
2.3.2	MpCCI GUI	5
2.3.3	MpCCI CouplingEnvironment	5
2.3.4	MpCCI Client	6
2.3.5	MpCCI FSIMapper	6
2.4	Further Enhancements of Existing Features	6
2.4.1	MpCCI GUI	6
2.4.2	MpCCI CouplingEnvironment	7
2.4.3	MpCCI Client	7
2.4.4	MpCCI FSIMapper	7
2.4.5	MpCCI Grid Morpher	7
2.4.6	MpCCI Batch	7
2.4.7	MpCCI Visualizer and MpCCI Monitor	8
2.4.8	MpCCI Tutorial	8
2.4.9	Code Specific Changes	8
2.5	Known Bugs and Limitations	10
3	Changes and New Features in the Earlier Releases	11
3.1	MpCCI 4.4.2-1	11
3.1.1	Further Enhancements of Existing Features	11
3.2	MpCCI 4.4.1-1	13
3.2.1	Further Enhancements of Existing Features	13
3.3	MpCCI 4.4.0-1	15
3.3.1	New Features	15
3.3.2	Further Enhancements of Existing Features	16
4	Prerequisites for MpCCI Installation	19
5	Supported Platforms in MpCCI 4.5	20
5.1	Platforms Supported by the MpCCI 4.5 Server	20
5.2	Codes Supported by MpCCI 4.5 on Different Platforms	21

1 Introduction MpCCI 4.5

The release notes relate to MpCCI 4.5.

MpCCI 4.5 represents a new release change over MpCCI 4.4. Please read the changes and features section
[▷ 2 Changes and New Features in MpCCI 4.5.0-1 ↳](#) for more details.

MpCCI 4.5 is available as an upgrade from an existing MpCCI 4.4 installation.

2 Changes and New Features in MpCCI 4.5.0-1

2.1 MpCCI Licensing

- New licensing software: for MpCCI 4.5.0 you will need the FlexNet Publisher 11.14 licensing software.

2.2 MpCCI Platforms

- Windows and Linux 32 bit platform is no longer supported.
- Linux 64 bit with glibc 2.3 is no longer supported.

2.3 New Features

2.3.1 MpCCI License Manager

- MpCCI license version has been updated to FlexNet Publisher 11.14 licensing software. An update of the license server is required.
- MpCCI license start command enhanced by
 - listing the used license file for setting up the license server.
 - listing the current license features if the license server correctly starts up. Otherwise it points the user to the log file to check the issues.

2.3.2 MpCCI GUI

- Support for automatic region building (see [▷ V-4.5.5 Automatic Generation of Regions by Rules](#)).
- Region specific definition of neighborhood search properties ([▷ V-4.5.6 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-2.5.2 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-4.5.7.4 Applying Operators to 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-4.5.9 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](#)).

2.3.3 MpCCI CouplingEnvironment

- A new relaxation approach by Quasi-Newton ([▷ V-3.3.3.1 Quasi-Newton Methods](#)) 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-4.5.7.4 Applying Operators to Mesh Based Components ◁](#)).
- Allow definition of neighborhood multiplicity and normal distance parameters locally per region ([▷ V-4.5.6 Applying Region Properties ◁](#)).
- Allow definition of the relaxation method for a quantity locally per region.

2.3.4 MpCCI Client

- Subcycling definition with a table function (see [▷ V-3.4.6.1 Subcycling Setting in the MpCCI GUI ◁](#)).
- Client library support for Visual Studio 2013, 2015.

2.3.5 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-4.3.1 Truncation of transient result data ◁](#)
- Introduction of “Harmonic Wizard” tab to map a set of harmonic turbomachinery CFD results at once to a CSM model. [▷ 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). [▷ X-5.10.1 Reading ◁](#)
- LS-Dyna added as target structural code format in transient force excitation mapping applications where a Fourier transformation is applied. [▷ X-5.11 LS-Dyna ◁](#)

2.4 Further Enhancements of Existing Features

2.4.1 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 [▷ IV-2.5 Coupling 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 ([▷ V-4.5.7.3 Orphans Settings for Mesh Based Components ◁](#)).

2.4.2 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 ([▷ V-3.3.3.2 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.

2.4.3 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.

2.4.4 MpCCI FSIMapper

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

2.4.5 MpCCI Grid Morpher

- Fix for supporting models without fixed boundary conditions.

2.4.6 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 ".baschrc" 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.

2.4.7 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.

2.4.8 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 [▷ VII-5 Elastic Flap in Water ◁](#).

2.4.9 Code Specific Changes

2.4.9.1 Abaqus

- Support of Abaqus 2016, 2017.
- Fixed iterative coupling to respect the maximum number of coupled iterations.
- Improved support for coupling with MBS code.

2.4.9.2 ANSYS

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

2.4.9.3 FINE/Hexa

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

2.4.9.4 FINE/Open

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

2.4.9.5 FINE/Turbo

- Support of FINE/Turbo 10.2, 11.1.

2.4.9.6 FLUX

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

2.4.9.7 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.

2.4.9.8 ANSYS Icepak

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

2.4.9.9 JMAG

- Support of JMAG 15.0, 15.1.

2.4.9.10 MSC Adams

- Support of MSC Adams 2015.1, 2016, 2017.
- Support MSC Adams Car application.
- Support static to dynamic simulation.

2.4.9.11 FMI

- New adapter for the Fraunhofer EAS Master, a simulation platform for FMI coupling (prototype on demand).

2.4.9.12 MSC Marc

- Support of MSC Marc 2015, 2016.

2.4.9.13 MSC NASTRAN

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

2.4.9.14 OpenFOAM

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

2.4.9.15 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.

2.4.9.16 STAR-CCM+

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

2.5 Known Bugs and Limitations

3 Changes and New Features in the Earlier Releases

3.1 MpCCI 4.4.2-1

3.1.1 Further Enhancements of Existing Features

3.1.1.1 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.

3.1.1.2 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.

3.1.1.3 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

- Support of SIMPACK 9.8.1.

STAR-CCM+

- Support of STAR-CCM+ 10.04.009.

3.2 MpCCI 4.4.1-1

3.2.1 Further Enhancements of Existing Features

3.2.1.1 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. [▷ 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.

3.2.1.2 MpCCI FSIMapper

- New option to select parts by using a string matching definition, cf. [▷ 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.

3.2.1.3 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 ([▷ VI-6.4 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-14.2.5.2 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.

3.3 MpCCI 4.4.0-1

3.3.1 New Features

3.3.1.1 MpCCI FSIMapper

- MpCCI FSIMapper is available as standalone installation and can be started with `fsimapp` (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.

3.3.1.2 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-3.3.3.3 Quantity Transformation for Cyclic Symmetric Meshes](#).
 - MpCCI GUI supports definition of periodicity, cf. [▷ V-4.5.2.3 Periodic Components](#).
 - Periodic model can be shown as full model in MpCCI Visualizer and written as .ccvx, cf. [▷ V-4.7 Edit Step](#).
 - A new tutorial [▷ VII-13 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-3.3.3.1 Quantity Relaxation](#).
 - Available for steady state or iteratively transient coupled simulations.
 - The used relaxation factor can be plotted in the MpCCI Monitor, cf. [▷ V-4.7 Edit 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-4.8.1 Configuring the MpCCI Coupling Server](#).

3.3.2 Further Enhancements of Existing Features

3.3.2.1 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-4.7.2 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.

3.3.2.2 Code Specific Changes

3.3.2.3 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 RelWall-Force).
 - 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.

3.3.2.4 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.

3.3.2.5 FINE/Open

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

3.3.2.6 Flowmaster

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

3.3.2.7 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-8.2.5.2 The MpCCI Run FSI ▲](#).
- 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.

3.3.2.8 JMAG

- Support scanning of JMAG model file with white spaces.

3.3.2.9 MSC Adams

- Support of MSC Adams 2014.

3.3.2.10 MSC Marc

- Support of MSC Marc 2014.

3.3.2.11 MD NASTRAN

- Support of MD NASTRAN 2014.

3.3.2.12 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-14.2.5.2 External 3rd party MPI◁.

3.3.2.13 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.

3.3.2.14 SIMPACK

- Support of SIMPACK 9.6, 9.7.

3.3.2.15 STAR-CCM+

- Support of STAR-CCM+ 9.04.009.
- Fix support of models with multiple region definitions.

3.3.2.16 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 4.5 ◁](#) 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 4.5

Platform lists for supported simulation codes are given in the [Codes Manual](#).

5.1 Platforms Supported by the MpCCI 4.5 Server

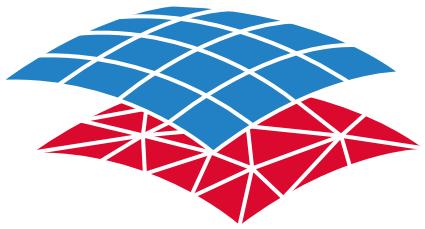
Platform	Bits	MpCCI arch.	Support
Linux with glibc 2.3 on AMD64 or EM64T processor	32/64	lnx3_x64	<i>not supported</i>
Linux with glibc 2.4 on AMD64 or EM64T processor	32/64	lnx4_x64	OK
Microsoft Windows 32 bit on AMD/Intel	32	windows_x86	<i>not supported</i>
Microsoft Windows 64 bit on AMD/Intel	32/64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

- ① 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 4.5 on Different Platforms

 Codes can only be supported on platforms they support themselves.

Linux with glibc 2.4 on AMD64 or EM64T processor		(<code>lnx4_x64</code>)
ANSYS	145, 150, 160, 161, 162, 170, 171, 172, 180	
Abaqus	6.14, 2016, 2017	
MSC Adams	2013, 2013.1, 2013.2, 2014, 2015, 2015.1, 2016, 2017	
FINE/Open	41, 42, 43, 51, 52, 61	
FINE/Turbo	91_1, 91_2, 91_3, 101, 102, 111	
FLUENT	14.5.0, 14.5.7, 15.0.0, 15.0.7, 16.0.0, 16.1.0, 16.2.0, 17.0.0, 17.1.0, 17.2.0, 18.0.0	
ANSYS Icepak	14.5.0, 14.5.7, 15.0.0, 15.0.7, 16.0.0, 16.1.0, 16.2.0, 17.0.0, 17.1.0, 17.2.0, 18.0.0	
JMAG	13.0, 13.1, 14.0, 14.1, 15.0, 15.1	
MATLAB	R2013b, R2014b, R2015a, R2016b	
MSC Marc	2013, 2013.1, 2014, 2014.1, 2014.2, 2015, 2016	
MSC NASTRAN	2013.0, 2013.1, 2014.0, 2014.1, 2016.0, 2016.1, 2017.0	
OpenFOAM	1.7, 2.0, 2.1, 2.2, 2.3, 2.4, 3.0.1, 1606+, 1612+	
RadTherm	11.0.0, 11.1.0, 11.1.1, 11.2.0, 11.3.0, 11.3.2, 12.0.0, 12.0.3, 12.1.0, 12.1.1, 12.2.0	
SIMPACK	9.7, 9.8.1	
STAR-CCM+	9.02, 9.04, 9.06, 10.02, 10.04, 10.06, 11.02, 11.04, 11.06	
STAR-CD	4.16.034, 4.18.044, 4.20.041, 4.22.005	
Microsoft Windows 64 bit on AMD/Intel		(<code>windows_x64</code>)
ANSYS	145, 150, 160, 161, 162, 170, 171, 172, 180	
Abaqus	6.14, 2016, 2017	
MSC Adams	2013, 2013.1, 2013.2, 2014, 2015, 2015.1, 2016, 2017	
FINE/Open	41, 42, 43, 51, 52, 61	
FINE/Turbo	91_1, 91_2, 91_3, 101, 102, 111	
FLUENT	14.5.0, 15.0.0, 15.0.7, 16.0.0, 16.1.0, 16.2.0, 17.0.0, 17.1.0, 17.2.0, 18.0.0	
Flowmaster	7.6, 7.7, 7.8, 8.0, 8.1, 8.2, 9.0, 9.2	
ANSYS Icepak	14.5.0, 15.0.0, 15.0.7, 16.0.0, 16.1.0, 16.2.0, 17.0.0, 17.1.0, 17.2.0, 18.0.0	
JMAG	13.0, 13.1, 14.0, 14.1, 15.0, 15.1	
MATLAB	R2013b, R2014b, R2015a, R2016b	
MSC Marc	2013, 2013.1, 2014, 2014.1, 2014.2, 2015, 2016	
MSC NASTRAN	2013.1, 2014.0, 2014.1, 2016.0, 2016.1, 2017.0	
RadTherm	11.0.0, 11.1.0, 11.1.1, 11.2.0, 11.3.0, 11.3.2, 12.0.0, 12.0.3, 12.1.0, 12.1.1, 12.2.0	
SIMPACK	9.7, 9.8.1	
STAR-CCM+	9.02, 9.04, 9.06, 10.02, 10.04, 10.06, 11.02, 11.04, 11.06	
STAR-CD	4.16.034, 4.18.044, 4.20.041, 4.22.005	



MpCCI
CouplingEnvironment

Part III

Installation Guide

Version 4.5.0

MpCCI 4.5.0-1 Documentation
Part III Installation Guide
PDF version
March 30, 2017

MpCCI is a registered trademark of Fraunhofer SCAI
www.mpcci.de



Fraunhofer Institute for Algorithms and Scientific Computing SCAI
Schloss Birlinghoven, 53754 Sankt Augustin, Germany

Abaqus and SIMULIA are trademarks or registered trademarks of Dassault Systèmes
ANSYS, FLUENT and ANSYS Icepak are trademarks or registered trademarks of Ansys, Inc.
Elmer is an open source software developed by CSC
FINE/Open and FINE/Turbo are trademarks of NUMECA International
Flowmaster is a registered trademark of Flowmaster Group NV
JMAG is a registered trademark of The JSOL Corporation
MATLAB is a registered trademark of The MathWorks, Inc.
MSC Adams, MSC Marc, MD NASTRAN and MSC NASTRAN are trademarks or registered trademarks of
MSC.Software Corporation
OpenFOAM is a registered trademark of OpenCFD Ltd.
PERMAS is a registered trademark of Intes GmbH
RadTherm, TAITherm is a registered trademark of ThermoAnalytics Inc.
SIMPACK is a registered trademark of SIMPACK AG.
STAR-CCM+ and STAR-CD are registered trademarks of CD adapco Group

ActivePerl has a Community License Copyright of Active State Corp.
FlexNet Publisher is a registered trademark of Flexera Software.
Java is a registered trademark of Oracle and/or its affiliates.
Linux is a registered trademark of Linus Torvalds
Mac OS X is a registered trademark of Apple Inc.
OpenSSH has a copyright by Tatu Ylonen, Espoo, Finland
Perl has a copyright by Larry Wall and others
UNIX is a registered trademark of The Open Group
Windows, Windows XP, Windows Vista and Windows 7 are registered trademarks of Microsoft Corp.

III Installation Guide – Contents

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	Configure a License Manager as Linux Service	19
5.2.3	Configure a License Manager as Windows Service	19
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
9.1	Linux	32
9.2	Windows	32
9.2.1	Strawberry Perl for Windows	32
9.2.2	ActivePerl for Windows	32
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.

See also [▷ 3.2 Local Windows Installation with the MSI ◁](#).

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 and navigate to the download area or go directly to <https://download.scai.fraunhofer.de>.
- Enter e-mail address (used as username) and password and press **Sign In**.
- 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 4.5 ◁.

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. At least a Java Runtime Environment 8 (JRE) or higher is required. This corresponds to Java 1.8.0.

A Java JRE 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.

- (!) In fact you should not have OpenSSH installed before the MpCCI installation.

Recommendations

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

- (!) Together with MpCCI some bug fixes for the OpenSSH are installed and the OpenSSH is automatically configured.

OpenSSH for Windows is already Installed

You are familiar with OpenSSH and already worked with OpenSSH.

- (!) Please save your existing "<Home>/.ssh" directory - e.g. rename it '_ssh' - to avoid the loss of your already generated and working RSA-keys.

Uninstall OpenSSH. After MpCCI was successfully installed OpenSSH will be installed on the fly.

After the final OpenSSH installation remove the directory "<Home>/.ssh".

- (!) Do not forget to rename your saved '_ssh' directory back to '.ssh'.

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**

```
set PATH=<MPCCI_HOME>\bin;%PATH%
```

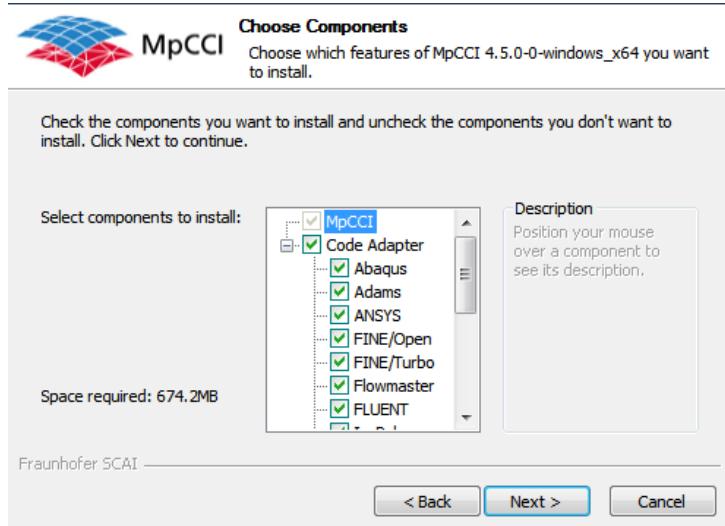
- !** Please keep in mind that a list separator is ':' under Linux and ';' under Windows.

3.2 Local Windows Installation with the MSI

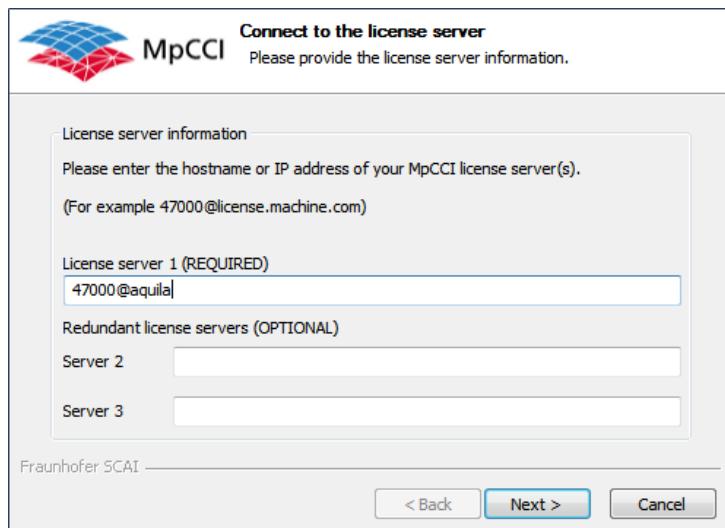
Log on as an administrator or make sure that you have administrative rights. You need administrative rights because of the required modification in the Windows registry.

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

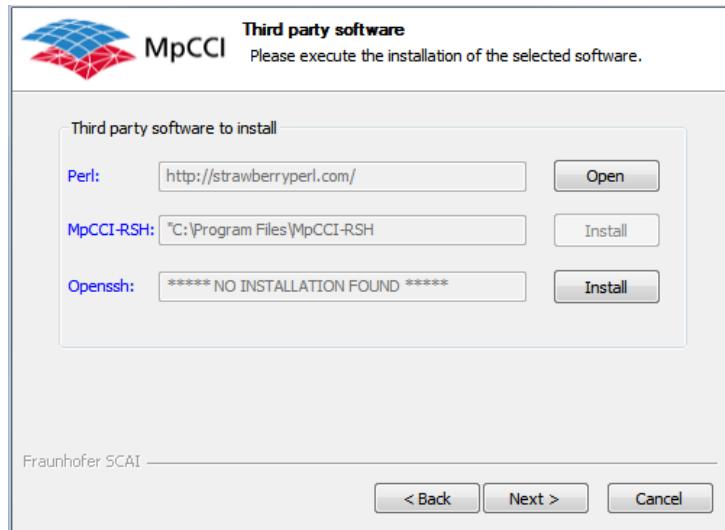
This installation procedure builds a proper MpCCI software environment under Windows.



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.



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.

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 [▷ 9 Installing Perl ◁](#)).

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=%HOMEPATH%\%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 its 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 these 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

Subcommands:	
ANSYS	Tools related to ANSYS.
Abaqus	Tools related to Abaqus.
Adams	Tools related to Adams.
FINEOpen	Tools related to FINEOpen.
FINETurbo	Tools related to FINETurbo.
FLUENT	Tools related to FLUENT.
Flowmaster	Tools related to Flowmaster.
IcePak	Tools related to IcePak.
JMAG	Tools related to JMAG.
MATLAB	Tools related to MATLAB.
MSC.Marc	Tools related to MSC.Marc.
MSC.Nastran	Tools related to MSC.Nastran.
OpenFOAM	Tools related to OpenFOAM.
RadTherm	Tools related to RadTherm.
SIMPACK	Tools related to SIMPACK.
STAR-CCM+	Tools related to STAR-CCM+.
StarCD	Tools related to StarCD.
codename	Tools related to codename.

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 abbreviated form `mpcci <code name> info`.

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 FLEXIm). The FlexNet Publisher license server has to be started on the license server host for which you require a license file.

- (!) For MpCCI 4.5 you will need the current FlexNet Publisher 11.14 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. [▷ 5.2.1 Troubleshooting License Start on Linux ↳](#)).

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 `mpcci license -sysid` or abbreviated `mpcci lic sys`. You should see an output line like

```
SERVER hostname 00087519576d 47000
```

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,

```
mpcci_sessions  
mpcci_clients
```

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

The feature `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 `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 the MpCCI 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 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:

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 "`<MpCCI 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 "`<MpCCI 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:

```
License server: 47000@aquila
Vendor daemon : SVD v11.14.0, status "UP"
```

Feature	Total	Used	Free
-----	-----	-----	-----

mpcci_sessions	2	0	2
mpcci_clients	10	0	10
mpcci_adapter_abaqus	1	0	1
mpcci_adapter_fluent	1	0	1
mpcci_adapter.....	1	0	1
mpcci_adapter.....	1	0	1

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

```
Can't exec license/lnx4_x64/lmgrd  No such file or directory.
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 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/Sxxxx", 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.

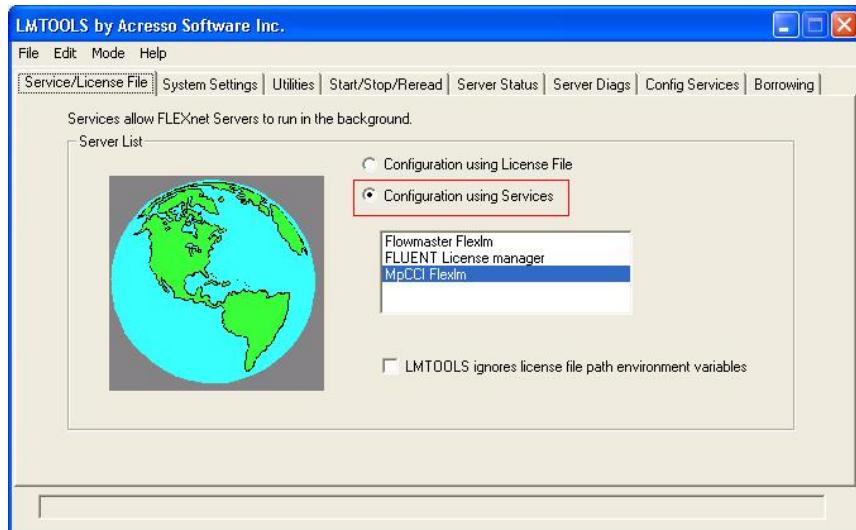
5.2.3 Configure a License Manager as Windows Service

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

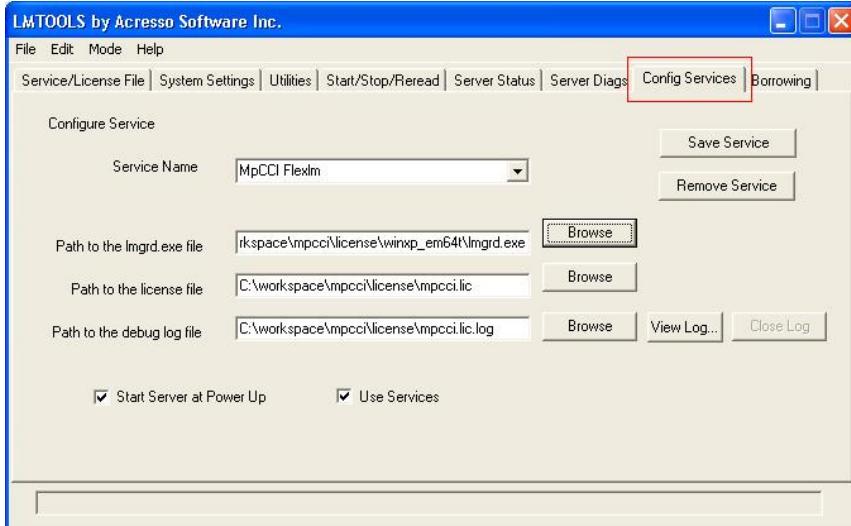
```
"<MpCCI_home>/license/<MpCCI_arch>/lmtools.exe"
```

<MpCCI_arch> corresponds to the output of the command:

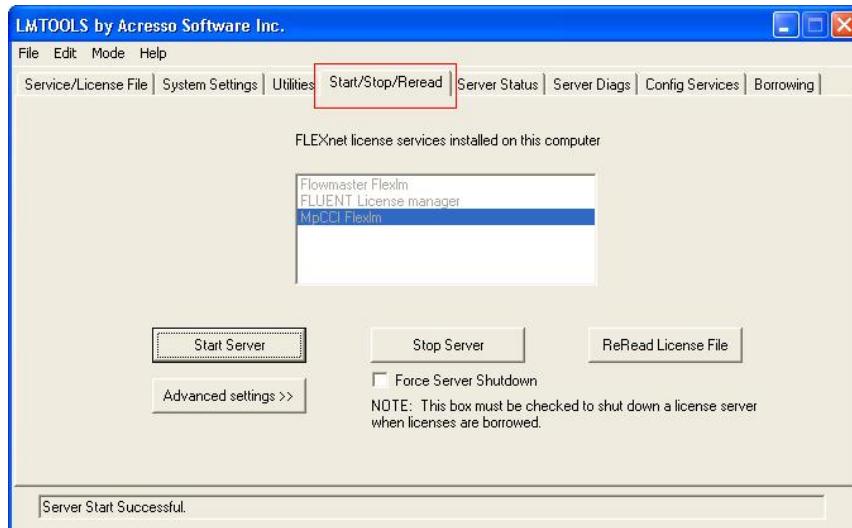
```
mpcci arch
```



- 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 FLEXIm.
- 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@quila
export MPCCI_LICENSE_FILE
```

Korn shell (`ksh`) and bash users

```
export MPCCI_LICENSE_FILE=47000@quila
```

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

```
setenv MPCCI_LICENSE_FILE 47000@quila
```

and under Windows

```
set MPCCI_LICENSE_FILE=47000@quila
```

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@quila;47000@einstei;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 "`.shosts`" 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 [▷ V-3.5.2 Hostlist File ◁](#)).

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 [▷ V-2.3 Environment and Environment Variables ◁](#).

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 [▷ II-5 Supported Platforms in MpCCI 4.5 ◁](#).

Please read [▷ V-2.3.1 MPCCI_ARCH - Architecture Tokens ◁](#) 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.

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

 If no remote access is required, you can define the environment variable `MPCCI_NOREMESH` (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](#))

 If no remote access is required, you can define the environment variable `MPCCI_NOREMESH` (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 address 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 PATH

The `rshd` and `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 `PATH` environment, are merged from the global system `PATH` defined for all users and the users individual `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 `PATH` environment variable contains a list with more than 1920 characters, not just the merge process fails, but also the final `PATH` prepared for the user gets clipped at a length of 1024 characters. In fact the users `PATH` value then contains only the first 1024 characters from the systems `PATH`.

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

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

Another workaround for the bug is the `PATH` file "`%USERPROFILE%\rsh\PATH`" which contains a free formatted list of directory search paths, separated by either the standard separator ";" or a newline character.

```
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\CMak 2.6\bin
```

If the file "`%USERPROFILE%\rsh\PATH`" exists the users `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 "`%userprofile%\bin`" is implemented.

9 Installing Perl

9.1 Linux

If you need to install or upgrade Perl under Linux please either download the source code of Perl 5.8.8 or later 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 5.24.

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://strawberrypperl.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](#) ▲.

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 5.24.

You may follow the link www.activestate.com/Products/ActivePerl to get an overview of the offered ActivePerl distributions.

Download the free ActivePerl community edition from www.activestate.com/activeperl/downloads.

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

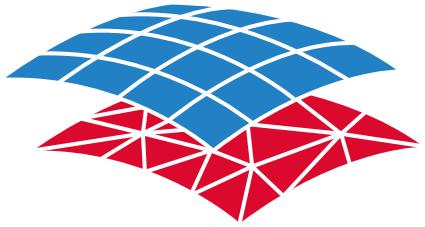
```
ftype perl
```

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
```

MpCCI
CouplingEnvironment

Part IV

Getting Started

Version 4.5.0

MpCCI 4.5.0-1 Documentation
Part IV Getting Started
PDF version
March 30, 2017

MpCCI is a registered trademark of Fraunhofer SCAI
www.mpcci.de



Fraunhofer Institute for Algorithms and Scientific Computing SCAI
Schloss Birlinghoven, 53754 Sankt Augustin, Germany

Abaqus and SIMULIA are trademarks or registered trademarks of Dassault Systèmes
ANSYS, FLUENT and ANSYS Icepak are trademarks or registered trademarks of Ansys, Inc.
Elmer is an open source software developed by CSC
FINE/Open and FINE/Turbo are trademarks of NUMECA International
Flowmaster is a registered trademark of Flowmaster Group NV
JMAG is a registered trademark of The JSOL Corporation
MATLAB is a registered trademark of The MathWorks, Inc.
MSC Adams, MSC Marc, MD NASTRAN and MSC NASTRAN are trademarks or registered trademarks of
MSC.Software Corporation
OpenFOAM is a registered trademark of OpenCFD Ltd.
PERMAS is a registered trademark of Intes GmbH
RadTherm, TAITherm is a registered trademark of ThermoAnalytics Inc.
SIMPACK is a registered trademark of SIMPACK AG.
STAR-CCM+ and STAR-CD are registered trademarks of CD adapco Group

ActivePerl has a Community License Copyright of Active State Corp.
FlexNet Publisher is a registered trademark of Flexera Software.
Java is a registered trademark of Oracle and/or its affiliates.
Linux is a registered trademark of Linus Torvalds
Mac OS X is a registered trademark of Apple Inc.
OpenSSH has a copyright by Tatu Ylonen, Espoo, Finland
Perl has a copyright by Larry Wall and others
UNIX is a registered trademark of The Open Group
Windows, Windows XP, Windows Vista and Windows 7 are registered trademarks of Microsoft Corp.

IV Getting Started – Contents

1 Multiphysics Computation with MpCCI	4
1.1 Multiphysics	4
1.2 Solution of Coupled Problems	4
1.3 Code Coupling with MpCCI	5
2 Setting up a Coupled Simulation	7
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
2.4 Models Step – Choosing Codes and Model Files	9
2.5 Coupling Step – Defining Coupling Regions and Quantities	10
2.5.1 Build Coupling Regions	11
2.5.2 Define Quantities to be Exchanged	13
2.5.3 Assign Defined Quantities to Built Coupling Regions	15
2.6 Monitors Step – Defining Quantities for Monitoring	17
2.7 Edit Step – Specifying Further Coupling Options	17
2.8 Go Step – Configuring the Application Start-Up and Starting Server and Codes	18
2.8.1 Configuring the Initial Exchange Mode	18
2.8.2 Setting Option Parameters	19
2.8.3 Checking the Application Configuration	19
2.8.4 Starting the Coupled Simulation	20
2.8.5 Interrupting the Computation	25
3 Checking the Results	26
3.1 The MpCCI Visualizer	26
3.2 Post-Processing	27

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).

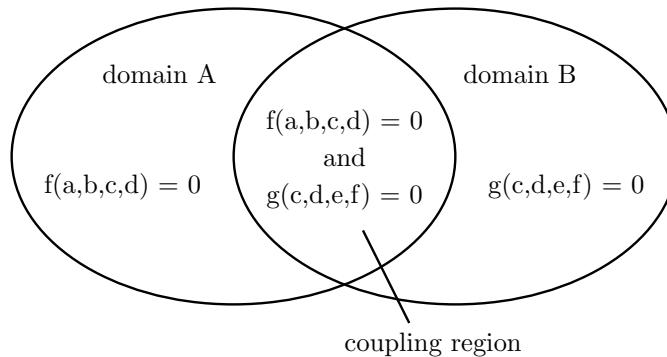


Figure 1: Coupled system: Two domains and a coupling region. Both systems share the variables c and d in the coupling region

[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 ($1 \rightarrow 2$), deformation($2 \rightarrow 1$)

Thermomechanical coupling

- First system: Solid mechanics (equilibrium)
- Second system: Heat conduction (Fourier's law)
- Quantities: Temperature ($2 \rightarrow 1$), deformation ($1 \rightarrow 2$)

Electrothermal coupling

- First system: Electrical conduction (Maxwell's equations)
- Second system: Heat conduction (Fourier's law)
- Quantities: Temperature/electric conductivity ($2 \rightarrow 1$), power loss/Joule heat ($1 \rightarrow 2$)

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.

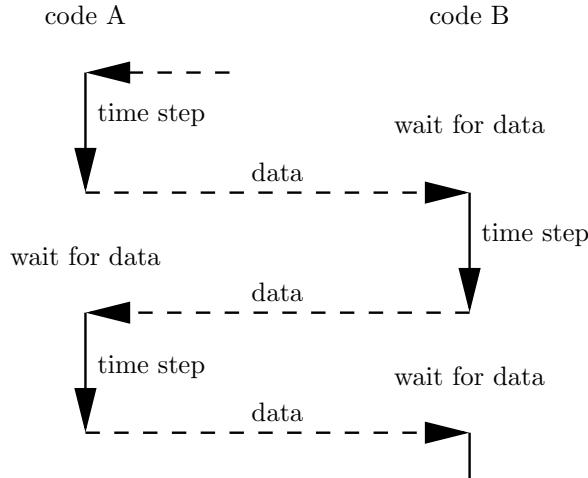


Figure 2: Staggered algorithm for solution of a coupled problem.

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 Start-Up and Starting Server and Codes <](#).

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

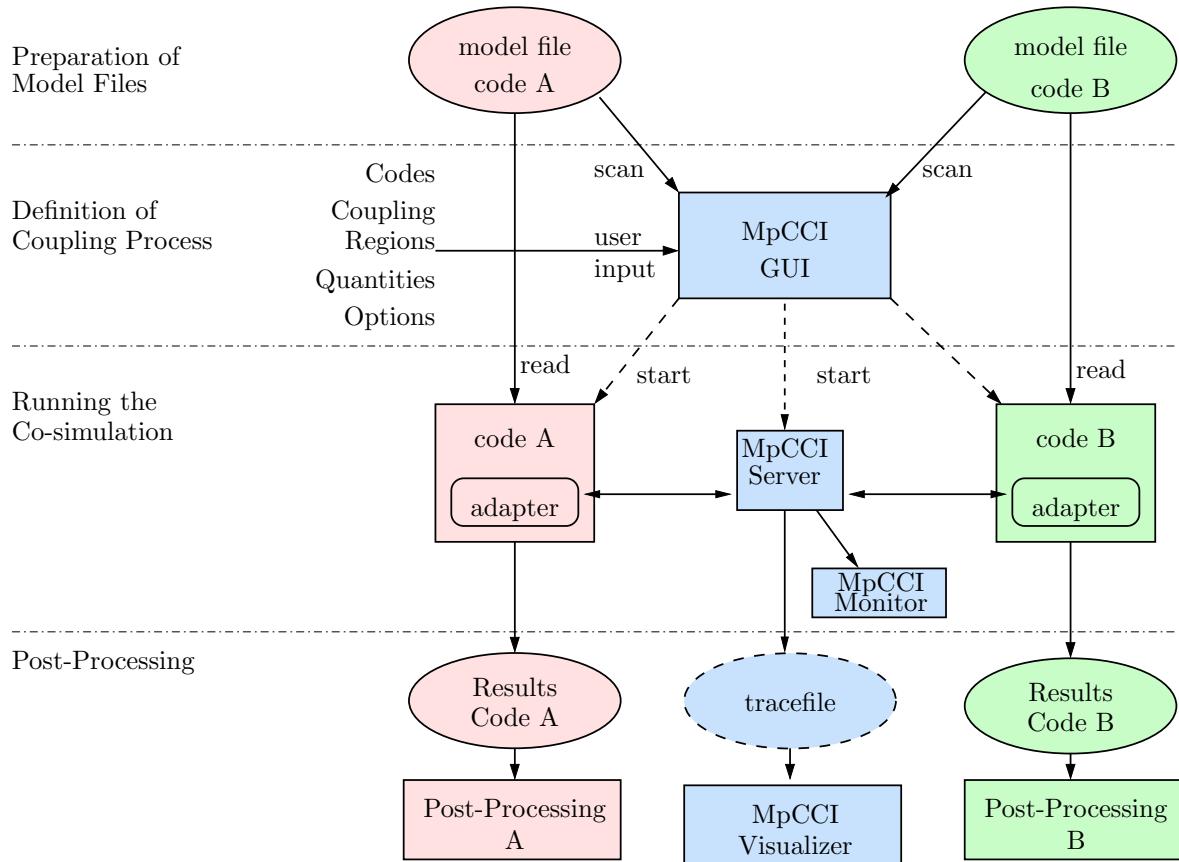


Figure 3: Co-simulation with MpCCI: Overview of the simulation process

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

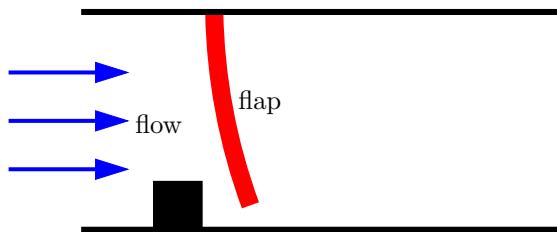


Figure 1: A simple coupled system

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](#).

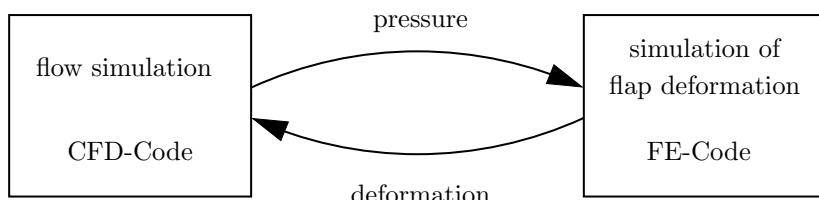


Figure 2: Exchange of quantities in the example FSI simulation.

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.

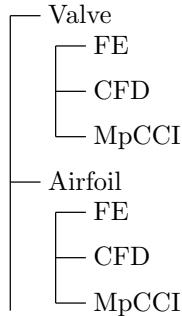


Figure 3: Organization of the directories

2.2.1 CFD Model

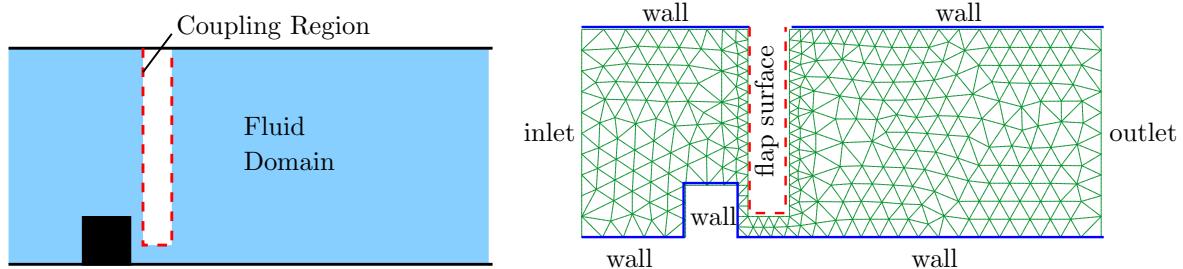


Figure 4: Fluid domain and CFD mesh

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.

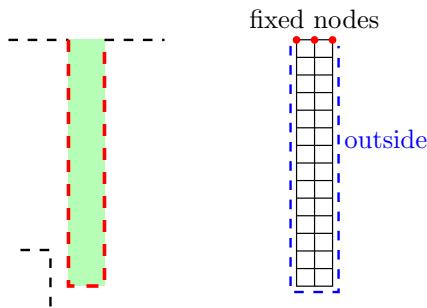


Figure 5: Finite Element model of the flap

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 run the coupled simulation.

At the bottom of each step you find the navigation panel where you see the current step and where you can navigate to the previous and next steps. Following steps must be finished to run a coupled simulation:

1. Models Step – choosing codes and model files.
2. Coupling Step – defining coupling regions and quantities.
3. Monitors Step – defining quantities for monitoring.
4. Edit Step – specifying further coupling options.
5. Go Step – configuring the application start-up and starting server and codes.

2.4 Models Step – Choosing Codes and Model Files

In the Models Step, the following steps should be accomplished:

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

For each code, `Code_1` and `Code_2` (Figure 6), a field for selecting and configuring the analysis code is provided in the MpCCI GUI.

For the CFD-Code: Select the desired CFD-Code - in this case `CFD` - from the pull-down menu providing the available codes. After your selection the parameters and settings of the picked solver are shown underneath (see Figure 6).

1. Select the CFD model file by clicking on the `Browse` button in order to choose the file via the file browser.
2. You may also configure some additional parameters like the CFD version or release.
3. Scan the model file by clicking on the `Start Scanner` button.

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).

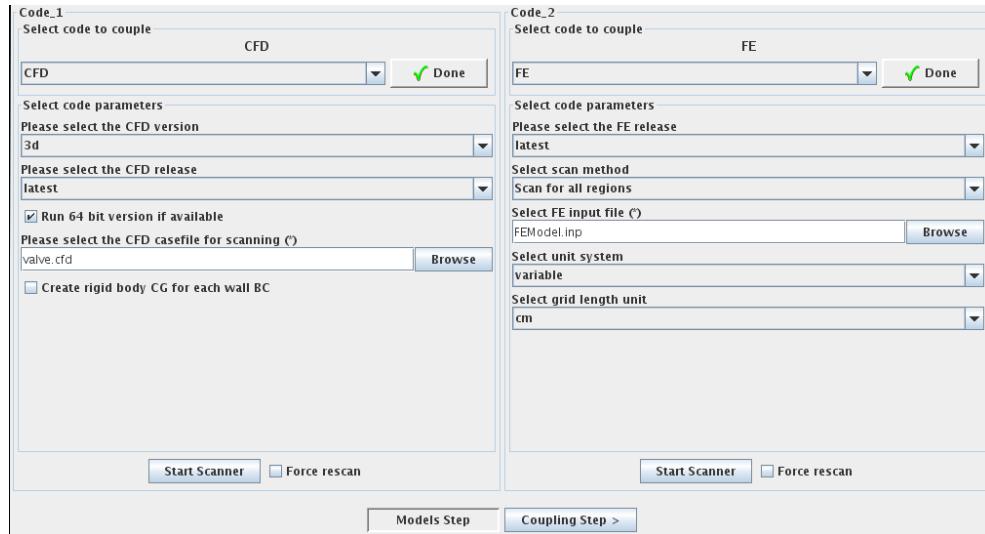


Figure 6: Models Step for CFD and FE codes

1. Select the FE model file via the file browser by clicking on the **[Browse]** button.
2. Select the unit system.
3. If the unit system is set to **variable** an additional parameter will be shown where you can select the grid length unit.
4. In addition you may change the settings for the **FE release** or the **scan method**.
5. Scan the model file by clicking on the **[Start Scanner]** button.

For each scanned model file the status is assigned to the corresponding analysis code:



Done if the extraction of the interface regions was successful.



Error 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.

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 **[Coupling Step >]** button.

2.5 Coupling Step – Defining Coupling Regions and Quantities

In the Coupling 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 coupling 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 7](#)).



Figure 7: Integration Point, Point, Line, Face and Volume element labels

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 8](#)).

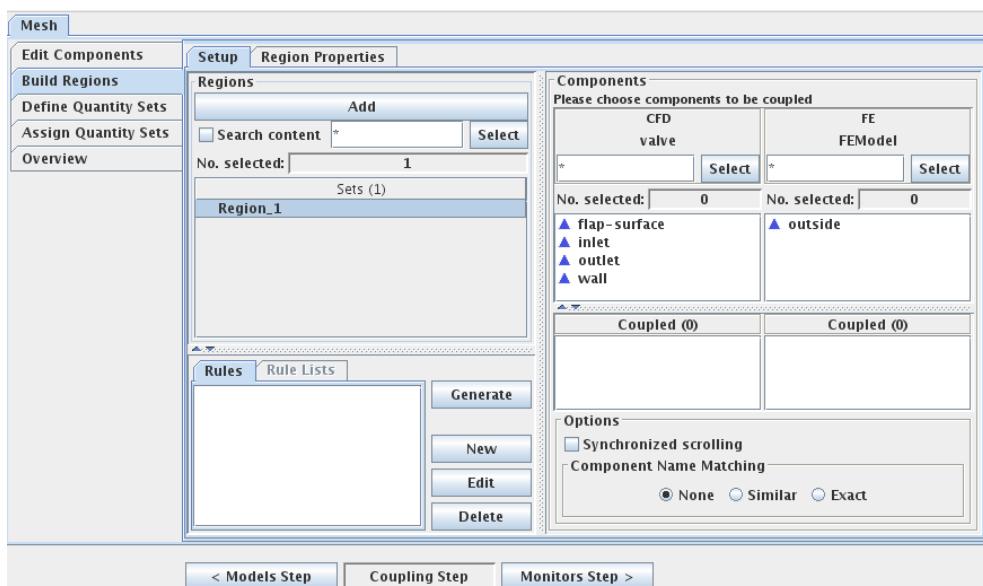


Figure 8: Coupling Step

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

2.5.1 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 8](#)).

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-4.5.5 Automatic Generation of Regions by Rules](#) for more information.

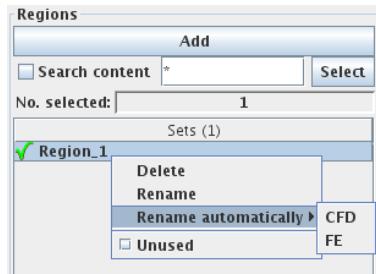


Figure 9: Regions with context menu

The “Regions” area (Figure 9) 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.

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

Regions may have different states (see Figure 10):

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.

2.5.1.1 Select Components for Each Interconnected Code

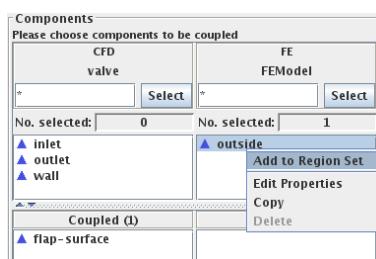


Figure 11: Coupled components selection

In the **Components** area (Figure 11) 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 11](#)).

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

No the region is complete we can go on with defining and assigning quantities to be exchanged.

2.5.2 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 12](#)).

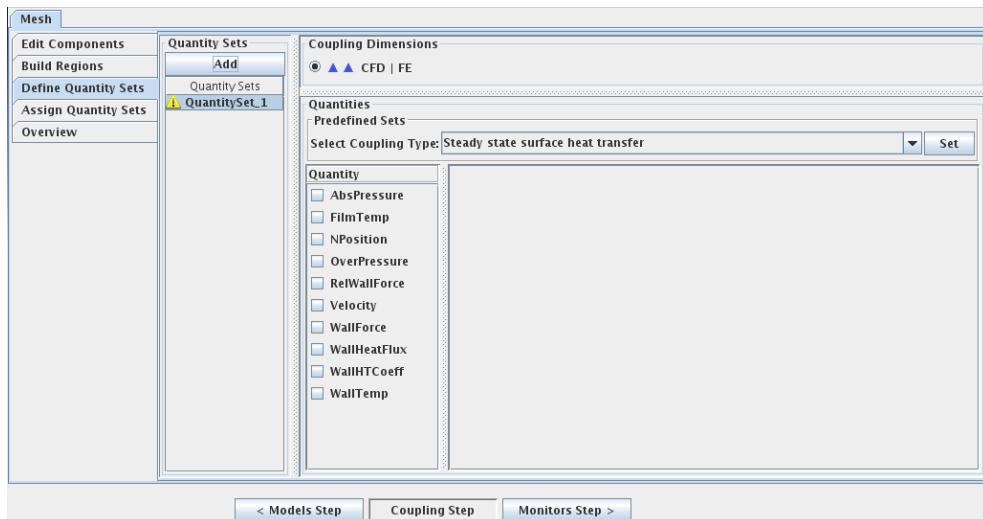


Figure 12: Definition of quantity sets

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.

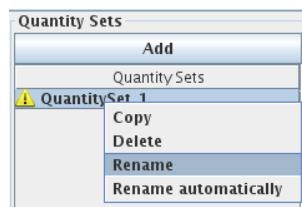


Figure 13: Quantity set with context menu

The “Quantity Sets” (see [Figure 13](#)) area is constructed similar to the regions area before. The default set **QuantitySet_1** already exists. It can be renamed using **Rename** or **Rename automatically** from the pop-up menu where **Rename automatically** builds a name depending on the quantities added to the set. 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.



Figure 14: Incomplete, Unidirectional and Bidirectional quantity set states

Quantity sets may also have different states (see [Figure 14](#)):

Incomplete needs quantities to be transferred

Unidirectional quantities are specified being transferred in only one direction: one code only sends, the other code only receives)

Bidirectional at least two quantities are specified being transferred bidirectional: each code sends and also receives

2.5.2.1 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:

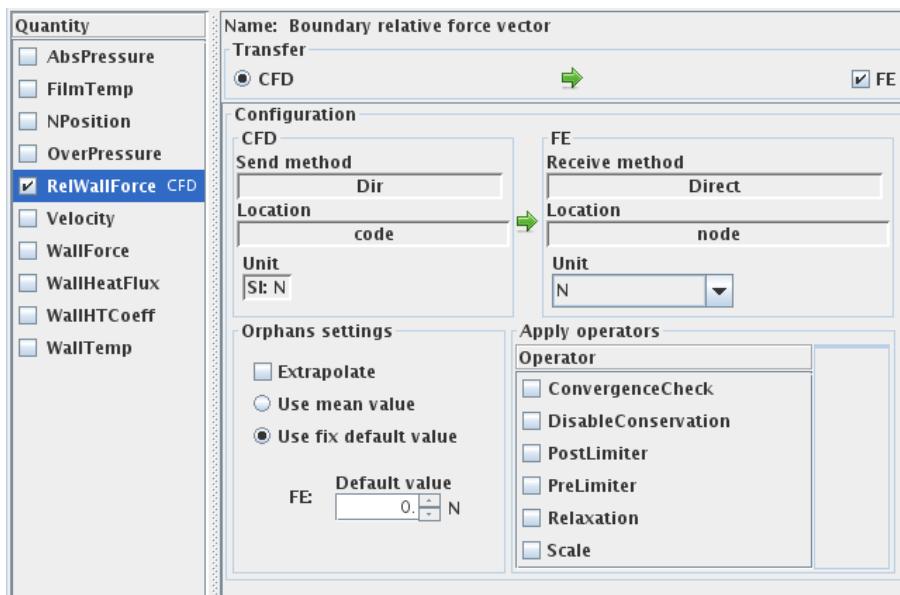


Figure 15: Pressure configuration

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).

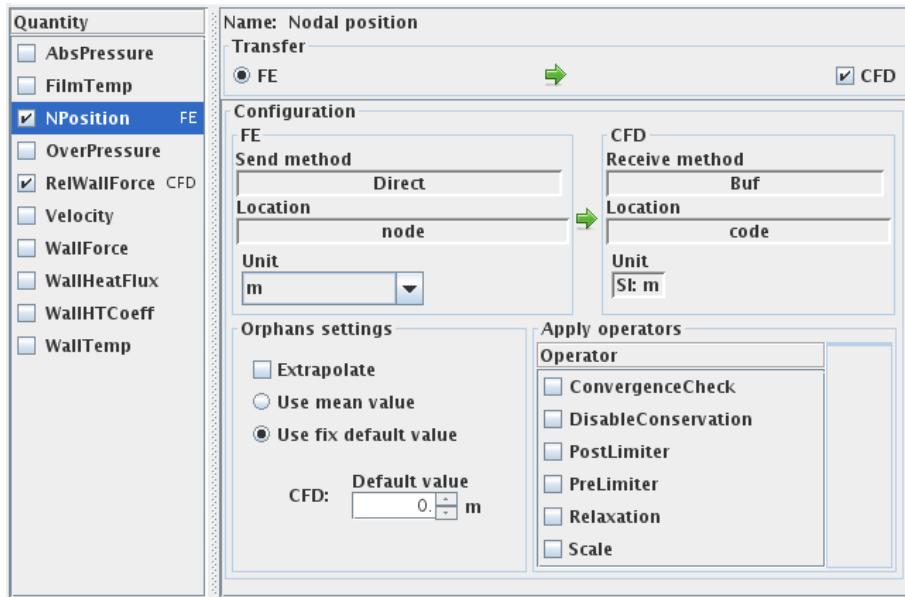


Figure 16: Deformation configuration

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

2.5.3 Assign Defined Quantities to Built Coupling Regions

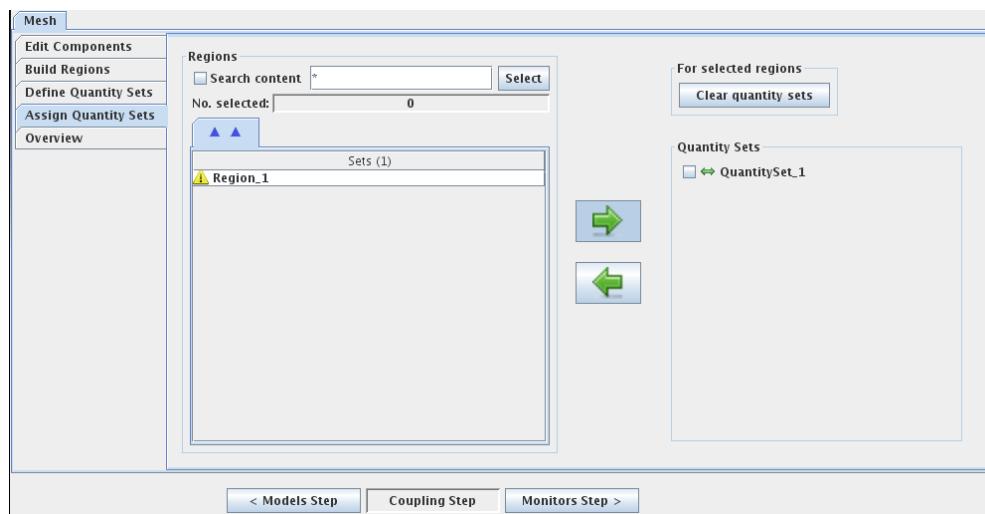


Figure 17: Assignment of quantity sets

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 Figure 14):

Incomplete needs quantity sets to be assigned

Unidirectional complete with assigned quantity sets where all quantities are transferred unidirectional: one code only sends, the other code only receives

Bidirectional complete with assigned quantity sets where the quantities are transferred bidirectional: each code sends and also receives

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.

2.5.3.1 Assign Quantity Set to Coupled Region

In our example we built the coupling region Region_1 and the quantity set QuantitySet_1. Now we assign the quantity set to the coupling region by selecting the region Region_1 and marking the quantity set QuantitySet_1 (see Figure 18).

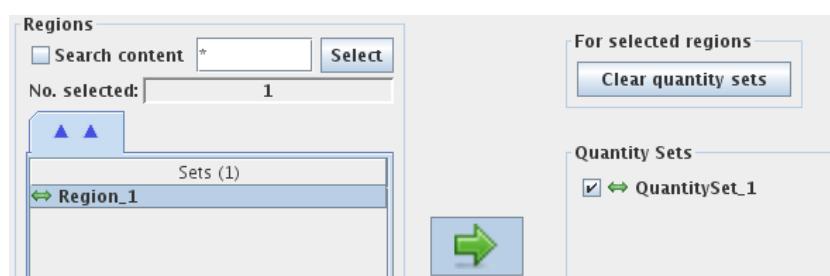


Figure 18: Assign quantity set

For further information have a look at [V-4.5 Coupling Step](#).

The definition of the coupling interface is completed, please click on the **Monitors Step >** button to continue.

2.6 Monitors Step – Defining Quantities for Monitoring

In the Monitors Step you can define some quantities for specific code components to be monitored. For the time being we won't apply monitors. Please click on the **Edit Step >** button to continue.

2.7 Edit Step – Specifying Further Coupling Options

In the Edit 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 settings is given in [V-4.7 Edit Step](#).

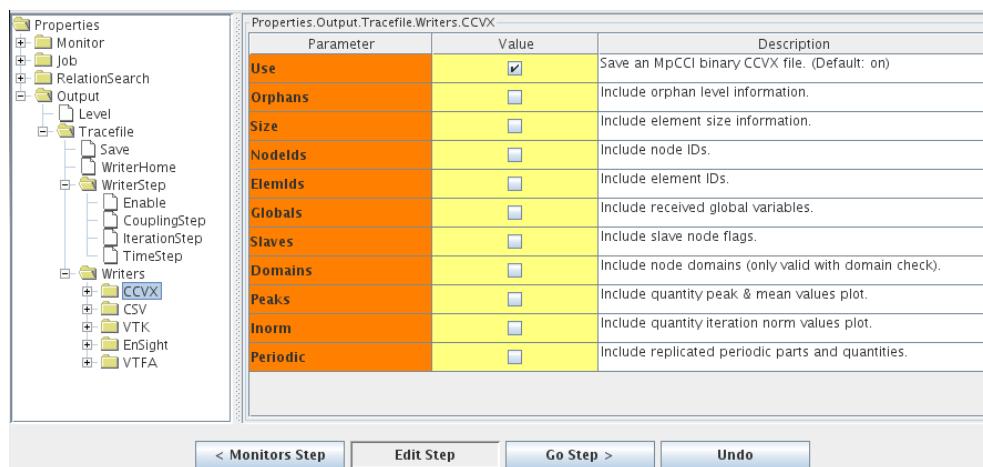


Figure 19: Edit Step

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

The Edit Step window displays a tree 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 the parameter name in the parameters tree. That followed edit the value by clicking the cell with the parameter and select or type in the desired value for this parameter.

For most cases the default values are appropriate, but for our example you may activate additional options for the CCVX tracefile format (Output → TraceFile → Writers → CCVX → Orphans) which is the option for writing orphan level information for the MpCCI Visualizer. Additionally you may edit the output level (Output → Level) which tells MpCCI how much output will be written during the simulation process.

Now click on the **Go Step >** button to proceed to the start-up window for the coupled simulation.

2.8 Go Step – Configuring the Application Start-Up and Starting Server and Codes

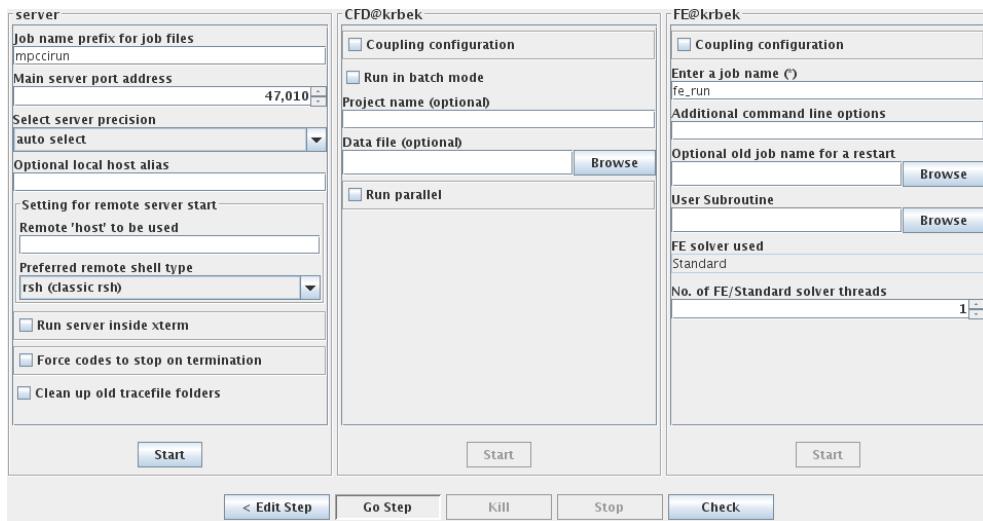


Figure 20: Go Step

In the 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](#)). Before starting the coupled simulation you have to configure the start-up of the applications. For the MpCCI coupling server you usually may retain the default values. For each analysis code you have to configure the initial exchange mode and set some option parameters if necessary.

2.8.1 Configuring the Initial Exchange Mode

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. After both codes have performed the initial data exchange they run the analysis computation until a coupled target time is reached. At this target time both analysis codes will exchange their solution quantities.

For both codes check the **Coupling configuration** option which populates additional settings (see [Figure 21](#)) and define an **Explicit-Transient** coupling scheme.

For the CFD code, set the Initial quantities transfer mode to send.

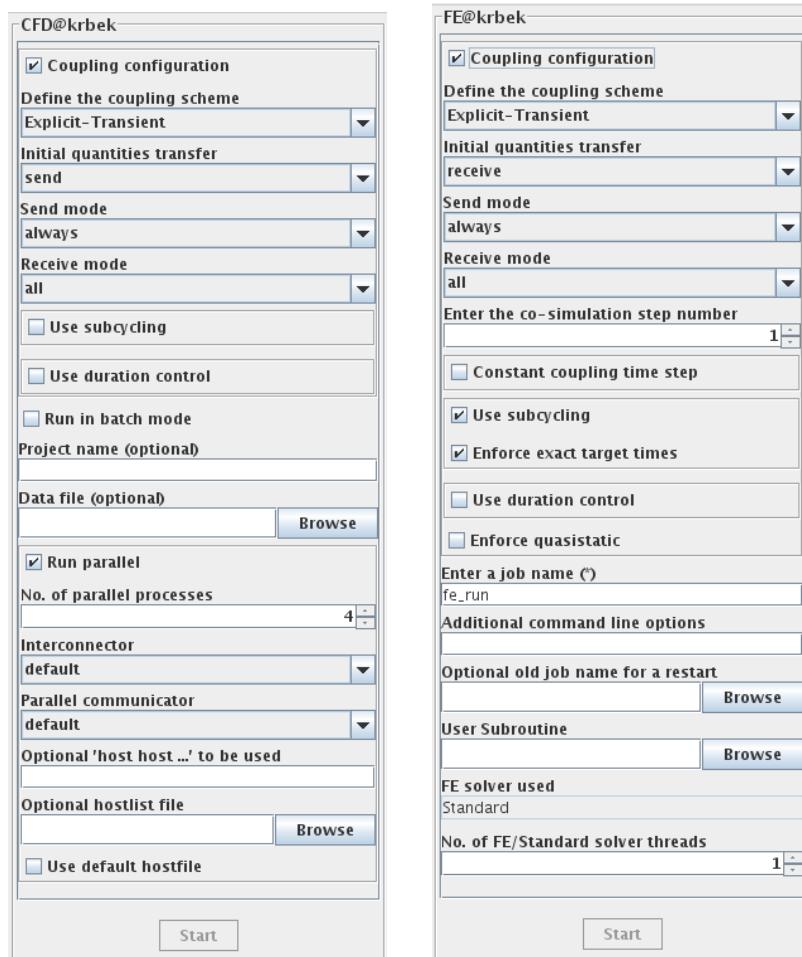


Figure 21: CFD code and FE code settings

For the *FE code*, set the Initial quantities transfer mode to receive.

With this Initial quantities transfer configuration for the CFD and FE code, the coupled simulation will use a parallel coupling algorithm (see [Figure 22](#)).

2.8.2 Setting Option 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 21](#). 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 options need to be set.

2.8.3 Checking the Application Configuration

To cross-check the application configuration a **Check** button at the bottom of the window is provided (see [Figure 20](#)).

For the general settings it checks

- Whether all required parameters for server and codes are set.
- Whether the selected initial quantities transfer is valid.
- Whether the selected initial quantities transfer fits to configuration of the exchanged quantities.

At the end of the check a report with the results will be shown.

This check allows that the formal conditions are fulfilled ahead of the simulation start.

Optional checkers may be provided for each single code. See [▷ V-4.8.3 Checking the configuration](#) for more information.

2.8.4 Starting the Coupled Simulation

Before starting a coupled simulation you have to establish a project by using the action **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.

Server and each application have their own **Start** button. But only the **Start** button from the MpCCI server is enabled because the server has to be launched in advance. After the server has been started its **Start** button will be replaced by a running symbol, all settings will be locked and the **Start** buttons for the next applications will be enabled one after the other.

For our valve example the start-up procedure will be:

1. Click on **Start** to launch the coupling server. Now the server waits to get response from all client codes.
2. The **Start** button of the next application (CFD) is enabled. Now click on **Start** to launch the CFD analysis code.
3. After that the last application (FE) has to be launched by a click on its **Start** button.

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_adapter_CFD: feature test done.
[MpCCI License] mpcci_clients: feature checkout...
[MpCCI License] mpcci_clients: feature checked out.
Waiting for incoming connections on "47010@nemo"...
[MpCCI License] mpcci_adapter_FE: feature test...
[MpCCI License] mpcci_adapter_FE: feature test done.
[MpCCI License] mpcci_clients: feature checkout...
[MpCCI License] mpcci_clients: feature checked out.
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

Det: 1

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

[...]

```

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
      NElements    : 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 send (CFD) and receive (FE) as initial exchange mode (see Figure 22 bottom). The CFD code is represented by “code B” and the FE code by “code A”. 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 Coupling Algorithms ◁.

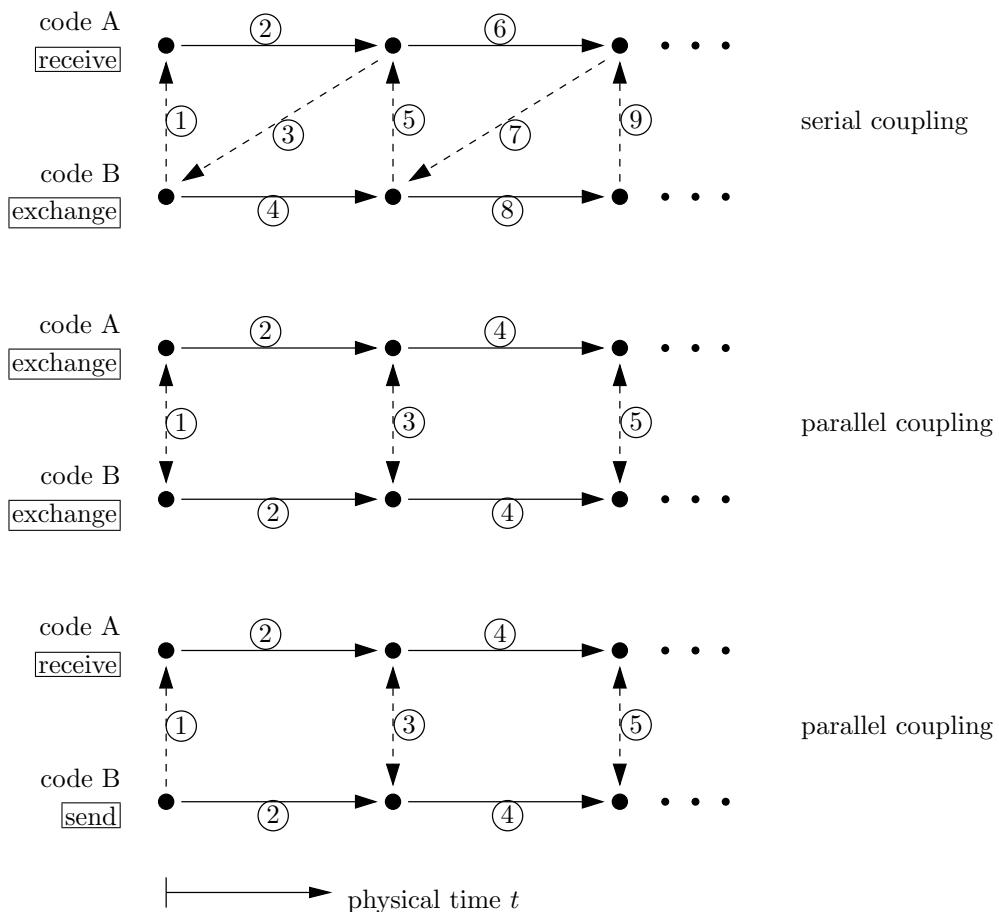


Figure 22: Initial exchange and resulting coupling algorithms for two codes with “exchange before solution”

When an application exits normally, a window displays information collected from the application output.

2.8.5 Interrupting the Computation

While the applications are running, the **Kill** and **Stop** buttons at the bottom of the window are enabled and provide following functionalities:

Kill to terminate all applications by sending a kill signal or by executing the "Killer" application module if available.

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.

To modify parameter settings you have to terminate the applications by pressing the **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-6 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 Start-Up and Starting Server and Codes](#) ◷). To obtain a tracefile, the checkbox **Use** must be selected in the Edit 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
```

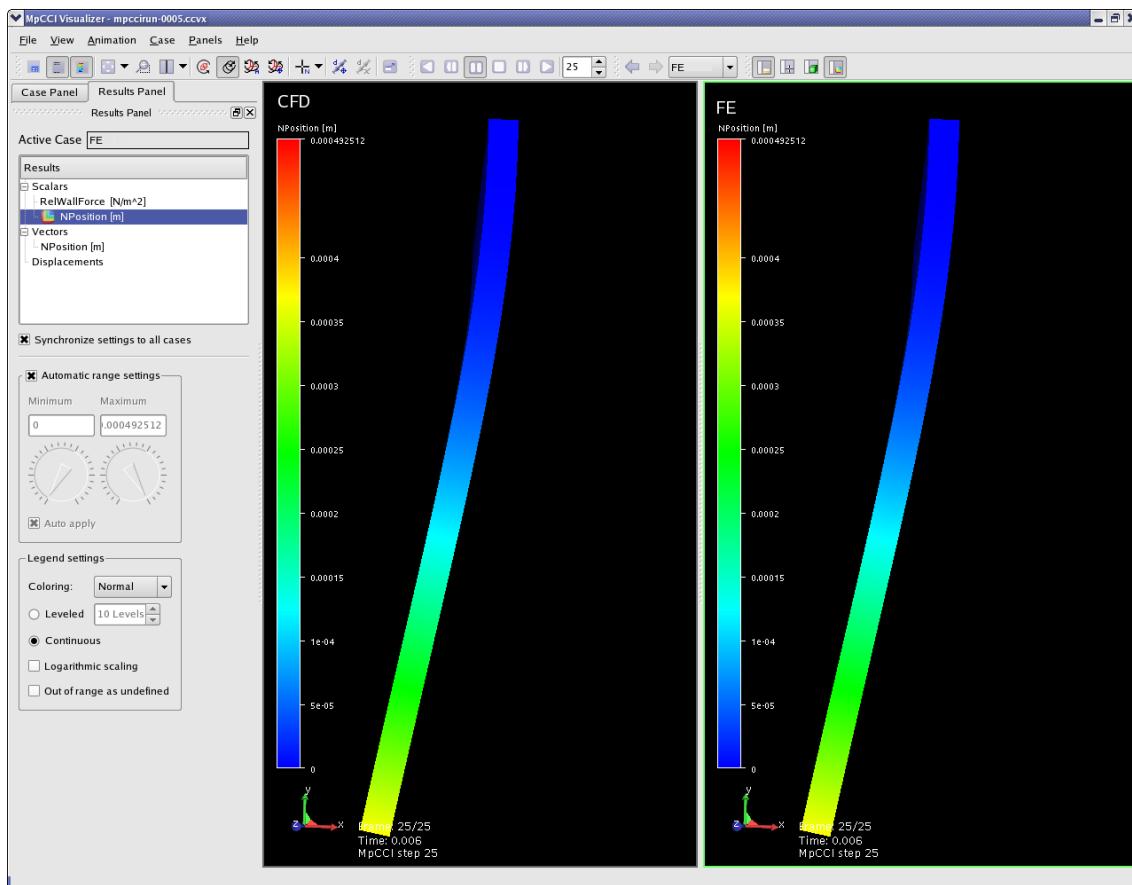


Figure 1: Main window of the MpCCI Visualizer

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.

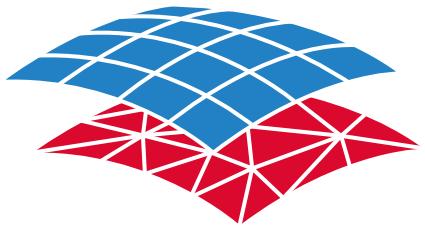
Transient analyses consist of several steps, the step number can be selected in the control window.

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.



MpCCI
CouplingEnvironment

Part V

User Manual

Version 4.5.0

MpCCI 4.5.0-1 Documentation
Part V User Manual
PDF version
March 30, 2017

MpCCI is a registered trademark of Fraunhofer SCAI
www.mpcci.de



Fraunhofer Institute for Algorithms and Scientific Computing SCAI
Schloss Birlinghoven, 53754 Sankt Augustin, Germany

Abaqus and SIMULIA are trademarks or registered trademarks of Dassault Systèmes
ANSYS, FLUENT and ANSYS Icepak are trademarks or registered trademarks of Ansys, Inc.
Elmer is an open source software developed by CSC
FINE/Open and FINE/Turbo are trademarks of NUMECA International
Flowmaster is a registered trademark of Flowmaster Group NV
JMAG is a registered trademark of The JSOL Corporation
MATLAB is a registered trademark of The MathWorks, Inc.
MSC Adams, MSC Marc, MD NASTRAN and MSC NASTRAN are trademarks or registered trademarks of
MSC.Software Corporation
OpenFOAM is a registered trademark of OpenCFD Ltd.
PERMAS is a registered trademark of Intes GmbH
RadTherm, TAITherm is a registered trademark of ThermoAnalytics Inc.
SIMPACK is a registered trademark of SIMPACK AG.
STAR-CCM+ and STAR-CD are registered trademarks of CD adapco Group

ActivePerl has a Community License Copyright of Active State Corp.
FlexNet Publisher is a registered trademark of Flexera Software.
Java is a registered trademark of Oracle and/or its affiliates.
Linux is a registered trademark of Linus Torvalds
Mac OS X is a registered trademark of Apple Inc.
OpenSSH has a copyright by Tatu Ylonen, Espoo, Finland
Perl has a copyright by Larry Wall and others
UNIX is a registered trademark of The Open Group
Windows, Windows XP, Windows Vista and Windows 7 are registered trademarks of Microsoft Corp.

V User Manual – Contents

1	Introduction	7
1.1	Basic Structure of MpCCI	7
2	The MpCCI Software Package	9
2.1	Introduction	9
2.2	The MpCCI Home Directory	10
2.3	Environment and Environment Variables	11
2.3.1	MPCCI_ARCH - Architecture Tokens	12
2.3.2	MPCCI_DEBUG - for Debugging	13
2.3.3	MPCCI_TINFO	13
2.4	MpCCI Project and Output Files	15
2.4.1	MpCCI Project Files	15
2.4.2	MpCCI Server Input Files	15
2.4.3	Log Files	15
2.4.4	Tracefile	15
2.5	The MpCCI Resource Directory	16
2.6	Temporary Files	17
2.7	Third Party Software Used by MpCCI	19
2.7.1	Perl	19
2.7.2	Java	19
2.7.3	Remote Shell and Remote Copy	19
3	Code Coupling	20
3.1	Multiphysics	20
3.1.1	Physical Domains	20
3.1.2	Coupling Types	21
3.2	Mesh Checks	22
3.2.1	Mesh Motion Checks	23
3.2.2	Bounding Box Checks	23
3.2.3	Domain Check	23
3.2.4	Slave Node Mark	25
3.3	Data Exchange	26
3.3.1	Association	27
3.3.2	Interpolation	29
3.3.3	Handling of Coupled Quantities	32
3.4	Coupling Algorithms	43
3.4.1	Course of the Coupling Process	43

3.4.2	Stationary Problems	44
3.4.3	Transient Problems	44
3.4.4	Iterative Coupling	50
3.4.5	Exchange of Time Step Size	54
3.4.6	Subcycling	55
3.4.7	Non-matching Time Steps	57
3.4.8	Restarting a Coupled Simulation	59
3.5	Running MpCCI in a Network	60
3.5.1	Client-Server Structure of MpCCI	60
3.5.2	Hostlist File	61
3.5.3	Remote Shell and Remote Copy	62
3.6	Coupled Analysis in Batch Mode	62
3.6.1	General Approach	62
3.6.2	Job Scheduler Environment	63
4	Graphical User Interface	76
4.1	Starting and Exiting MpCCI GUI	76
4.1.1	Starting MpCCI GUI	76
4.1.2	Exiting MpCCI GUI	77
4.2	MpCCI GUI Properties	77
4.3	MpCCI GUI Menus	77
4.3.1	File Menu	78
4.3.2	Batch Menu	78
4.3.3	License Menu	79
4.3.4	Tools Menu	80
4.3.5	Codes Menu	80
4.3.6	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	Coupling Step	83
4.5.1	Global Variables	83
4.5.2	Editing Components	84
4.5.3	Coupling Components with Different Dimensions	88
4.5.4	Simplified Selection	88
4.5.5	Automatic Generation of Regions by Rules	89
4.5.6	Applying Region Properties	93
4.5.7	Quantity Specifications	94
4.5.8	Assigning Preconfigured Quantity Settings	97

4.5.9	Overview of the Coupled Regions	100
4.5.10	Requirements	101
4.6	Monitors Step	101
4.7	Edit Step	102
4.7.1	Monitor	102
4.7.2	Job	104
4.7.3	Relation Search	106
4.7.4	Output	107
4.8	Go Step	109
4.8.1	Configuring the MpCCI Coupling Server	109
4.8.2	Coupling Configuration Parameters	110
4.8.3	Checking the configuration	111
4.9	Remote File Browser	111
4.9.1	File Browser Handling	111
4.9.2	How to mount a new file system	113
5	Command Line Interface	115
5.1	Using the Command Line Interface	115
5.2	Overview of All Subcommands	116
5.3	Starting MpCCI	118
5.3.1	<code>mpcci fsimapper</code>	119
5.3.2	<code>mpcci gui</code>	120
5.3.3	<code>mpcci logviewer</code>	121
5.3.4	<code>mpcci monitor</code>	122
5.3.5	<code>mpcci visualize</code>	123
5.4	MpCCI Tools	125
5.4.1	<code>mpcci ccvxcat</code>	126
5.4.2	<code>mpcci cosimpre</code>	127
5.4.3	<code>mpcci morpher</code>	128
5.4.4	<code>mpcci observe</code>	131
5.4.5	<code>mpcci xterm</code>	132
5.5	Information and Environment	133
5.5.1	<code>mpcci arch</code>	134
5.5.2	<code>mpcci doc</code>	135
5.5.3	<code>mpcci info</code>	136
5.5.4	<code>mpcci env</code>	138
5.5.5	<code>mpcci home</code>	139
5.5.6	<code>mpcci where</code>	140
5.6	Installation and Licensing	141
5.6.1	<code>mpcci license</code>	142

5.6.2 <code>mpcci list</code>	143
5.6.3 <code>mpcci lmutil</code>	144
5.6.4 <code>mpcci ssh</code>	145
5.6.5 <code>mpcci test</code>	146
5.7 Job Control	149
5.7.1 <code>mpcci backup</code>	150
5.7.2 <code>mpcci batch</code>	151
5.7.3 <code>mpcci batch LSF</code>	153
5.7.4 <code>mpcci batch PBS</code>	154
5.7.5 <code>mpcci batch N1GE</code>	155
5.7.6 <code>mpcci batch LoadLeveler</code>	156
5.7.7 <code>mpcci batch GLOBUS</code>	157
5.7.8 <code>mpcci clean</code>	158
5.7.9 <code>mpcci kill</code>	159
5.7.10 <code>mpcci ps</code>	161
5.7.11 <code>mpcci ptoj</code>	162
5.7.12 <code>mpcci server</code>	163
5.7.13 <code>mpcci top</code>	167
6 MpCCI Visualizer	168
6.1 Using the MpCCI Visualizer	168
6.1.1 Data Flow	168
6.1.2 Supported Platforms	168
6.1.3 Starting the Monitor	169
6.1.4 Starting the Visualizer	169
6.2 MpCCI Visualizer for .ccvx and online monitoring	169
6.2.1 Introduction	169
6.2.2 Main Window	170
6.2.3 Menus and Toolbars	170
6.2.4 Panels	176
6.2.5 Viewport Area	182
6.2.6 Preferences Viewer Dialog	185
6.2.7 Preferences Server Dialog	187
6.2.8 Store Animated Files Dialog	187
6.2.9 Error Dialog	189
6.2.10 Command Line Parameters	190
6.3 Frequently Asked Questions	191
7 MpCCI Grid Morpher	192
7.1 Using the MpCCI Grid Morpher	192
7.1.1 Options description	192

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](#).

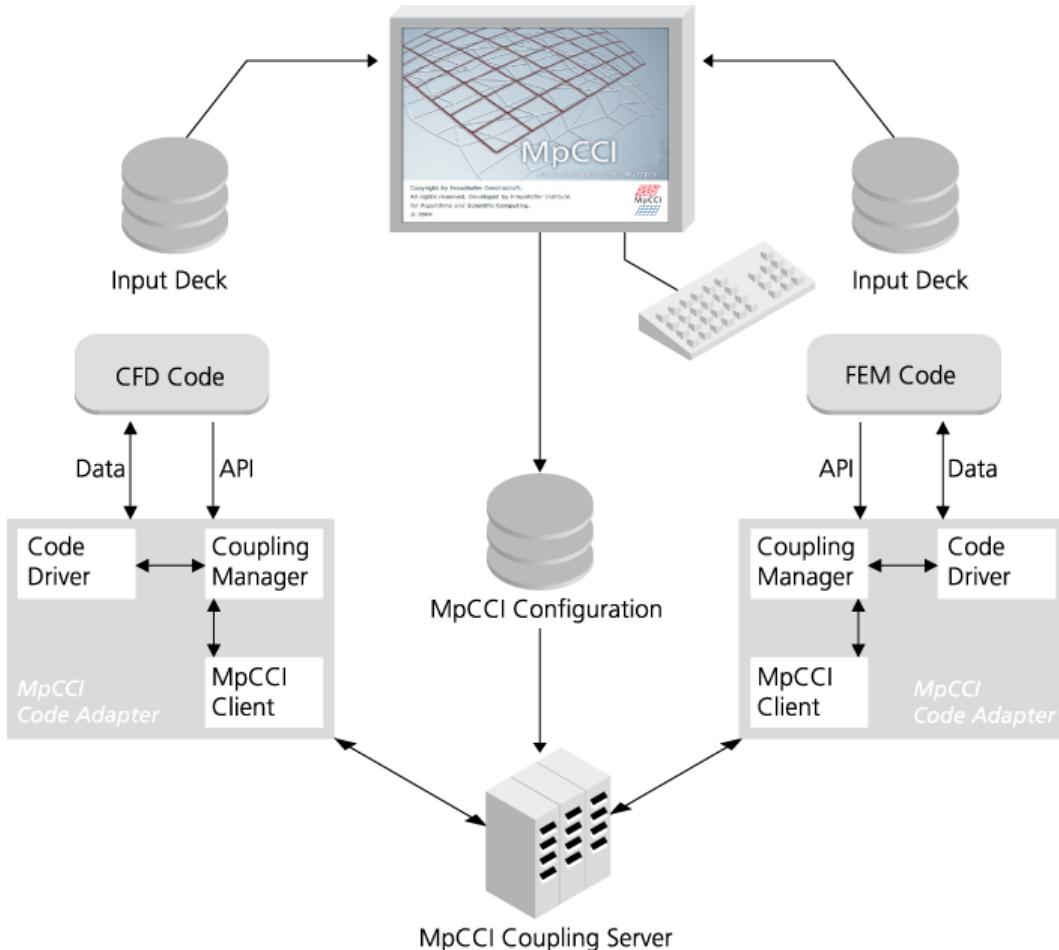


Figure 1: MpCCI architecture

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 [▷ 4 Graphical User Interface ◁](#)
- 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.

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 [▷ 3 Code Coupling ◁](#).

MpCCI Visualizer. The MpCCI visualizer is a tool for checking the coupling process. Meshes and transferred quantities can be displayed. See [▷ 6 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 [▷ 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 [▷ 7 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 [▷ 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 [▷ 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 135](#)

"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>_....`.

- ① 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:

MPCCI_ARCH	Contains the MpCCI architecture token, which is used to identify the platform	▷ 2.3.1 MPCCI_ARCH - Architecture Tokens ◁
MPCCI_DEBUG	If defined, MpCCI runs in debug mode	▷ 2.3.2 MPCCI_DEBUG - for Debugging ◁
MPCCI_HOSTLIST_FILE	Path to file which contains a list of possible hosts for parallel runs	▷ 3.5.1 Client-Server Structure of MpCCI ◁
MPCCI_TINFO	Passed to the code adapter to determine the coupling algorithm	▷ 2.3.3 MPCCI_TINFO ◁
MPCCI_LICENSE_FILE	Location of the MpCCI license server	▷ III-5.3 Defining the License Server ◁
MPCCI_LICENSE_TIMEOUT	Timeout for license check	▷ III-5.3 Defining the License Server ◁
MPCCI_REMOTE_HOSTNAME	Name of the host seen by a remote system	▷ 3.5 Running MpCCI in a Network ◁
MPCCI_RSHTYPE	Type of remote shell used by MpCCI	▷ 3.5.3 Remote Shell and Remote Copy ◁
MPCCI_NOREMSH	Disable errors if remote shell tools is not available on the local machine if defined	
MPCCI_TIMEOUT	Client connection timeout	▷ 3.5.1 Client-Server Structure of MpCCI ◁
MPCCI_TMPDIR	Path to temporary directory	▷ 2.6 Temporary Files ◁

System Variables

In addition to the control variables, MpCCI evaluates some system variables:

General variables (all OS)

JAVA_BINDIR JAVA_HOME JAVA_ROOT JDK_HOME JRE_HOME	These variables are evaluated by MpCCI to find the Java installation.	
PATH	The list of directories which is searched for executables. "MPCCI_HOME/bin" must be included in the path.	▷ III-2 Before the Installation ◁

Windows variables

ALLUSERSPROFILE	location of All Users Profile	
COMSPEC	secondary command interpreter	
HOMEDRIVE HOME PATH	These variables define the home directory, which is referred to as HOME in this manual.	
PROCESSOR_ARCHITECTURE PROCESSOR_ARCHITEW6432	The MpCCI architecture token is defined according to the values of these variables	
USERPROFILE	Used to determine the home directory of a user on a remote machine.	▷ 2.7.3 Remote Shell and Remote Copy ◁ on page 19

UNIX/Linux variables

HOME	The home directory, which is referred to as HOME in this manual.	
------	--	--

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 4.5 ◁](#).

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.

2.3.3 `MPCCI_TINFO`

This variable is required and is only used by the adapter for the application. It is set automatically within the MpCCI GUI before starting an application. This variable is composed of 10 information for the coupling configuration. The `MPCCI_TINFO` must be set by using this syntax:

```
MPCCI_TINFO=coupling scheme:coupling algorithm:initial exchange:send mode:receive mode:
duration control:start time:end time:start iter:end iter:
post iter: implicit timestep: implicit iter:implicit max iter:
subcycle:number of subcycle
```

with these available values:

- coupling scheme = Implicit transient (I), Explicit-SteadyState (ES) , Explicit-Transient (ET) (default = ET)
- coupling algorithm = Gauss Seidel GS, Jacobi J
- initial exchange = send|exchange|receive (default = exchange)
- send mode = always|onewait|allwait (default = always)
- receive mode = all|available|any|complete (default = all)
- duration control = activate duration control none|duration (default = none)
- start time = time to begin the coupling (default = -1)
- end time = time to end the coupling (default = -1)
- start iter = iteration number to begin the coupling (default = -1)
- end iter = iteration number to end the coupling (default = -1)
- post iter = number of post iterations after end of coupling (default = -1)
- implicit timestep = time step for implicit coupling (default = -1)
- implicit iter = number of iterations without coupling (default = 1)
- implicit iter max = maximum number of iterations (default = -1)
- subcycle = activate subcycle none|subcycle (default = none)
- number of subcycle = number of subcycle step (default = 2)

If you use the `mpcci server ...` command to start the server and invoke the applications "by hand", please do not forget to set this variable to for example:

```
MPCCI_TINFO=ET:J:send:always:all:none:2:none:-1:-1:-1:-1:-1:-1:-1:-1:-1
```

before you start the applications.

2.4 MpCCI Project and Output Files

2.4.1 MpCCI Project Files

The MpCCI project files are named "`*.csp`" and contain all relevant data for a coupled computation in XML format.

2.4.2 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!

2.4.3 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.

"`<job name prefix>_<writer type>`" If you choose to activate some writers in the Edit Step of the MpCCI GUI (see ▷4.7 Edit 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 ▷4.8 Go Step) and the suffix of the writer (e. g. `ccvx, vtk`).

2.4.4 Tracefile

MpCCI server can write tracefile for other visualizer tools:

- The tracefile "`*.ccvx`" is used for the MpCCI Visualizer, see ▷6 MpCCI Visualizer.
- 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.

2.5 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.

2.6 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 the MpCCI 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.

2.7 Third Party Software Used by MpCCI

2.7.1 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.

2.7.2 Java

The MpCCI GUI is based on Java. To run the MpCCI GUI a Java virtual machine is required.

2.7.3 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 Algorithms](#).

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:

Domain	MpCCI Token
Solid mechanics	<code>SolidStructure</code>
Fluid mechanics	<code>Fluid</code>
Acoustics	<code>SolidAcoustics</code>
Fluid pipe systems	<code>Network</code>
Solid heat transfer	<code>SolidThermal</code>
Fluid heat transfer	<code>FluidThermal</code>
Heat radiation	<code>Radiation</code>
Electromagnetism	<code>ElectroMagnetism</code>
Multibody dynamics	<code>mbdynamics</code>

 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 [4.5.8 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-9 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-6 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 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-7 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 at the MpCCI GUI Edit Step ([▷2 Job◁](#)).

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 [▷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 Overlapcheck in the [▷4.7 Edit Step◁](#).

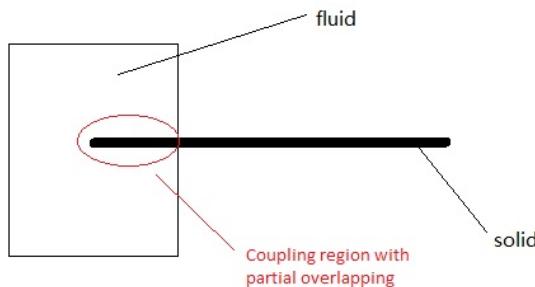


Figure 1: Partial overlapping for the coupling region exceeding 2.5 times the overlapping size in one direction

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 [▷2 Job◁](#).

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 writer. 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 domain (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 [▷ 2 Job](#) in MpCCI GUI or later in the MpCCI Monitor by using the server preferences menu dialog ([▷ 6.2.7 Preferences Server Dialog](#)). You have also the option to save this information in the tracefile by activating the Domains for the corresponding writer used.

 All of these options requires that the DomainCheck at the job level 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.

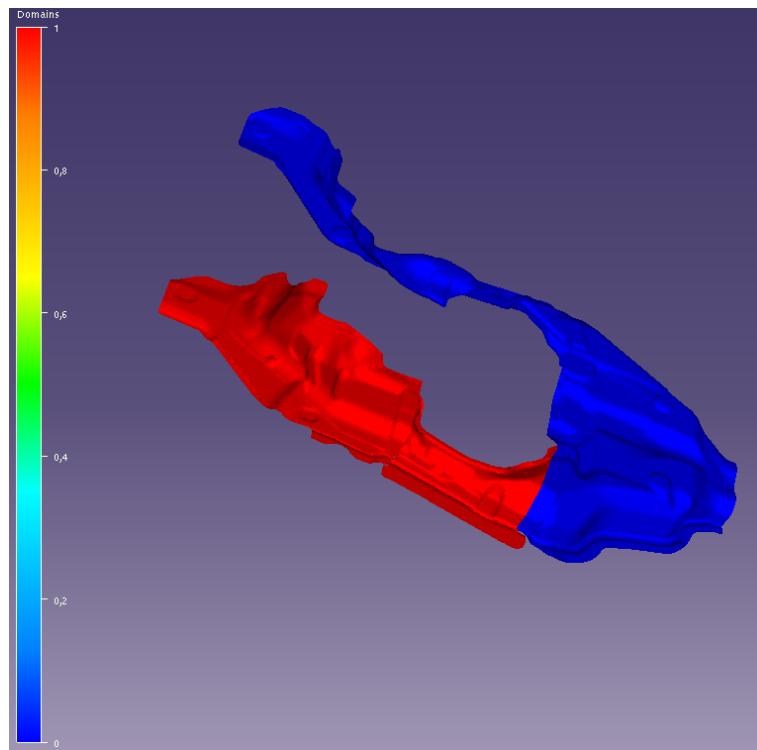
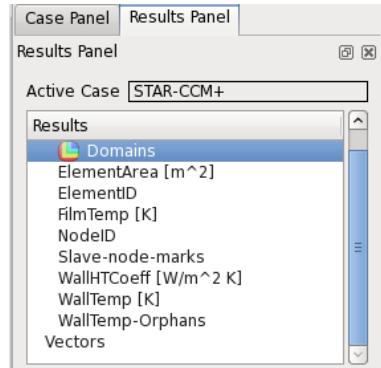


Figure 2: Example of domains visualization

3.2.4 Slave Node Mark

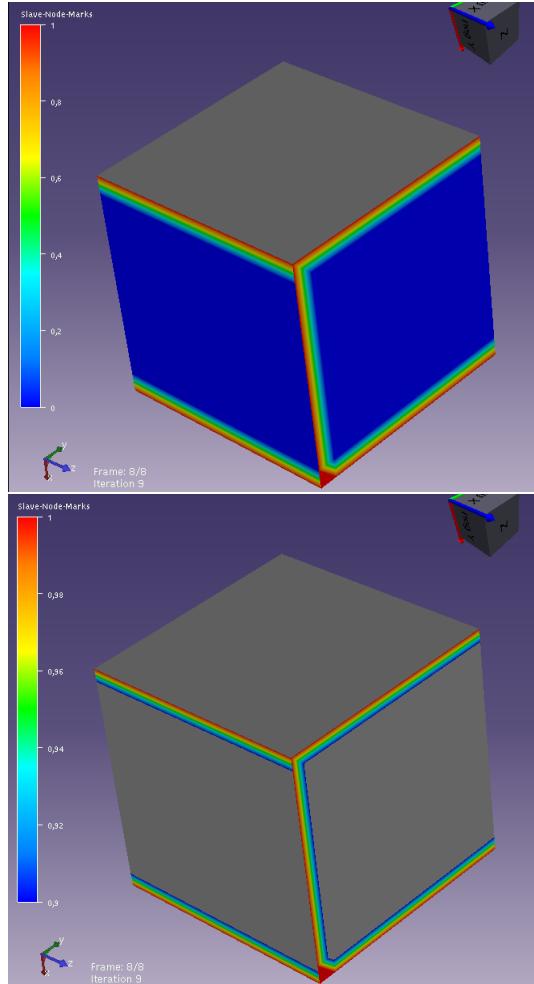
For a mesh composed of several partitions it is possible to visualize the nodes relation at the partition boundary. The partitions connection can be identified. For using this feature the parameter **Slaves** in [►1](#)

Monitor \triangleleft 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 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.



3.3 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 three 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.

Interpolation The data which shall be transferred must be adapted to the target mesh, while e.g. conserving the sum of forces.

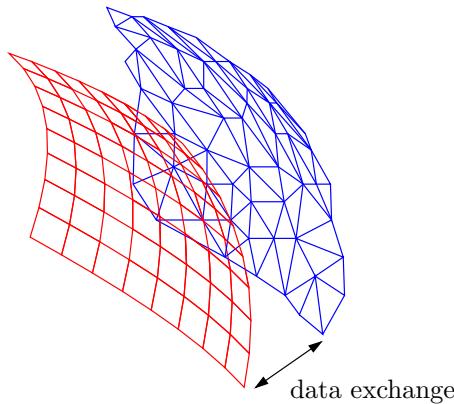


Figure 3: Data exchange between two non-matching grids (distance exaggerated)

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

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.

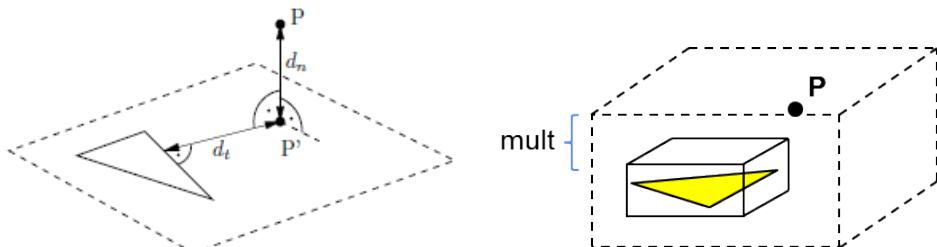


Figure 5: a)Normal distance d_n and tangential distance d_t for a point P b) Multiplicity factor parameter

MpCCI adjusts the neighborhood search parameters depending on the mesh discretization by computing

the geometric parameters according to this procedure:

- Characteristic mesh length:
 - $l_{\min} = 0.5 * (l_{\minSource} + l_{\minTarget})$
 - $l_{\max} = \max(l_{\maxSource}, l_{\maxTarget})$
 - $l_{\mean} = 0.5 * (l_{\meanSource} + l_{\meanTarget})$
- Node tolerance: $ntol = \min(l_{\minSource}, l_{\minTarget})/5.0$
- Precision of neighborhood search: $prec = (0.9 * l_{\minSource} + 0.1 * l_{\meanSource})/20$
- Normal distance: $d_n = l_{\mean}$
- Tangential distance: $d_t = l_{\min}$
- Distance: $vdist = 0.9 * l_{\min} + 0.1 * l_{\mean}$
- Multiplicity:
 - $mult = \frac{l_{\max}}{l_{\min}}$ and is clipped to 10.0
 - $mult = mult * mfac$ mfac is the user multiplicity factor.
 - $correction = \frac{\max(l_{\minSource}, l_{\minTarget})}{l_{\min}}$ This correct the averaging from l_{\min} in case of degenerated elements and limits to a maximum value of 2.
 - Final multiplicity factor value: $mult = mult * correction$

The only parameter to control the neighborhood search is **multiplicity** (see [▷ 4.7 Edit Step ▲](#)). 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 the **normal distance** (see [▷ 4.7 Edit Step ▲](#)). 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.

All parameters for association can be set in the Edit Step of the MpCCI GUI, see [▷ 4.7 Edit Step ▲](#).

3.3.1.2 Association between a System Network Code and a Mesh Based Code

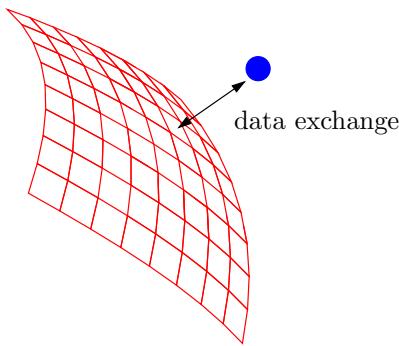


Figure 6: Representation of an integration point coupling with a surface mesh

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.



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 Flux and Field Interpolation

The interpolation of quantities requires the distinction of two different cases, which can be characterized as flux and 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.

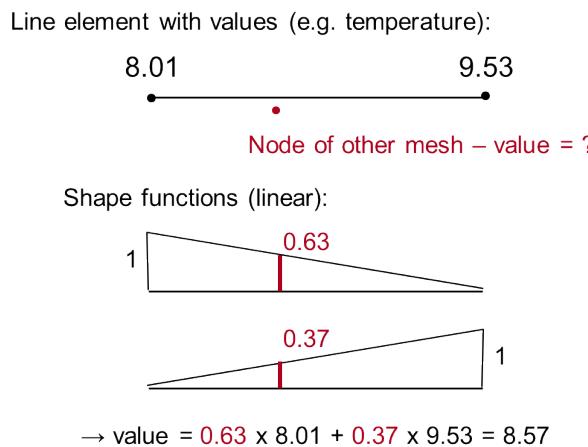


Figure 8: Field interpolation example

In *flux integral* interpolation (Figure 9), the value is adapted to the element sizes to preserve the integral. Flux interpolation is e. g. used for forces.

In MpCCI a third case, *flux density*, is considered. 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](#).

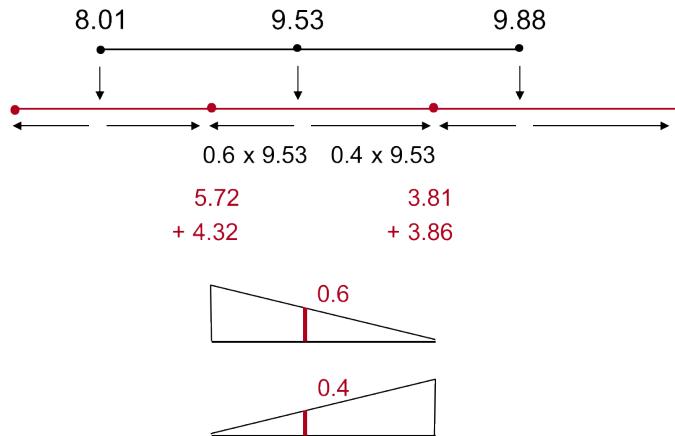


Figure 9: Flux interpolation example

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 [▷ 4.5.7.3 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 [▷ 6 MpCCI Visualizer ◁](#). In the MpCCI Visualizer orphans information is associated with different values in the color range legend ([Figure 10](#)).

- 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 contains any orphaned nodes.
- A weakly mapped node gets the value 1.
- All orphans information value greater than 1 has 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 value 2 or 6:
 - 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 as absolute value and in percent of the mesh size.

The information provides a list of:

- fully orphaned nodes
- weakly mapped nodes

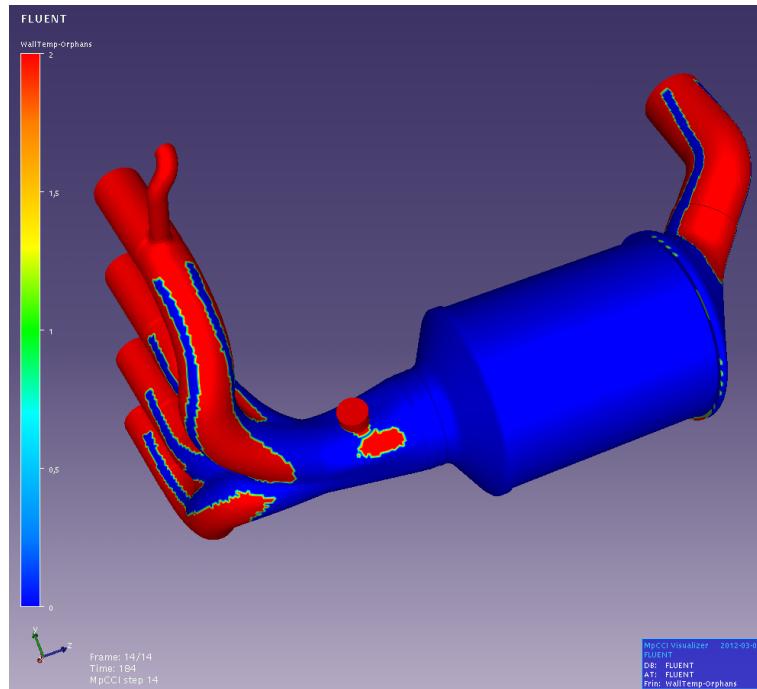


Figure 10: Orphaned nodes representation example

- 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 11](#)

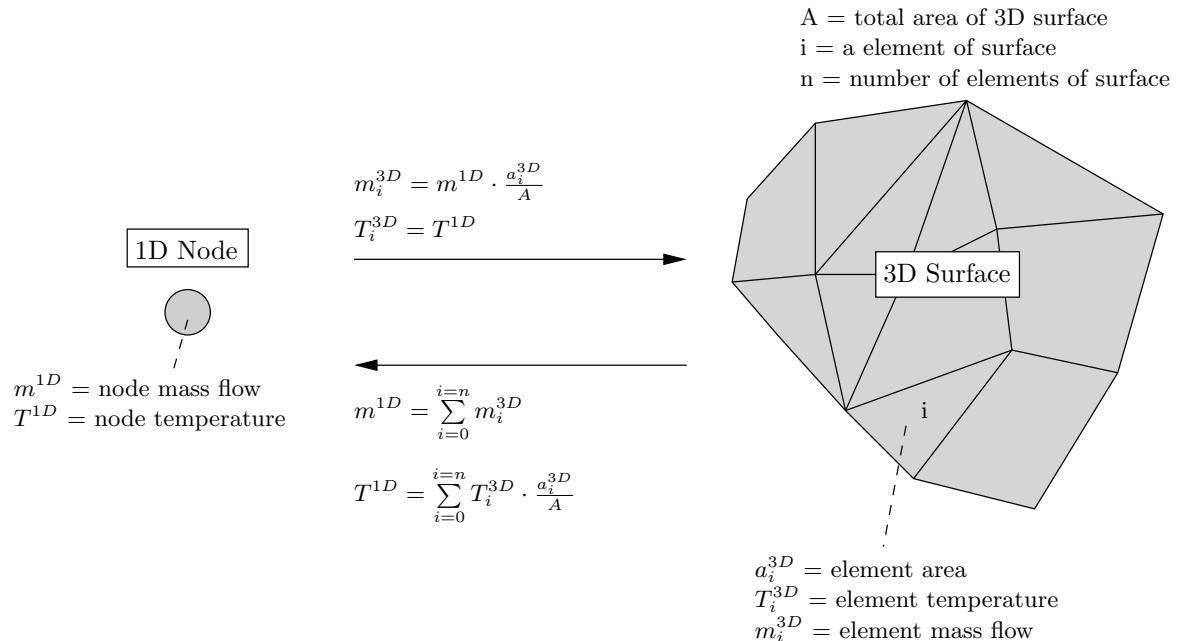


Figure 11: Exemplary calculation of flux quantity mass flow m and field quantity temperature T between 1D and 3D codes

3.3.3 Handling of Coupled Quantities

Each simulation code may work in its own unit system definition. In order to couple models created in different unit systems, MpCCI internally uses a 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 be also provided.

Additionally to the different unit systems different 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 ramped.

3.3.3.1 Quantity Relaxation

Relaxation of quantities is a method to stabilize and improve the convergence of the coupled process for some cases.

MpCCI offers the capability to relax a quantity on the receiver side by assigning a relaxation operator (see [4.5.7.4 Applying Operators to Mesh Based Components](#)). The offered relaxation methods are described in what follows.

Fixed Relaxation

The relaxation factor α – which is provided by the user – is used by MpCCI for an under- or over-relaxation of the quantity value which is sent to the receiver:

$$Q_{\text{relax}}^{n+1} = \alpha \cdot Q_{\text{new}}^{n+1} + (1 - \alpha) \cdot Q_{\text{relax}}^n$$

where

- α is the relaxation factor provided in MpCCI GUI.
- Q_{new}^{n+1} is the current mapped value (unrelaxed).
- Q_{relax}^{n+1} is the relaxed value sent to the receiver.
- Q_{relax}^n is the last relaxed value sent to the receiver.

Relaxation is available for steady state or iteratively coupled transient simulations. Under-relaxation with a value smaller than one might lead to a more stable solution for some problems, e.g. with large quantity spikes. Some steady-state couplings might benefit from an over-relaxation 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.

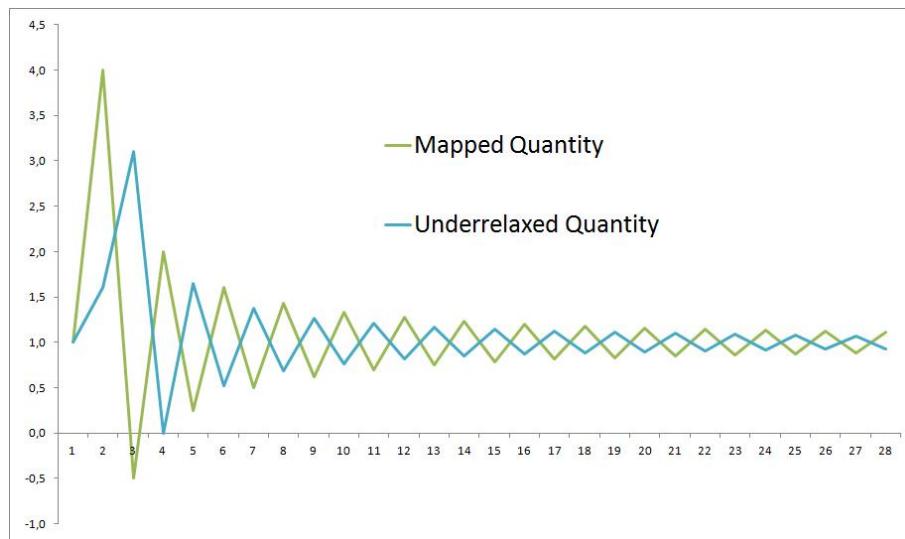


Figure 12: Under Relaxation in case of oscillating quantity values

The fixed relaxation option for the quantity is activated by selecting the operator Relaxation and choosing the method Fixed via the panel shown in Figure 14.

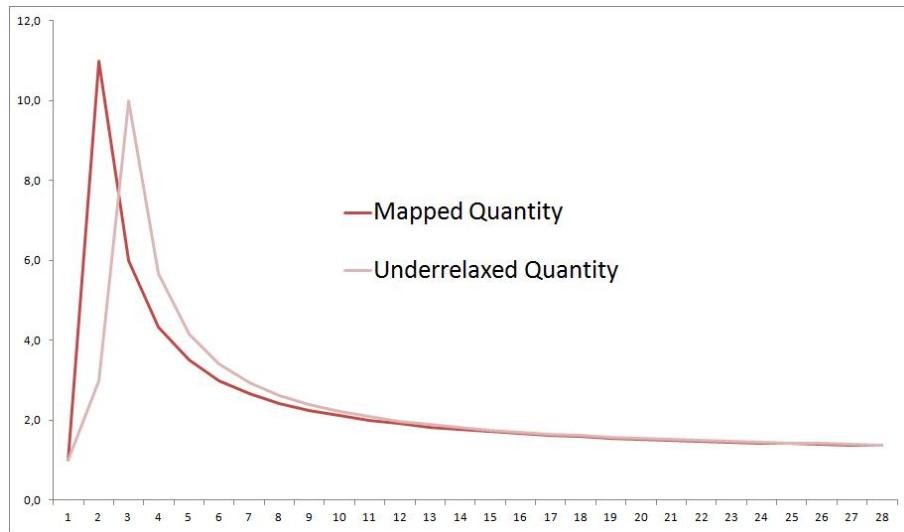


Figure 13: Under relaxation in case of an initial spike

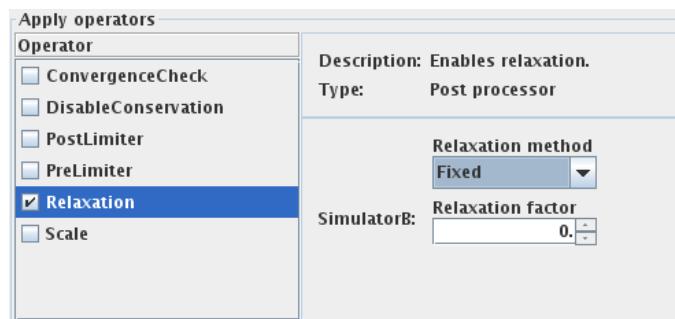


Figure 14: Relaxation quantity setting

Ramping

A special kind of relaxation is called **ramping** in MpCCI. There, the relaxation factor α is linearly increasing or decreasing over the iteration counter until a value of 1, i. e. no relaxation, is reached.

This is available for under-relaxation (starting with a value smaller than 1 and increasing to 1) or over-relaxation (starting with a value bigger than 1 and decreasing to 1).

The **ramping factor** β is computed according to the linear ramping function:

$$\begin{aligned}\beta &= \min(1, \beta_0 + qtid \cdot \beta_d) && \text{for under-relaxation ramping} \\ \beta &= \max(1, \beta_0 - qtid \cdot \beta_d) && \text{for over-relaxation ramping}\end{aligned}$$

where

- β is the ramping factor.
- β_0 provides the initial relaxation value and refers to **Initial ramp value**. For under-relaxation, $\beta_0 \in [0, 1]$,

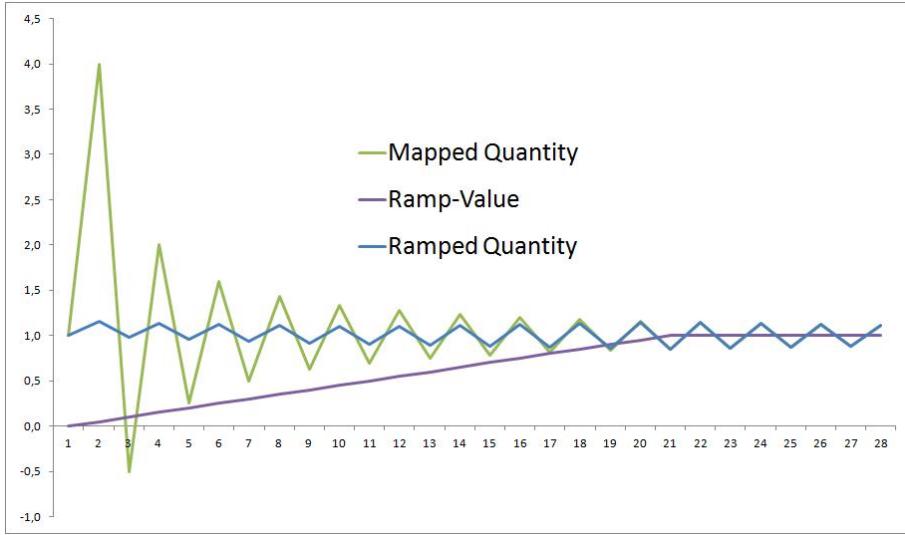


Figure 15: Ramping effect in case of oscillating quantity values

for over-relaxation, $\beta_0 \in [1, 2]$.

- β_d provides the ramping step value and refers to **ramp delta**, $\beta_d \in [0.001, 1]$.
- $qtid$ is an incremental counter identifying the data exchange.

If ramping is activated, the changes of the received quantity between two iterations or time steps are multiplied by the **ramp value**.

$$\begin{aligned} Q_{\text{relax}}^{n+1} &= Q_{\text{relax}}^n + \beta \cdot (Q_{\text{new}}^{n+1} - Q_{\text{relax}}^n) \\ &= \beta \cdot Q_{\text{new}}^{n+1} + (1 - \beta) \cdot Q_{\text{relax}}^n \end{aligned}$$

which in fact is identical with a classical under- (or over-)relaxation, except that the relaxation factor is increasing (decreasing for over-relaxation) with each new coupling step until the final value 1, i. e. no relaxation, is reached.

Ramping is available for all quantities and for all types of coupling algorithms. In the iterative transient coupling scenario the ramping value is again re-initialized to its start value β_0 at the first iteration at the same time level and then increased during the iterations - at the same time level.

Ramping might help to avoid the exchange of any unrealistic spikes during the first few iterations (see Figure 16).

Ramping is activated by applying the Relaxation operator and choosing the Ramping method (see Figure 17). Depending on the **Initial ramp value** an increasing ramping (starting with an under-relaxation for an **Initial ramp value** smaller than 1) or a decreasing ramping (starting with over-relaxation for an **Initial ramp value** bigger than 1) is used.

A ramping is recommended for forces or pressure to avoid initial pressure spikes.

When the option **relax** is activated in the Monitor settings in the Edit Step of the MpCCI GUI, the used relaxation factor β is plotted in the MpCCI Visualizer.

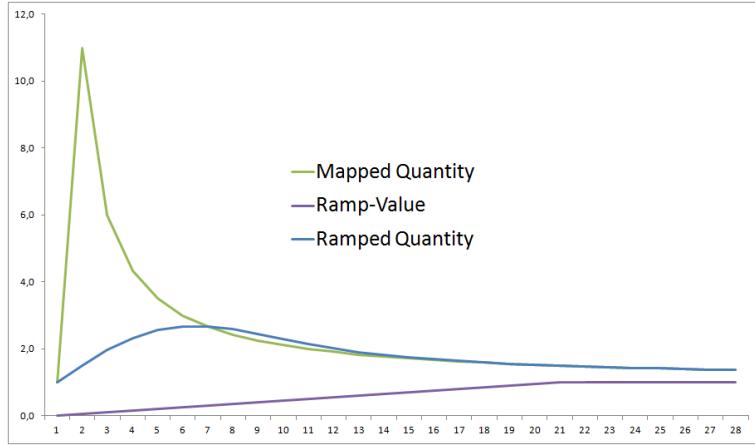


Figure 16: Ramping effect in case of an initial spike

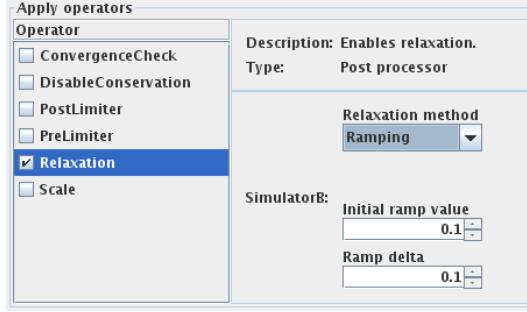


Figure 17: Ramping quantity setting

Aitken Relaxation

MpCCI offers the possibility to use an adaptively calculated relaxation factor. Aitken's method is used to get the optimal relaxation factor; the unrelaxed and relaxed quantity values from the current and the last step are needed:

$$\Delta Q^{k+1} = Q_{\text{relax}}^k - Q_{\text{new}}^{k+1} \quad \Delta Q^k = Q_{\text{relax}}^{(k-1)} - Q_{\text{new}}^k$$

This can be used to get the current Aitken factor μ^k for $k > 0$:

$$\mu^k = \mu^{(k-1)} + (\mu^{(k-1)} - 1) \frac{(\Delta Q^k - \Delta Q^{k+1})^T \Delta Q^{k+1}}{(\Delta Q^k - \Delta Q^{k+1})^2}$$

For the first step $k = 0$ the Aitken factor is set to 0.

The relaxation factor is then defined as

$$\omega^k = 1 - \mu^k$$

and used the following way:

$$Q_{\text{relax}}^{k+1} = \omega^k \cdot Q_{\text{new}}^{k+1} + (1 - \omega^k) \cdot Q_{\text{relax}}^k$$

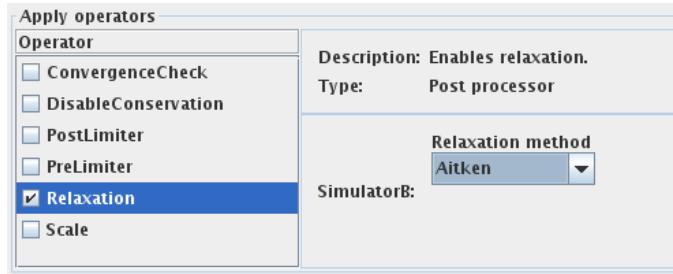


Figure 18: Adaptive relaxation quantity setting

To use adaptive relaxation Aitken has to be chosen as method for the Relaxation operator (see [Figure 18](#)). The calculated optimal relaxation factor ω 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.

When the option `relax` is activated in the Monitor settings in the Edit Step of the MpCCI GUI, the used relaxation factor ω^k is plotted in the MpCCI Visualizer.

Quasi-Newton Methods

The Quasi-Newton methods can strongly accelerate and stabilize the convergence of the mapped quantities, cf. [Koch \[2016\]](#). They can be applied to steady and iterative transient couplings.

In a multiphysical coupling, the goal is to achieve an equilibrium between both coupled codes

$$\begin{aligned}\mathcal{A} : Q_B &\mapsto Q_A, \\ \mathcal{B} : Q_A &\mapsto Q_B.\end{aligned}$$

within each time step respectively at the end of the static simulation.

Using the serial coupling algorithms, described in [▷ 3.4.3 Transient Problems ◁](#), this is equivalent to solving a fixed point problem

$$\begin{aligned}H(Q_A) &= \mathcal{A}(\mathcal{B}(Q_A)) = Q_A, \text{ or} \\ H(Q_B) &= \mathcal{B}(\mathcal{A}(Q_B)) = Q_B.\end{aligned}$$

The Quasi-Newton methods compute iteratively the fixed point Q_* of H , i. e. $H(Q_*) = Q_*$, by reformulating the equation into a root problem of a nonlinear function $R = H - \text{Id}$:

$$R(Q) = H(Q) - Q = 0$$

The general formula for the relaxed quantity sent to the receiver is given by the classical Newton iteration applied to $R(Q) = 0$:

$$\begin{aligned}Q_{\text{relax}}^{n+1} &= Q_{\text{relax}}^n - J^n(R(Q_{\text{relax}}^n)) \\ &= Q_{\text{relax}}^n - J^n(H(Q_{\text{relax}}^n) - Q_{\text{relax}}^n), \\ &= Q_{\text{relax}}^n - J^n(Q_{\text{new}}^{n+1} - Q_{\text{relax}}^n),\end{aligned}$$

where J^n is an approximation of the inverse Jacobian of $R = H - \text{Id}$.

To construct the approximate inverse Jacobian matrix, the mapped (Q_{new}) and relaxed (Q_{relax}) quantities from previous Quasi-Newton iterations are used.

The concrete computation of J^n differs for each Quasi-Newton method (see [Koch \[2016\]](#) and its references):

- The Broyden method uses the update formula

$$J^n = J^{n-1} + \frac{J^{n-1} R^n}{\|R^n - R^{n-1}\|_2^2} (R^n - R^{n-1})^T,$$

where $R^n = Q_{\text{new}}^n - Q_{\text{relax}}^n$. At the start of a new time step, the initial guess J^0 is set to the last computed Jacobian approximation from the previous time step. In the very first time step a scaled unit matrix $J^0 = -\omega I$, $\omega > 0$ is taken as initial guess. This approach computes an approximation, which minimizes $\|J^n - J^{n-1}\|$, under the constraint $J^n(R^n - R^{n-1}) = Q_{\text{relax}}^n - Q_{\text{relax}}^{n-1}$.

- The Jacobi method and Anderson Mixing employ the update formula

$$J^n = J^0 + (W - J^0 V)(V^T V)^{-1} V^T,$$

with the difference matrices

$$\begin{aligned} W &= [Q_{\text{relax}}^1 - Q_{\text{relax}}^0, \dots, Q_{\text{relax}}^n - Q_{\text{relax}}^{n-1}], \\ V &= [R^1 - R^0, \dots, R^n - R^{n-1}]. \end{aligned}$$

This approximation minimizes $\|J^n - J^0\|$, under the constraint $J^n V = W$. Anderson Mixing sets the initial guess in each time step to a scaled unit matrix, i.e. $J^0 = -\omega I$, $\omega > 0$. The Jacobi method reuses information from earlier time steps, by using the last computed approximation in the previous time step as initial matrix J^0 .

These methods build the standard versions of Jacobi and Anderson Mixing.

- Anderson Mixing and Jacobi are capable to solve conceptually the nonlinear “inverse” system $Id - H^{-1} = 0$. In this case, the new quantity sent to the receiver is computed by

$$Q_{\text{relax}}^{n+1} = Q_{\text{new}}^n - J^n(Q_{\text{new}}^n - Q_{\text{relax}}^n).$$

The approximation J^n of the Jacobian of $Id - H^{-1}$ is computed in nearly the same manner as above. The only difference is the matrix

$$W = [Q_{\text{new}}^1 - Q_{\text{new}}^0, \dots, Q_{\text{new}}^n - Q_{\text{new}}^{n-1}]$$

is built by the unrelaxed quantity values; the matrix V stays the same.

The LeastSquares method is a special case of the inverse Anderson Mixing where $\omega = 0$.

Anderson Mixing and the Jacobi method have high memory and computational cost, which depends quadratically on the degrees of freedom of the quantity. In exchange, both methods have good convergence behaviour. The complexity of the Anderson Mixing is linear in the quantity’s degrees of freedom, but usually converges slower than the other two methods. Thus, for a not too large number of degrees of freedom, i.e. low tens of thousands, the Jacobi method is the best choice. For higher discretized interfaces, it is advisable to use the LeastSquares type of the Anderson Mixing method.

- (!) For most fluid-structure interactions the inverse approach converges faster than the standard approach.
It is recommended to use Anderson Mixing or Jacobi prior to Broyden.
- (!) The Jacobi Quasi-Newton method transfers the Jacobian from one time step to the next as initial guess. This usually accelerates the convergence, but in some cases this can lead to divergence. In this case try Anderson Mixing.

The Quasi-Newton methods for the quantity are activated by choosing the method Quasi-Newton for the Relaxation operator (see [Figure 19](#)). After that the desired Quasi-Newton method can be chosen and configured (give the value of ω and choose between standard or inverse version) as described in [▷ 4.5.7.4 Applying Operators to Mesh Based Components ↲](#).

- (!) The methods can be applied to only one quantity, either Q_A or Q_B .

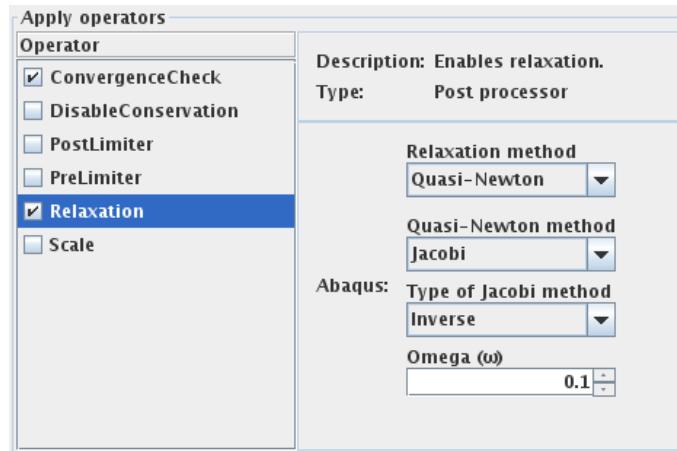


Figure 19: Quasi-Newton quantity setting

! As the Quasi-Newton methods are implemented only for serial coupling schemes, the Initial quantities transfer of the coupled codes have to be chosen as a combination of receive and exchange or as a combination of receive and skip.

If the Quasi-Newton method is applied to steady problems, a maximal column number of V and W are given via the parameter `MaxIterQuasiNewton` in the Edit Step Job section. When the iteration counter reaches this number, the column corresponding to the ancientest quantity value is replaced.

The divergence of the Quasi-Newton methods is checked by the condition $\|R^{n+1}\|_\infty > 10 \cdot \|R^n\|_\infty$. In this case, the methods are restarted, i. e. the matrices are built up anew starting from the current iterate. If the parameter `ResidualsQuasiNewton` (Edit Step Job section) is chosen as $X \geq 2$, the divergence is additionally checked by considering the last X residuals' trend. If its slope is positive, a restart is triggered. In steady simulations, a choice of `ResidualsQuasiNewton = 0` leads to an automatic restart when `MaxIterQuasiNewton` has been reached; if no restart is desired, choose `ResidualsQuasiNewton = 1`.

An example using Quasi-Newton methods is [VII-3 Driven Cavity](#).

3.3.3.2 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.

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.

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

Motion type	Application area
Single/Multiple Reference Frame	Steady State approximations
Mixing Plane or Frozen Rotor	Weak rotor-stator interation
	Initial solution for unsteady flow
Sliding Mesh Model	Unsteady flow
Nonlinear Harmonic Method	Strong rotor-stator interaction
	Vortex shedding

- Solid mechanics physical domain

Motion type	Application area
Rotating reference frame	Stress analysis induced from rotation and flow force Operational vibrations
Stationary reference frame	Unsteady flow Strong rotor-stator interaction Vortex shedding

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 MeshMotion and MRFCorrect features in the Edit Step Job section of MpCCI GUI ([▷ 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

Motion Properties	Use MeshMotion	use MRFCorrect
Identical	NO	NO
Different	YES	NO

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

Motion Properties	Use MeshMotion	use MRFCorrect
Different	YES	NO

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

Motion Properties	Use MeshMotion	use MRFCorrect
Identical	NO	NO

List of codes implementing the mesh motion definition

Abaqus Model defining a rotating reference frame and stationary reference frame are supported.

- MpCCI recognized the keyword *dload with the option CENT or CENTRIF as a rotating reference frame.

(!) The keyword CENT needs some additional hand-on, 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 Edit Step. Before adding the components into the list of coupled components you should provide the CENT using the following syntax by using the edit with the right mouse click ([▷ 4.5.2.2 Editing Component Properties ◁](#)):

```
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 recognized 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.

3.3.3.3 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 20](#).

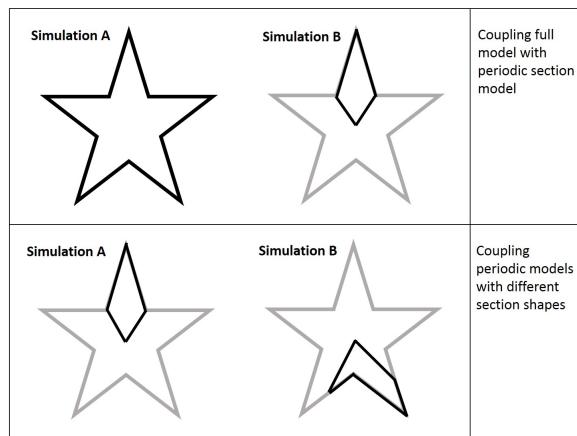


Figure 20: Periodicity panel

The necessary information is inputted into the GUI, cf. [4.5.2.3 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.

3.3.3.4 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. [4.5.7 Quantity Specifications](#)).

The following table shows the different operators available at different stages where the sequence numbers describe the workflow for a quantity:

Sequence No.	Stage	Operators	Comment
1	Receive operation		MpCCI receives the data.
2	Pre operation	ConvergenceCheck PreLimiter	Selectable operators.
3	Mapping operation		Automatic mapping process controlled by MpCCI.
4	Orphans operation	Orphans settings	You can select the <code>orphans</code> treatment.
5	Adjust operation	DisableConservation	You can disable conservative mapping correction.
6	Post operation	PostLimiter Scale	You can define their rank when applying these operators.
7	Relaxation operation	Relaxation	You can select the operator and specify the relaxation method.
8	Send operation		MpCCI sends the data.

3.4 Coupling Algorithms

3.4.1 Course of the Coupling Process

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 or 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.

During the iteration, which also consists of several time steps in a transient problem, 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. The data is also not associated with certain time steps or iterations.

A consequence of this is also that the possible coupling algorithms are mainly determined by the capabilities of the simulation codes and the corresponding code adapters. The code adapters call send- and/or receive-functions. These calls can appear at different states of the computation:

- At the beginning of each time step, i. e. before the iteration of the time step.
- At the end of each time step, i. e. after the iteration.
- Before or after an iteration step.
- On direct demand of the user.

Most codes only offer exchanges at the beginning or end of a time step, which limits the choice of coupling algorithms. Exchange on direct demand usually is only used in stationary computations. Most simulation codes do not support exchanges during iterations.

Please check the [Codes Manual](#) for information on the supported exchange times of the codes.

MpCCI supports weak coupling. Strong 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.

In MpCCI the selection of a coupling algorithm is based on the selection of the initial exchange. There are four choices, which are defined in the Go Step of the MpCCI GUI (see [4.8 Go Step](#)): **exchange**, **receive**, **send** and **skip**. The setting of this option defines what happens if the exchange routine is called for the first time. All further calls of the exchange routing will yield a complete exchange, i. e. sending and receiving of data.

- (!) 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.
- If one code is receiving data, it waits until data is available from the other code if its coupling configuration for the receiving mode was set to wait.

3.4.2 Stationary Problems

For stationary problems, it is assumed that there is exactly one solution of the coupled problem, which shall be found. The coupling algorithm does not have a big influence on the solution in this case.

3.4.3 Transient Problems

Two general cases can be distinguished: Either both codes send and receive data, or one code only sends while the other receives only.

In any case it is important to know whether each code performs the data exchange at the beginning of the time step before the iteration, which we call “exchange before solution”, or at the end of a time step after the iteration, “exchange after solution”.

3.4.3.1 Construction of Coupling Algorithms

Coupling algorithms can be constructed in a straight-forward way. The three basic steps are shown in [Figure 21](#). It is important to check first, whether the codes exchange before or after solution. In this example, code A exchanges after the solution, code B before the solution.

The construction of the algorithm works as follows:

1. Draw time lines for both codes, including boxes which indicate the data exchanges. For code A, the box is at the end of the steps, for code B always at the beginning. The first box in each code represents the initial transfer, which can be chosen in the MpCCI GUI. We assume, that for A exchange (marked by “x”) and for B receive (marked by “r”) were selected. All further exchanges *always* get an “x” as data is received and sent in all further steps.
2. Now, the data transfers are constructed, we proceed from the left to the right. Find the first sending operation in code A, which is found in the first box ($x = \text{send and receive}$). Thus a transfer line starts here. Now find the first receiving operation in code B, which is also the first box. This is the end of the line, which yields the first transfer.

Now we proceed with the next sending operation, which is found in the second box in code B, which is connected to the first box of code A.

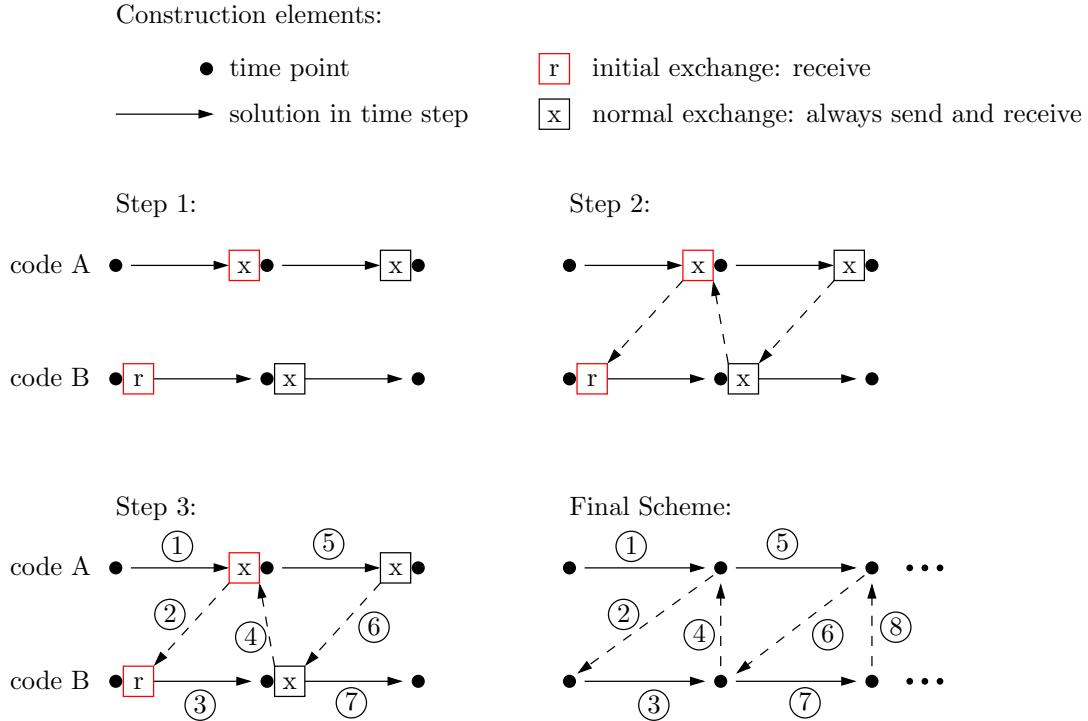


Figure 21: Construction of a coupling algorithm: Code A exchanges after solution, while code B exchanges before solution.

In this way all arrows can be found.

3. The only thing missing now is the order of operations. For this we remember:

- “receive” always waits for data
- “send” never waits (data is buffered)
- exchanges “x” can be carried through in any order: Send first or receive first, as necessary.

In our example, this means code A can start the computation (1), as it does not need to wait, while code B receives data, thus it will wait. After code A has finished computing the first time step, it can send data (2) and then waits to receive data. Now, code B got its data and can start computing (3), then sends data to code A (4), which can then proceed (5), sends data (6) and code B continues (7).

In this way we obtain the final algorithm depicted in [Figure 21](#), which can be identified as a sequential coupling algorithm. Note that not all combinations of the initial transfer yield reasonable coupling algorithms.

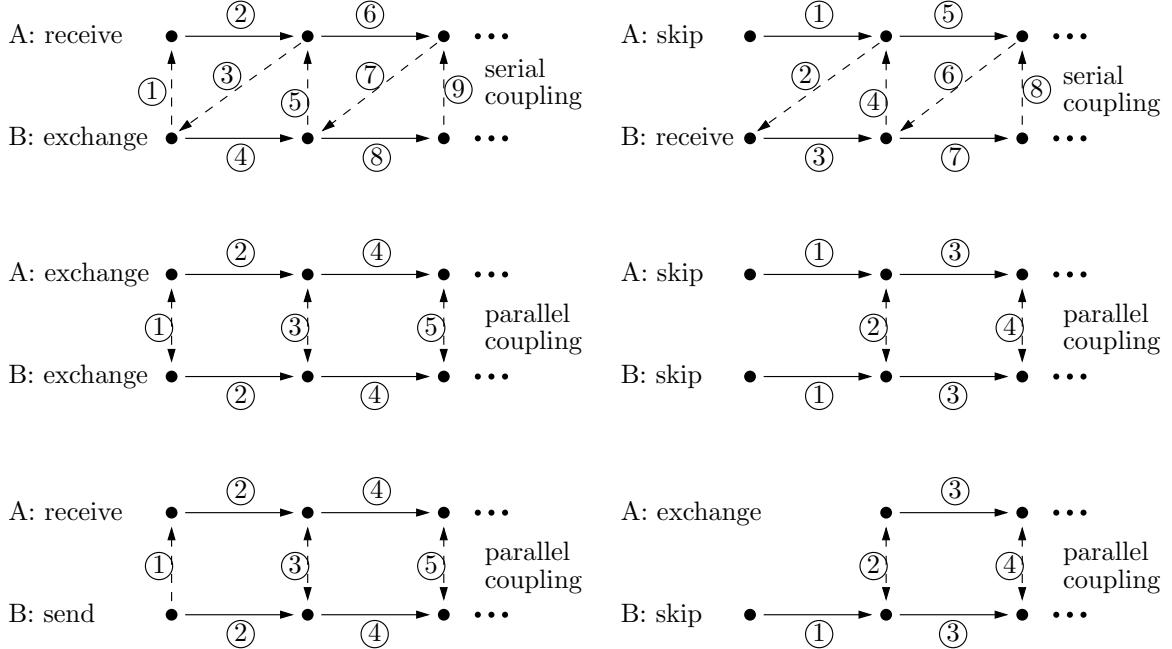
3.4.3.2 Standard Case: Bidirectional Transfer

In most coupled simulations, data is transferred between the codes in both directions. The coupling algorithms depend on the order of solution and transfer and the selected initial transfer.

There is only a limited number of reasonable settings. Three cases can be regarded:

- Both codes exchange before iteration. The possible combinations are sketched in [Figure 22 a\)](#).

a) Both codes exchange before iteration



b) Both codes exchange after iteration

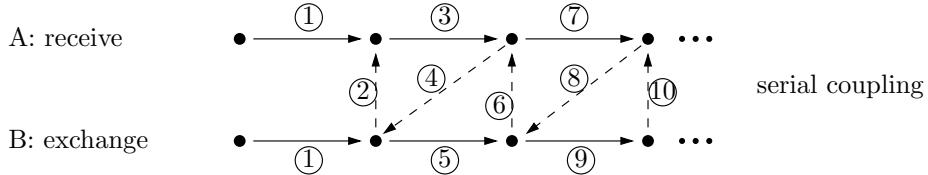


Figure 22: Coupling algorithms for different combinations of Initial quantities transfer: The difference between cases a) and b) is only the additional step of both codes before the initial exchange in case b).

- b) Both codes exchange after the iteration. The only difference to the previous case is that no data is transferred immediately before the first solution step. Otherwise the algorithms remain the same. One example is shown in [Figure 22 b\)](#).
- c) One code exchanges after the iteration, and one before the iteration, which yields a wide variety of algorithms. However, only few can actually be recommended for use. To add further algorithms, one can allow a “dummy step” in the code with exchange after solution, i.e. the “clocks” of both codes are not synchronous. One code is always one time step ahead. Possible algorithms are depicted in [Figure 23](#).

c) Code A exchanges after iteration and code B before

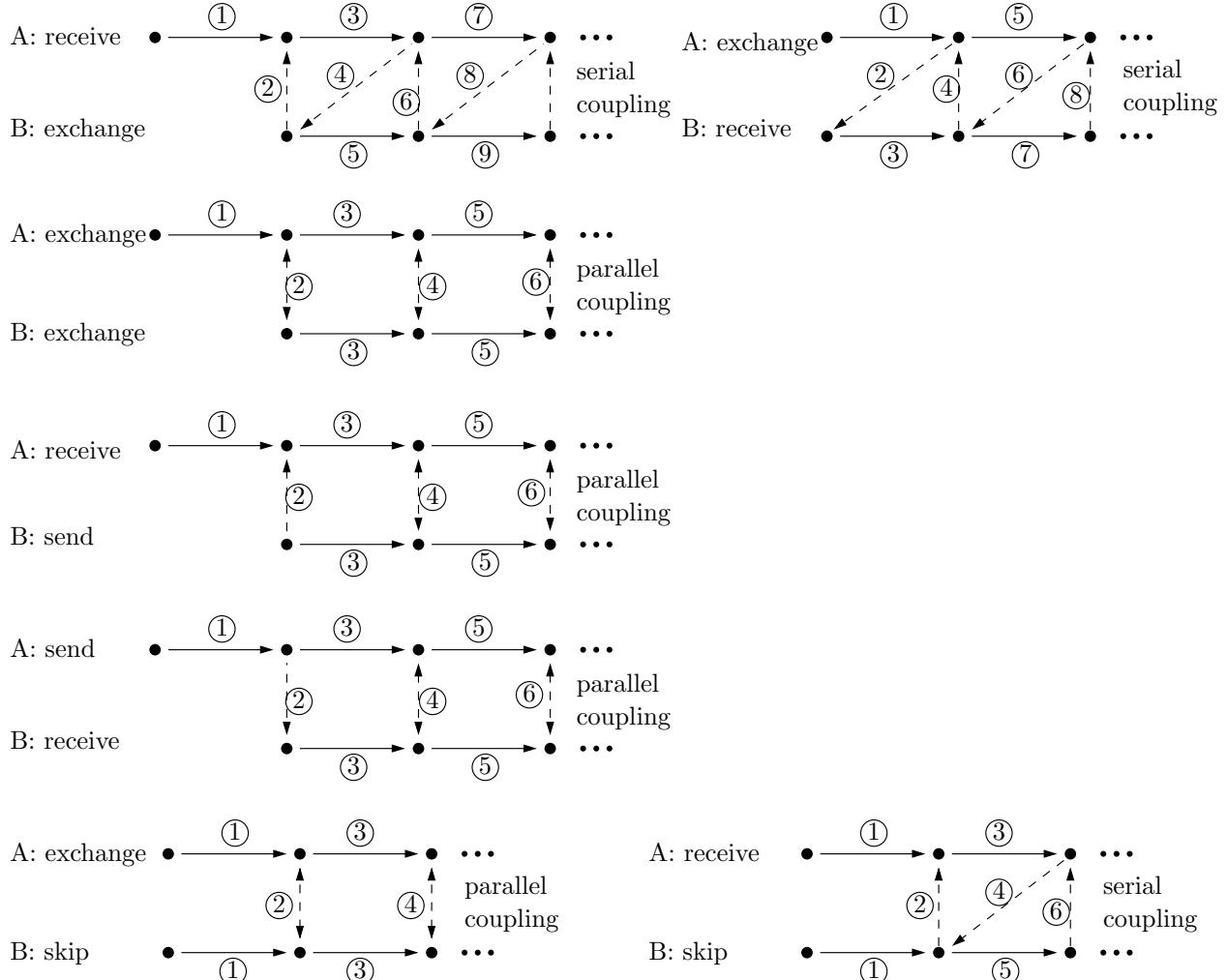


Figure 23: Coupling algorithms for code A with exchange after solution and code B with exchange before solution.

3.4.3.3 Special Case: Unidirectional Transfer

In some coupling applications, only unidirectional transfer is used, i.e. one code only sends data to the other code, which only receives. This reduces the possible coupling algorithms, there is only one basic coupling algorithm which should be used if possible. If A is the sending code and B the receiving code, this algorithm can be described by the algorithm in [Figure 24](#).

Depending on the coupling properties of the codes, the user should select the following settings:

Exchange before/after solution	Initial Exchange
1 A: before, B: before	→ A: skip B: receive
2 A: before, B: after	→ A: skip B: skip with dummy step
3 A: after, B: before	→ A: send B: receive
4 A: after, B: after	→ A: skip B: receive

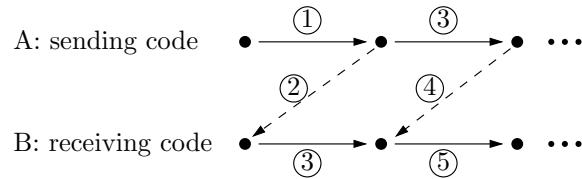


Figure 24: Coupling algorithm for unidirectional transfer

In case 2, the coupling algorithm can only be realized with a dummy step in code B. The best combinations are 3 and 4.

Additionally to the setting of the `Initial quantities transfer` you can set the `Send mode` option from the `Coupling configuration`. The `Send mode` provides you the capability to define the communication scheme:

- **asynchronous** communication: in that case use the mode `always`. The code sends data without taking care of the partner codes.
- **synchronous** communication: in that case use the mode `onewait` or `allwait`. The code sends the data if one or all partner codes are ready to receive the data.

3.4.3.4 Special Case: Coupling with Mixed Solution 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 *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 states solution and in the solid the fields evolve in a fully transient manner.

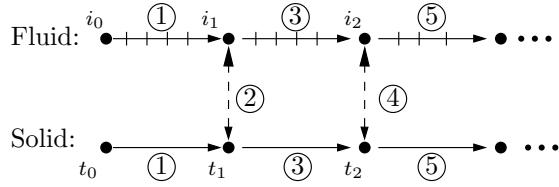
As in [Figure 25](#) explains, the exchange of temperature (`Temperature` or `WallTemp`) and the heat flux (`WallHeatFlux`) is performed after the solid time step and a fluid steady state step. The fluid side may optionally perform some subcycling iterations before transmitting the heat fluxes.

- *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 states solution whereas in the fluid the Navier-Stockes equations evolve in fully transient manner.

As the electromagnetic phenomena are faster solved compare to the gas dynamic phenomena you may employ an asynchronous coupling scheme. The electromagnetic code may update its boundary conditions with the available temperature results from the fluid code. To achieve this coupling procedure you should select from the `Receive mode` option from [4.8.2 Coupling Configuration Parameters](#) one of the following option:

Parallel coupling scheme:



Serial coupling scheme:

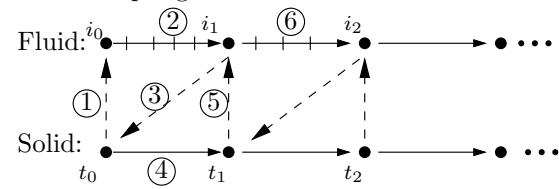


Figure 25: Quasi-transient fluid solid coupling algorithm

- `available` gets what is available from the server without waiting.
- `any` gets all data if at least one quantity is available. Otherwise the receive action is skipped.
- `complete` gets data only if all quantities are available. Otherwise the receive action is skipped.

With one of the asynchronous receive option you may achieve following coupling flow as shown in [Figure 26](#) by using the `available`.

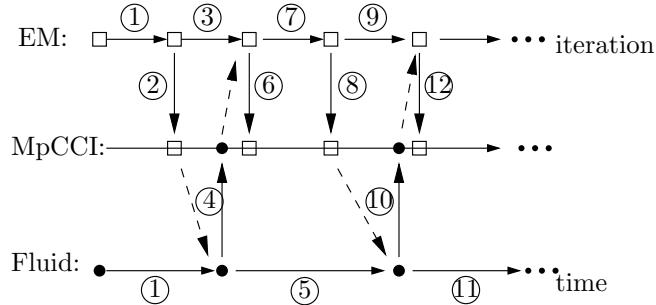


Figure 26: Coupling algorithm with asynchronous receive mode

1. The electromagnetic code (EM) sends its data (2) to MpCCI server and skips the receive of data because nothing is available from the fluid code. The EM continues to iterate (3).
2. The fluid code sends its data (4) and receives the data from state (2) and process to the next time step (5).
3. The EM reaches its next data exchange and sends at (6) the data. The data receive action can be performed by using the data from(4). The EM updates its coupled interface values and continues to iterate (7).
4. At (8) the EM sends its data and receives nothing. The iteration continues (9).
5. The fluid sends its data at (10), gets the data from state (8).
6. The EM reaches its next data exchange, sends at (12) the data and updates its boundaries with state (10).

For the MpCCI procedure it leads to select two problems from different solution types (transient and steady state). By selecting such model you will be warned by the MpCCI GUI that you are currently mixing the solution types ([Figure 27](#)). You can process to the next step and setup the coupled quantities as usual.

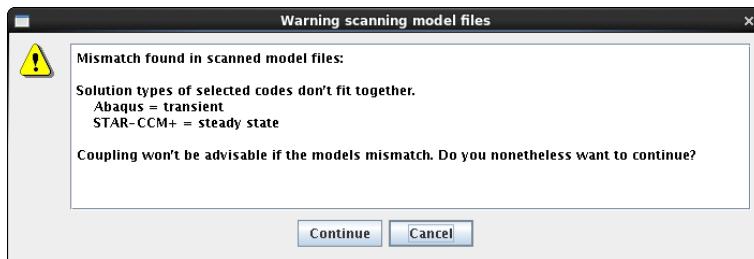


Figure 27: Coupling with mixed solution types

For each code you need to select the corresponding coupling scheme for each solution type: Explicit-Transient, Explicit-SteadyState (see [▷ 4.8.2 Coupling Configuration Parameters](#)).

3.4.4 Iterative Coupling

3.4.4.1 Applications

Transient coupling algorithms that were introduced in [▷ 3.4.3 Transient Problems](#) exchange data once during each time step. For some problems this explicit transient coupling results in severe stability problems. Using iterative transient coupling schemes it is possible to reach a stable solution with moderate time step sizes for these problems.

Iterative 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 iterative coupling schemes because of bigger possible time step sizes.

3.4.4.2 Iterative Coupling Algorithms

The construction of iterative coupling algorithms follows the pattern of the construction of explicit transient coupling schemes.

Depending on the **Initial quantities transfer** settings two different coupling schemes are possible: a Gauss-Seidel and a Jacobi algorithm corresponding to the usual serial and parallel coupling schemes which are described in [▷ 3.4.3.1 Construction of Coupling Algorithms](#).

[Figure 28](#) and [Figure 29](#) show the two coupling algorithms. During the Gauss-Seidel algorithm – e. g. with the **Initial quantities transfer** options “exchange” and “receive” – one code leads the inner iterations and then sends data back to the previous iteration of the other code. When a Jacobi algorithm (**Initial quantities transfer** settings “send” and “receive” or “exchange” and “exchange”) is used both codes advance to the next inner iteration in parallel.

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 inner iterations for each time step are computed in a serial or in a parallel way – depending on the **Initial quantities transfer** options.

For some applications – especially fluid-structure interactions with incompressible fluids and moving walls – the Gauss-Seidel algorithm should produce a more stable solution.

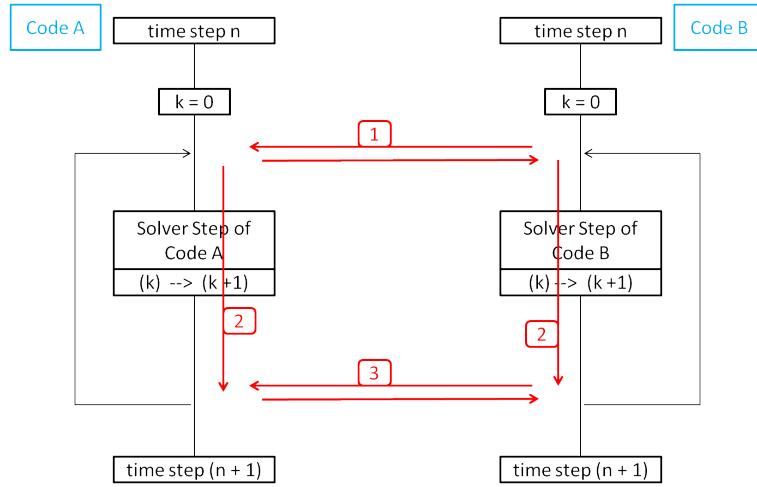


Figure 28: Jacobi algorithm for iterative coupling with initial “Exchange”, “Exchange” settings.

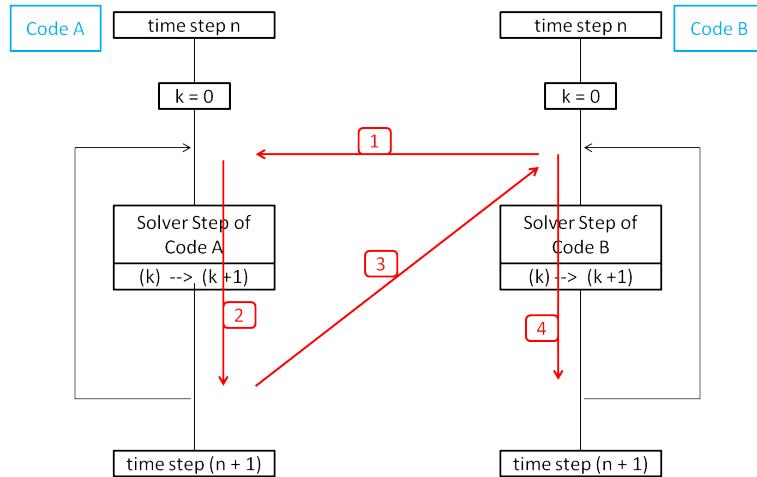


Figure 29: Gauss-Seidel algorithm for iterative coupling with initial “Exchange”, “Receive” settings

3.4.4.3 Convergence Control for Inner Iterations

Since the inner iterations are computationally expensive, the possibility to stop the inner iterations adaptively is essential to keep computation times as low as possible. When the coupled quantity values remain more or less constant between consecutive data exchanges, the coupled solution cannot be improved by continuing the inner iterations; the simulation can proceed to the next time step.

MpCCI monitors the relative difference between two successively coupled quantities. If q^n and q^{n+1} are the quantity values for the inner iterations n and $n + 1$ respectively, the relative difference in the maximum norm – bounded by 0 and 2 – is computed as

$$0 \leq 2 \frac{\|q^n - q^{n+1}\|_\infty}{\|q^n\|_\infty + \|q^{n+1}\|_\infty} \leq 2.$$

This difference is calculated for every iteratively coupled quantity and can be plotted during the simulation

in the MpCCI Monitor. It is possible to use this relative difference to define an adaptive stopping criterion for the inner iterations (cf. the next section).

- (!) It is not recommended to use the quantity NPosition for the definition of a convergence criterion. In the case of the coupled quantity NPosition the relative maximum norm is calculated for the displacements, not for the position of the coordinates. If the inner iterations are converged, the displacements for two consecutive iterations are very small (and quite close to each other). This leads to a relatively big value of the calculated norm. Therefore, only the quantity sent by the CFD code, e.g. RelWallForce, should be used to judge the convergence of the inner loop.

3.4.4.4 Settings for Iterative Coupling in the MpCCI GUI and Supported Codes

Currently, iterative coupling is supported for FLUENT, MATLAB, MSC NASTRAN, OpenFOAM and Abaqus. Usage of iterative coupling algorithms is quite simple: in the Go Step the coupling scheme can be set to “Implicit-Transient”. This adds the “Iteration control” panel to the Go Step.

If the Initial quantities transfer settings are set to a parallel coupling method (e.g. “send” and “receive”, or “exchange” and “exchange”), this will result in a Jacobi-type iterative coupling algorithm. Using the Initial quantities transfer settings “receive” and “exchange” results in a serial Gauss-Seidel-type coupling method.

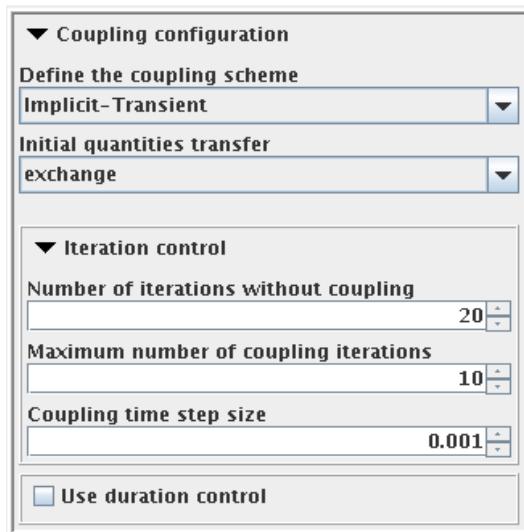


Figure 30: Go Step for iterative coupling example

Expand “Iteration control” gives access to all information relevant to iterative coupling: The “Coupling time step size” has to be set for every code. This ensures that both codes use the same time step. If the simulation models contain another time step value, this value will be patched to match the “Coupling time step size” at the start of the coupled simulation.

Furthermore, a “Maximum number of coupling iterations” can be specified for each code. If no adaptive stopping criterion is used or the convergence tolerance value is not reached, the inner iteration loop will be stopped when this iteration number is reached.

The “Number of iterations without coupling” can be specified for MSC NASTRAN and FLUENT. Both codes will compute this number of iterations between two data exchanges. OpenFOAM and Abaqus always solve the complete time step for each inner iteration, so the number of iterations without coupling cannot

be set for these simulation codes. In [Figure 30](#) twenty iterations without coupling have been chosen, leading to a data exchange every twenty iterations of the MSC NASTRAN time step.

At quantity configuration in the Define Quantity Sets tab of the Coupling Step an adaptive stopping criterion for the inner iterations can be defined, cf. [Figure 31](#). For each coupled quantity the operator ConvergenceCheck can be applied and a tolerance value can be entered for the sending code: the inner iterations stop when the relative difference between two consecutive quantity values lies beneath this value. The relative difference between two consecutive quantity values is computed as described in [▷ 3.4.4.3 Convergence Control for Inner Iterations ◁](#).

If this convergence criterion for the inner iterations is not met, the inner cycle will stop when the maximum number of coupling iterations – defined in the Go Step – has been reached.

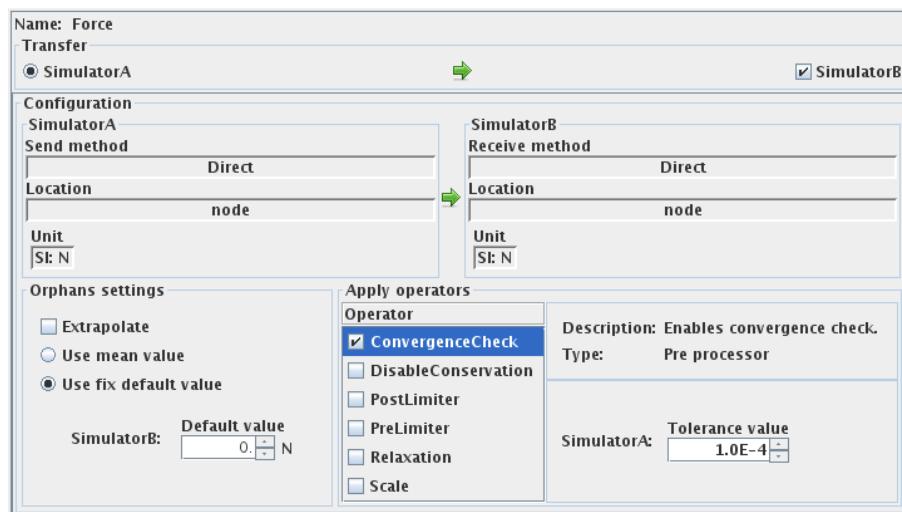


Figure 31: Convergence tolerance for inner iterations

The coupling configuration in [Figure 30](#) also shows the possibility to Use duration control. This is used as described in [▷ 3.4.6.1 Subcycling Setting in the MpCCI GUI ◁](#).

3.4.4.5 Settings for Iterative Coupling in Abaqus

For Abaqus no special settings need to be made when using iterative coupling.

3.4.4.6 Settings for Iterative Coupling in MSC NASTRAN

For MSC NASTRAN no special settings need to be made when using iterative coupling.

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 in the “Go Step”.

3.4.4.7 Settings for Iterative Coupling in FLUENT

If an implicitly coupled simulation is started with the FLUENT GUI, the settings from the MpCCI GUI – concerning the time step size and the number of inner iterations – are automatically imported into FLUENT.

After loading the **FLUENT** simulation the convergence check of the residual monitors is turned off. This ensures that all required inner iterations are computed.

The coupled simulation can be started either through the control panel **MpCCI run Implicit FSI** or by using the text interface.

The control panel for implicitly coupled simulations can be found under **Solve** and **MpCCI Run Implicit FSI**. [Figure 32](#) shows a screenshot of the panel. The panel allows to change the number of time steps, the reporting interval and the profile update interval. After clicking “Run” the coupled simulation starts.

In the text interface the simulation can be started with the command:

```
> (mpcci-solve n)
```

where n represents the number of time steps to be calculated.

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.

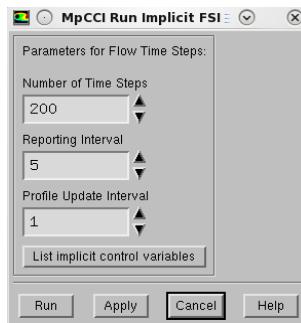


Figure 32: MpCCI Run Implicit FSI panel in FLUENT

3.4.5 Exchange of Time Step Size

Instead of using a fixed coupling time step size, it is also possible to use adaptive time stepping. 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” (see 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.

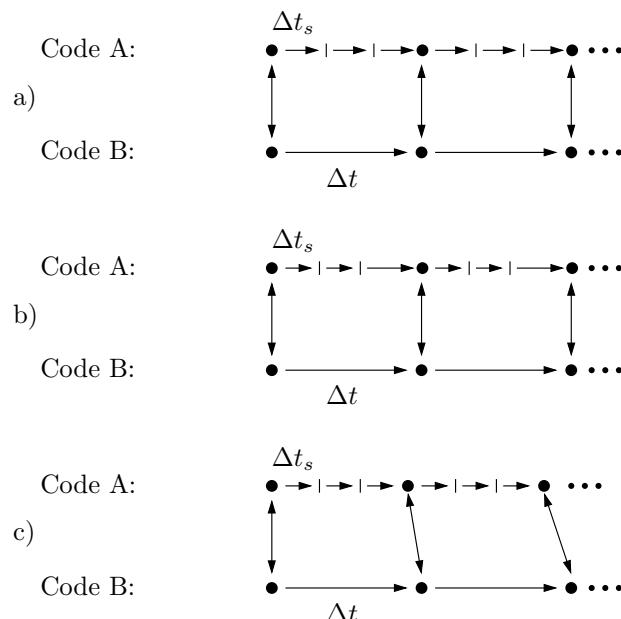
Convergence Problems: If the time step size is determined by one code it may be simply too large for the partner code. Please choose carefully which code shall determine the size and set reasonable limits.

! It is strongly recommended to always define a default time step size in each simulation code, because the value of “PhysicalTime” may not be defined at the beginning of a simulation! This depends on the initial exchange settings and how exactly each code evaluates the received value. The default value should be the same for both partner codes.

3.4.6 Subcycling

The idea of subcycling is the definition of extra iteration steps inside a coupling step, which is schematically depicted in [Figure 33](#). 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 33](#): Three examples for coupling algorithms with subcycling: a) constant subcycling steps Δt_s and exact coupling steps sizes of both codes, b) not necessarily constant subcycling steps Δt_s and exact coupling step sizes enforced, c) not necessarily constant subcycling steps Δt_s and exact coupling step sizes not enforced

A tutorial where subcycling is applied for a thermal simulation is [▷ VII-6 Exhaust Manifold ◁](#).

3.4.6.1 Subcycling Setting in the MpCCI GUI

MpCCI GUI offers the possibility to configure the subcycling for each solver. The setting is available under the **Coupling configuration** option at the Go Step.

Depending on the coupling scheme defined the settings for the subcycling differ.

For a transient problem – setting the coupling scheme to **Explicit-Transient** – you can set the following options:

Use subcycling enables the subcycling (see [Figure 34](#)).

No. of steps without coupling defines the fixed number of time steps without coupling. The elapsed time between two data exchanges depends on the time step size used by the solver.

Only the time step count will trigger the data exchange.

Use variable no. of steps without coupling enables to set the number of steps without coupling depending on the coupling step number. For Variable no. of steps without coupling MpCCI accepts a list of `couplingStepNo=nofSteps` pairs each defining the coupling step number (`couplingStepNo`) from where the number of 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.

Use duration control enables the setting for starting and ending the coupling (see [Figure 35](#) for transient problems).

Time for starting the coupling sets the start time for the coupling. When the simulation code has reached this start time in exact manner or has exceeded it, the data transfer is allowed.

Time for ending the coupling sets the end time for the coupling. When the simulation code has reached this end time in exact manner or has exceeded it, the data transfer is stopped. The simulation code is now uncoupled.

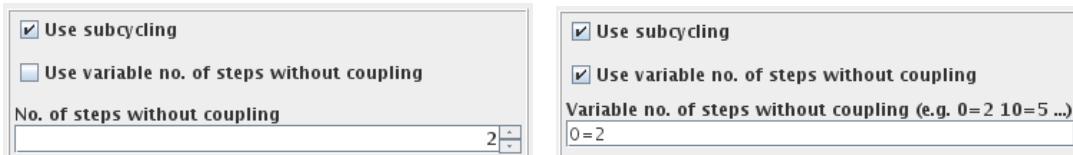


Figure 34: Subcycling configuration with fix and variable number of steps.

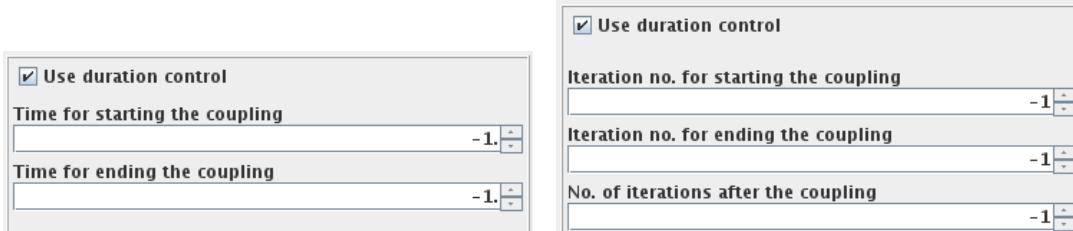


Figure 35: Duration configuration for transient and stationary problems.

For a stationary problem – setting the coupling scheme to `Explicit-SteadyState` – you can set the following options:

Use subcycling enables the subcycling (see [Figure 34](#)).

No. of steps without coupling defines the fixed number of iterations without coupling. A data transfer is triggered automatically after the target number of iterations has been reached.

Use variable no. of steps without coupling enables the definition of a variable subcycling function depending on the coupling step number. For Variable no. of steps without coupling MpCCI accepts a list of `couplingStepNo=nofIterations` pairs each defining the coupling step number (`couplingStepNo`) from where the number of iterations without coupling will be set to the given value `nofIterations`. This value remains until a new coupling step number with another value is specified or the end of the coupling is reached. The `couplingStepNo=nofIterations` pairs are separated by whitespace.

Use duration control enables the setting for starting and ending the coupling (see [Figure 35](#) for stationary problems).

Iteration no. for starting the coupling sets the start iteration for the coupling. When the simulation code has reached this start iteration, the data transfer is allowed.

Iteration no. for ending the coupling sets the end iteration for the coupling. When the simulation code has reached this end iteration, the data transfer is stopped. The simulation code is now uncoupled.

No. of iterations after the coupling sets the number of iterations after the coupling. When the simulation code has reached end iteration it will run these number of iterations uncoupled.

(!) For all the field options, a negative value will deactivate the option.

3.4.7 Non-matching Time Steps

3.4.7.1 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 36](#). In this example codes A and B each use their own fixed, non-matching time step sizes.

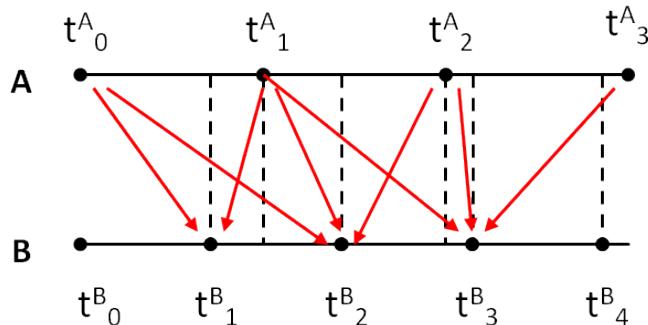


Figure 36: Non-matching time step sizes. Red arrows show which values of code A are used for the interpolation (using a “higher-order” interpolation)

The time step of code B is smaller; the quantity data MpCCI sends to code B at the first step t_1^B is the result of a linear interpolation using the two quantity values sent by code A at time steps t_0^A and t_1^A . This is indicated by the two red arrows pointing from code A to code B.

For time step two of code B (t_2^B) the quantity values from up to three (depending on the selected interpolation method) steps by code A – t_0^A, t_1^A and t_2^A – are used for the interpolation. Obviously, the values that are sent from code B to A also have to be interpolated. [Figure 36](#) contains only the interpolation for one direction (from A to 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 Edit Step of the MpCCI GUI. The different methods are described in the next section.

3.4.7.2 Interpolation Methods

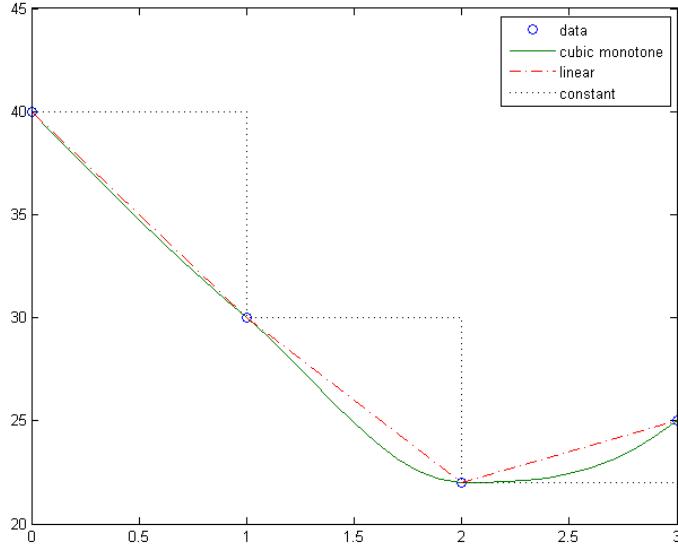


Figure 37: The three different interpolation methods: a piecewise monotone cubic hermite spline, a linear interpolation and a constant interpolation for four data points.

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 edit step. A constant, linear and higher order interpolation are available. The different interpolation schemes are shown in [Figure 37](#)

The constant interpolation (i. e. zero order) uses the last available value; the first order interpolation uses one value before and one after the requested value for a standard linear interpolation.

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 37](#), 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.

3.4.7.3 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 "Initial quantity transfer" option or the "HistorySize" setting.

This happens if deformed coordinates (i.e. MpCCI quantity NPosition) are sent from a code with a big time step to a code with a small time step size.

Increasing the HistorySize in the "Edit 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 (Δt_{big} and Δt_{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 Initial quantity transfer settings are not appropriate for the selected coupling algorithms. When using Jacobi-type algorithms both codes should use "Exchange" as Initial quantity transfer. Using other combinations (e.g. "Send" and "Receive") might result in the fatal message from above because the first grid information (if the coordinates are a coupled quantity) is not received.

Gauss-Seidel-type coupling algorithms Currently, MpCCI does not support extrapolation in time, which leads to some restrictions when codes with different time step sizes are coupled.

Gauss-Seidel-type coupling algorithms with non-matching time step sizes can only be used when the leading code (initial quantity transfer "Receive") uses a bigger time step size than the lagging code (initial quantity transfer "Exchange"). This can be handled by adjusting the Initial quantities transfer settings. 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 Gauss-Seidel schemes for non-matching time step sizes.

3.4.8 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 algorithms (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.

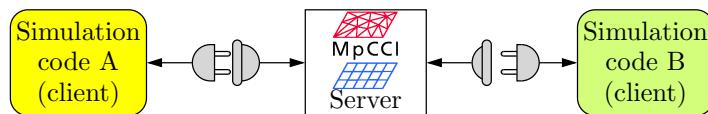
Restarting a coupled simulation is an advanced analysis procedure. It is difficult to choose the initial exchanges such that the time in both codes is still synchronous after a restart. Mostly the initial exchanges should be set to exchange for the restart. This is also code-dependent, there is information 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

a) Coupling without parallel execution



b) Coupling with parallel execution of simulation codes

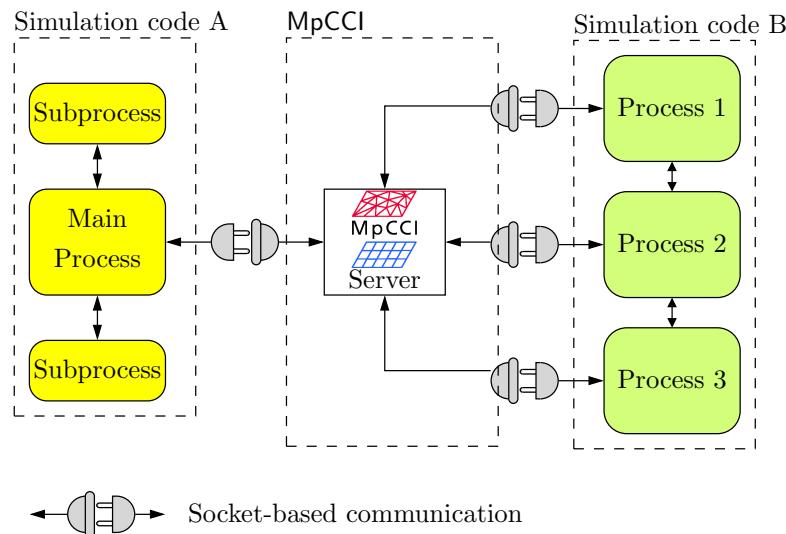


Figure 38: Client-server structure of MpCCI

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 38 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 38 b\)](#) on the left.

- Each subprocess of a simulation code communicates with the MpCCI server as shown in [Figure 38 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 [▷ 4.8 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 ([▷ 4.9 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 ([▷ 4.8.1 Configuring the MpCCI Coupling Server](#)) or when starting the server with `mpcci server` ([▷ 5.7.12 mpcci server](#)).

In the example sketched in [Figure 38 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 38 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.

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 MPCCLARCH - 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 ([▷ 2.5 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:

<code>MPCCI_RSHTYPE</code>	<code>remote shell</code>	<code>file copy</code>
<code>rsh</code>	<code>rsh</code> or <code>remsh</code>	<code>ftp</code> or <code>rcp</code>
<code>ssh</code>	<code>ssh</code>	<code>scp</code> or <code>sftp</code>

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 [▷ III-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.

 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.
- 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.
- useAbsolutePath** Use the current local MpCCI installation directory as reference for any remote computer access.

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. . 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 syntax should be used: VARIABLE=value.

For example:

```
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, SGE, LoadLeveler and GLOBUS.

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 [▷ 4.8.1 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`
- `LOADL_PROCESSOR_LIST`

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

- 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 ..."

are set, the MpCCI startup procedure will allocate the listed hosts for the code represented by the name <CODENAME> or for the server with 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:

MPCCI_<CODENAME-1>.HOSTS = "/pattern/".

 <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:

```
mpcci -batch [OPTIONS] < BATCH_NAME > submit [BATCH-OPTIONS] < project file >
```

with the arguments meaning:

[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](#)).

< BATCH_NAME > represents the name of the batch system e.g. PBS, SGE, etc.

[BATCH-OPTIONS] represents a list of the original options of the submit command.

<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>/mpcci/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:

```
SGE 1532
PBS 8470
```

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. [Figure 3](#) in [▷4.3.2 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 39](#))

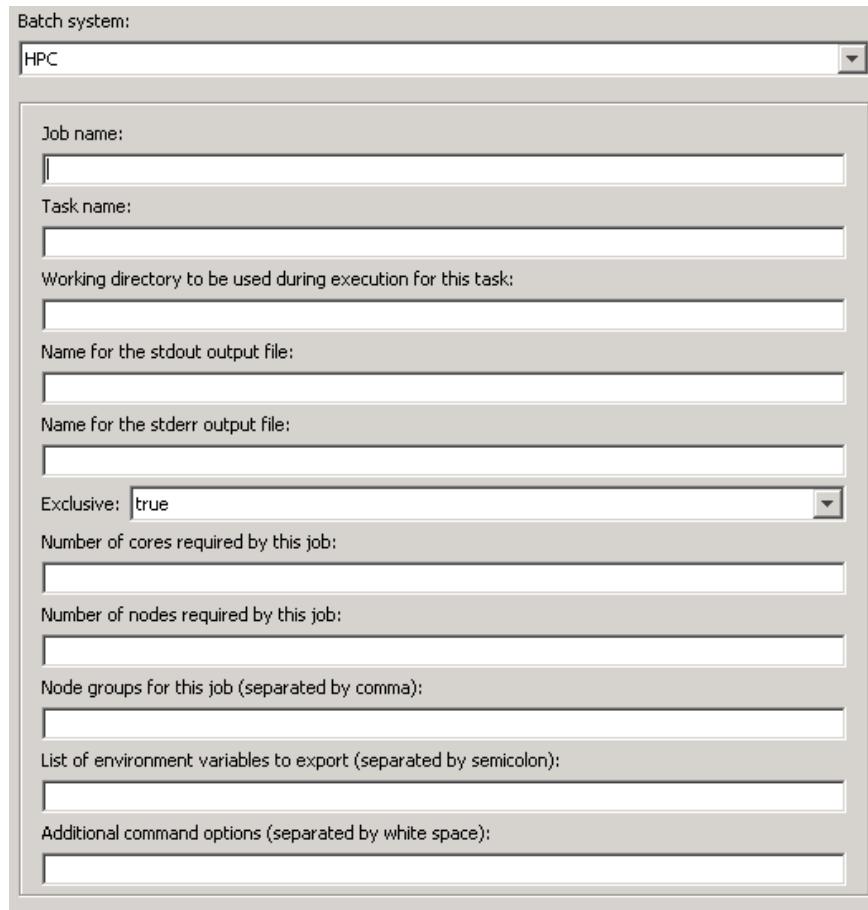


Figure 39: MS HPC options

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 40)

The screenshot shows a configuration interface for an LSF batch system. At the top, a dropdown menu labeled "Batch system:" is set to "LSF". Below this are eight input fields, each with a label to its left:

- Job name:** (input field)
- Queue name:** (input field)
- Name for the stdout output file:** (input field)
- Name for the stderr output file:** (input field)
- Number of hosts/processors:** (input field)
- Host selection (option -m):** (input field)
- Resource requirements (option -R):** (input field)
- Additional command options:** (input field)

Figure 40: 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 41)

The screenshot shows a configuration dialog for the OpenPBS batch system. At the top, a dropdown menu is set to "OpenPBS". Below it, there are several input fields and checkboxes:

- Job name:** A text input field.
- Queue name:** A text input field.
- Name for the stdout output file:** A text input field.
- Name for the stderr output file:** A text input field.
- Join both standard output and standard error**: A checkbox.
- Keep both standard output and standard error**: A checkbox.
- Defines the resources that are required by the job (option: -l):** A text input field.
- Additional command options:** A text input field.

Figure 41: OpenPBS options

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 `-j oe` 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 `-k oe` 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 42)

The screenshot shows a configuration window for a SGE batch system. At the top, a dropdown menu labeled "Batch system:" is set to "SGEEE". Below it is a series of input fields and controls:

- "Job name:"
- "Queue name:"
- "Name for the stdout output file:"
- "Name for the stderr output file:"
- A checked checkbox labeled "Change to the working directory of call"
- A field labeled "Resource requirements (option: -l):" containing a placeholder "[]"
- A field labeled "Define the parallel environment option (option: -pe):" containing a placeholder "[]"
- A field labeled "Additional command options:" containing a placeholder "[]"

Figure 42: SGE options

The options are identical for the SGE, N1GE and SGEEE 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 LoadLeveler System (Figure 43)

The screenshot shows a configuration window for a LoadLeveler batch system. At the top, a dropdown menu labeled "Batch system:" is set to "LoadLeveler". Below this, there are four input fields: "Job name:" (empty), "Job type:" (set to "pvm3"), "Name for the stdout output file:" (empty), and "Name for the stderr output file:" (empty). There is also a field for "Number of nodes [min,max]" which is currently empty.

Figure 43: LoadLeveler options

Job name You can provide a job name for the coupled analysis. This setting represents the option `job_name` of the submit command.

Job type You can specify the type of the job by selecting the option `pvm3`, `parallel` or `serial`. This setting represents the option `job_type` 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 [min,max] You can provide the number of nodes to be reserved for the job by specifying an interval[min,max]. This setting represents the option `node` of the submit command.

Options Details for a **GLOBUS** System ([Figure 44](#))

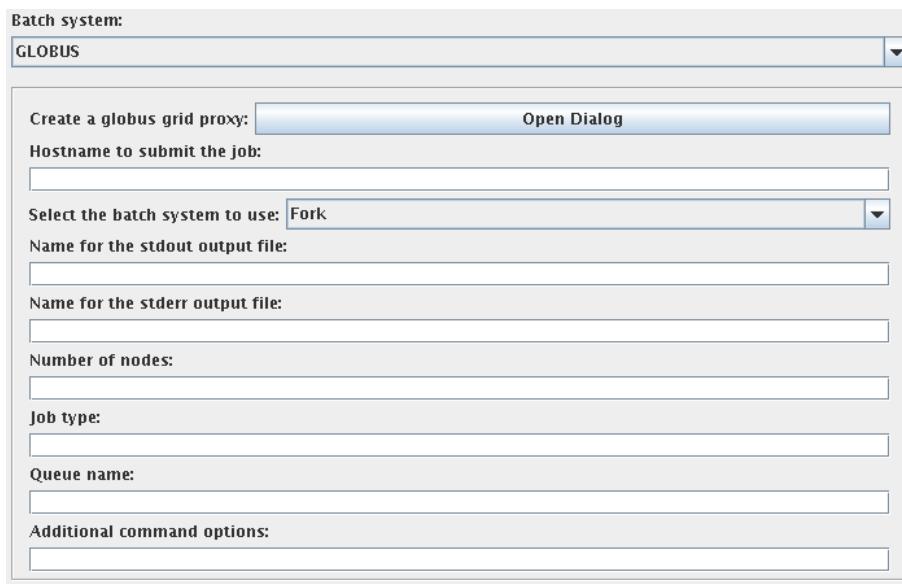


Figure 44: GLOBUS options

Create a GLOBUS grid proxy By clicking on Open Dialog you get a dialog where you can specify the options for creating a grid proxy for the GLOBUS system.

Hostname to submit the job You can provide the hostname of the GLOBUS factory for job submission. This setting represents the option -F of the submit command.

Select the batch system to use You can select the type of scheduler to be used by GLOBUS. This setting represents the option -Ft 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 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.

Number of nodes You can provide the number of nodes to be reserved for the job. This setting represents the option count of the submit command.

Job type You can provide the type of the job. This setting represents the option jobtype of the submit command.

Queue name You can provide the queue name to submit the coupled analysis job. This setting represents the option queue 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.

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
```

4 Graphical User Interface

4.1 Starting and Exiting MpCCI GUI

4.1.1 Starting MpCCI GUI

When you create a coupled simulation project, the MpCCI 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:
  -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.
```

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 `[Alt]+[X]` 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 the defined rules for region building (see [▷ 4.5.5 Automatic Generation of Regions by Rules](#)) and the selected columns to be viewed in the region properties table (see [▷ 4.5.6 Applying Region Properties](#)) are saved.

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 version of MpCCI you are running,
- the name of the currently coupled simulation project or “noname” if no project is specified and
- the steps of the coupling process with the currently worked on step marked by angle brackets.



Figure 1: MpCCI GUI title and main menus

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.

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.

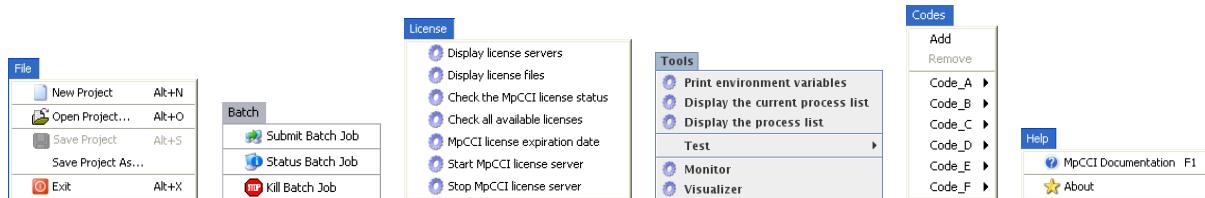


Figure 2: File, Batch, License, Tools, Codes and Help menus

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 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 split into three parts (see [Figure 3](#)):

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 4](#)).

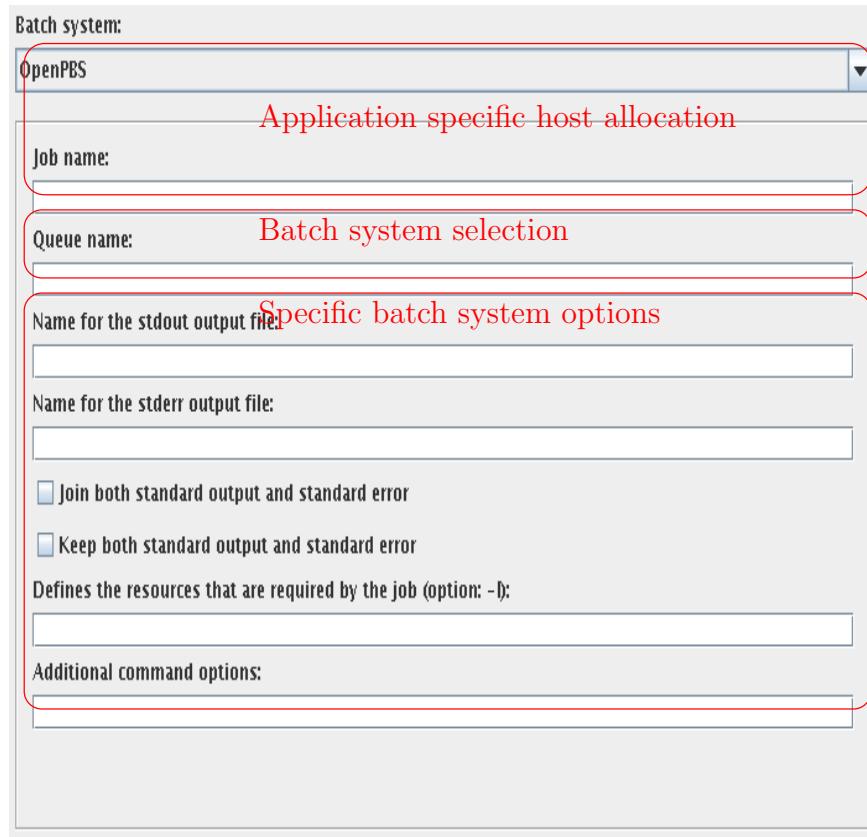


Figure 3: Batch submit configuration dialog



Figure 4: Batch status dialog

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 5](#)).

4.3.3 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.

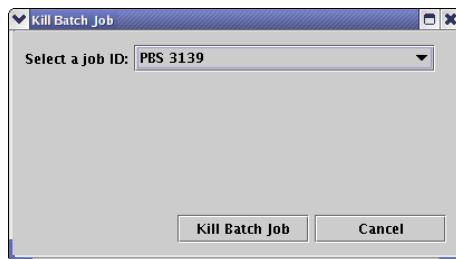


Figure 5: Batch kill dialog

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.

4.3.4 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→MpCCI simple test runs a simple test-case delivered with MpCCI.

Tools→Monitor starts the MpCCI monitor.

Tools→Visualizer starts the MpCCI Visualizer.

4.3.5 Codes Menu

Codes→Add Adds a new code selection to the Models Step and switches the view to the Models Step. Needed when coupling more than two codes. There is no limit given by MpCCI for the number of codes to be coupled. But coupling more than three codes hasn't been tested because of missing use cases.

Codes→Remove Removes the rightmost code selection from the Models Step and adjusts the settings done in the other steps. The view will be switched to the Models Step. The Remove entry is only enabled if at least three code selections are present.

Codes→Codename Provides code specific commands which are defined in the respectively code configuration file (see [VIII-2.4.2 Codes Menu: <CodesMenuEntries>](#)).

4.3.6 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 of four steps for setting up a coupled simulation. Its aim is

- to select the codes involved in the coupling,
- to choose their corresponding model files and
- to start the scanner for each code which will extract the components needed to build the coupling regions in the next step, the Coupling 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 the Coupling 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 static, 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 clicking on the **Coupling Step** button 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 next 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 region and monitor sets 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 region and monitor sets removed

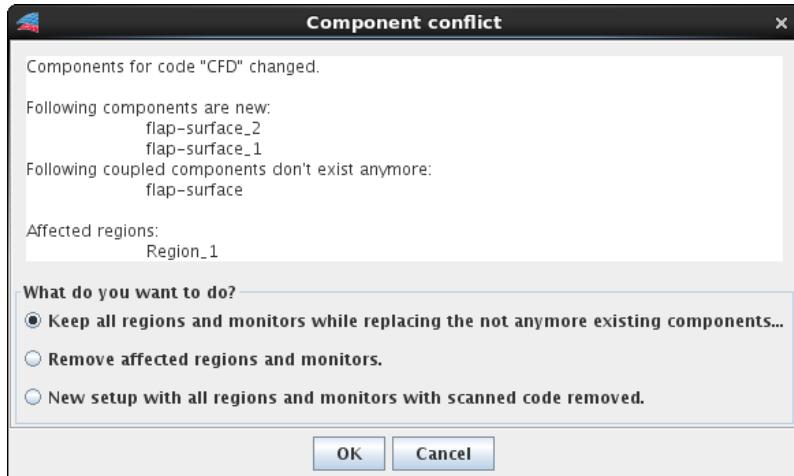


Figure 6: Component conflict after changing the model

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)

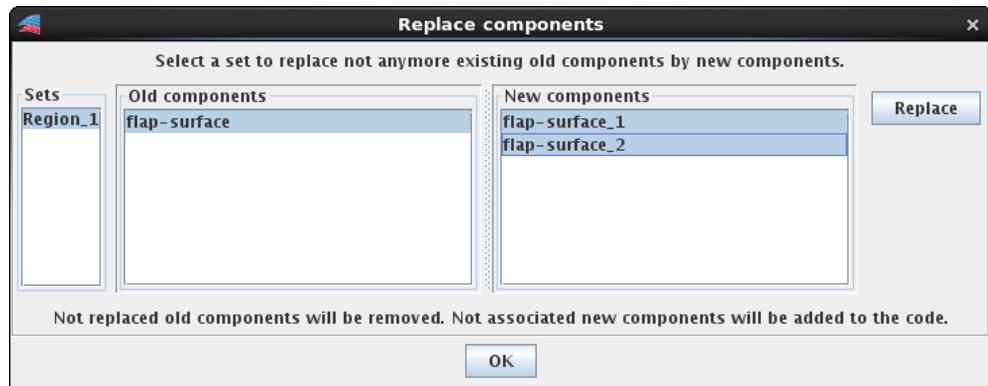


Figure 7: Replacing components in a component conflict

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 Coupling Step

In the MpCCI Coupling 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 Coupling Step provides an overview over the coupled setup for the mesh based components.

The basic steps are described in [▷ IV-2.5 Coupling Step – Defining Coupling Regions and Quantities ◁](#).

Additional features are described in the following:

4.5.1 Global Variables

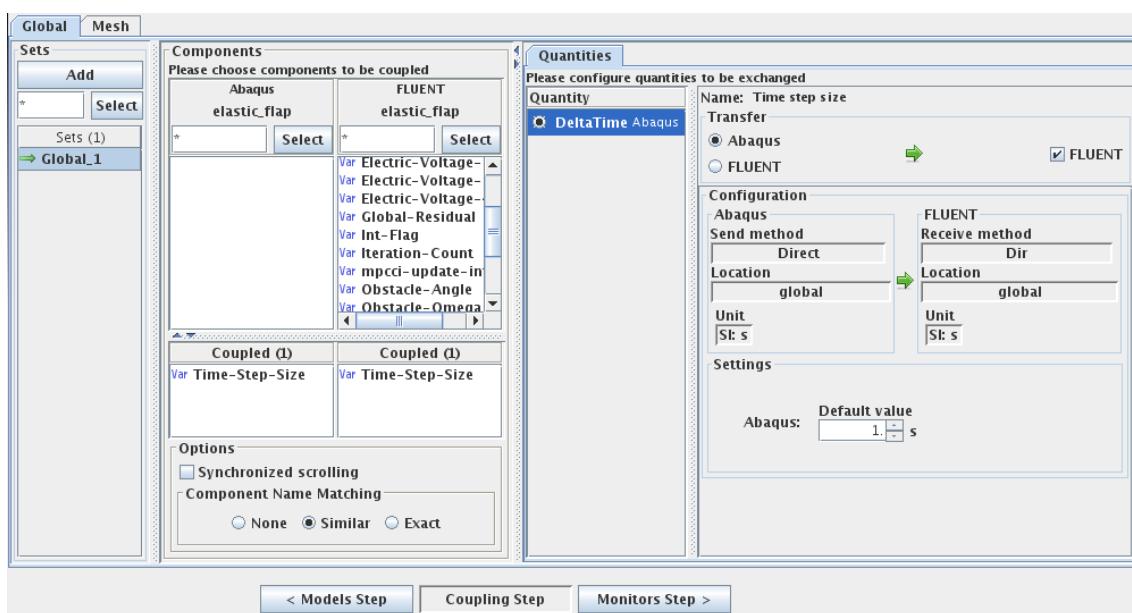


Figure 8: Defining global variables

The definition of sets with global variables to be exchanged differs a little from that of mesh based coupling regions. [Figure 8](#) 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 [section 2.5.1 \(part IV\)](#) 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-2.5.1.1 Select Components for Each Inter-connected Code ◁](#)). Their type is depicted by the label **Var**. 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-2.5.2.1 Select Quantities to be Transferred](#)). But for global variables only one quantity may be selected per set which is why they are selected by radio buttons instead of checkboxes.

4.5.2 Editing Components

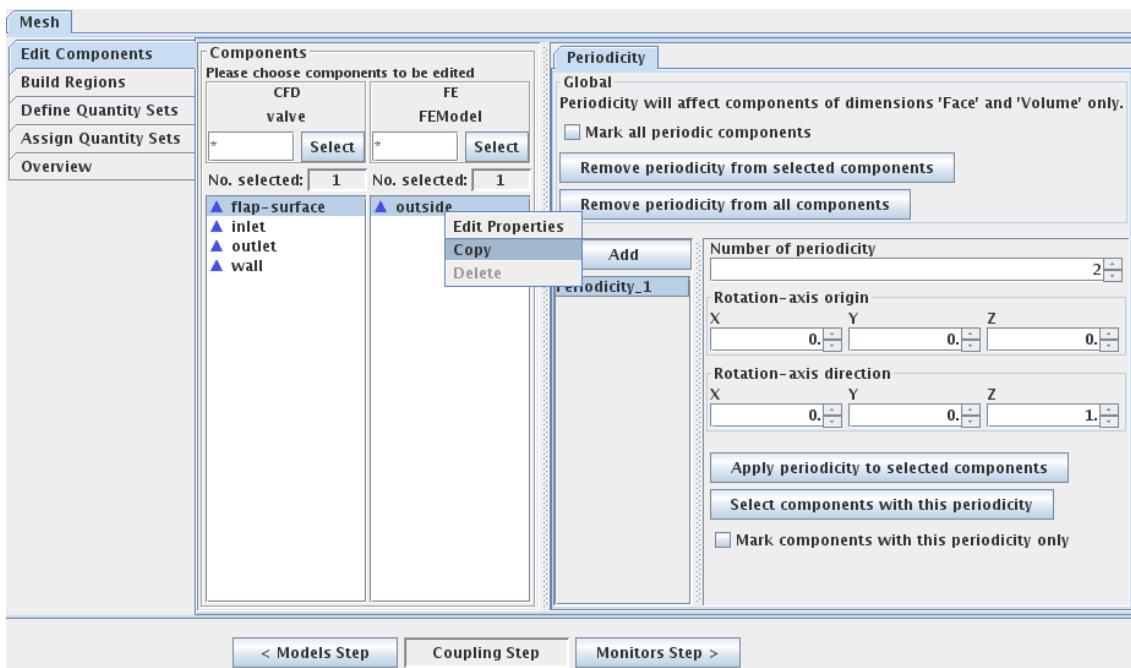


Figure 9: Edit Components tab

In the **Edit Components** tab for mesh based components (see [Figure 9](#)) 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 pop-up menu with the entries **Edit Properties**, **Copy** and **Delete** is also available in the components lists of the **Build Regions** tab.

4.5.2.1 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** ([gui.xcf](#)).

After that you can choose **Copy** from the component's pop-up menu (see [Figure 9](#), 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.

4.5.2.2 Editing Component Properties

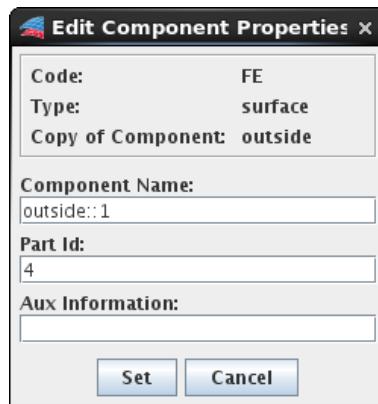


Figure 10: Dialog for editing component properties

The pop-up menu in the components list (Figure 9) 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 10).

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 (▷ 4.5.2.3 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.

4.5.2.3 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 ▷ 3.3.3.3 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 11) shown in the right area of the Edit Components tab (see Figure 9).

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.

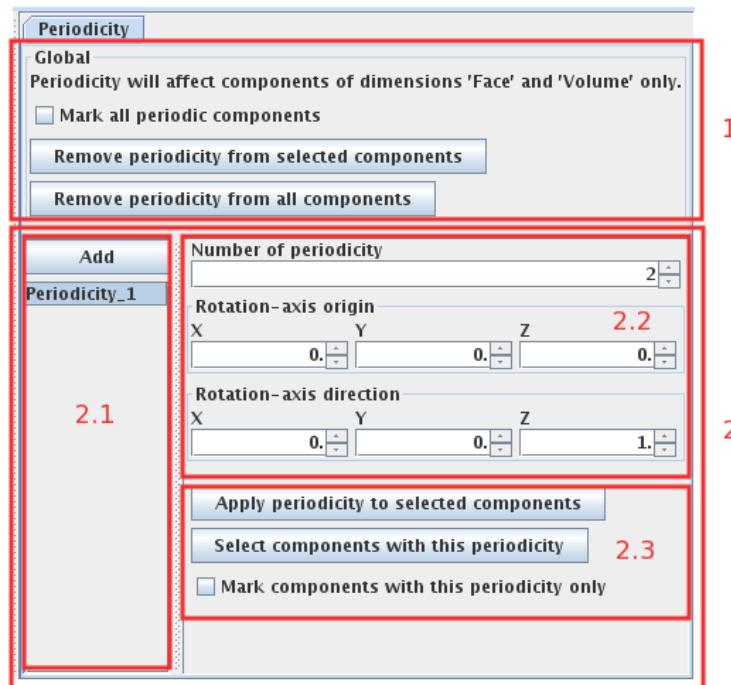


Figure 11: Periodicity panel

The specification of a periodicity is done in the specification area (2.2) by defining the parameters:

- 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 12](#)) or from the action area (2.3 in [Figure 11](#)).

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 12](#), 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:

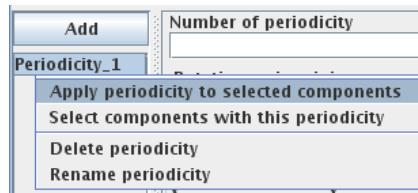


Figure 12: Periodicity pop-up menu

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 13](#)):

- **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 13](#).

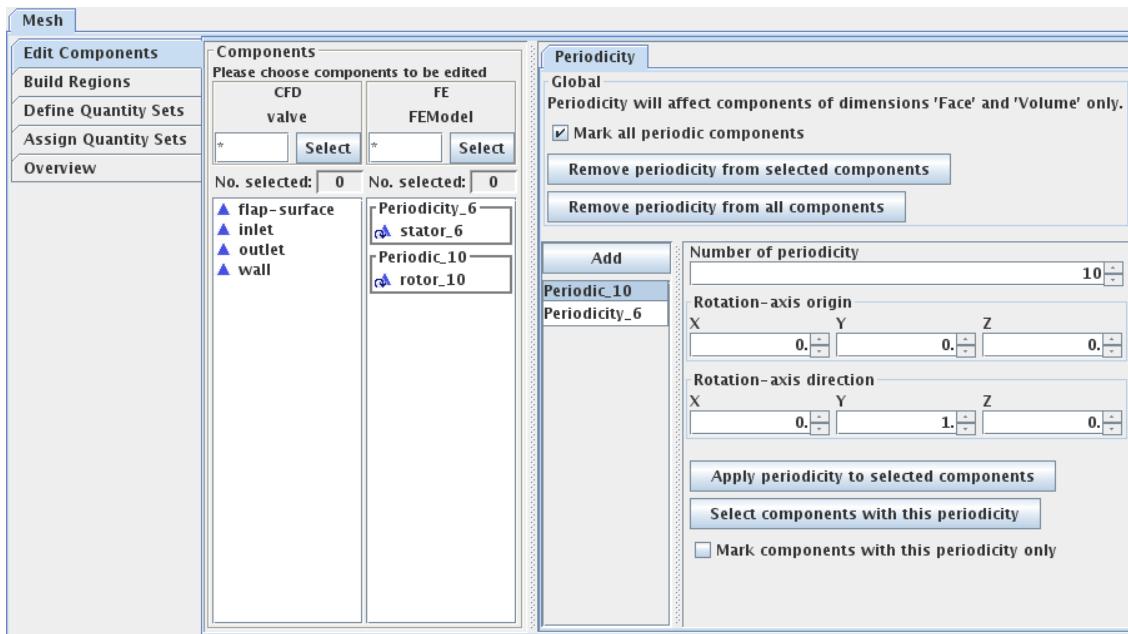


Figure 13: Marked periodic components

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 12](#)) and also in the action area 2.3 of the periodicity panel ([Figure 11](#)).

Removing periodicity from components is part of the global options (1 in [Figure 11](#)) 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

4.5.3 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.

(i) 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.

4.5.4 Simplified Selection

4.5.4.1 Simplified Component Selection

The selection of components out of a large list sometimes may be confusing. To simplify this a selection area for selecting components by their name (see [Figure 14](#)) and an Options area (see [Figure 15](#)), located

Global: below the coupled components lists (see [Figure 8](#)),

Mesh: in the Build Regions tab below the coupled components lists (see [Figure 8 \(part IV\)](#)), exist.



Figure 14: Component selection area

The component selection area consists of a textfield where a pattern with wildcards - “*” for none or more characters and “?” for at most one character - 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.

The option area provides



Figure 15: Option area for choosing components

Synchronized scrolling useful for scrolling large component lists when you are looking for similar named components in both codes. If enabled and you scroll one component list, the list with the other code components will be automatically scrolled to the same alphabetical position.

Component Name Matching which configures the behavior of the automatic component selection. Automatic component selection means that if you select a component of one code, components of the other code with matching names will automatically be selected, too. Choose

None to disable the automatic selection,

Similar to automatically select components with similar names (names starting with equal initial characters) and

Exact to automatically select a component with the exact equal name of the manually selected component.

4.5.4.2 Simplified Region Selection

Dealing with a large number of regions may require to select them by their name or the name of their contained components. For this an area for simplified region selection (see [Figure 16](#)) is located at the top of several region lists namely in the tabs Build Regions, Assign Quantity Sets and Overview.



Figure 16: Area for simplified region selection

This area provides a textfield where a pattern with wildcards - “*” for none or more characters and “?” for at most one character - can be given. If the **Search content** field is marked the patterns relate to the names of the contained components of each region, otherwise the patterns will be applied to the name of the regions themselves. Via the **Select** button those regions matching the given pattern will be selected.

4.5.5 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 17](#) 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 8 \(part IV\)](#)).

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.

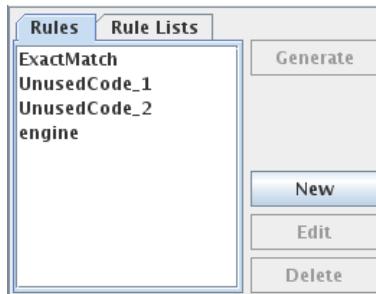


Figure 17: Rules area for automatic region generation

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 18](#)).

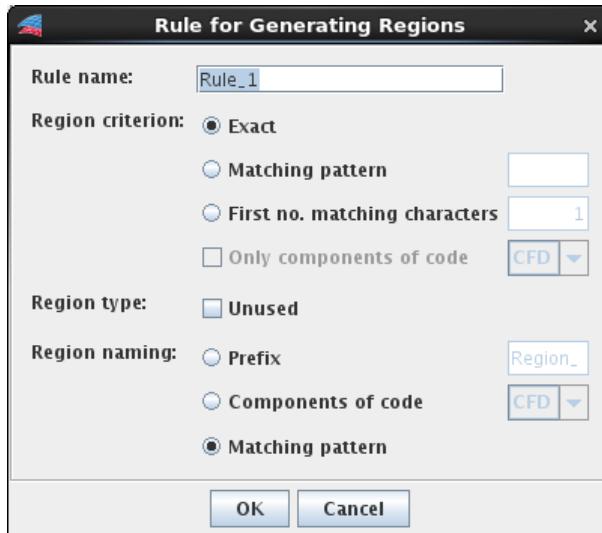


Figure 18: Dialog for setting rule properties

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 where one component's name may be prepended by a prefix e. g. :

"flap-surface" ≡ "Flap-Surface" Match exactly because both names are equal when they are compared to case-insensitive.

"flap-surface" ≡ "V001_Flap-surface" Match exactly because "V001_" is a prefix to the equal part "flap-surface".

"flap-surface" ≠ "V002_flap-surface-3" Don't match exactly because of "-3" appended to the second name.

"V001_flap-surface" ≠ "V002_flap-surface" Don't match exactly because of the different pre-

fixes "V001_" and "V002_".

This region criterion results in building several regions each with exactly one component from each code.

Matching pattern Components which name match the same pattern build a region. The pattern is handled as a regular expression. Multiple patterns are separated by whitespace. Each pattern will be used to generate one region.

If a matching pattern could not be found in the component names no region for this pattern will be generated.

First no. matching characters Components which are named equally at the beginning build a region. The length of the equally named string has to be entered.

This region criterion may result in several regions, each with a group of components equally named by the first n characters at the beginning 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 needs components of two 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 equal part 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 a rule and clicking the **Edit** button in the rules area (see [Figure 17](#)). After that the dialog for setting rule properties opens (see [4.5.5 To define a new rule](#)).

Deleting a rule is done by selecting a rule and clicking the **Delete** button in the rules area (see [Figure 17](#)). A dialog pops up to confirm the deletion. If the rule is part of a rule list (see [4.5.5.1 Rule Lists](#)), the user will be informed and has to confirm the removal from that list, too.

Generate regions by selecting a rule and clicking the **Generate** button in the rules area (see [Figure 17](#)). The rule will be applied to the code components lists selected in the Components area (see [Figure 8 \(part IV\)](#)). 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.

4.5.5.1 Rule Lists

The defined region rules can be concatenated so that one rule after another will be automatically applied by one call. These sequenced rule lists are accessed by the Rule Lists tab in the rules area (see [Figure 19](#)).



Figure 19: Example rule Lists in rules area

Handling rule lists is similar to that of rules. The action buttons New, Edit, Delete and Generate are the same. The defined region rule 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.

Editing a rule list, while creating a new one ([New](#)) or editing an existing one ([Edit](#) or double clicking a rule list), is done via the dialog shown in [Figure 20](#). A rule list is defined by its name and the sequenced rules. Therefore the dialog offers following input and editing options:

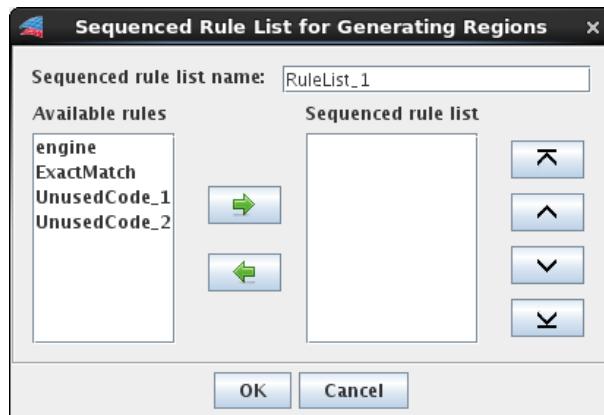


Figure 20: Dialog for editing sequenced rule lists

Sequenced rule list name The name of the rule list.

Available rules List of rules which are available meaning listed in the Rules tab and not already used in this rule list.

Sequenced rule list List of rules building this sequenced rule 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 rule lists.

→ moves the selected available rules to the end of the sequenced rule list.

removes the selected rules from the sequenced 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 sequence of the rule list is done with the buttons at the right of the sequenced rule 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.

Deleting a rule list is done by selecting a rule list and clicking the **Delete** button in the rules area (see [Figure 19](#)). A dialog pops up to confirm the deletion.

Generate regions by selecting a rule list and clicking the **Generate** button in the rules area (see [Figure 19](#)). Each rule of the rule list will be applied as described in [4.5.5 Generate regions](#). A dialog will show a summary with the results of the applied rules.

Save the rule list by clicking the **OK** button. **Cancel** or closing the dialog will dismiss the editing and won't create a new rule resp. leave the rule list unchanged.

4.5.6 Applying Region Properties

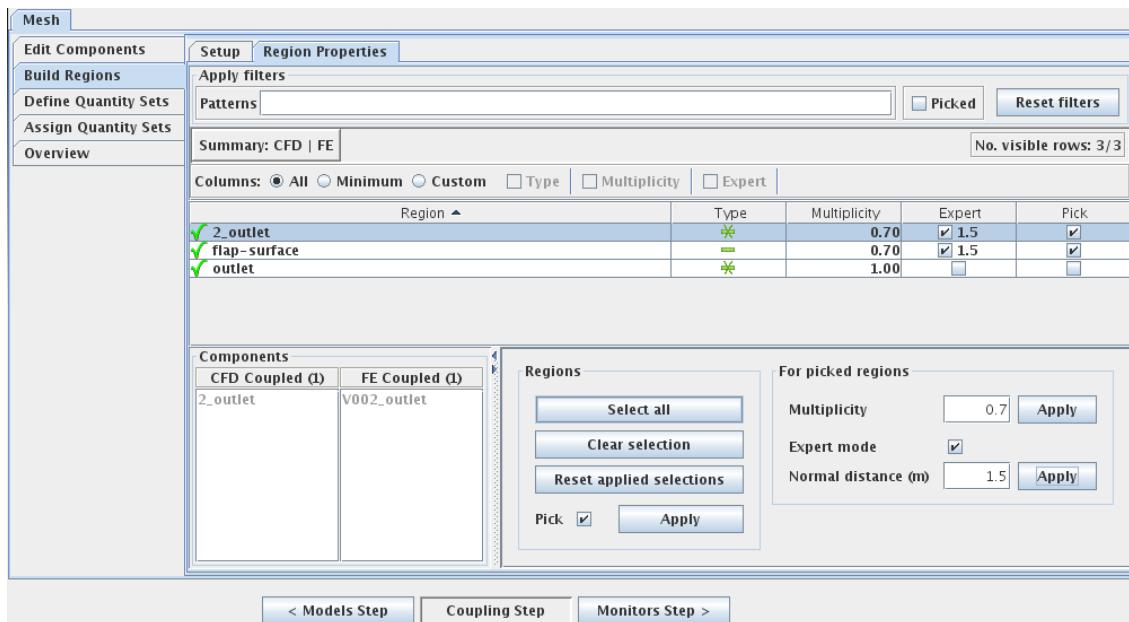


Figure 21: Region properties tab

The attributes for the relation search are globally defined among the control parameters for MpCCI in the EditStep (see [4.7.3 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 21](#).

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
 - Patterns** For specifying a whitespace separated list of patterns which will be applied to the names of the visible regions. Wildcards “*” and “?” are allowed to match none, one or more characters.
 - 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.

4.5.7 Quantity Specifications

4.5.7.1 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 22](#).



Figure 22: Selection of the sender/receiver for a quantity

4.5.7.2 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 8](#)).

4.5.7.3 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 23](#)) by

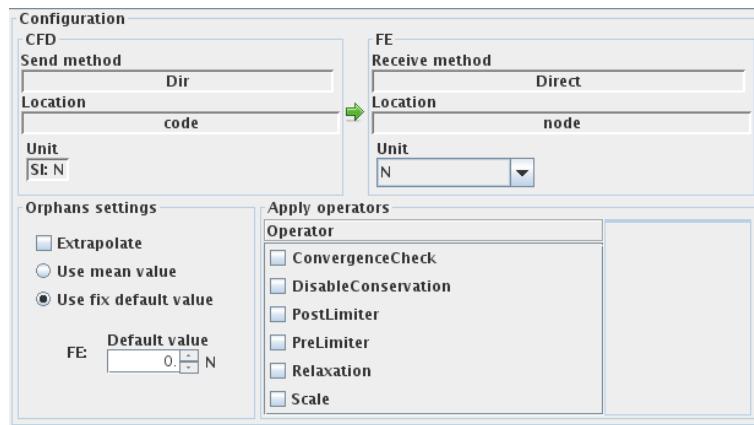


Figure 23: Quantity configuration areas

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.

4.5.7.4 Applying Operators to Mesh Based Components

The MpCCI server provides some operators in the Apply operators area which can be applied to the transferred quantities (see [Figure 23](#)). 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 24](#)).

The type of an operator is either pre processor or post processor and the necessary parameters are operator dependant. Following operators are supported by MpCCI.

ConvergenceCheck Pre processor

Enables convergence check.

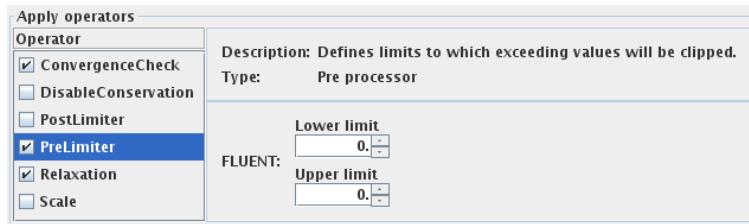


Figure 24: Operators area

Parameters: **Tolerance value** Specifies the convergence criterium. If the variance of a quantity value is below this tolerance between two coupling steps, the quantity is considered as converged.

! If this operator is assigned to more than one quantity, the convergence of one quantity will lead to a converged simulation even if the other quantities are not yet considered as converged. Thus, applying the convergence check operator may only be useful for one of the transferred quantities.

DisableConservation Post processor

Disables conservative mapping correction (default: enabled).

<No parameters>

PostLimiter Post processor

Defines limits to which exceeding values will be clipped.

Parameters: **Sequence no.** 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.

Parameters: **Lower limit** The lower limit.

Upper limit The upper limit.

Relaxation Post processor

Enables relaxation. For steady state or iterative transient simulations MpCCI offers the possibility to over- or under-relax the sent quantity values. Several relaxation methods are offered. See [3.3.3.1 Quantity Relaxation](#) for more information on how to use the different relaxation methods.

Parameters: **Relaxation method** The method used for relaxation:

Aitken Uses an adaptively calculated relaxation factor.

Fixed Uses the given fixed Relaxation factor.

Quasi-Newton Uses the chosen Quasi-Newton method for relaxation.

Anderson Mixing Provides the types **Inverse** and **Standard** with a given omega value $\omega > 0$ and the type **LeastSquares** where the omega value is automatically set to 0.

Broyden Uses the given omega value $\omega > 0$ as scale factor for the initial Jacobian.

Jacobi Provides the two types **Inverse** and **Standard** with an omega value $\omega > 0$.

Ramping This is a special kind of relaxation with a relaxation factor varied over the iteration resp. time. Under-relaxation is achieved by starting with a value smaller than 1 whereas over-relaxation is achieved by starting with a value bigger than 1. The ramping value will be increased to 1 for under-relaxation and decreased to 1 for over-relaxation. The necessary parameters are the **Initial ramp value** and the **Ramp delta**.

Scale Post processor

Defines a scaling function applied like a ramping (for steady state solution).

Parameters: **Sequence no.** The rank of this operator in the list of applied post processors.

Scaling at start The scaling factor at the start.

Scaling delta The delta by which the scaling factor is increased.

Number of constant scaling steps The number of steps with one constant scaling factor.

Upper scaling limit The upper limit where the increasing stops. A zero value here means that the increasing never stops.

For example, with the following scale parameters definition:

Scaling at start: 0.5, Scaling delta: 0.2, Number of constant scaling steps: 3, Upper scaling limit: 1 you will have the scaling factor plotted on [Figure 25](#).

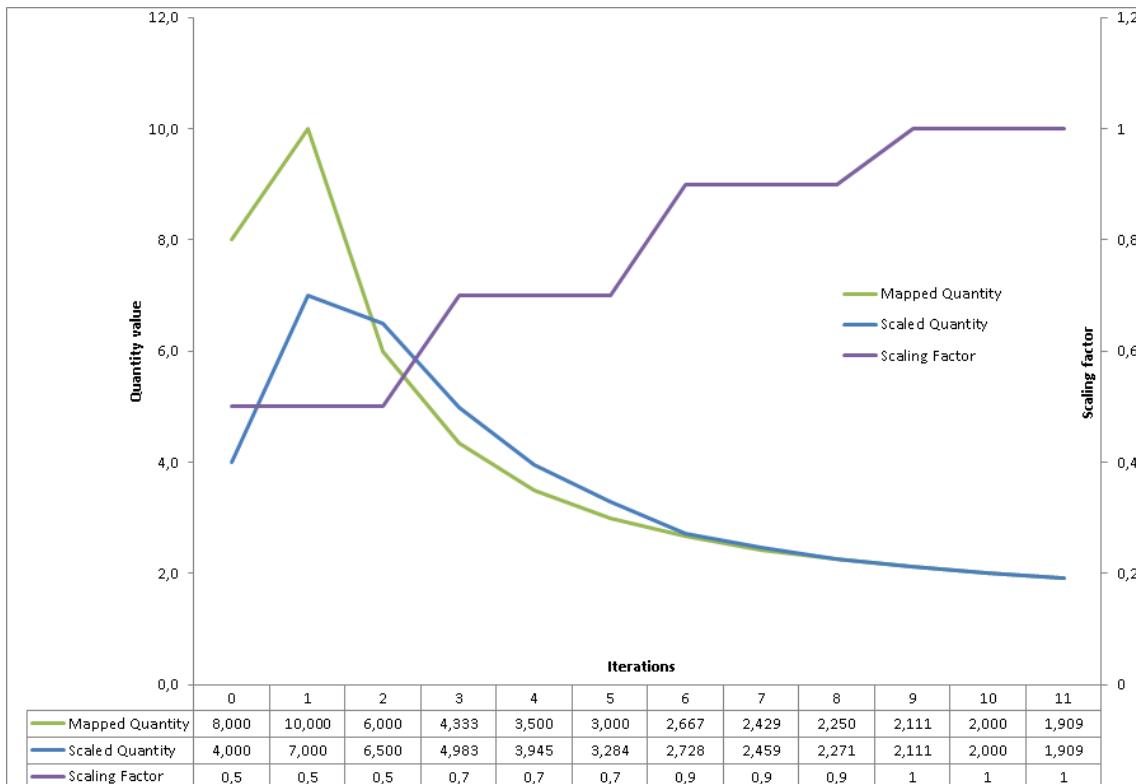


Figure 25: Scaling function on quantity

4.5.8 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 26](#)). 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:



Figure 26: Predefined quantity settings

- 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

<i>Code type</i>	<i>Quantities to send</i>
FluidPlasma	ElectrRes3
ElectroMagnetism	LorentzForce JouleHeat

Transient electric arc coupling: Method 2

Dimension of coupling region: 3

<i>Code type</i>	<i>Quantities to send</i>
FluidPlasma	ElectrRes3
ElectroMagnetism	MagneticField CurrentDensity

Coupled magnetic field and thermal analysis

Dimension of coupling region: 3

<i>Code type</i>	<i>Quantities to send</i>
CFD	Temperature
ElectroMagnetism	JouleHeat

One way force mapping

Dimension of coupling region: 2

<i>Code type</i>	<i>Quantities to send</i>
CFD	RelWallForce

One way pressure mapping

Dimension of coupling region: 2

<i>Code type</i>	<i>Quantities to send</i>
CFD	OverPressure

Steady state radiative heat transfer

Dimension of coupling region: 2

<i>Code type</i>	<i>Quantities to send</i>
Radiation	WallTemp
FluidThermal	FilmTemp WallHTCoeff

Steady state surface heat transfer

Dimension of coupling region: 2

<i>Code type</i>	<i>Quantities to send</i>
SolidThermal	WallTemp
FluidThermal	FilmTemp WallHTCoeff

Transient MHD Coupling

Dimension of coupling region: 3

<i>Code type</i>	<i>Quantities to send</i>
FluidPlasma	ElectrRes3
ElectroMagnetism	LorentzForce

Transient radiative heat transfer

Dimension of coupling region: 2

<i>Code type</i>	<i>Quantities to send</i>
Radiation	WallTemp
FluidThermal	WallHeatFlux

Transient surface heat transfer

Dimension of coupling region: 2

<i>Code type</i>	<i>Quantities to send</i>
SolidThermal	WallTemp
Radiation	WallTemp
FluidThermal	WallHeatFlux
InjectionMoulding	WallHeatFlux

Absolute pressure based Fluid-Structure Interaction

Dimension of coupling region: 2

<i>Code type</i>	<i>Quantities to send</i>
SolidStructure	NPosition
CFD	WallForce

Gauge pressure based Fluid-Structure Interaction

Dimension of coupling region: 2

Code type	Quantities to send
SolidStructure	NPosition
CFD	RelWallForce

4.5.9 Overview of the Coupled Regions

To get an overview of the mesh based coupled regions the Mesh tab of the Coupling Step provides an Overview tab (see [Figure 27](#)).

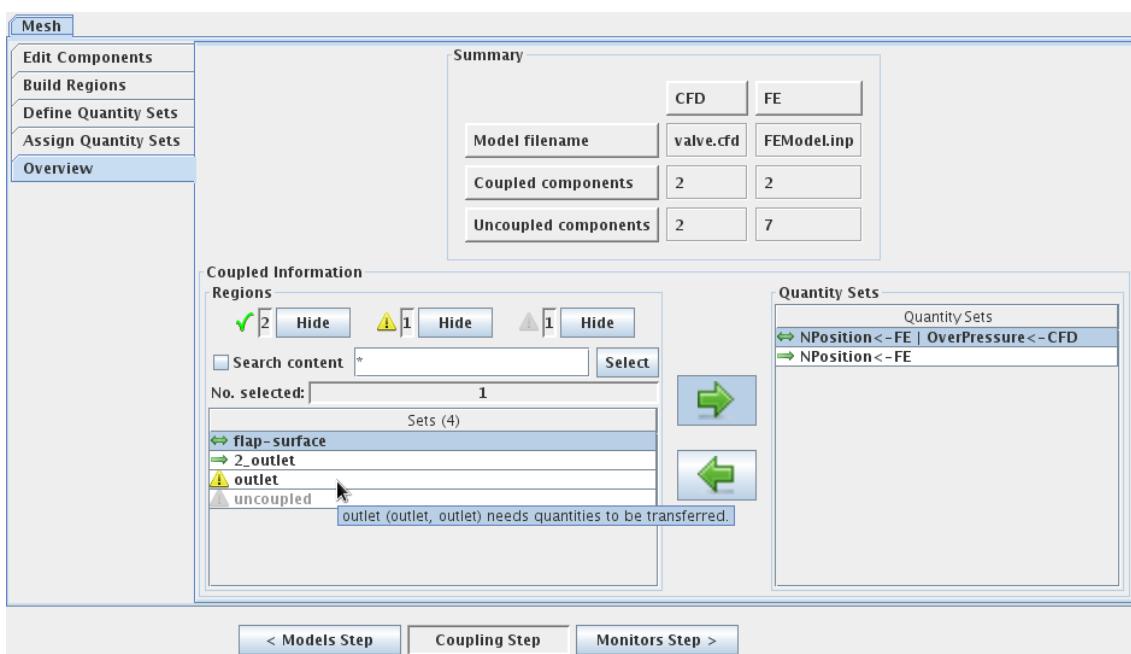


Figure 27: Overview tab

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- (\rightarrow) or bidirectional (\leftrightarrow) 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 27](#) shows for region **outlet**, the tooltip of a region displays its contained components and an additional explanation for not ready regions. By the direction button \rightarrow all quantity sets will be marked which are assigned overall to the selected regions. The button \leftarrow on the other hand marks the regions which have assigned the selected quantity sets.

4.5.10 Requirements

In the Coupling 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 the **Monitors Step** button will cause a dialog popping up with an appropriate message and will inhibit the next step.

4.6 Monitors Step

In the MpCCI Monitors Step you create and configure a region to monitor (see Figure 28).

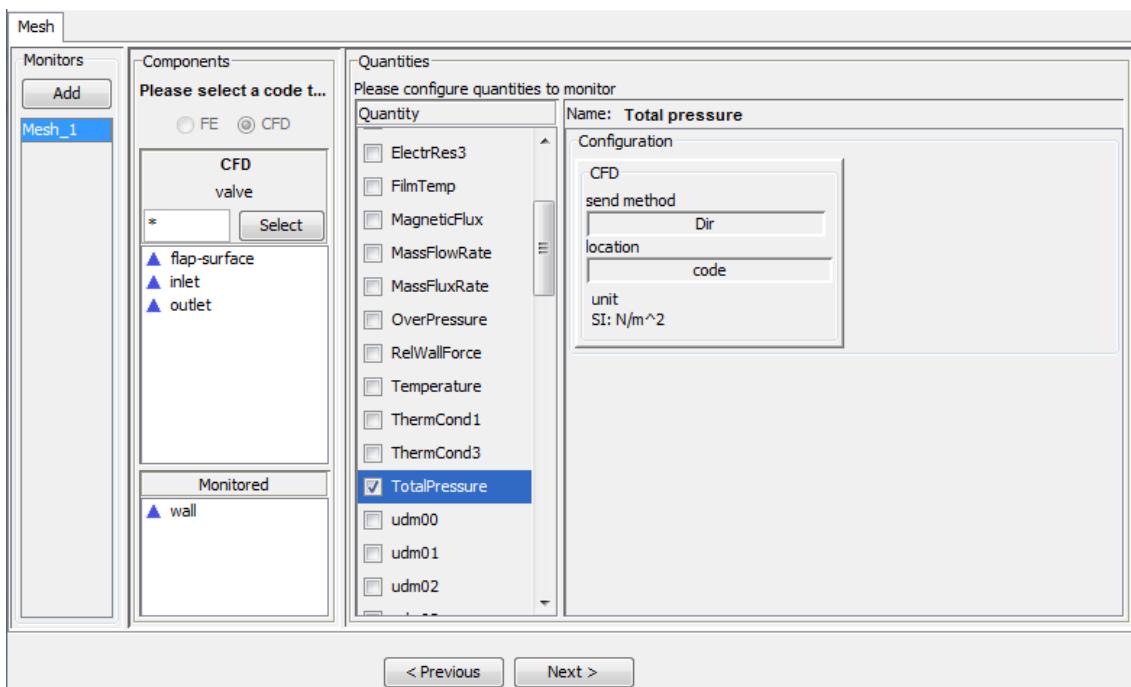


Figure 28: Monitors Step

Create a monitor set 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 which will be monitored.

Add additional monitors by clicking the **Add** button and repeating the procedure.

4.7 Edit Step

The options in the Edit Step are the control parameters for MpCCI. They are grouped into **Monitor**, **Job**, **Relation Search** and **Output** sections.

4.7.1 Monitor

In this section you may specify some parameters used by MpCCI to control the online monitor.

Parameter	Value	Description
Enable	<input checked="" type="checkbox"/>	Activate the MpCCI monitor. (Default: on)
Nodelds	<input type="checkbox"/>	Send the node IDs to the monitor.
Elemlds	<input type="checkbox"/>	Send the element IDs to the monitor.
Size	<input type="checkbox"/>	Send the element sizes to the monitor.
Orphans	<input checked="" type="checkbox"/>	Send the nodal orphan level to the monitor. (Default: on)
Globals	<input type="checkbox"/>	Monitor global variables.
Slaves	<input type="checkbox"/>	Monitor slave node flags.
Domains	<input type="checkbox"/>	Include node domains (only valid with domain check).
Normals	<input type="checkbox"/>	Include face normal vectors.
Peaks	<input type="checkbox"/>	Include quantity peak & mean values plot.
AbsDiff	<input type="checkbox"/>	Display the absolute differences between iterations or time steps.
RelDiff	<input type="checkbox"/>	Display the relative differences between iterations or time steps.
Inorm	<input type="checkbox"/>	Include quantity iteration norm values plot.
Relax	<input type="checkbox"/>	Include quantity relaxation factor plot.
Periodic	<input type="checkbox"/>	Display the replicated periodic parts and quantities.
Modulo	100	Set the maximum number of steps kept by the monitor. (Default: 100 steps)
Memory	50	Set the maximum memory in MB used by the monitor. (Default: 50 MB)
AutoLaunch	<input checked="" type="checkbox"/>	Auto launch the MpCCI monitor. (Default: on)
Host	localhost	Specifies the name of the host machine of the MpCCI monitor.
Port	47,002	Main monitor port address
NetDevice		Specify a network device or IP address.

Figure 29: Monitor settings

Parameter Description

Enable	Activate the MpCCI Monitor. (Default: on)
Nodelds	Send the node IDs to the MpCCI Monitor. (Default: off)
Elemlds	Send the element IDs to the MpCCI Monitor. (Default: off)
Size	Send the element sizes information to the MpCCI Monitor. (Default: off)
Orphans	Send the nodal orphan quality level (0...6) to the MpCCI Monitor.

Parameter	Description
Globals	(Default: on) Monitor global variables. (Default: off)
Slaves	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)
Domains	Send the node domains' IDs to the MpCCI Monitor (only active in conjunction with a domain check - see ▶4.7.2 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 "diffusive" extrapolation.
Normals	(Default: off) Send face normal vectors to the MpCCI Monitor. (Default: off)
Peaks	Send the peak and mean values of a quantity on a partition to the MpCCI Monitor. (Default: off)
AbsDiff	Display the absolute differences between iterations or time steps. (Default: off)
RelDiff	Display the relative differences between iterations or time steps. (Default: off)
Inorm	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)
Relax	Include quantity relaxation factor plot. (Default: off)
Periodic	Display the replicated periodic parts and quantities. (Default: off)
Modulo	Set the maximum number of steps kept by the MpCCI Monitor. (Default: 100 steps)
Memory	Set the maximum memory in MB used by the MpCCI Monitor. (Default: 50 MB)
AutoLaunch	Auto launch the MpCCI Monitor by the server. If set to off, the user can use the command <code>mpcci monitor [options]</code> to launch the MpCCI Monitor "manually". (Default: on)
Host	Specifies the name of the host machine where the MpCCI Monitor runs. (Default: localhost)
Port	 The MpCCI Monitor is only supported on Windows or Linux operating systems.
NetDevice	Provide the MpCCI Monitor port address to use. (Default: 47002) Specify a network device or 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)

4.7.2 Job

In this section you specify the attributes of the coupled job.

Properties Job		
Parameter	Value	Description
Interpolation	Higher Order	Select the interpolation method. (Default: Higher Order)
AngularInterpolation	NLERP	Select the interpolation method for angular coordinates. (Default: NLERP) -NLERP: Normalized Linear INTERPolation (default). 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. -Zero Order: Constant interpolation.
MeshMotion	<input type="checkbox"/>	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 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. BEWARE: 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	<input type="checkbox"/>	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	<input type="checkbox"/>	Use the initial mesh-mesh relationship during the moving mesh transformation (Default: off) Only valid with the mesh motion option.
OverlapCheck	<input checked="" type="checkbox"/>	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 complaints and an abortion by the MpCCI server.
DomainCheck	<input type="checkbox"/>	Enable a check on the domains of parts or meshes. (Default: off) With an activated domain check the server finds the no. of domains of a mesh/part and also searches for isolated strap elements/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	<input type="checkbox"/>	Enable the shifting of baffle front and rear side for a geometrical separation of the two sides of a baffle (Default: off) In the 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.
ConformalMesh	<input type="checkbox"/>	Enable the conformal mesh mapping algorithm (Default: off) In case two partner meshes are conformal (geometry and element/cell type) a special mapping algorithm can be applied. The relationship is much faster, the mapping does not produce any mapping errors. ATTENTION: 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	1E-3	Define the (relative) tolerance of any time difference/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 ≥ 0.5 resets the value to its default. Note: This value is only relevant for transient cases with multiple buffers.
HistorySize	4	Define the minimum number of buffers for storing the quantity history. (Default: 4)
MaxIterQuasiNewton	20	Define the maximal number of saved iteration values to be used for the approximation of the Jacobian in the Quasi-Newton relaxation. Applies only for steady simulations. (Default: 20)
ResidualsQuasiNewton	0	Define the number of residuals which are checked to be increasing in a Quasi-Newton relaxation. (Default: 0) A positive slope of the linear least squares fit to these residuals leads to a restart of the Quasi-Newton method, which means to re-initialize the Jacobian. If the parameter is set to 0 in steady simulations, the Quasi-Newton method is restarted automatically after 'MaxIterQuasiNewton' and the residual trend is not considered; if no automatic restart is desired, choose 1.

Figure 30: Job settings

Parameter	Description
Interpolation	Select the interpolation method. (Default: Higher Order) For more details see 3.4.7 Non-matching Time Steps
Higher Order	Interpolation with a cubic Hermite spline using up to four available quantity values and finite difference schemes to approximate the necessary tangential values.
First Order	Linear Interpolation using two available quantity values.
Zero Order	Constant Interpolation. The last available quantity value is sent to the receiving code.

Parameter	Description
AngularInterpolation	<p>Select the interpolation method for angular coordinates. (Default: NLERP)</p> <p>NLERP Normalized Linear intERPolation. This is computationally cheap, but the angular velocity during a time interval is not constant.</p> <p>SLERP Spherical Linear intERPolation on the unit sphere in 4 dimensions. This guarantees constant angular velocity in each time interval.</p> <p>SQUAD Spherical QUADrangle interpolation. The spherical analogon to cubic splines. This interpolation method yields smooth rotation.</p> <p>Zero Order Constant interpolation.</p>
MeshMotion	<p>Enable moving mesh transformation within the MpCCI server. (Default: off)</p> <p>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.</p> <p>Please enable mesh motion under the following coupling scenarios:</p> <ul style="list-style-type: none"> • 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. <p>Disabling motions will result in a better mapping performance and numerical stability, since the mesh-mesh relationship is determined only once.</p> <p> 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.</p>
MRFCorrect	Enable MRF corrections within the MpCCI server. (Default: off)
UseInitialNBH	 NOT YET DOCUMENTED. Currently this is an experimental feature and might not work as expected.
OverlapCheck	Use the initial mesh-mesh relationship during the moving mesh transformation. (Default: off)
DomainCheck	<p>Only valid with the mesh motion option.</p> <p>Enable a quick overlap check during the creation of mesh pairs. (Default: on)</p> <p>In case you have only tiny overlapping regions between two meshes you should deactivate this check to avoid complains and an abortion by the MpCCI server.</p> <p>Enable a check on the domains of parts or meshes. (Default: off)</p> <p>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.</p> <p> A domain check is expensive and should be deactivated in case of production runs with remeshing.</p>
BaffleShift	Enable the shifting of baffle front and rear side for a geometrical separation of the two sides of a baffle. (Default: off)

Parameter	Description
ConformalMesh	<p>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.</p> <p>The code-adapter has to support the baffle shift feature and must define the thickness of a baffle to the MpCCI server.</p>
TimeTolerance	<p>Enable the conformal mesh mapping algorithm (Default: off)</p> <p>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.</p> <p>! 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.</p> <p>Currently this is an experimental feature and might not always work as expected.</p>
HistorySize	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)
MaxIterQuasiNewton	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 ≥ 0.5 resets the value to its default.
ResidualsQuasiNewton	<p>! This value is only relevant for transient cases with multiple buffers.</p> <p>Define the minimum number of buffers for storing the quantity history. A number between 4 and 32 may be chosen. (Default: 4)</p> <p>Define the maximal number of saved iteration values to be used in the Quasi-Newton relaxation applied to steady simulations (Default: 20). For more details see 3.3.3.1 Quasi-Newton Methods.</p> <p>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 is performed (restart). 0 and 1 lead to no residual check. In the case of steady simulations a choice of 0, leads to a restart each MaxIterQuasiNewton. For more details see 3.3.3.1 Quasi-Newton Methods.</p>

4.7.3 Relation Search

In this section you specify the attributes of the relation search.

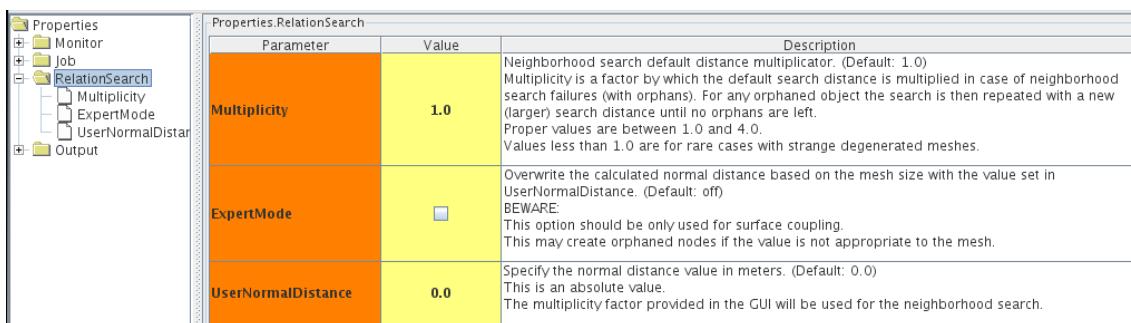


Figure 31: Relation search settings

Parameter	Description
Multiplicity	<p>Neighborhood search default distance factor. (Default: 1.0)</p> <p>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.</p> <p>Proper values are between 1.0 and 4.0.</p> <p>Values less than 1.0 are for rare cases with strange degenerated meshes.</p>
ExpertMode	<p>Overwrite the calculated normal distance based on the mesh size with the value set in UserNormalDistance. (Default: off)</p> <p> This option should be only used for surface coupling. This may create orphaned nodes if the value is not appropriate to the mesh.</p>
UserNormalDistance	<p>Specify the normal distance value in meters. (Default: 0.0)</p> <p>This is an absolute value. The multiplicity factor provided in the MpCCI GUI will be used for the neighborhood search.</p>

4.7.4 Output

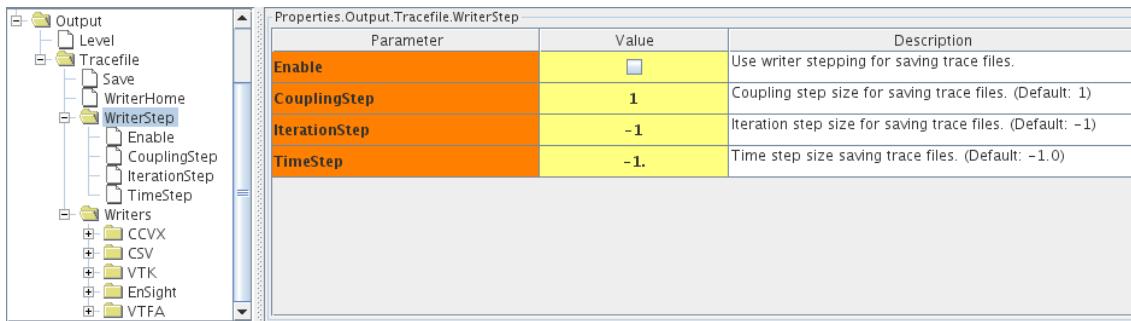


Figure 32: Output settings

In this section you specify the output written by MpCCI and select and configure the writers for which trace files shall be written.

Parameter	Description
Level	Determines how much output MpCCI writes: 0 no output, 1 minimal output (default), 2 additional output, 3 maximal output - for debug only.
Tracefile	
Save	Save trace files for visualization. (Default: on)
WriterHome	Enter a path to save the trace files. (Default: \$MPCCI_JOBDIR)
WriterStep	
Enable	Use writer stepping for saving trace files. (Default: off) It allows saving the results at the specified step interval.
CouplingStep	Coupling step size for saving trace files. (Default: 1)
IterationStep	Iteration step size for saving trace files. (Default: -1)
TimeStep	Time step size for saving trace files. (Default: -1.0)

Parameter	Description
Writers	
CCVX	
Use	Save an MpCCI binary CCVX file. (Default: on)
Orphans	Include orphan level information. (Default: off)
Size	Include element size information. (Default: off)
Nodelds	Include node IDs information. (Default: off)
Elemlds	Include element IDs information. (Default: off)
Globals	Include received global variables. (Default: off)
Slaves	Include slave node flags. (Default: off)
Domains	Include node domains (only valid with domain check - see 4.7.2 Job). (Default: off)
Peaks	Include quantity peak and mean values plot. (Default: off)
Inorm	Include quantity iteration norm values plot. (Default: off)
Periodic	Include replicated periodic parts and quantities. (Default: off)
CSV	
Use	Save a comma-separated-values file (CSV) for point coupling. (Default: off)
Globals	Include received global variables. (Default: off)
VTK	
Use	Save Paraview VTK files. (Default: off)
Orphans	Include orphan level information. (Default: off)
Size	Include element size information. (Default: off)
Binary	Save in big endian binary format. (Default: off)
Nodelds	Include node IDs information. (Default: off)
Elemlds	Include element IDs information. (Default: off)
Midedge	Include mid-edge nodes on quadratic elements. (Default: off)
Group	File content grouped by mesh or part numbers. (Default: mesh)
Slaves	Include slave node flags. (Default: off)
Domains	Include node domains (only valid with domain check - see 4.7.2 Job). (Default: off)
Periodic	Include replicated periodic parts and quantities. (Default: off)
EnSight	
Use	Save EnSight Gold binary files. (Default: off)
Sflush	Save all EnSight Gold files at each coupling step. (Default: off)
Orphans	Include orphan level information. (Default: off)
Size	Include element size information. (Default: off)
Nodelds	Include node IDs information. (Default: off)
Elemlds	Include element IDs information. (Default: off)
Midedge	Include mid-edge nodes on quadratic elements. (Default: off)
Slaves	Include slave node flags. (Default: off)
Domains	Include node domains (only valid with domain check - see 4.7.2 Job). (Default: off)
Periodic	Include replicated periodic parts and quantities. (Default: off)
VTFA	
Use	Save Ceetron VTF ASCII files. (Default: off)
Orphans	Include orphan level information. (Default: off)
Size	Include element size information. (Default: off)
Periodic	Include replicated periodic parts and quantities. (Default: off)

4.8 Go Step

In the Go Step you will configure the start-up of the applications. Following is a detailed description of the MpCCI coupling server parameters and the coupling configuration which is similar to all analysis codes. The further configuration for the analysis codes is described in the appropriate code section of the [Codes Manual](#). How to start, stop and kill the coupled simulation can be found in [IV-2.8 Go Step – Configuring the Application Start-Up and Starting Server and Codes](#).

4.8.1 Configuring the MpCCI Coupling Server

For the MpCCI coupling server the following parameters can be configured (see also [Figure 33](#)):

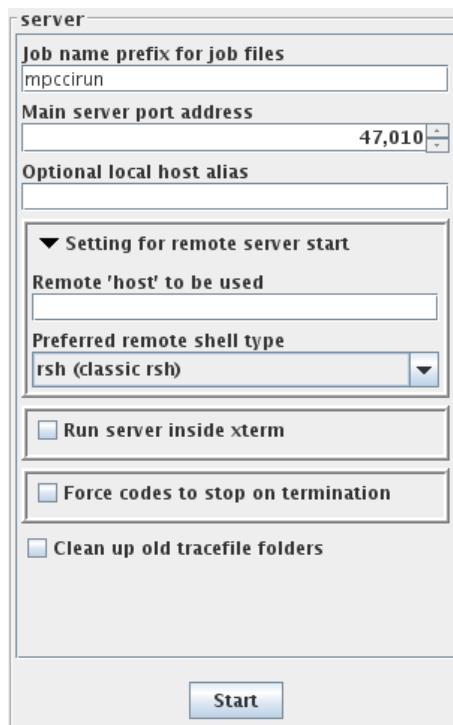


Figure 33: Server settings

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 34](#)):

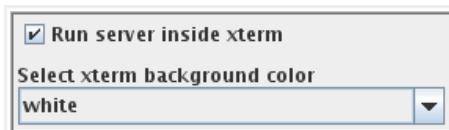


Figure 34: Run server inside xterm settings

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 35](#)):



Figure 35: Force codes to stop on termination settings

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.

4.8.2 Coupling Configuration Parameters

In the coupling configuration ([Figure 36](#)) you can configure the settings for the used coupling algorithm which are further described in [3.4 Coupling Algorithms](#).

Define the coupling scheme offers up to three coupling schemes depending on the code used and depending on the solution type the code uses for the chosen model file:

Explicit-SteadyState for stationary problems

Explicit-Transient for transient problems

Implicit-Transient for iterative coupling

Initial quantities transfer defines the first transfer of the quantities for this code. Select one of receive, send, exchange or skip. The successive transfers will be exchange.

Send mode defines the send communication scheme

always is used for asynchronous communication. The code sends data without taking care of the partner code.

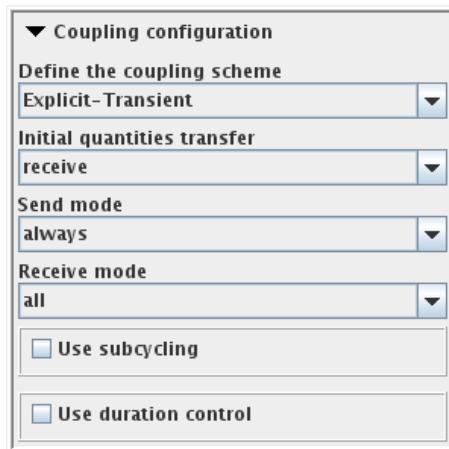


Figure 36: Coupling configuration for a transient problem

onewait is used for synchronous communication. The code sends the data if at least one partner code is ready to receive the data.

allwait is also used for synchronous communication, but the code sends the data only when all partner codes are ready to receive the data.

Receive mode defines the receive communication scheme

all gets data for all requested quantities waiting for them.

available gets what is available from the server without waiting.

any gets all data if at least one quantity is available. Otherwise the receive action is skipped.

complete gets data only if all quantities are available. Otherwise the receive action is skipped.

The other options depend on the coupling scheme used. For transient and stationary problems subcycling and duration control can be used as shown in [▷3.4.6.1 Subcycling Setting in the MpCCI GUI](#). For iterative coupling one can configure iteration and duration control settings as described in [▷3.4.4.4 Settings for Iterative Coupling in the MpCCI GUI and Supported Codes](#).

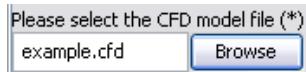
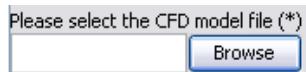
4.8.3 Checking the configuration

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-2.4.7 Environments for Scanner, Checker, Starter, Stopper and Killer](#) for a detailed description.

4.9 Remote File Browser

4.9.1 File Browser Handling

By clicking on the button **[Browse]** 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.

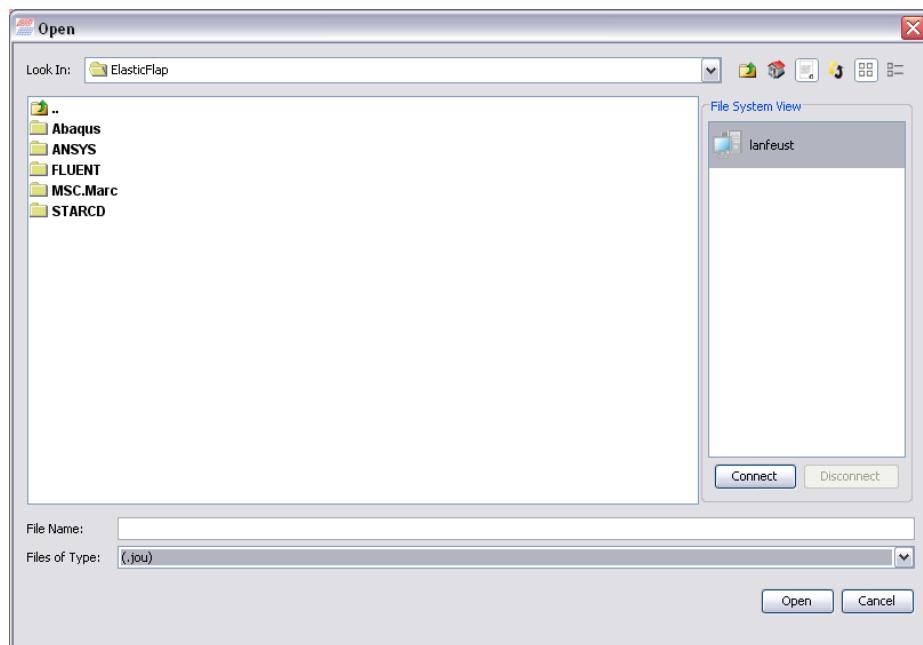


Figure 37: File browser

The remote file browser (Figure 37) 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 provided text field followed by a click on the **[Save]** button.

4.9.2 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 38](#)



Figure 38: 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.



Figure 39: 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 40](#) 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 40: 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 4.5.0-1, Build (Thu Mar 30 07:10:05 2017)
(c) 2004-2017 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:

Subcommand	Short Description	Discussed in
<code>arch [-n]</code>	Print the MpCCI base architecture without a newline or with a newline [<code>-n</code>] at the end and exit. This is different from <code>mpcci info arch</code> which prints the used architecture.	▷ 5.5.1 mpcci arch <small>▷ on page 134</small>
<code>backup <file ...></code>	Make a backup copy of a list of files.	▷ 5.7.1 mpcci backup <small>▷ on page 150</small>
<code>batch <project></code>	Start an MpCCI batch job with project file <code><project></code> .	▷ 5.7.2 mpcci batch <small>▷ on page 151</small>
<code>ccvxcat <port file ...></code>	Catenate and/or send .ccvx files to the MpCCI visualizer.	▷ 5.4.1 mpcci ccvxcat <small>▷ on page 126</small>
<code>clean</code>	Remove all files from the temporary MpCCI directory <code><Home>/mpcci/tmp</code> .	▷ 5.7.8 mpcci clean <small>▷ on page 158</small>
<code>cosimpre</code>	Use co-simulation pre-check tools	▷ 5.4.2 mpcci cosimpre <small>▷ on page 127</small>
<code>doc</code>	View the MpCCI documentation.	▷ 5.5.2 mpcci doc <small>▷ on page 135</small>
<code>env</code>	Print out the environment used by MpCCI in various formats for further processing.	▷ 5.5.4 mpcci env <small>▷ on page 138</small>
<code>fsimapper</code>	Launch the MpCCI visualizer with FSI mapping plugin.	▷ 5.3.1 mpcci fsimapper <small>▷ on page 119</small>
<code>gui</code>	Launch the MpCCI GUI.	▷ 5.3.2 mpcci gui <small>▷ on page 120</small>

Subcommand	Short Description	Discussed in
<code>home [-n]</code>	Print the MpCCI home directory without a newline or with a newline <code>[-n]</code> at the end and exit. This is in fact a shortcut for <code>mpcci info home</code> .	▷ 5.5.5 mpcci home ◄ on page 139
<code>info</code>	Print general information about MpCCI.	▷ 5.5.3 mpcci info ◄ on page 136
<code>kill</code>	Platform independent process kill based on command line pattern matching.	▷ 5.7.9 mpcci kill ◄ on page 159
<code>license</code>	Manage the license server and print license related information.	▷ 5.6.1 mpcci license ◄ on page 142
<code>list</code>	List information about the MpCCI installation and the supported codes.	▷ 5.6.2 mpcci list ◄ on page 143
<code>loutil</code>	Run the FLEXnet <code>loutil [OPTIONS]</code> command delivered with MpCCI. Avoid running the loutil command installed by codes other than MpCCI.	▷ 5.6.3 mpcci loutil ◄ on page 144
<code>logviewer</code>	Launch the MpCCI Logfile Viewer.	▷ 5.3.3 mpcci logviewer ◄ on page 121
<code>monitor</code>	Launch the MpCCI online monitor.	▷ 5.3.4 mpcci monitor ◄ on page 122
<code>morpher</code>	Launch the MpCCI grid morpher.	▷ 5.4.3 mpcci morpher ◄ on page 128
<code>observe</code>	Start the MpCCI file observer.	▷ 5.4.4 mpcci observe ◄ on page 131
<code>ps</code>	Unix <code>ps -ef</code> compatible ps for all platforms.	▷ 5.7.10 mpcci ps ◄ on page 161
<code>ptoj</code>	Convert an MpCCI project file into an MpCCI job properties file.	▷ 5.7.11 mpcci ptoj ◄ on page 162
<code>server</code>	Start the MpCCI server.	▷ 5.7.12 mpcci server ◄ on page 163
<code>ssh</code>	Check/fix your ssh installation.	▷ 5.6.4 mpcci ssh ◄ on page 145
<code>test</code>	Run some install/communication tests.	▷ 5.6.5 mpcci test ◄ on page 146
<code>top</code>	Display process top list / Launch the taskmanager.	▷ 5.7.13 mpcci top ◄ on page 167
<code>visualize</code>	Launch the MpCCI visualizer.	▷ 5.3.5 mpcci visualize ◄ on page 123
<code>where <CMD></code>	Find all locations of the executable <code><CMD></code> in the PATH.	▷ 5.5.6 mpcci where ◄ on page 140
<code>xterm</code>	Start a process inside an xterm.	▷ 5.4.5 mpcci xterm ◄ on page 132

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.

Subcommand	Short Description	Discussed in
<code>fsimapper</code>	Launch the MpCCI visualizer with FSI mapping plugin.	▷ 5.3.1 mpcci fsimapper ◁ on page 119
<code>gui</code>	Launch the MpCCI GUI.	▷ 5.3.2 mpcci gui ◁ on page 120
<code>logviewer</code>	Launch the MpCCI Logfile Viewer.	▷ 5.3.3 mpcci logviewer ◁ on page 121
<code>monitor</code>	Launch the MpCCI online monitor.	▷ 5.3.4 mpcci monitor ◁ on page 122
<code>visualize</code>	Launch the MpCCI visualizer.	▷ 5.3.5 mpcci visualize ◁ on page 123

5.3.1 mpcci fsimapper

```
Usage:  
    mpcci fsimapper [-]option  
  
Synopsis:  
'mpcci 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.  
  
-convert <FORMAT> <model>  
    Use this option to convert native ANSYS models into MapLib format.  
  
-help  
    This screen.  
  
-maxmem <MB>  
    Set the maximum memory used by the visualizer.  
  
-scan <FORMAT> <model>  
    Use this option to scan models with the MpCCI FSIMapper.  
    Supported scanner formats are ABAQUS, ANSYS, LSDYNA, FLUENT, NASTRAN,  
    CFXCSV, FLOTHERMMAPLIB, FLOEFDMAPIB, FLOTHERMXT, FINETURBO and ENSIGHT.  
    The scanner output will be written to stdout.
```

The MpCCI Visualizer is started with the FSI mapping plugin activated. The use of the FSIMapper is described in [▷ 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 Logviewer.
```

```
Options:  
-help This screen.
```

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 ▷ 6 MpCCI Visualizer ◁.

5.3.5 mpcci visualize

Usage:

```
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

```
[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.

```
job-000011.ccvx  
job-000012.ccvx  
.....
```

Examples:

Display a single file

```
mpcci visualize name.ccvx  
mpcci visualize my-tracefiles/name
```

Display a file located on a remote host

```
mpcci visualize remotehost.network.com:/home/user/jobs/tracefiles/last.ccvx
```

Display a series of files [job0000.ccvx job0001.ccvx ...]

```
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.
-skipstep <step>	Define how many steps should be skipped.
-skiptime <dt>	Define the time intervall 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 ▷6 MpCCI Visualizer ◁.

5.4 MpCCI Tools

The commands in this section are several helper applications to be used with MpCCI.

Subcommand	Short Description	Discussed in
<code>ccvxcat <port file ...></code>	Catenate and/or send .ccvx files to the MpCCI visualizer.	▷ 5.4.1 mpcci ccvxcat ◁ on page 126
<code>cosimpre</code>	Use co-simulation pre-check tools	▷ 5.4.2 mpcci cosimpre ◁ on page 127
<code>morpher</code>	Launch the MpCCI grid morpher.	▷ 5.4.3 mpcci morpher ◁ on page 128
<code>observe</code>	Start the MpCCI file observer.	▷ 5.4.4 mpcci observe ◁ on page 131
<code>xterm</code>	Start a process inside an xterm.	▷ 5.4.5 mpcci xterm ◁ on page 132

5.4.1 mpcci ccvxcat

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  
.....
```

Options:

-connect <port@host>	Define the socket of the visualizer (default=47003).
-help	This screen.
-jobname <jobname>	Set the job name for the visualizer communication.
-nopcheck	Do not check and warn about used port numbers.
-replace <old> <new>	Replace string "old" with "new" in part names.
-single	Load a single file.
-skipstep <step>	Define how many steps should be skipped.
-skiptime <dt>	Define the time intervall which should be skipped.

5.4.2 mpcci cosimpre

Usage:

```
mpcci cosimpre [-]OPTIONS
```

Synopsis:

'mpcci cosimpre' is used to prepare the co-simulation coupled regions setup

Options:

- 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 mpcci morpher

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/m4.4/bin/lnx4_x64/mpcci_morpher.exe"
```

with 'LD_LIBRARY_PATH'

```
"<Home>/work/m4.4/bin/lnx4_x64"
"/home/compeng/compiler/icc10.1/linux_em64t/lib"
```

Grid morpher daemon 3.1B, double precision
Build Mar 30 2017, 03:22:02
Copyright (c) 2004-2013, Fraunhofer SCAI.

License checkout...
Your license will expire in 1372 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

-enocheck	Skip all edge checks
-rlen[gth] #f #f	Min/max allowed relative length change of an edge 0.0 < Min length < 1.0 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 < 1.0
                  1.0 < Max area < ?
-aar[ea] #f #f Min/max allowed absolute face area
-fas[pect] #f   Max. allowed face aspect ratio [1..500]
-fsk[ew]    #f   Max. allowed skewness of faces [0.5..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

```

-dbl[ayers] #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...0.3]
-initlast
-residual
-mrelax    #f Morphing relaxation factor: ]0.0...2.0]
-local
-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
-listen      #d  Port number for socket communications
-debug        Save a -morph.vrt file after morphing
-h[elp]       This screen and exit
-out[put]     #c  Information output level: [f|v|d|q]
                f = full, verbose + displacements received
                v = verbose
                q = quiet
                d = default between quiet and verbose
-s[etup]       List setup and exit
-csca[le]     #f  Scale vertices read by factor #f
-dsca[le]     #f  Scale displacements by factor #f
-wait         #d  Set waiting timeout in seconds, 0=never timeout
-noparamcheck Do not check the limits of any parameter
-trace        #c  Save received vertices in a trace file
-vis[ualize]   Visualize grid while morphing
```

5.4.4 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.

5.4.5 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.

Subcommand	Short Description	Discussed in
<code>arch [-n]</code>	Print the MpCCI base architecture without a newline or with a newline <code>[-n]</code> at the end and exit. This is different from <code>mpcci info arch</code> which prints the used architecture.	▷ 5.5.1 mpcci arch on page 134
<code>doc</code>	View the MpCCI documentation.	▷ 5.5.2 mpcci doc on page 135
<code>env</code>	Print out the environment used by MpCCI in various formats for further processing.	▷ 5.5.4 mpcci env on page 138
<code>home [-n]</code>	Print the MpCCI home directory without a newline or with a newline <code>[-n]</code> at the end and exit. This is in fact a shortcut for <code>mpcci info home</code> .	▷ 5.5.5 mpcci home on page 139
<code>info</code>	Print general information about MpCCI.	▷ 5.5.3 mpcci info on page 136
<code>where <CMD></code>	Find all locations of the executable <code><CMD></code> in the PATH.	▷ 5.5.6 mpcci where on page 140

5.5.1 `mpcci arch`

`mpcci arch` is simply a shortcut for `mpcci info arch`, see also [▷ 5.5.3 `mpcci info` ◁ on page 136](#).

`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 4.5 ◁](#).

5.5.2 **mpcci doc**

This command can be used to view MpCCI documentation, the available documentation can be listed with the option **-list**. The documentation is located in the "<MpCCI home>/doc" directory and can be accessed directly as well.

Currently two kinds of documentation are available:

mpcci doc LicenseAdministration The FlexNet Publisher License Administration

mpcci doc MpCCIdoc The MpCCI documentation.

5.5.3 `mpcci info`

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:

<code>-arch</code>	MpCCI basic architecture token.
<code>-arch32</code>	MpCCI 32 bit architecture token.
<code>-arch64</code>	MpCCI 64 bit architecture token.
<code>-build</code>	Build date of the installed MpCCI release.
<code>-help</code>	This screen.
<code>-home</code>	MpCCI home path.
<code>-java</code>	Java command used by MpCCI.
<code>-javaver</code>	Java version of the java command.
<code>-jobid</code>	Jobid used by MpCCI.
<code>-liba32</code>	Pathname of the 32 bit libmpcci.a (if available).
<code>-liba64</code>	Pathname of the 64 bit libmpcci.a (if available).
<code>-libdl32</code>	Pathname of the 32 bit libmpcci.so (if available).
<code>-libdl64</code>	Pathname of the 64 bit libmpcci.so (if available).
<code>-make</code>	Make command used by MpCCI.
<code>-patches</code>	Patchlevel of the installed MpCCI release.
<code>-perl</code>	Pathname of the perl command used by MpCCI.
<code>-perlinc</code>	@INC list used by Perl.
<code>-perlver</code>	Version of the running Perl.
<code>-release</code>	Full MpCCI release token.
<code>-remcp</code>	Pathname of remote copy command used by MpCCI.
<code>-remsh</code>	Pathname of remote shell command used by MpCCI.
<code>-rshtype</code>	MpCCI server remote shell type used.
<code>-userid</code>	Userid used by MpCCI.
<code>-version</code>	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 4.5 ▷](#).

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 [▷ 2.7.3 Remote Shell and Remote Copy ◁](#) for more information on remote shells.

rcp Gives the path to the rcp command, see also [▷ 2.7.3 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.

Subcommand	Short Description	Discussed in
<code>license</code>	Manage the license server and print license related information.	▷ 5.6.1 <code>mpcci license</code> ◁ on page 142
<code>list</code>	List information about the MpCCI installation and the supported codes.	▷ 5.6.2 <code>mpcci list</code> ◁ on page 143
<code>lmutil</code>	Run the FLEXnet <code>lmutil [OPTIONS]</code> command delivered with MpCCI. Avoid running the lmutil command installed by codes other than MpCCI.	▷ 5.6.3 <code>mpcci lmutil</code> ◁ on page 144
<code>ssh</code>	Check/fix your ssh installation.	▷ 5.6.4 <code>mpcci ssh</code> ◁ on page 145
<code>test</code>	Run some install/communication tests.	▷ 5.6.5 <code>mpcci test</code> ◁ on page 146

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 client license version.

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) 1989-2016 Flexera Software LLC. All Rights Reserved.  
usage:           lmutil lmborrow -status  
                lmutil lmborrow -purge  
                lmutil lmborrow -purge -status  
                lmutil lmborrow -clear  
                lmutil lmborrow {all|vendor} dd-mmm-yyyy:[time]           lmutil lmborrow -return [-c licfile]  
-fqdn] [-vendor name] feature [-bv version]  
                lmutil lmdiag [-c licfile] [-n]  
                lmutil lmdown [-c licfile] [-q] [-all] [-vendor name] [-force] [-help]  
                lmutil lmhostid [-ptype (PHY|AMZN|VM)] [-ether|-internet (v4|v6)|-user|-n|-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}]  
                    [-odecimal] [-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 lmremove [-c licfile] feature user host display  
                lmutil lmremove [-c licfile] -h feature host port handle  
                lmutil lmremove [-c licfile] [-tsborrow <client_host>] | [-tsborrowstat]  
                lmutil lmreread [-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  
                lmutil lmminfo [-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.

Subcommand	Short Description	Discussed in
<code>backup <file ...></code>	Make a backup copy of a list of files.	▷ 5.7.1 mpcci backup ◄ on page 150
<code>batch <project></code>	Start an MpCCI batch job with project file <code><project></code> .	▷ 5.7.2 mpcci batch ◄ on page 151
<code>clean</code>	Remove all files from the temporary MpCCI directory <code><Home>/mpcci/tmp</code> .	▷ 5.7.8 mpcci clean ◄ on page 158
<code>kill</code>	Platform independent process kill based on command line pattern matching.	▷ 5.7.9 mpcci kill ◄ on page 159
<code>ps</code>	Unix <code>ps -ef</code> compatible ps for all platforms.	▷ 5.7.10 mpcci ps ◄ on page 161
<code>ptoj</code>	Convert an MpCCI project file into an MpCCI job properties file.	▷ 5.7.11 mpcci ptoj ◄ on page 162
<code>server</code>	Start the MpCCI server.	▷ 5.7.12 mpcci server ◄ on page 163
<code>top</code>	Display process top list / Launch the taskmanager.	▷ 5.7.13 mpcci top ◄ on page 167

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>`.

Usage:

```
mpcci batch [OPTIONS] <projectname>
mpcci batch [OPTIONS] OPENPBS ...
mpcci batch [OPTIONS] LSF ...
mpcci batch [OPTIONS] PBS ...
mpcci batch [OPTIONS] TORQUE ...
mpcci batch [OPTIONS] N1GE ...
mpcci batch [OPTIONS] PBSPRO ...
mpcci batch [OPTIONS] GLOBUS ...
mpcci batch [OPTIONS] LOADLEVELER ...
```

Synopsis:

- 'mpcci batch' is used to start an MpCCI job in batch mode.
- 'mpcci batch OPENPBS' is used to control a OPENPBS 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 TORQUE' is used to control a TORQUE batch job.
- 'mpcci batch N1GE' is used to control a N1GE batch job.
- 'mpcci batch PBSPRO' is used to control a PBSPRO batch job.
- 'mpcci batch GLOBUS' is used to control a GLOBUS batch job.
- 'mpcci batch LOADLEVELER' is used to control a LOADLEVELER 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.
-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.

-np <codeA:N> <codeB:M>
Bind each code to the specified total number of cores.

-useAbsolutePath
Use current local MpCCI installation path on remote access.

globus
GLOBUS batch system control.

loadleveler
LOADLEVELER batch system control.

lsf
LSF batch system control.

n1ge
N1GE batch system control.

openpbs
OPENPBS batch system control.

pbs
PBS batch system control.

pbspro
PBSPRO batch system control.

torque
TORQUE batch system control.
```

See ▷ 3.6 Coupled Analysis in Batch Mode◀ for details.

5.7.3 mpcci batch LSF

Usage:

```
mpcci batch LSF submit [BATCH-OPTIONS] <projectname>
mpcci batch LSF status <jobid>
mpcci batch LSF kill <jobid>
```

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 system are also 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 N1GE**

The following Sun Grid Engine family queuing system are also available and supported by MpCCI:

- SGE
- N1GE
- SGEEE

Usage:

```
mpcci batch N1GE submit [BATCH-OPTIONS] <projectname>
mpcci batch N1GE status <jobid>
mpcci batch N1GE kill <jobid>
```

Synopsis:

'mpcci batch N1GE' is used to manage an MpCCI job running under N1GE.

Batch commands:

-help	This screen.
kill <jobid>	Kill a N1GE batch job.
status <jobid>	Display a N1GE batch job status.
submit [BATCH-OPTIONS] <projectname>	Submit a N1GE MpCCI job.

See ▷ 3.6 Coupled Analysis in Batch Mode ◁ for details.

5.7.6 mpcci batch LoadLeveler

Usage:

```
mpcci batch LOADLEVELER submit [BATCH-OPTIONS] <projectname>
mpcci batch LOADLEVELER status <jobid>
mpcci batch LOADLEVELER kill <jobid>
```

Synopsis: 'mpcci batch LOADLEVELER' is used to manage an MpCCI job running under LOADLEVELER.

Batch commands:

-help	This screen.
kill <jobid>	Kill a LOADLEVELER batch job.
status <jobid>	Display a LOADLEVELER batch job status.
submit [BATCH-OPTIONS] <projectname>	Submit a LOADLEVELER MpCCI job.

See ▷ 3.6 Coupled Analysis in Batch Mode ◁ for details.

5.7.7 mpcci batch GLOBUS

Usage:

```
mpcci batch GLOBUS submit [BATCH-OPTIONS] <projectname>
mpcci batch GLOBUS status <jobid>
mpcci batch GLOBUS kill <jobid>
```

Synopsis:

'mpcci batch GLOBUS' is used to manage an MpCCI job running under GLOBUS.

Batch commands:

-help	This screen.
kill <jobid>	Kill a GLOBUS batch job.
status <jobid>	Display a GLOBUS batch job status.
submit [BATCH-OPTIONS] <projectname>	Submit a GLOBUS MpCCI job.

See ▷ 3.6 Coupled Analysis in Batch Mode ◁ for details.

5.7.8 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.

5.7.9 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 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.

5.7.10 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 <user name>"

Options:
-help This screen.

5.7.11 `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.
```

5.7.12 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.

```
-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 has 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.

-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.

5.7.13 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 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.

6.1 Using the MpCCI Visualizer

6.1.1 Data Flow

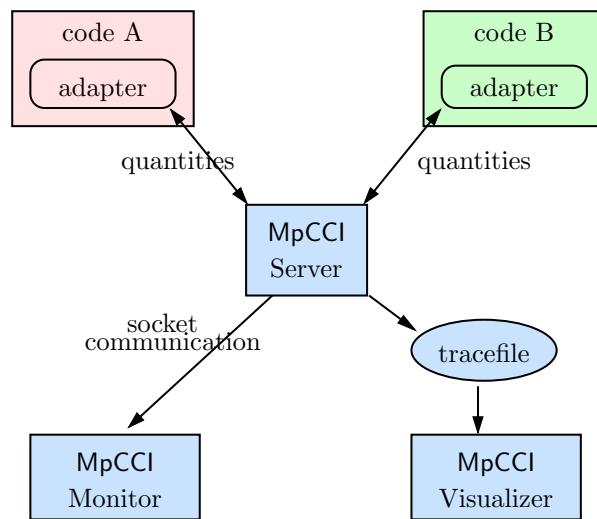


Figure 1: Data flow for the MpCCI Visualizer. The MpCCI Server Process collects the exchanged data and writes them to the tracefile.

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 Edit Step. This is achieved by selecting the writer's (e.g. CCVX, VTK or VTFA) option Use in the Edit Step of the MpCCI GUI. See [▷4.7 Edit Step◁](#) for a description of the Edit Step and “[Getting Started](#)” for a general description of the coupling process.

The name of the tracefile is set in the Edit 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.

6.1.2 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.

6.1.3 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 ▷ 5.3.4 `mpcci monitor` ◁ for a detailed command description.

6.1.4 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 ▷ 5.3.5 `mpcci visualize` ◁ for a detailed command description.

6.2 MpCCI Visualizer for .ccvx and online monitoring

6.2.1 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 `[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.

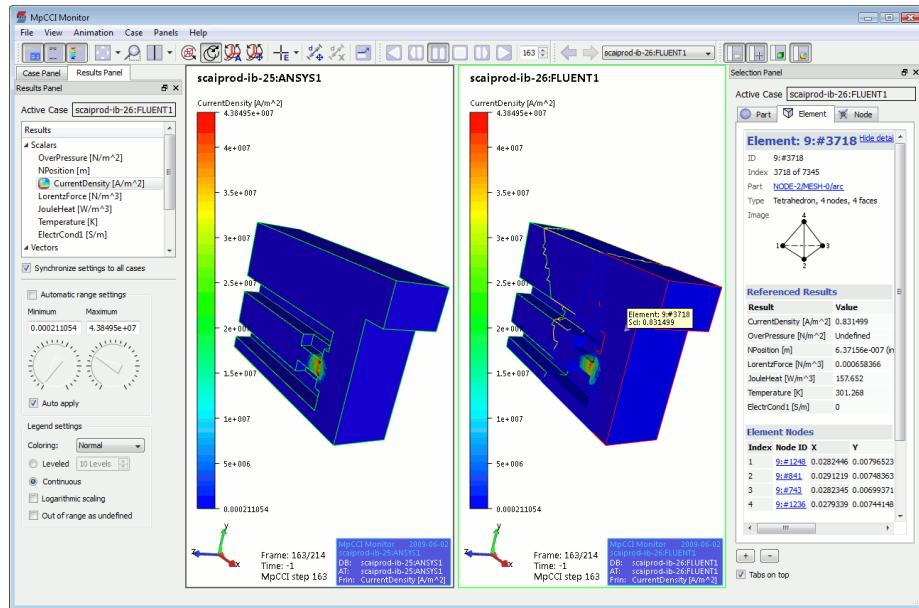


Figure 2: Main window of the MpCCI Visualizer for .ccvx and online monitoring

6.2.2 Main Window

The main window of the MpCCI Visualizer for .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.

6.2.3 Menus and Toolbars

6.2.3.1 "File" Menu

Icon	Option	Key	Description
File→Open...		Ctrl+O	Browse for a local ccvx file or type a remote ccvx file in the form [[user]@hostname:]path/to/file[.ccvx(.gz)]. This option is available in the visualizer mode.
File→Connect to Host...		Ctrl+H	Connect to an MpCCI server for monitoring. This menu is available in the monitor mode.
File→Disconnect from Host...			Disconnect from an MpCCI server. This menu is available in the monitor mode.
File→Close			Close the current file or monitored job.
File→Export Image...		Ctrl+S	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.

- 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.

6.2.3.2 "View" Menu

Icon	Option	Key	Description
	View→Info Box		- Show/hide the info box in the active view.
	View→Text		- Show/hide textual information in the active view.
	View→Legend		- Show/hide part and color legends in the active view.
	View→Modify→Reset View		- Reset the viewpoint of the active view to its initial value.
	View→Modify→Frame Model		- Reset the viewpoint so that the whole model fits the active view.
	View→Rubber Band Zoom		- Active rubber band zooming. Define new zoom by dragging a rectangle in one of the views using the left mouse button.
	View→Viewports...		- Select a viewport configuration. Several configurations are available as illustrated by the icons. A maximum number of four views is supported.
	View→Perspective Projection		- Switch between perspective and orthographic projection.
	View→Move as Outline		- Draw the model as outlines when it is being panned or rotated.
	View→Synchronous Navigation		- Apply navigation to all views synchronously.
	View→Automatic Rotation Point		- The nearest model surface point will be used as rotation point instead of a fixed point.
	View→Set Rotation Point		- Select a new fixed rotation point by clicking into one of the views using the left mouse button.
	View→Navigation Mode→Zoom		- The model will never be penetrated or clipped when zooming into it.
	View→Navigation Mode→Walk		- The model will be penetrated and clipped when zooming into it. Only compatible with perspective projection.
	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	[F10] Enable/disable quad buffered active stereo rendering for all views. This option requires a suitable hardware and driver installation.
	View→Full Screen	[F11] 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.

6.2.3.3 "Animation" Menu

Icon	Option	Key	Description
	Animation→Play Backward	-	Play the animation of coupling steps in backward direction.
	Animation→Step Backward	-	Go one animation step backward.
	Animation→Pause	-	Pause the animation of coupling steps.
	Animation→Stop	-	Stop the animation of coupling steps and go back to the first step.
	Animation→Step Forward	-	Go one animation step forward.
	Animation→Play	-	Play the animation of coupling steps in forward direction.
	Animation→Record	-	Record the animation of coupling steps as an animated GIF, AVI, WMV, MP4.

6.2.3.4 "Case" Menu

Icon	Option	Key	Description
	Case→Next Case	-	Load the next available case (i.e. coupling results of a code) into the active view.
	Case→Previous Case	-	Load the previous available case into the active view.

6.2.3.5 "Panels" Menu

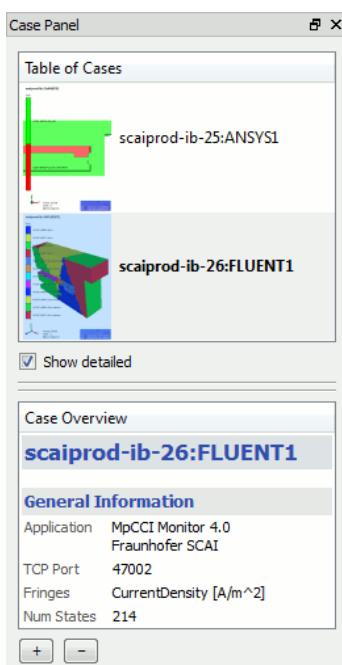
Icon	Option	Key	Description
	Panels→Case Panel	-	Show/hide the panel containing the list of available cases.
	Panels→Selection Panel	-	Show/hide the panel containing information on the currently selected items in the active view.
	Panels→Settings Panel	-	Show/hide the panel for modification of part draw styles and superimposition offset.
	Panels→Results Panel	-	Show/hide the panel for selection of available quantities for visualization.
	Panels→Extraction Panel	-	Show/hide the panel for extraction features, i.e. cut planes and iso surfaces.
	Panels→FSIMapper	-	Show the FSIMapper panel for file based mapping; only available in FSIMapper mode.

6.2.3.6 "Help" Menu

Icon	Option	Key	Description
	Help→Help Contents	[F1]	Show help for MpCCI Visualizer.
	Help→About	-	Show application information and system diagnostics.

6.2.4 Panels

6.2.4.1 "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.

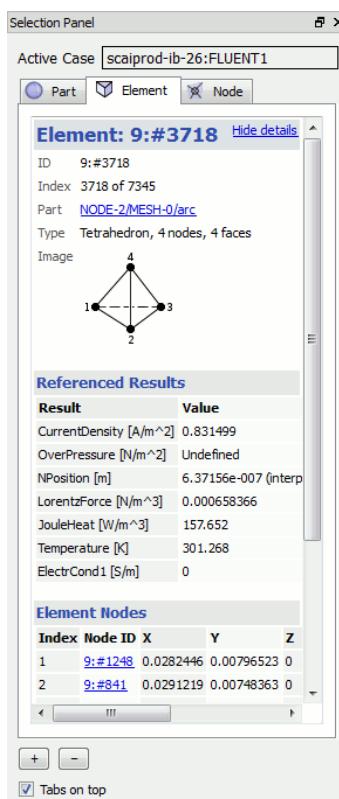
The **Show detailed** checkbox allows switching to a more compact listing. The description shown in the *Case Overview* area is HTML formatted. Use the context menu to copy it to the clipboard or save it to a file. To modify the font size, one either presses the buttons **[+]** or **[-]** or presses the keys **[Ctrl +]** and **[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.

Figure 3: Case panel.

Mouse action	Description
Left mouse button click	Select case and show further case information in the <i>Case Overview</i> area.
Left mouse button double click	Load case into the active view as primary case.
Left mouse button drag	Drag case entry onto a view to show it as primary case.
Left mouse button drag + [ALT]	Drag case entry onto a view to superimpose it as additional case.
Right mouse button click	Use the context menu to assign or superimpose the selected case.

6.2.4.2 "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 **[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 **[-]** or presses the keys **[Ctrl +]** and **[Ctrl -]**.

The **Tabs on top** checkbox allows moving the *Part*, *Element*, and *Node* tabs to the side.

Figure 4: Selection panel.

6.2.4.3 "Settings" Panel

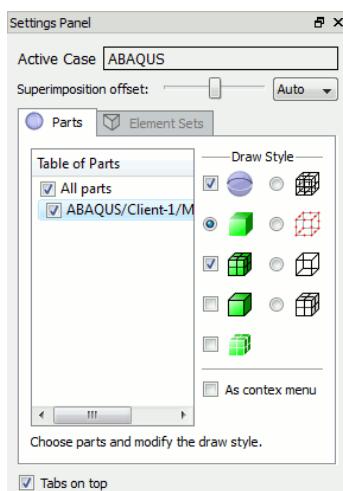


Figure 5: Settings panel.

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 **Table of Parts** lists the parts of the current active case in a hierachic 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. 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.

Icon	Option	Description
	Visible	Show or hide the selected parts.
	Show as surface	Show surface geometry of selected parts as surface.
	Show as lines	Show surface geometry of selected parts as lines.
	Show as points	Show surface geometry of selected parts as points.
	Show as outlines	Show surface geometry of selected parts as outlines.
	Show as hidden lines	Show surface geometry of selected parts as lines, with all hidden lines removed.
	Overlay mesh lines	Show the edges of finite elements as lines on top of the surface geometry of parts. Only applies to surface draw style.
	Overlay mesh outlines	Show the outlines over on top of the surface geometry of parts. Only applies to surface draw style.
	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.

6.2.4.4 "Results" Panel

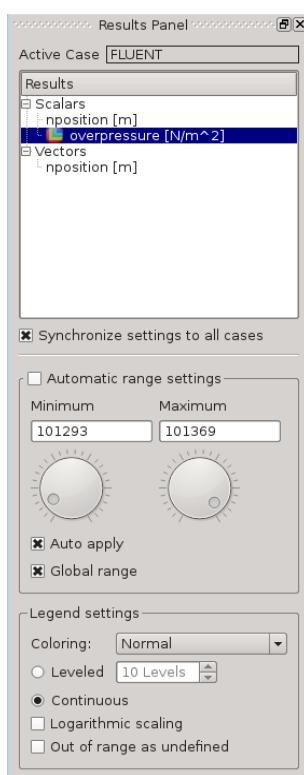


Figure 6: Results panel.

also synchronized for all cases.

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.

Initially, the minimum and maximum range values can be different for individual cases, thus also the color fringes will appear different. To overcome this issue, set the range values manually, so that color fringes are

Settings for Scalar Results

Option	Description
Automatic range settings	Choose <i>Minimum</i> and <i>Maximum</i> range for color fringes automatically, otherwise define range values manually.
Minimum	Define the minimum scalar value for the color legend. One can either use the numeric input field or dial widget.
Maximum	Define the maximum scalar value for the color legend. One can either use the numeric input field or dial widget. The maximum cannot be lower than the minimum value.
Auto apply	Automatically apply new <i>Minimum</i> and <i>Maximum</i> range values. One should disable this option for large datasets, when a color fringes update takes a lot of time.

Global range	Use the minimum and maximum values from all frames as global range value. If this option is deactivated, the range will be adapted to the local minimum and maximum values of the current displayed frame.
Legend settings	Modify legend coloring used for color fringes.
Coloring	Select a pre-defined color scale for color fringes. The most common color scales are "Default", a rainbow color scale, and "Metal casting".
Leveled	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.
Continuous	Use a continuous color scale, with smooth transitions from color to color.
Logarithmic	Apply logarithmic scaling to color fringes. This is only supported for leveled color scales.
Out of range as undefined	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

Option	Description
Show absolute vector length	Vector length representation uses the absolute length value.
Draw vectors as lines	Change the vector arrows representation to lines.
Scaling	Scaling factor for vector glyphs. The scaling factor relates the maximum possible glyph size with the dataset's bounding box diagonal.
Draw skip	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.
Coloring	Determines glyph coloring, using either a <i>Constant</i> color, as defined by the colored button, or <i>Color by scalar</i> in which case the glyph color corresponds to the color fringes of the active scalar result.

Settings for Displacement Results

Option	Description
Scaling	Scaling factor for displacement vectors. A scaling factor greater than one visually overemphasizes the displacement.

6.2.4.5 "Extraction" Panel

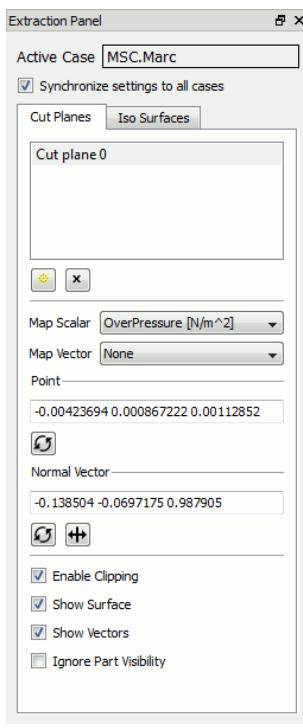


Figure 7: Extraction panel.

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.

6.2.5 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.

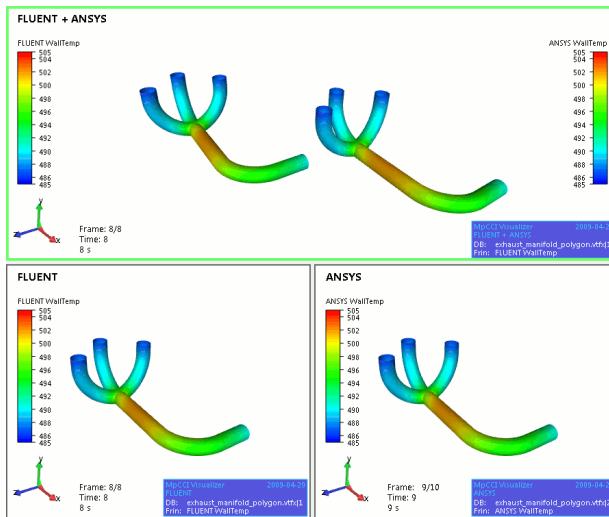


Figure 8: Viewports area. The active view is highlighted by a green border.

6.2.5.1 View Mouse Interaction

Mouse Button	Operation	Key	Description
Left	Drag	-	Rotate
Middle	Drag	-	Zoom
Right	Drag	-	Pan
Mouse wheel	Scroll	-	Zoom
Right	Click	-	Show context menu.
Left	Click	[Ctrl]	Select item. Item information will be shown in the selection panel. Click into empty view area to clear the item selection.
Right	Click	[Ctrl]	Show/hide part under cursor. Click into empty view area to show all parts again.
Left	Drag	[Ctrl]+[Shift]	Rotate active cut plane
Middle	Drag	[Ctrl]+[Shift]	Displace active cut plane into normal direction
Right	Drag	[Ctrl]+[Shift]	Pan active cut plane
Mouse wheel	Scroll	[Ctrl]+[Shift]	Displace active cut plane into normal direction

6.2.5.2 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.

Option	Description
 Select Node	Activates node selection mode. Click into the view using the left mouse button to select a node.
 Select Part	Activates part selection mode. Click into the view using the left mouse button to select a part.
 Select Element	Activates element selection mode. Click into the view using the left mouse button to select an element.
 Reset View	Reset the viewpoint of the view to its initial value.
 Frame Model	Reset the viewpoint so that the whole model fits the view.
 Show/Hide→Text	Show/hide textual information in the view.

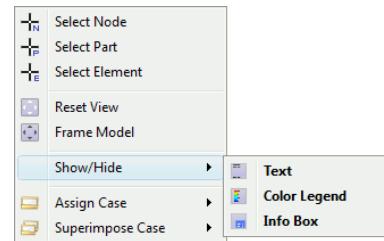


Figure 9: View context menu.

-  **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.

6.2.6 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.

Option	Description
Application Settings	
Navigation profile	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.
Toolbar icon size	
Synchronized navigation	Dis/Enable synchronized navigation on application start.
Rendering Settings	
Fast single frame drawing	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.
Use flipbook animation	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.
Maximum animation FPS	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.
Use software OpenGL	Use software OpenGL rendering, bypassing the graphics hardware. This option is only available for Windows platforms.

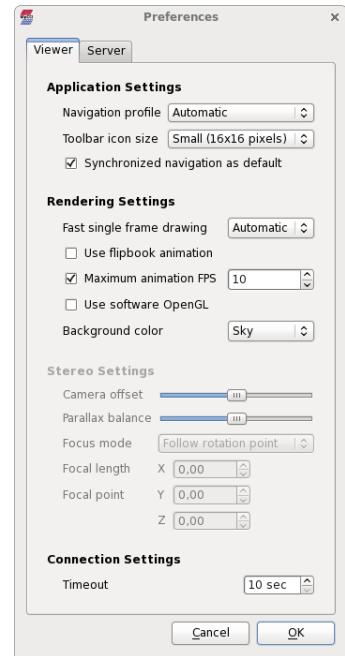


Figure 10: Preferences viewer dialog.

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 point 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.

6.2.7 Preferences Server Dialog

The preferences dialog allows setting several monitor preferences related to the co-simulation. Details for each setting can be found at [4.7.1 Monitor](#).

Option	Description
Server Information	
Orphans	Dis/Enable the send of orphans information.
Node Ids	Dis/Enable the send of node ids.
Element Ids	Dis/Enable the send of element ids.
Element Sizes	Dis/Enable the send of element sizes.
Global Variables	Dis/Enable the send of global variables values.
Slave Nodes	Dis/Enable the send of slaves node.
Node Domains	Dis/Enable the send of node domains information.
Peak Values	Dis/Enable the send of peak values.
Iteration Norm And Mean Values	Dis/Enable the send of iteration norm and mean values.
Element Normals	Dis/Enable the send of element normals.
Absolute Quantity Difference	Dis/Enable the send of absolute quantity difference.
Relative Quantity Difference	Dis/Enable the send of relative quantity difference.
Periodic Replicates	Dis/Enable the send of the replicated periodic parts and quantities.
Aitken Relaxation Value	Dis/Enable the send of the Aitken relaxation value.

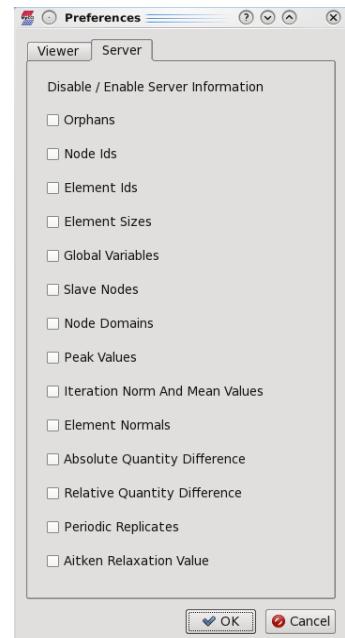


Figure 11: Preferences server dialog.

6.2.8 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.

Option	Description
Store Animated Files	
GIF	Select GIF export format.
AVI	Select AVI export format.

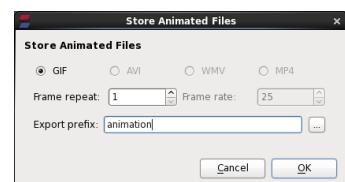


Figure 12: Store Animated Files panel.

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.

A copy of the Libav can be downloaded via <http://libav.org/download.html>. The library offers a simple `./configure` and `./install` mechanism on Linux systems whereas on Windows already compiled binaries can be downloaded from <http://win32.libav.org/win64/> for 64bit operating system resp. <http://win32.libav.org/win32/> for Windows 32bit.

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

Option	Description ([] = default)
<code>-prefix=PREFIX</code>	install binaries in PREFIX []

Licensing options:

<code>-enable-gpl</code>	allow use of GPL code, the resulting libs and binaries will be under GPL [no]
<code>-enable-version3</code>	upgrade (L)GPL to version 3 [no]
<code>-enable-nonfree</code>	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.

6.2.9 Error Dialog

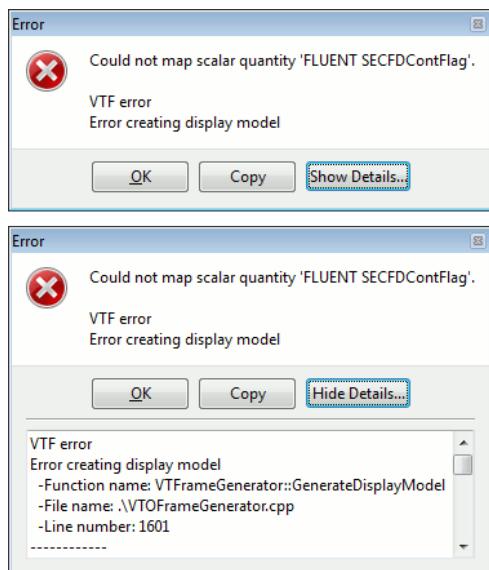


Figure 13: Error dialog with optional details.

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 *bt_mpcci_visualizer.txt* in the execution path which provides useful debugging information for the MpCCI support.

6.2.10 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):

-help	: Overview of command line options
-listen <port>	: Start listening for incoming connections. Default Monitor port 47002. Default Visualize port 47003.
-connect <port@host>	: Connect to one or more servers
-monitor	: Switch from Visualize mode (default) to Monitor mode
-netdevice <string>	: Use the specified network device for monitoring
-modulo <number>	: Save memory by showing only <number> steps
-maxmem <number>	: Save memory by using at most <number> MB for steps
-jobname <name>	: Default job name
-updatedelay <ms>	: Scene update after SYNC command limited to once per <ms> milliseconds, default is 0, i.e. no limit

6.3 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.

7 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.

7.1 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 ([▷ VI-3 ANSYS ↴](#))
- OpenFOAM ([▷ VI-14 OpenFOAM ↴](#))
- STAR-CD ([▷ VI-18 STAR-CD ↴](#))

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 [▷ 5.4.3 mpcci_morpher ↴](#)) 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.

7.1.1 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.

 MpCCI Grid Morpher is multi threaded parallelized on the following platforms: lnx4_x64, windows_x64

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. The options file contains floating text. You may comment out a line with the # sign.

Please select the output level Activate output level of the MpCCI Grid Morpher.

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),

List of cell zone names (strings 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 STAR-CD, 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.

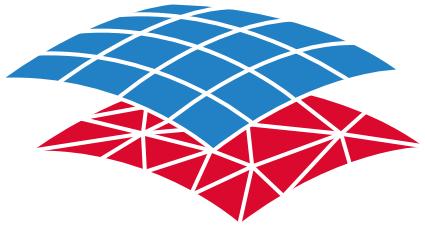
Optional list of fixed boundary regions Enter the boundary region numbers corresponding to the STAR-CD, OpenFOAM,... model, where a floating of vertices is explicitly not permitted (e.g. in the case where you might want to fix vertices on a symmetry plane).

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.



MpCCI
CouplingEnvironment

Part VI

Codes Manual

Version 4.5.0

MpCCI 4.5.0-1 Documentation
Part VI Codes Manual
PDF version
March 30, 2017

MpCCI is a registered trademark of Fraunhofer SCAI
www.mpcci.de



Fraunhofer Institute for Algorithms and Scientific Computing SCAI
Schloss Birlinghoven, 53754 Sankt Augustin, Germany

Abaqus and SIMULIA are trademarks or registered trademarks of Dassault Systèmes
ANSYS, FLUENT and ANSYS Icepak are trademarks or registered trademarks of Ansys, Inc.
Elmer is an open source software developed by CSC
FINE/Open and FINE/Turbo are trademarks of NUMECA International
Flowmaster is a registered trademark of Flowmaster Group NV
JMAG is a registered trademark of The JSOL Corporation
MATLAB is a registered trademark of The MathWorks, Inc.
MSC Adams, MSC Marc, MD NASTRAN and MSC NASTRAN are trademarks or registered trademarks of
MSC.Software Corporation
OpenFOAM is a registered trademark of OpenCFD Ltd.
PERMAS is a registered trademark of Intes GmbH
RadTherm, TAITherm is a registered trademark of ThermoAnalytics Inc.
SIMPACK is a registered trademark of SIMPACK AG.
STAR-CCM+ and STAR-CD are registered trademarks of CD adapco Group

ActivePerl has a Community License Copyright of Active State Corp.
FlexNet Publisher is a registered trademark of Flexera Software.
Java is a registered trademark of Oracle and/or its affiliates.
Linux is a registered trademark of Linus Torvalds
Mac OS X is a registered trademark of Apple Inc.
OpenSSH has a copyright by Tatu Ylonen, Espoo, Finland
Perl has a copyright by Larry Wall and others
UNIX is a registered trademark of The Open Group
Windows, Windows XP, Windows Vista and Windows 7 are registered trademarks of Microsoft Corp.

VI Codes Manual – Contents

1	Overview	11
1.1	Common MpCCI Subcommands for Simulation Codes	12
1.2	Unit Systems	14
2	Abaqus	15
2.1	Quick Information	15
2.1.1	Supported Coupling Schemes	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	17
2.2.1	Model Preparation	17
2.2.2	Restart	17
2.2.3	Models Step	17
2.2.4	Coupling Step	18
2.2.5	Go Step	19
2.2.6	Running the Computation	22
2.2.7	Post-Processing	23
2.3	Code-Specific MpCCI Commands	24
2.4	Code Adapter Reference	25
2.4.1	Patched Input File	25
2.4.2	SIMULIA's Co-Simulation Engine (CSE)	25
2.4.3	Update Code Adapter Installation	26
2.5	Co-Simulation Restart	28
2.6	Known limitations	29
3	ANSYS	30
3.1	Quick Information	30
3.1.1	Supported Coupling Schemes	30
3.1.2	Supported Platforms and Versions	30
3.1.3	References	30
3.1.4	Adapter Description	30
3.1.5	Prerequisites for a Coupled Simulation	30
3.1.6	Supported ANSYS product variable	31
3.2	Coupling Process	33
3.2.1	Model Preparation	33

3.2.2	APDL Script	36
3.2.3	Models Step	40
3.2.4	Coupling Step	40
3.2.5	Go Step	43
3.2.6	Running the Computation	44
3.3	Code-Specific MpCCI Commands	45
3.4	Code Adapter Reference	47
3.5	Frequently Asked Questions	48
4	ANSYS Icepak	50
4.1	Quick Information	50
4.1.1	Supported Platforms and Versions	50
4.1.2	Supported Quantities	50
4.2	Code-Specific MpCCI Commands	51
5	FINE/Open	53
5.1	Quick Information	53
5.1.1	Supported Coupling Schemes	53
5.1.2	Supported Platforms and Versions	53
5.1.3	References	53
5.1.4	Adapter Description	53
5.1.5	Prerequisites for a Coupled Simulation	53
5.2	Coupling Process	53
5.2.1	Model Preparation	53
5.2.2	Models Step	54
5.2.3	Coupling Step	54
5.2.4	Go Step	55
5.2.5	Running the Computation	56
5.2.6	Post-Processing	58
5.3	Code-Specific MpCCI Commands	58
5.4	Code Adapter Reference	59
5.5	Limitations	59
6	FINE/Turbo	60
6.1	Quick Information	60
6.1.1	Supported Coupling Schemes	60
6.1.2	Supported Platforms and Versions	60
6.1.3	References	60
6.1.4	Adapter Description	60
6.1.5	Prerequisites for a Coupled Simulation	60

6.2	Coupling Process	60
6.2.1	Model Preparation	60
6.2.2	Models Step	62
6.2.3	Coupling Step	62
6.2.4	Go Step	63
6.2.5	Running the Computation	64
6.2.6	Post-Processing	65
6.3	Code-Specific MpCCI Commands	65
6.4	Trouble shooting, open issues and known bugs	67
7	Flowmaster	68
7.1	Quick Information	68
7.1.1	Supported Coupling Schemes	68
7.1.2	Supported Platforms and Versions	68
7.1.3	References	68
7.1.4	Adapter Description	68
7.1.5	Prerequisites for a Coupled Simulation	68
7.2	Coupling Process	69
7.2.1	Model Preparation	69
7.2.2	Models Step	71
7.2.3	Coupling Step	71
7.2.4	Go Step	72
7.3	Code-Specific MpCCI Commands	73
7.4	Code Adapter Description	74
7.5	Trouble Shooting, Open Issues and Known Bugs	75
8	FLUENT	76
8.1	Quick Information	76
8.1.1	Supported Coupling Schemes	76
8.1.2	Supported Platforms and Versions	76
8.1.3	References	76
8.1.4	Adapter Description	76
8.1.5	Prerequisites for a Coupled Simulation	77
8.2	Coupling Process	77
8.2.1	Model Preparation	77
8.2.2	Models Step	79
8.2.3	Coupling Step	79
8.2.4	Go Step	81
8.2.5	Running the Computation	82
8.3	Code-Specific MpCCI Commands	88

8.4	Code Adapter Reference	90
8.4.1	The MpCCI UDF Library	90
8.4.2	UDF-Hooks	90
8.5	Trouble shooting, open issues and known bugs	93
8.6	Frequently Asked Questions	94
9	JMAG	95
9.1	Quick Information	95
9.1.1	Supported Coupling Schemes	95
9.1.2	Supported Platforms and Versions	95
9.1.3	References	95
9.1.4	Adapter Description	95
9.1.5	Prerequisites for a Coupled Simulation	95
9.1.6	Supported JMAG Modules	95
9.2	Coupling Process	96
9.2.1	Model Preparation	96
9.2.2	Models Step	99
9.2.3	Coupling Step	99
9.2.4	Go Step	100
9.3	Code-Specific MpCCI Commands	100
9.4	Code Adapter Description	101
10	MATLAB	102
10.1	Quick Information	102
10.1.1	Supported Coupling Schemes	102
10.1.2	Supported Platforms and Versions	102
10.1.3	References	102
10.1.4	Adapter Description	102
10.1.5	Prerequisites for a Coupled Simulation	102
10.1.6	Supported MATLAB Modules	102
10.2	Coupling Process	103
10.2.1	Model Preparation	103
10.2.2	MpCCI MEX Function	105
10.2.3	Models Step	110
10.2.4	Coupling Step	110
10.2.5	Go Step	113
10.3	Code-Specific MpCCI Commands	113
10.4	Code Adapter Description	114

11 MSC Adams	115
11.1 Quick Information	115
11.1.1 Supported Coupling Schemes	115
11.1.2 Supported Platforms and Versions	115
11.1.3 References	115
11.1.4 Adapter Description	115
11.1.5 Prerequisites for a Coupled Simulation	115
11.2 Coupling Process	116
11.2.1 Model Preparation	116
11.2.2 Dynamic or Kinematic Simulation	116
11.2.3 Static Simulation	117
11.2.4 Multiple Statements	117
11.2.5 Iterative Coupling	117
11.2.6 MSC Adams Template Products	117
11.2.7 Models Step	118
11.2.8 Coupling Step	119
11.2.9 Go Step	121
11.2.10 Running the Computation	123
11.2.11 Post-Processing	124
11.3 Code-Specific MpCCI Commands	125
11.4 Code Adapter Reference	126
11.4.1 Patched Input File	126
12 MSC Marc	128
12.1 Quick Information	128
12.1.1 Supported Coupling Schemes	128
12.1.2 Supported Platforms and Versions	128
12.1.3 References	128
12.1.4 Adapter Description	128
12.1.5 Prerequisites for a Coupled Simulation	128
12.2 Coupling Process	130
12.2.1 Model Preparation	130
12.2.2 Models Step	130
12.2.3 Coupling Step	131
12.2.4 Go Step	131
12.2.5 Running the Computation	132
12.2.6 Post-Processing	133
12.3 Code-Specific MpCCI Commands	134
12.4 Trouble Shooting, Open Issues and Known Bugs	135

13 MSC NASTRAN	136
13.1 Quick Information	136
13.1.1 Supported Coupling Schemes	136
13.1.2 Supported Platforms and Versions	136
13.1.3 Adapter Description	136
13.1.4 Prerequisites for a Coupled Simulation	136
13.2 Coupling Process	137
13.2.1 Model Preparation	137
13.2.2 Models Step	138
13.2.3 Coupling Step	139
13.2.4 Go Step	139
13.2.5 Running the Computation	140
13.2.6 Post-Processing	141
13.3 Code-Specific MpCCI Commands	142
13.4 Code Adapter Reference	143
13.5 Trouble Shooting, Open Issues and Known Bugs	144
14 OpenFOAM	145
14.1 Quick Information	145
14.1.1 Supported Coupling Schemes	145
14.1.2 Supported Platforms and Versions	145
14.1.3 References	145
14.1.4 Adapter Description	145
14.1.5 Prerequisites for a Coupled Simulation	145
14.2 Coupling Process	146
14.2.1 Model Preparation	146
14.2.2 Models Step	148
14.2.3 Coupling Step	149
14.2.4 Go Step	150
14.2.5 Running the Computation	151
14.2.6 Post-Processing	153
14.3 Grid Morphing	153
14.3.1 MpCCI Grid Morpher	153
14.3.2 OpenFOAM Grid Morpher	153
14.4 Code-Specific MpCCI Commands	156
14.5 Code Adapter Reference	158
15 RadTherm/TAITherm	159
15.1 Quick Information	159
15.1.1 Supported Coupling Schemes	159

15.1.2 Supported Platforms and Versions	159
15.1.3 References	159
15.1.4 Adapter Description	159
15.1.5 Prerequisites for a Coupled Simulation	159
15.2 Coupling Process	159
15.2.1 Model Preparation	160
15.2.2 Models Step	160
15.2.3 Coupling Step	161
15.2.4 Go Step	161
15.2.5 Checking the Computation	167
15.2.6 Running the Computation	167
15.2.7 Post-Processing	170
15.3 Code-Specific MpCCI Commands	170
15.4 Code Adapter Reference	171
15.4.1 Quantity handling	171
16 SIMPACK	172
16.1 Quick Information	172
16.1.1 Supported Coupling Schemes	172
16.1.2 Supported Platforms and Versions	172
16.1.3 References	172
16.1.4 Adapter Description	172
16.1.5 Prerequisites for a Coupled Simulation	172
16.2 Coupling Process	173
16.2.1 Model Preparation	173
16.2.2 Simulation	173
16.2.3 Models Step	173
16.2.4 Coupling Step	175
16.2.5 Go Step	176
16.2.6 Running the Computation	177
16.2.7 Post-Processing	177
16.3 Code-Specific MpCCI Commands	178
17 STAR-CCM+	179
17.1 Quick Information	179
17.1.1 Supported Coupling Schemes	179
17.1.2 Supported Platforms and Versions	179
17.1.3 References	179
17.1.4 Adapter Description	179
17.1.5 Prerequisites for a Coupled Simulation	180

17.2 Coupling Process	180
17.2.1 Model Preparation	180
17.2.2 Models Step	181
17.2.3 Coupling Step	181
17.2.4 Go Step	182
17.2.5 Running the Computation	184
17.2.6 Post-Processing	187
17.3 Code-Specific MpCCI Commands	187
17.4 Grid Morphing	188
17.5 Code Adapter Reference	188
17.5.1 Java Macro Script	188
17.6 Trouble Shooting, Open Issues and Known Bugs	202
18 STAR-CD	203
18.1 Quick Information	203
18.1.1 Supported Coupling Schemes	203
18.1.2 Supported Platforms and Versions	203
18.1.3 References	204
18.1.4 Adapter Description	204
18.1.5 Prerequisites for a Coupled Simulation	204
18.2 Coupling Process	204
18.2.1 Model Preparation	204
18.2.2 Models Step	206
18.2.3 Coupling Step	207
18.2.4 Go Step	209
18.2.5 Running the Computation	210
18.2.6 Post-Processing	211
18.3 Code-Specific MpCCI Commands	212
18.4 Grid Morphing	213
18.4.1 MpCCI Grid Morpher	214
18.4.2 Restart with MpCCI Grid Morpher	215
18.4.3 pro-STAR Grid Morpher	216
18.5 Code Adapter Reference	219
18.5.1 STAR-CD 4.x	219
18.5.2 Automatic Model Preparation STAR-CD 4.x	219
18.6 Trouble Shooting, Open Issues and Known Bugs	221

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

```
mpcci <codename> -releases
```

The `releases` subcommand prints a short list of all available releases of a simulation code.

```
mpcci <codename> -info
```

The `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. `HOME` is the main directory of a code installation or adapter, `EXEC` the code executable and `PATH` the path to the code adapter, which does not exist if an adapter is not available. The architecture token given under `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.

mpcci <codename> -diff <model file 1> <model file 2>

The `diff` subcommand runs the scanner on the given two model files and prints the differences.

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.

unit system	mass	length	time	forces	el. current	temperature
SI	kg	m	s	N	A	K
British*	lbf	ft	s	lbf	A	K
cgs	g	cm	s	dyn	Bi	K
SI-mm-t-s	t	mm	s	N	A	K
US-ft-lbf-s	slug	ft	s	lbf	A	K
US-in-lbf-s	lbf s ² /in	in	s	lbf	A	K

Alternatively, you can also select the option `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 `variable` an additional parameter will be shown where you can select the `grid length unit`, which corresponds to the length unit used in your model.

(!): 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

Physical Domains	SolidStructure, SolidAcoustics, SolidThermal
Company Name	SIMULIA
Company Homepage	www.3ds.com/products/simulia/overview
Support	www.3ds.com/products-services/simulia/support/support-overview/
Tutorials	▷ VII-2 Vortex-Induced Vibration of a Thin-Walled Structure ◁ ▷ VII-3 Driven Cavity ◁ ▷ VII-4 Elastic Flap in a Duct ◁ ▷ VII-5 Elastic Flap in Water ◁ ▷ VII-6 Exhaust Manifold ◁ ▷ VII-9 Pipe Nozzle ◁

2.1.1 Supported Coupling Schemes

The coupling schemes supported by Abaqus depend on the solver type:

- Abaqus/Explicit performs an initial data exchange prior to the first increment. Then after each increment solution a co-simulation step occurs.
- Abaqus/Standard performs a co-simulation after each converged increment.

Abaqus supports subcycling, unidirectional and bidirectional transfer.

For Abaqus after completing the last increment a final exchange, which is inverted regarding the initial exchange, is accomplished.

Initial Transfer	Final Transfer
receive	send
send	receive
skip	exchange
exchange	exchange

2.1.2 Supported Platforms and Versions

The following versions of Abaqus are supported by MpCCI (only the main version is listed):

MpCCI ARCH: Code platform	Supported versions		
	6.14	2016	2017
lnx4_x64: lnx86-64	X	X	X
windows_x64: win86-64	X	X	X

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 home-page abaqus.custhelp.com: Log into Abaqus Answers and search for “FSI guide”.

Besides general information, the FSI guide contains several examples of coupled simulations with Abaqus and FLUENT, Abaqus and STAR-CD.

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.

 The code adapters delivered with Abaqus 6.12, Abaqus 6.13, Abaqus 6.14 software are compatible with MpCCI 4.2. Check the following section ▷2.4.3 Update Code Adapter Installation◀ to update the code adapters.

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 requires one additional enabling token “cosimulation” for Abaqus 6.12, 6.13.
- Co-simulation based on the SIMULIA’s Co-Simulation Engine (CSE) requires the token “multi-physics” for Abaqus 6.14 and higher.

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 Coupling 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 the MpCCI 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.

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:

Please select the 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.

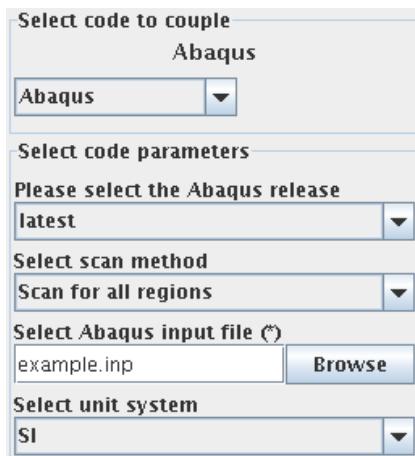


Figure 1: Abaqus options in the Models Step

Select 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.

Select Abaqus input file Select the Abaqus input file "***.inp**".

Select 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 Coupling Step

Abaqus supports the following quantities for coupling:

Quantity	Dim.	Default Value	Integration Type	Coupling Dimension	Location	Send Option	Receive Option
AbsPressure	Scalar	0.0 N/m ²	flux dens.	Face	Code		Buffer
Acceleration	Vector	0.0 m/s ²	field	Point	Code	Direct	Buffer
AngularAcceleration	Vector	0.0 rad/s ²	field	Point	Code	Direct	Buffer
AngularCoordinate	Quaternion	0.0	mesh coordinate	Point	Code	Direct	Buffer
AngularVelocity	Vector	0.0 rad/s	field	Point	Code	Direct	Buffer
BodyForce	Vector	0.0 N/m ³	flux dens.	Volume	Code	Direct	Buffer
DeltaTime	Scalar	1.0 s	g-min	Global	global	Direct	Direct
FilmTemp	Scalar	300.0 K	field	Face	Code		Buffer
Force	Vector	0.0 N	flux integral	Point, Face	Code	Direct	Buffer
HeatRate	Scalar	0.0 W	field	Face	Code		Buffer
NPosition	Vector	0.0 m	mesh coordinate	Face, Volume	Code	Direct	
OverPressure	Scalar	0.0 N/m ²	flux dens.	Face	Code		Buffer
PointPosition	Vector	0.0 m	mesh coordinate	Point	Code	Direct	Buffer

Quantity	Dim.	Default Value	Integration Type	Coupling Dimension	Location	Send Option	Receive Option
PorePressure	Scalar	0.0 N/m ²	flux dens.	Face, Volume	Code	Direct	Buffer
PorousFlow	Scalar	0.0 m/s	flux dens.	Face, Volume	Code	Direct	Buffer
RelWallForce	Vector	0.0 N	flux integral	Face	Code		Buffer
Temperature	Scalar	300.0 K	field	Volume	Code	Direct	Buffer
Torque	Vector	0.0 N m	flux integral	Point	Code	Direct	Buffer
Velocity	Vector	0.0 m/s	field	Point, Face, Volume	Code	Direct	Buffer
WallForce	Vector	0.0 N	flux integral	Face	Code	Direct	Buffer
WallHeatFlux	Scalar	0.0 W/m ²	flux dens.	Face	Code		Buffer
WallHTCoeff	Scalar	0.0 W/m ² K	field	Face	Code		Buffer
WallTemp	Scalar	300.0 K	field	Face	Code	Direct	Buffer

(!) 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.

2.2.5 Go Step

In the Go Step, the following options can be selected (see [Figure 2](#)):

Coupling configuration The general coupling configuration is described in [V-4.8.2 Coupling Configuration Parameters](#). For Abaqus there are following deviations:

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.

Constant coupling time step and Coupling time step This button must be checked if the time step size DeltaTime is not selected as a global quantity to be sent or received by Abaqus. In this case you must specify the value of the coupling time step size in the field below.

(!) For steady state analysis you need to activate that field and set the value to 1 in order to exchange after each increment for example. Otherwise the default value used is 1e-5

Use subcycling and Enforce exact target times 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-3.4.6 Subcycling](#). If you select this option, you can also choose whether Abaqus has to meet the coupling target times in an exact or loose manner by toggling the Enforce exact target times button.

Enforce quasistatic Define a quasi static analysis procedure as a transient analysis for the coupling. The pseudo-time is sent to MpCCI instead of the iteration number.

Enter a job name This is the name of the Abaqus job, which is also used as base for the Abaqus output files.

Additional command line options Additional command line options for Abaqus can be given here, they

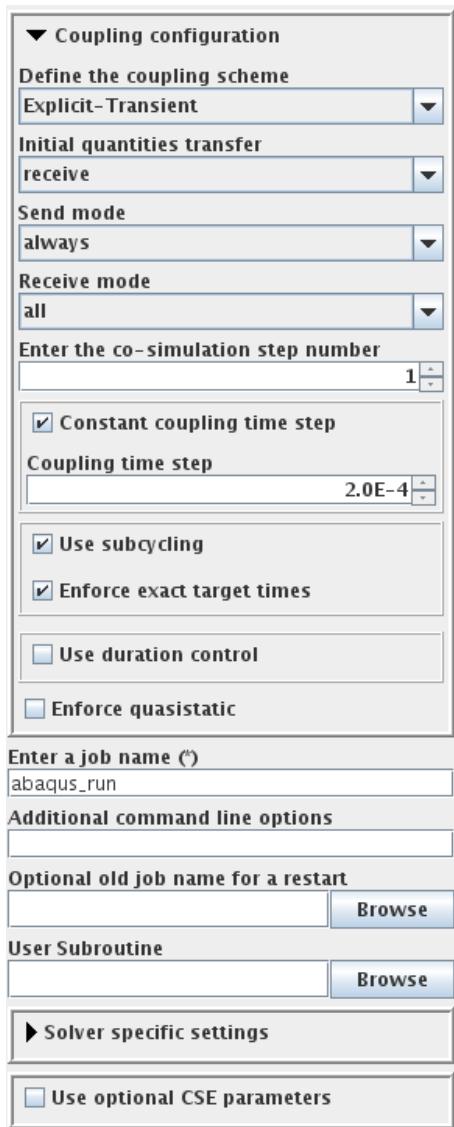


Figure 2: Abaqus options in the Go Step

will directly be used when Abaqus is started.

(!) MpCCI uses the alternate syntax "-option value" for the Abaqus command line options.

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.

User Subroutine is the name of a FORTRAN file (source or object file), which is handed to Abaqus as `user=<file>`. See "User subroutines: overview" of the Abaqus Analysis User's Manual.



Figure 3: Abaqus solver specific options for solvers Standard and Explicit

Solver specific settings groups the options which can be set for the used solver (see [Figure 3](#)).

Abaqus solver used shows the Abaqus solver used for this computation. The solver type is retrieved from the scanner run.

Standard is the default solver type and provides the option (cf. [Figure 3](#) Standard):

No. of Abaqus/Standard solver threads Enter the number of threads for an Abaqus/Standard run.

Explicit solver type provides following options (cf. [Figure 3](#) Explicit):

Double precision for Abaqus/Explicit Use this option to select the precision.

Run Abaqus/Explicit in parallel Select this to start a parallel run. A panel with additional options appears, which are further described in [2.2.6.1 Parallel Execution](#).

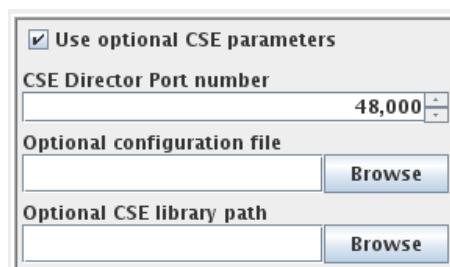


Figure 4: Abaqus CSE parameters in the Go Step

Use optional CSE parameters Activate this option in order to access some additional parameters. It is recommended to provide different initial port number 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 convention name for the co-simulation region interface. You may be able to add, 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`

2.2.6 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.

2.2.6.1 Parallel Execution

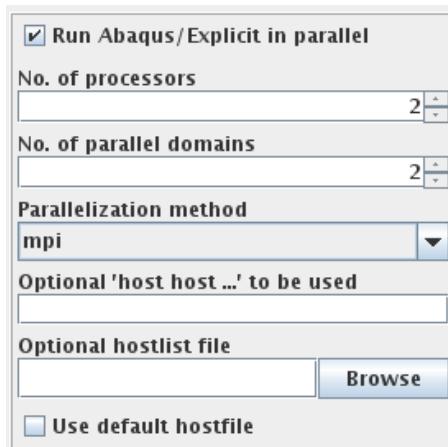


Figure 5: Abaqus options for a parallel run

Parallel runs of Abaqus are supported by MpCCI.

In the Go Step, for Abaqus/Explicit a set of additional options can be chosen for a parallel run as shown in [Figure 5](#).

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**.

Parallelization method Select the parallelization method either `mpi` or leave blank which results into the Abaqus default. The corresponding Abaqus command line option is `-mp_mode`.

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.

2.2.6.2 Batch Execution

Abaqus is always run as a batch process, therefore no special settings are necessary for batch execution.

2.2.7 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.

-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 -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:

```
**
** MpCCI automatically inserted this *CO-SIMULATION keyword.
** Therefore any existing *CO-SIMULATION keyword was removed.
**
*CO-SIMULATION, NAME=COSIMULATION_1, PROGRAM=MPCCI,
    CONTROLS=COSIMULATION_CONTROLS_1
*CO-SIMULATION REGION, TYPE=SURFACE, EXPORT
ASSEMBLY_BLOCK-1_WALL, COORD
*CO-SIMULATION REGION, TYPE=SURFACE, IMPORT
ASSEMBLY_BLOCK-1_WALL, CF
*CO-SIMULATION CONTROLS, NAME=COSIMULATION_CONTROLS_1,
    TIME INCREMENTATION=SUBCYCLE, TIME MARKS=YES, STEP SIZE=EXPORT
```

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 time incrementation scheme is selected in the Go Step of the MpCCI GUI.
 Behind the `*CO-SIMULATION CONTROLS` keyword you find:
`TIME INCREMENTATION=SUBCYCLE` if subcycling was selected or `TIME INCREMENTATION=LOCKSTEP`, which means Abaqus uses the coupling time step also for the computation of the Abaqus part of the simulation. If `SUBCYCLE` is selected, the option `TIME MARKS` determines whether the target times are met in an exact (`YES`) or loose (`NO`) manner.
- If the time step size is exchanged, you find either `STEP SIZE=EXPORT` if Abaqus is the sender or `STEP SIZE=IMPORT` if Abaqus receives the time step size from the partner code.
- If neither the time step size is exchanged nor a constant coupling time step is activated, you find `STEP SIZE=EXPORT`. Abaqus performs the co-simulation step at its own time stepping method.

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.

2.4.3 Update Code Adapter Installation

2.4.3.1 Error Message Using Old Code Adapter

If you are using Abaqus with MpCCI 4.3 without having updated the code adapter you will get the following error message:

```
ABAQUS-MpCCI: amppci_tinfo_init: The coupling configuration environment variable
ABAQUS-MpCCI:
ABAQUS-MpCCI:"MPCCI_TINFO=ET:J:receive:always:all:none:-1.0:-1.0:-1:-1:-1.0:1:1:none:2"
ABAQUS-MpCCI:
ABAQUS-MpCCI: is invalid.
```

2.4.3.2 Upgrade Instructions

MpCCI 4.3, MpCCI 4.4 provides the code adapter update for Abaqus 6.12, Abaqus 6.13 for Linux/x86-64 and Windows/x86-64 Abaqus platform.

You will find the code adapter update under the following directory:

"MPCCI_HOME/codes/Abaqus/adapters/<ABAQUS_RELEASE/<ABAQUS_ARCH>".

- "MPCCI_HOME" designates the home installation directory of MpCCI. You can use the command `mpcci home` to figure out this information.
- "ABAQUS_RELEASE" is for example 6.13 (only major and minor release is take in account).
- "ABAQUS_ARCH" is either win86-64 or lnx86-64.

Installation procedure:

1. First make a backup of the MpCCI code adapter library you find under the directory:
"ABAQUS_INST/ABAQUS_RELEASE/code/bin".

To figure out the value of "ABAQUS_INST" you can execute the following command: `mpcci abaqus info`.

```
Abaqus release "6.13-1":
  HOME: "/etc/soft/abaqus/linux_x86_64"
  EXEC: "/etc/soft/abaqus/linux_x86_64/6.13-1/code/bin/abq6131"
  ARCH: "lnx86-64"
  Static linked MpCCI adapter.
```

```
Abaqus release "6.12-1":
  HOME: "/etc/soft/abaqus/linux_x86_64"
  EXEC: "/etc/soft/abaqus/linux_x86_64/6.12-1/code/bin/abq6121.exe"
  ARCH: "lnx86-64"
  Static linked MpCCI adapter.
```

Latest valid Abaqus release for MpCCI: "6.13-1"

The label "HOME" of the command corresponds to the "ABAQUS_INST".

On Linux platform the code adapter is named "libABQMpCCI_Adapter.so" and on Microsoft Windows platform "ABQMpCCI_Adapter.dll".

Backup one of this file according to your installation platform.

2. The new MpCCI Abaqus code adapter library may be found under the MpCCI installation directory: "MPCCI_HOME/codes/Abaqus/adapters/<ABAQUS_RELEASE/<ABAQUS_ARCH>". Please copy either the "libABQMpCCI_Adapter.so" or "ABQMpCCI_Adapter.dll" according to your

current platform to "ABAQUS_INST/ABAQUS_RELEASE/code/bin" directory .

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:

```
**
** MpCCI automatically inserted this *CO-SIMULATION keyword.
** Therefore any existing *CO-SIMULATION keyword was removed.
**
*CO-SIMULATION, NAME=COSIMULATION_1, PROGRAM=MPCCI,
    CONTROLS=COSIMULATION_CONTROLS_1
*CO-SIMULATION REGION, TYPE=SURFACE, EXPORT
ASSEMBLY_BLOCK-1_WALL, COORD
*CO-SIMULATION REGION, TYPE=SURFACE, IMPORT
ASSEMBLY_BLOCK-1_WALL, CF
*CO-SIMULATION CONTROLS, NAME=COSIMULATION_CONTROLS_1,
    TIME INCREMENTATION=SUBCYCLE, TIME MARKS=YES, STEP SIZE=EXPORT
```

- 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.

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 Abaqus 6.14 SIMULIA Co-Simulation Engine. The workaround consists of formulate your problem in a 3D cyclic symmetric model.

3 ANSYS

3.1 Quick Information

Physical Domains	SolidStructure, SolidThermal
Company Name	ANSYS, Inc.
Company Homepage	www.ansys.com
Support	ANSYS Customer portal at support.ansys.com
Tutorials	▷ VII-2 Vortex-Induced Vibration of a Thin-Walled Structure ◁ ▷ VII-4 Elastic Flap in a Duct ◁ ▷ VII-6 Exhaust Manifold ◁ ▷ VII-7 Busbar System ◁ ▷ VII-9 Pipe Nozzle ◁

3.1.1 Supported Coupling Schemes

ANSYS is run via an APDL script, therefore all coupling algorithms are possible.

3.1.2 Supported Platforms and Versions

The following versions of ANSYS are supported by MpCCI:

MpCCI_ARCH: Code platform	Supported versions								
	145	150	160	161	162	170	171	172	180
lnx4_x64: linx64	X	X	X	X	X	X	X	X	X
windows_x64: winx64	X	X	X	X	X	X	X	X	X

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:

```
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.

Product	Feature Names
ANSYS Multiphysics/LS-DYNA	ane3flds
ANSYS Multiphysics 1	ane3f11
ANSYS Multiphysics 2	ane3f12
ANSYS Multiphysics 3	ane3f13
ANSYS Multiphysics/LS-DYNA PrepPost	ane3fldp
ANSYS Mechanical	ansys, ansysul, ansysuh, ansysrf
ANSYS Mechanical/Emag	ane3
ANSYS Mechanical/FLOTRAN	anfl
ANSYS Mechanical/LS-DYNA	ansysds
ANSYS Mechanical/Emag/LS-DYNA	ane3ds
ANSYS Mechanical/CFD-Flo/LS-DYNA	anflds
ANSYS Mechanical/LS-DYNA PrepPost	ansysdp
ANSYS Mechanical/Emag/LS-DYNA PrepPost	ane3dp
ANSYS Mechanical/CFD-Flo/LS-DYNA PrepPost	anfldp
ANSYS Structural	struct, struct1, struct2, struct3
ANSYS Structural/Emag	ste3
ANSYS Structural/FLOTRAN	stfl
ANSYS Structural/Emag/CFD-Flo	ste3fl
ANSYS Structural/LS-DYNA	structds
ANSYS Structural/Emag/LS-DYNA	ste3ds
ANSYS Structural/CFD-Flo/LS-DYNA	stflds
ANSYS Structural/Emag/CFD-Flo/LS-DYNA	ste3flds
ANSYS Structural/LS-DYNA PrepPost	structdp
ANSYS Structural/Emag/LS-DYNA PrepPost	ste3dp
ANSYS Structural/CFD-Flo/LS-DYNA PrepPost	stfldp
ANSYS Structural/Emag/CFD-Flo/LS-DYNA PrepPost	ste3fldp
ANSYS Professional NLT	prf
ANSYS Professional/Emag	prfe3
ANSYS Professional/FLOTRAN	prffl
ANSYS Professional/Emag/FLOTRAN	prfe3fl
ANSYS Emag	emag
ANSYS Emag/FLOTRAN	emagfl
ANSYS Mechanical PrepPost	prepost
ANSYS PrepPost/LS-DYNA PrepPost	prpostdy
ANSYS FLOTRAN	flowtran
ANSYS Academic Teaching Advanced	aa_t_a, aa_mcad, aa_ds
ANSYS Academic Research	aa_r, aa_mcad

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 ◁](#).

ANSYS Element Types	Comments
BEAM3, BEAM4, BEAM23, BEAM24, BEAM44, BEAM54, BEAM188, BEAM189, SHELL51, SHELL61, SHELL208, SURF151, SURF153, SURF251 PIPE288, PIPE289, ELBOW290, MPC184,	MPCCI_ETYP_LINE2, MPCCI_ETYP_LINE3
PLANE13, PLANE25, PLANE42, PLANE55, PLANE67, PLANE181, PLANE182, SHELL28, SHELL41, SHELL43, SHELL57, SHELL63, SHELL131 SHELL143, SHELL157, HYPER56, VISCO106, CPT212, SURF252,	MPCCI_ETYP_TRIA3, MPCCI_ETYP_QUAD4
PLANE2, PLANE35	MPCCI_ETYP_TRIA3, MPCCI_ETYP_TRIA6
SHELL91, SHELL93, SHELL99, SHELL132, SHELL150, SHELL281 PLANE53, PLANE145, PLANE223, PLANE77, PLANE78, PLANE82, PLANE83, PLANE121, PLANE183, PLANE146, PLANE230 SURF152, SURF154 CPT213, HYPER74, VISCO88, VISCO108,	MPCCI_ETYP_TRIA6, MPCCI_ETYP_QUAD4, MPCCI_ETYP_QUAD8
SOLID5, SOLID45, SOLID46, SOLID62, SOLID64, SOLID65, SOLID69, SOLID70, SOLID96, SOLID97, SOLID164, SOLID285, SOLID185, HYPER58, HYPER86, VISCO107, SOLSH190, CPT215	MPCCI_ETYP_TET4, MPCCI_ETYP_WEDGE6, MPCCI_ETYP_HEX8, MPCCI_ETYP_PYRAM5
SOLID87, SOLID92, SOLID98, SOLID123, SOLID127, SOLID148, SOLID168, SOLID186, SOLID187, SOLID227, SOLID232, SOLID237, CPT217	MPCCI_ETYP_TET4, MPCCI_ETYP_TET10
SOLID90, SOLID95, SOLID122, SOLID117, SOLID122, SOLID128, SOLID147, SOLID186, SOLID191, SOLID226, SOLID231, SOLID236, VISCO89, CPT216	MPCCI_ETYP_TET4, MPCCI_ETYP_WEDGE6, MPCCI_ETYP_HEX8, MPCCI_ETYP_PYRAM5, MPCCI_ETYP_PYRAM13
MESH200	MPCCI_ETYP_LINE2, MPCCI_ETYP_TRIA3, MPCCI_ETYP_QUAD4, MPCCI_ETYP_QUAD8

Table 1: Supported ANSYS element types

<i>Quantity</i> <i>Element</i>	Joule Heat Density	Lorentz Force Density	Electric Resistivity	Nodal Positions	Wall Forces	Relative Wall Forces	Film Tempera- ture	Wall Heat- transfer Coefficient	Wall Tempera- ture
BEAM3				+	+	+			
SOLID5	+		+	+	+	+			
PLANE13	+	+	+						
SOLID45			+	+	+	+			
SHELL57							+	+	+
SHELL63				+	+	+			
SOLID69	+	+	+						
SHELL93				+	+	+			
SOLID97	+	+	+						
SOLID117	+	+	+						
SURF151							+	+	+
MESH200				+					
PLANE230	+		+						
SOLID231	+		+						
SOLID236	+	+	+						

Table 2: Quantities supported for different ANSYS elements.

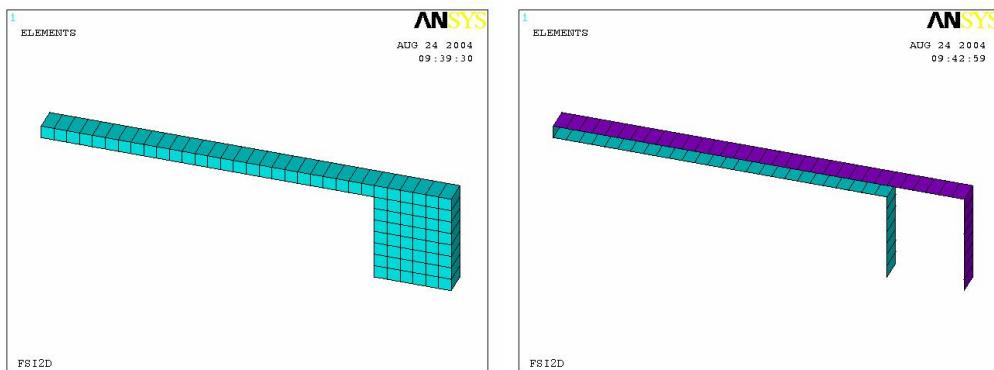


Figure 1: ANSYS Flap and Additional SHELL63 elements for coupling

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-Raphson

/com > > >
/com > > > initial data transfer controlled via 'Initial quantities transfer' setting:
/com > > > send / receive / exchange

~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
*enddo

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 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, WRCPPL, 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, WRCPPL, <filename>.cpl`

Writes a component list file with all defined components and variables into "`<filename>.cpl`". This file is normally generated and used by the MpCCI 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, WRGMD, <filename>.gmd, 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, INIT [, 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.
- If **ALLWAIT** is given, **ANSYS** is waiting for all partner codes (which is using the quantities) before sending all quantities.

***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]**

Notifies MpCCI 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 a element table fulfilling the naming convention:

```
etab, MPCCI_<sindex>, <item>, <component>
```

For example: To get joule heat density from storage index 0 the APDL command should be:

```
etab, MPCCI_00, jheat
```

(see ANSYS documentation of **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:

```
etab, MPCCI_<sindex>, <item>, <component>
etab, MPCCI_<sindex+1>, <item>, <component>
etab, MPCCI_<sindex+2>, <item>, <component>
```

For example: to get define Lorentz force density vector into element tables starting with storage index 5 the APDL command could be:

```
etab, volu, volu           ! element volume
sexp, MPCCI_05, lfx, volu,,,-1   ! lorentz force density
sexp, MPCCI_06, lfy, volu,,,-1
sexp, MPCCI_07, lfz, volu,,,-1
```

3.2.3 Models Step

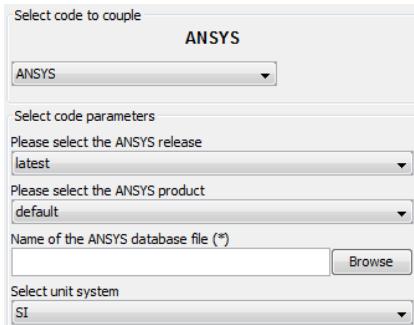


Figure 2: ANSYS options in the Models Step

In the Models Step, the following options must be chosen:

Select the 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.

Select ANSYS product to run Choose one of the ANSYS products you have licensed, see also [section 3.1.6](#).

Name of the ANSYS database file (*) Select the ANSYS database which contains your model definitions.

Select unit system Select the unit system which was used in ANSYS (see [1.2 Unit Systems](#)).

3.2.4 Coupling Step

ANSYS supports the following quantities for coupling:

Quantity	Dim.	Default Value	Integration Type	Coupling Dimension	Location	Send Option	Receive Option
AbsPressure	Scalar	0.0 N/m ²	flux dens.	Line, Face, Volume	Element	Direct, ETAB	Direct
AngularCoordinate	Quaternion	0.0	mesh coordinate	Point	Node	Direct	Direct
AngularVelocity	Vector	0.0 rad/s	field	Point	Node	Direct	Direct
BodyForce	Vector	0.0 N/m ³	flux dens.	Volume	Node, Element	Direct, ETAB	Direct
CGAngle	Vector	0.0 rad	g-max	Global	global	APDL	APDL
CGOmega	Vector	0.0 rad/s	g-max	Global	global	APDL	APDL
CGPosition	Vector	0.0 m	g-max	Global	global	APDL	APDL
CGVelocity	Vector	0.0 m/s	g-max	Global	global	APDL	APDL
ChargeDensity	Scalar	0.0 C/m ³	field	Line, Volume	Element	Direct	Direct
Current1	Scalar	0.0 A	g-max	Global	global	APDL	APDL
Current2	Scalar	0.0 A	g-max	Global	global	APDL	APDL
Current3	Scalar	0.0 A	g-max	Global	global	APDL	APDL
Current4	Scalar	0.0 A	g-max	Global	global	APDL	APDL
CurrentDensity	Vector	0.0 A/m ²	flux dens.	Line, Face, Volume	Node, Element	Direct	Direct
DeltaTime	Scalar	1.0 s	g-min	Global	global	APDL	APDL

Quantity	Dim.	Default Value	Integration Type	Coupling Dimension	Location	Send Option	Receive Option
ElectrCond1	Scalar	0.0 S/m	field	Line, Face, Volume	Element	Direct	Direct
ElectrCond3	Vector	0.0 S/m	field	Line, Face, Volume	Element	Direct	Direct
ElectrCondX	Scalar	0.0 S/m	field	Line, Face, Volume	Element	Direct	Direct
ElectrCondY	Scalar	0.0 S/m	field	Line, Face, Volume	Element	Direct	Direct
ElectrCondZ	Scalar	0.0 S/m	field	Line, Face, Volume	Element	Direct	Direct
ElectricField	Vector	0.0 V/m	field	Line, Volume	Element	Direct	Direct
ElectricFlux	Vector	0.0 C/m ²	flux dens.	Line, Volume	Element	Direct	Direct
ElectricPot	Scalar	0.0 V	field	Line, Face, Volume	Node	Direct	Direct
ElectrRes1	Scalar	0.0 ohm m	field	Line, Face, Volume	Element	Direct	Direct
ElectrRes3	Vector	0.0 ohm m	field	Line, Face, Volume	Element	Direct	Direct
ElectrResX	Scalar	0.0 ohm m	field	Line, Face, Volume	Element	Direct	Direct
ElectrResY	Scalar	0.0 ohm m	field	Line, Face, Volume	Element	Direct	Direct
ElectrResZ	Scalar	0.0 ohm m	field	Line, Face, Volume	Element	Direct	Direct
Enthalpy	Scalar	0.0 W/m ³	flux dens.	Volume	Node, Element	Direct	Direct
FilmTemp	Scalar	300.0 K	field	Face	Element		Direct
Force	Vector	0.0 N	flux integral	Point	Node	Direct	Direct
HeatFlux	Vector	0.0 W/m ²	flux dens.	Volume	Node, Element	Direct, ETAB	Direct
HeatSource	Scalar	0.0 W/m ³	flux dens.	Volume	Node, Element	Direct, ETAB	Direct
IntFlag	Scalar	0	g-max	Global	global	APDL	APDL
IterationNo	Scalar	0	g-max	Global	global	APDL	APDL
JouleHeat	Scalar	0.0 W/m ³	flux dens.	Volume	Node, Element	Direct, ETAB	Direct
LorentzForce	Vector	0.0 N/m ³	flux dens.	Volume	Node, Element	Direct, ETAB	Direct
MagneticField	Vector	0.0 A/m	field	Line, Volume	Element	Direct	Direct
MagneticFlux	Vector	0.0 T	flux dens.	Line, Face, Volume	Node, Element	Direct	Direct
NPosition	Vector	0.0 m	mesh coordinate	Line, Face, Volume	Node	Direct	
OverPressure	Scalar	0.0 N/m ²	flux dens.	Line, Face, Volume	Element	Direct, ETAB	Direct
PhysicalTime	Scalar	0.0 s	g-min	Global	global	APDL	APDL
PointPosition	Vector	0.0 m	mesh coordinate	Point	Node	Direct	Direct

Quantity	Dim.	Default Value	Integration Type	Coupling Dimension	Location	Send Option	Receive Option
RealFlag	Scalar	0.0	g-max	Global	global	APDL	APDL
ReffPressure	Scalar	1.12e5 N/m ²	g-max	Global	global	APDL	APDL
RelWallForce	Vector	0.0 N	flux integral	Face	Node		Direct
Residual	Scalar	0.0	g-max	Global	global	APDL	APDL
SpecificHeat	Scalar	1.0 J/kg K	field	Volume	Element	Direct	Direct
Temperature	Scalar	300.0 K	field	Line, Volume	Node, Element	Direct, ETAB	Direct
ThermCond1	Scalar	0.0 W/m K	field	Line, Face, Volume	Element	Direct	Direct
ThermCond3	Vector	0.0 W/m K	field	Line, Face, Volume	Element	Direct	Direct
ThermCondX	Scalar	0.0 W/m K	field	Line, Face, Volume	Element	Direct	Direct
ThermCondY	Scalar	0.0 W/m K	field	Line, Face, Volume	Element	Direct	Direct
ThermCondZ	Scalar	0.0 W/m K	field	Line, Face, Volume	Element	Direct	Direct
TimeStepNo	Scalar	0	g-max	Global	global	APDL	APDL
Torque	Vector	0.0 N m	flux integral	Point	Node	Direct	Direct
Velocity	Vector	0.0 m/s	field	Volume	Element		Direct
Voltage1	Scalar	0.0 V	g-max	Global	global	APDL	APDL
Voltage2	Scalar	0.0 V	g-max	Global	global	APDL	APDL
Voltage3	Scalar	0.0 V	g-max	Global	global	APDL	APDL
Voltage4	Scalar	0.0 V	g-max	Global	global	APDL	APDL
WallForce	Vector	0.0 N	flux integral	Face	Node		Direct
WallHeatFlux	Scalar	0.0 W/m ²	flux dens.	Face	Element	Direct	Direct
WallHTCoeff	Scalar	0.0 W/m ² K	field	Face	Element		Direct
WallTemp	Scalar	300.0 K	field	Face	Node, Element	Direct	Direct

3.2.5 Go Step

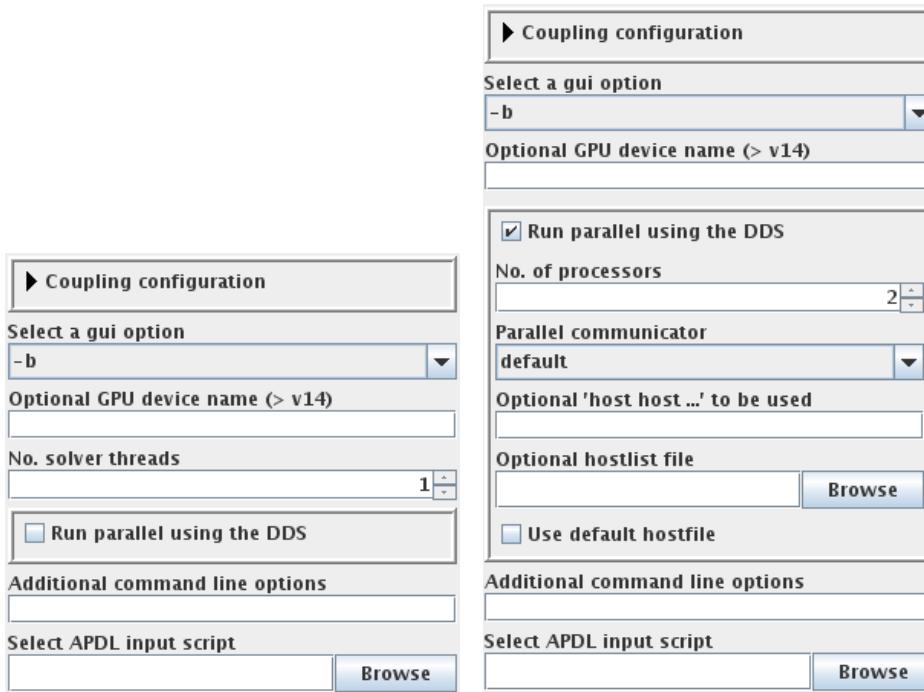


Figure 3: ANSYS options in the Go Step with SMP (left) and DDS (right)

In the Go Step, the following options can be selected:

Coupling configuration See ▷V-4.8.2 Coupling Configuration Parameters◀.

Define the coupling scheme ANSYS only supports Explicit-Transient and Explicit-SteadyState coupling schemes.

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 (>v14) 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 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 ◁](#).

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 ◁](#)).

3.2.6 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 ◁](#).

mpcci ansys -convert <db file> [product] [release]

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  
ane3f1  
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

Question:

Could ANSYS use automatic time stepping scheme for the coupling?

Answer:

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 [▷ VII-4 Elastic Flap in a Duct ◁](#).

Question:

In MpCCI Coupling Step I can't find any component for coupling the time step size in the Global Panel. How can I activate the Global Panel?

Answer:

In ANSYS model you need to create a parameter name DELTATIME for instance with the command:
`*SET,DELTATIME,1` 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 [▷ VII-4 Elastic Flap in a Duct ◁](#)).

You will be to save your ANSYS model and rescan it in the MpCCI GUI.

Question:

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.

Answer:

By using shell elements without solid for the modeling you do not need dummy elements as recommended in [▷ 3.2.1 Model Preparation ◁](#) 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.

Question:

I noticed that the result of "OverPressure" transferred is quite different than with "RelWallForce". I used shell elements for modeling my wall and have defined a default load on the surface. What happens?

Answer:

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.

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 FLEXIm 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 `ANSYSLIC_ADMIN` is set in my user setting environments to `1056@license.com` and MpCCI did not find any products.

Answer:

Set the environment variable `ANSYSLMD_LICENSE_FILE` to the FLEXIm license server for the ANSYS product.

The FLEXIm message error:

`Error getting status: Invalid returned data from license server system. (-12,16)` indicates that the port 1056 is not associate with a FLEXIm 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

Physical Domains	CFD, Fluid, FluidThermal
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 Platforms and Versions

MPCCI_ARCH: Code platform	Supported versions										
	14.5.0	14.5.7	15.0.0	15.0.7	16.0.0	16.1.0	16.2.0	17.0.0	17.1.0	17.2.0	18.0.0
lnx4_x64: lnamd64	X	X	X	X	X	X	X	X	X	X	X
windows_x64: win64	X		X	X	X	X	X	X	X	X	X

4.1.2 Supported Quantities

ANSYS Icepak supports the following quantities for coupling:

Quantity	Dim.	Default Value	Integration Type	Coupling Dimension	Location	Send Option	Receive Option
DeltaTime	Scalar	1.0 s	g-min	Global	global	Dir	Dir
Enthalpy	Scalar	0.0 W/m ³	flux dens.	Volume	Code	Dir	UDM
FilmTemp	Scalar	300.0 K	field	Face	Code	Dir	UDM
HeatFlux	Vector	0.0 W/m ²	flux dens.	Volume	Code	UDM	UDM
HeatSource	Scalar	0.0 W/m ³	flux dens.	Volume	Code	UDM	UDM
IntFlag	Scalar	0	g-max	Global	global	Dir	Dir
IterationNo	Scalar	0	g-max	Global	global	Dir	Dir
JouleHeat	Scalar	0.0 W/m ³	flux dens.	Volume	Code	UDM	UDM
JouleHeatLin	Scalar	0.0 W/m ³ K	flux dens.	Volume	Code	UDM	UDM
PhysicalTime	Scalar	0.0 s	g-min	Global	global	Dir	Dir
RealFlag	Scalar	0.0	g-max	Global	global	Dir	Dir
Residual	Scalar	0.0	g-max	Global	global	Dir	Dir
SpecificHeat	Scalar	1.0 J/kg K	field	Volume	Code	Dir	UDM
Temperature	Scalar	300.0 K	field	Volume	Code	Dir	UDM
ThermCond1	Scalar	0.0 W/m K	field	Face, Volume	Code	Dir	UDM
ThermCond3	Vector	0.0 W/m K	field	Face, Volume	Code	Dir	UDM
ThermCondX	Scalar	0.0 W/m K	field	Volume	Code	Dir	UDM
ThermCondY	Scalar	0.0 W/m K	field	Volume	Code	Dir	UDM
ThermCondZ	Scalar	0.0 W/m K	field	Volume	Code	Dir	UDM

Quantity	Dim.	Default Value	Integration Type	Coupling Dimension	Location	Send Option	Receive Option
TimeStepNo	Scalar	0	g-max	Global	global	Dir	Dir
WallHeatFlux	Scalar	0.0 W/m ²	flux dens.	Face	Code	Dir	UDM
WallHTCoeff	Scalar	0.0 W/m ² K	field	Face	Code	Dir	UDM
WallTemp	Scalar	300.0 K	field	Face	Code	Dir	UDM

4.2 Code-Specific MpCCI Commands

The MpCCI subcommands available for ANSYS IcePak are:

Usage:

```
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

Physical Domains	CFD, Fluid, FluidThermal
Company Name	NUMECA International
Company Homepage	www.numeca.com
Support	support @numeca.com
Tutorials	▷ VII-2 Vortex-Induced Vibration of a Thin-Walled Structure ◁ ▷ VII-4 Elastic Flap in a Duct ◁

5.1.1 Supported Coupling Schemes

FINE/Open supports exchange after solution. Unidirectional and bidirectional transfer is possible.

5.1.2 Supported Platforms and Versions

MPCCI_ARCH: Code platform	Supported versions					
	41	42	43	51	52	61
lnx4_x64: x86_64	X	X	X	X	X	X
windows_x64: win64	X	X	X	X	X	X

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_<bit>.so”, 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 for coupled simulation using MpCCI and for mesh deformation if needed.

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.
- The FINE/Open computation must run in standalone.

5.2.2 Models Step

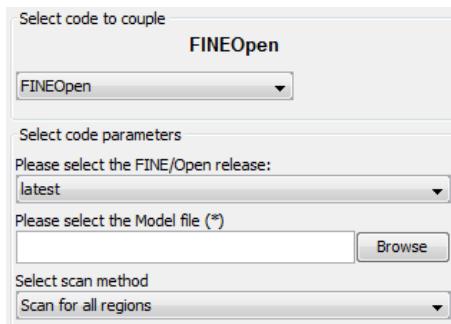


Figure 1: FINE/Open options in the Models Step

In the Models Step, the following options must be chosen:

Please select the 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).

Please select the run file Select the FINE/Open input file "*.run".

Select 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/Turbo GUI).

5.2.3 Coupling Step

FINE/Open supports the following quantities for coupling:

Quantity	Dim.	Default Value	Integration Type	Coupling Dimension	Location	Send Option	Receive Option
AbsPressure	Scalar	0.0 N/m ²	flux dens.	Face	Node, Element	Direct	
DeltaTime	Scalar	1.0 s	g-min	Global	global	Direct	
Density	Scalar	1.0 kg/m ³	field	Volume	Node, Element	Direct	
IterationNo	Scalar	0	g-max	Global	global	Direct	
Position	Vector	0.0 m	mesh coordinate	Face	Node		Direct
OverPressure	Scalar	0.0 N/m ²	flux dens.	Face	Node, Element	Direct	
PhysicalTime	Scalar	0.0 s	g-min	Global	global	Direct	
RefPressure	Scalar	1.12e5 N/m ²	g-max	Global	global	Direct	
RelWallForce	Vector	0.0 N	flux integral	Face	Element	Direct	
TimeStepNo	Scalar	0	g-max	Global	global	Direct	
Velocity	Vector	0.0 m/s	field	Face	Node, Element	Direct	
WallHeatFlux	Scalar	0.0 W/m ²	flux dens.	Face	Element	Direct	
WallTemp	Scalar	300.0 K	field	Face	Element	Direct	Direct

 It is not possible for FINE/Open solver to receive DeltaTime quantity. The time step can only be sent to the counterpart code.

5.2.4 Go Step

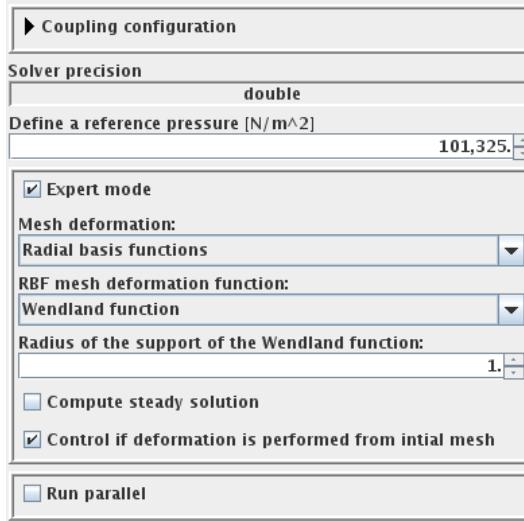


Figure 2: FINE/Open options in the Go Step

Coupling configuration See [V-4.8.2 Coupling Configuration Parameters](#).

Define the coupling scheme FINE/Open only supports Explicit-Transient and Explicit-SteadyState coupling schemes.

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 $\frac{N}{m^2}$.

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).

Compute steady solution This is an advanced parameter setting the FINE/Open parameter `iMpCCIStadyMode_`. 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.

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 [▷ 5.2.5.1 Parallel Execution ◁](#).

5.2.5 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.

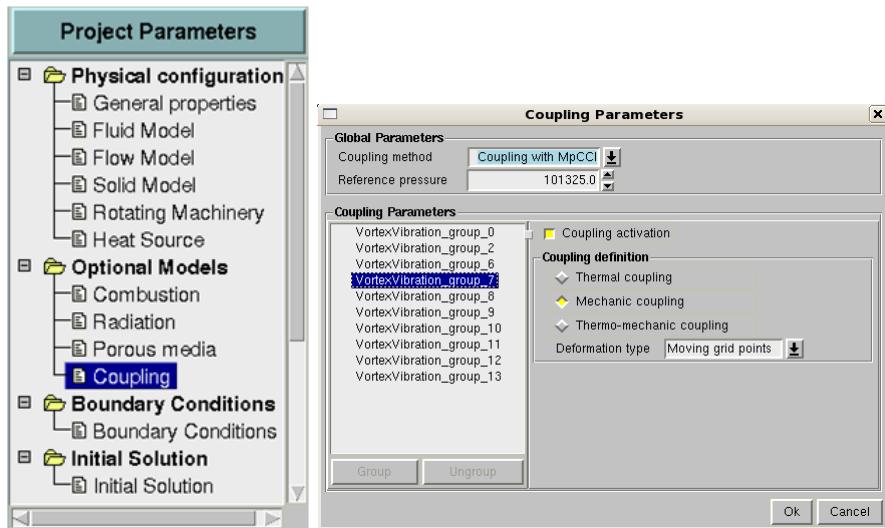


Figure 3: Identifying boundaries in FINE/Open GUI

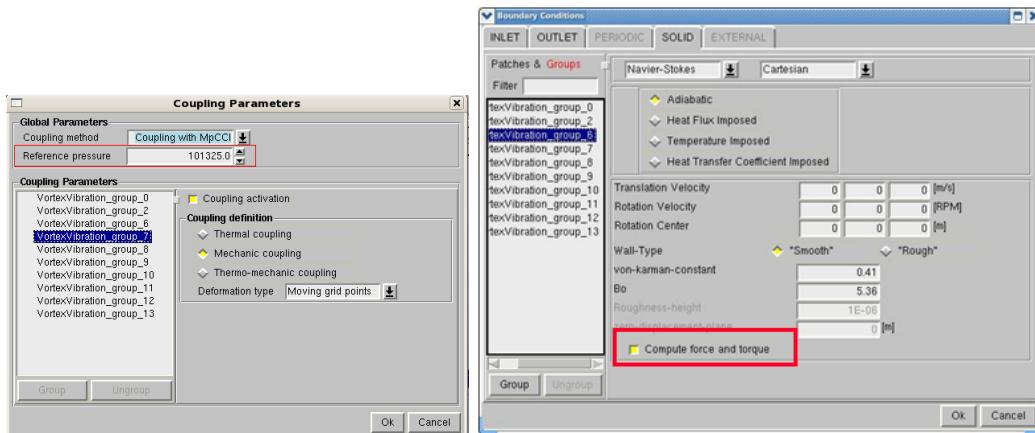


Figure 4: Setting a reference pressure and Force and Torque in FINE/Open GUI

5.2.5.1 Parallel Execution

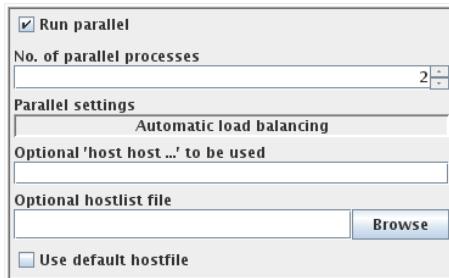


Figure 5: Parallel options

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 [▷ 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 ◁](#).

5.2.5.2 Batch Execution

FINE/Open always runs as a batch process.

5.2.6 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:
<code>mpcci FINEOpen [-]option</code>
Synopsis:
<code>'mpcci FINEOpen'</code> is used to get information about FINEOpen.
Options:
<code>-diff <run1> <run2></code>
Run the scanner on two .run files and print the differences.

```
-help
    This screen.

-info
    List verbose information about all FINEOpen releases.

-releases
    List all FINEOpen releases which MpCCI can find on the local system.

-scan <input-file>
    Run the scanner and create an output file.
```

The subcommands `diff`, `info`, `releases` and `scan` are described in [▷ 1.1 Common MpCCI Subcommands for Simulation Codes ◁](#).

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.

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

Physical Domains	CFD, Fluid, FluidThermal, Turbomachinery
Company Name	NUMECA International
Company Homepage	www.numeca.com
Support	support@numeca.com
Tutorials	▷ VII-2 Vortex-Induced Vibration of a Thin-Walled Structure ◁ ▷ VII-4 Elastic Flap in a Duct ◁

6.1.1 Supported Coupling Schemes

FINE/Turbo supports exchange before solution. Unidirectional and bidirectional transfer is possible.

6.1.2 Supported Platforms and Versions

MPCCI_ARCH: Code platform	Supported versions					
	91_1	91_2	91_3	101	102	111
lnx4_x64: x86_64	X	X	X	X	X	X
windows_x64: win64	X	X	X	X	X	X

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 for coupled simulation using MpCCI and for mesh deformation if needed.

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 ↳](#)).

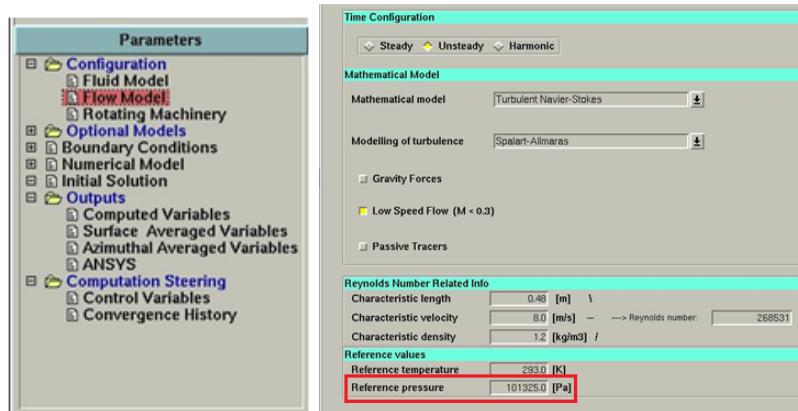


Figure 1: Setting the reference pressure in FINE/Turbo GUI

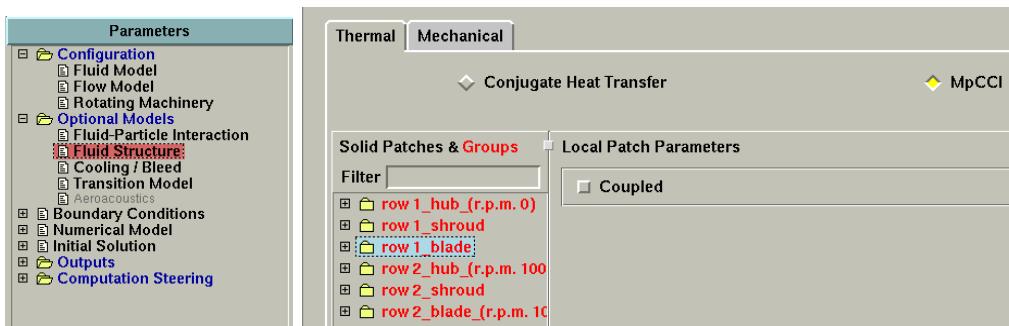


Figure 2: Setting coupled thermal boundaries in FINE/Turbo GUI

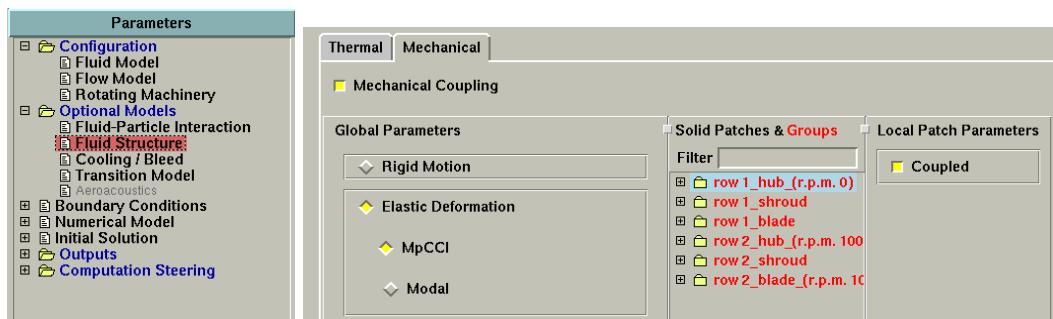


Figure 3: Setting coupled mechanical boundaries in FINE/Turbo GUI

6.2.2 Models Step

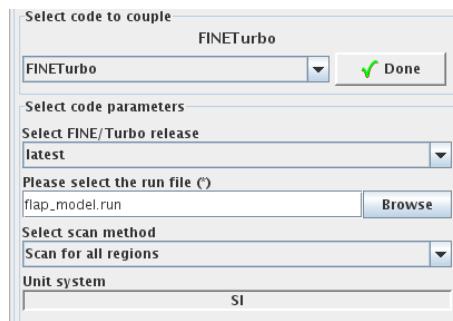


Figure 4: FINE/Turbo options in the Models Step

In the Models Step, the following options must be chosen:

Please select the 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).

Please select the run file Select the FINE/Turbo input file "`*.run`".

Select 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/Turbo GUI).

Unit system As FINE/Turbo works in SI units, no choice is given for the unit system to the user.

6.2.3 Coupling Step

FINE/Turbo supports the following quantities for coupling:

Quantity	Dim.	Default Value	Integration Type	Coupling Dimension	Location	Send Option	Receive Option
AbsPressure	Scalar	0.0 N/m ²	flux dens.	Face	Code	Direct	
DeltaTime	Scalar	1.0 s	g-min	Global	global	Direct	

Quantity	Dim.	Default Value	Integration Type	Coupling Dimension	Location	Send Option	Receive Option
Density	Scalar	1.0 kg/m ³	field	Volume	Code	Direct	
NPosition	Vector	0.0 m	mesh coordinate	Face	Code		Direct
OverPressure	Scalar	0.0 N/m ²	flux dens.	Face	Code	Direct	
PhysicalTime	Scalar	0.0 s	g-min	Global	global	Direct	
RefPressure	Scalar	1.12e5 N/m ²	g-max	Global	global	Direct	
RelWallForce	Vector	0.0 N	flux integral	Face	Code	Direct	
WallForce	Vector	0.0 N	flux integral	Face	Code	Direct	
WallHeatFlux	Scalar	0.0 W/m ²	flux dens.	Face	Code	Direct	
WallTemp	Scalar	300.0 K	field	Face	Code	Direct	Direct

 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.

6.2.4 Go Step

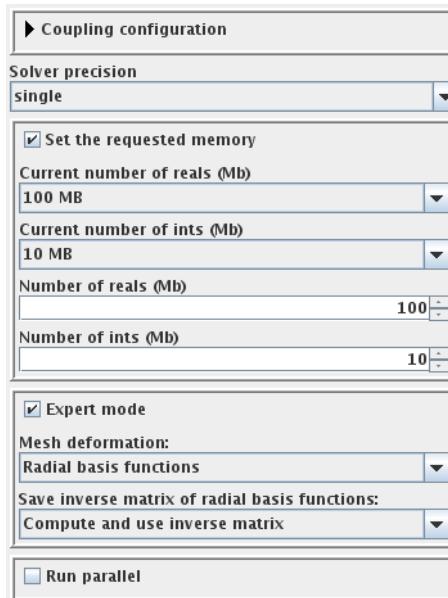


Figure 5: FINE/Turbo options in the Go Step

Coupling configuration See ▷V-4.8.2 Coupling Configuration Parameters◀.

Define the coupling scheme FINE/Turbo only supports Explicit-Transient and Explicit-SteadyState coupling schemes.

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 [6.2.5.1 Parallel Execution](#).

6.2.5 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.

6.2.5.1 Parallel Execution

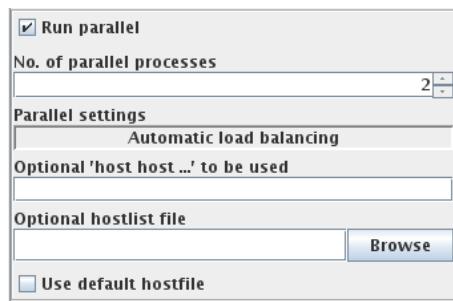


Figure 6: FINE/Turbo options for a parallel run

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 [▷ 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 ◁](#).

(!) 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.

6.2.5.2 Batch Execution

FINE/Turbo always runs in batch mode.

6.2.6 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:

<p>Usage:</p> <pre>mpcci FINETurbo [-]option</pre> <p>Synopsis:</p> <pre>'mpcci FINETurbo' is used to get information about FINETurbo.</pre> <p>Options:</p> <ul style="list-style-type: none"> -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. -intmem <RUN-file> Show the number of ints used by this computation. -realmem <RUN-file> Show the number of reals used by this computation. -releases List all FINETurbo releases which MpCCI can find on the local system. -scan <RUN-file>

Run the scanner and create an output file.

The subcommands `diff`, `info`, `releases` and `scan` are described in [▷ 1.1 Common MpCCI Subcommands for Simulation Codes](#).

mpcci fineturbo -intmem <RUN-file>

The `intmem` subcommand shows the number of ints used by the computation for the given run file.

mpcci fineturbo -realmem <RUN-file>

The `realmem` subcommand shows the number of reals used by the computation for the given run file.

6.4 Trouble shooting, open issues and known bugs

Feature:

Thermal coupling simulation

Version:

FINE/Turbo 9.x releases

Problem:

The quantity WallTemp is not properly imported by FINE/Turbo

Workaround:

The user should manually divide the mesh bloc in order to have surface bloc with only one patch for the co-simulation. This issue should be fixed in FINE/Turbo 10.1.

7 Flowmaster

7.1 Quick Information

Physical Domains	Fluid and Thermal System simulation
Company Name	Flowmaster
Company Homepage	www.flowmaster.com
Support	http://supportnet.mentor.com
Tutorials	▷ VII-11 Y-Junction ▷

7.1.1 Supported Coupling Schemes

MpCCI supports coupling with Flowmaster 7.6 and higher.

Depending on the Flowmaster analysis type the following scheme is applied:

- Steady-State Analysis
 1. Flowmaster receives the data.
 2. Flowmaster analysis is requested.
 3. Flowmaster sends the data.
- Transient
 1. Flowmaster exchanges the data (send and receive).
 2. Flowmaster analysis is requested.

7.1.2 Supported Platforms and Versions

The following versions of Flowmaster are supported by MpCCI:

MPCCI_ARCH: Code platform	Supported versions							
	7.6	7.7	7.8	8.0	8.1	8.2	9.0	9.2
windows_x64: mswin	X	X	X	X	X	X	X	X

7.1.3 References

Flowmaster documentation is part of the Flowmaster distribution.

7.1.4 Adapter Description

The Flowmaster adapter is an executable based on the Flowmaster COM interface.

7.1.5 Prerequisites for a Coupled Simulation

To run a coupled simulation you need the following:

- Ordinary Flowmaster installation.

7.2 Coupling Process

You have to create and validate a Flowmaster 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 Flowmaster. The connection to boundaries in a CFD model is realized within a Flowmaster 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.

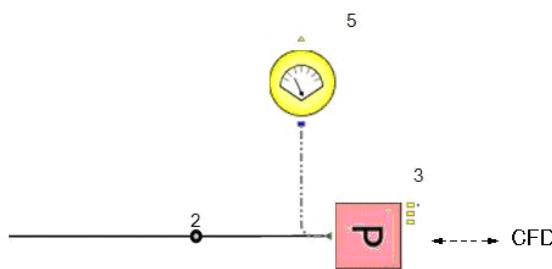


Figure 1: A Flowmaster pressure source connects to CFD mass flow boundary

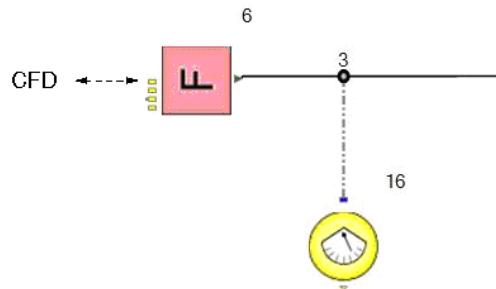


Figure 2: A Flowmaster flow source connects to CFD pressure boundary

! Note that the CFD prescribed flow boundary incorporates in Flowmaster a pressure source whereas the CFD pressure boundary requires a Flowmaster flow source as coupling partner. Components within the Flowmaster 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 Flowmaster 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**.

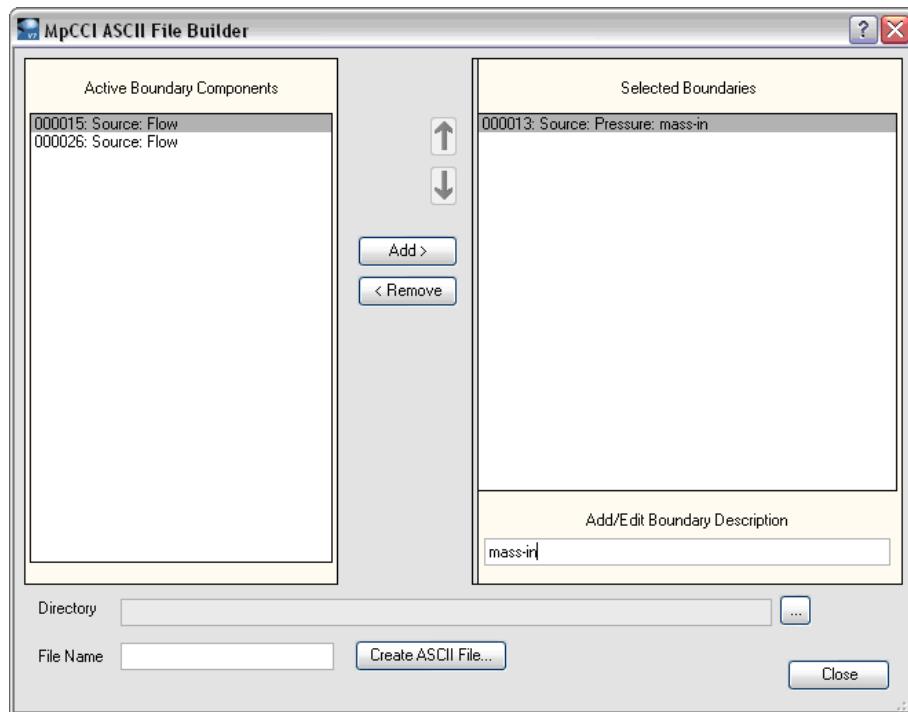


Figure 3: Export Flowmaster boundaries dialog

7.2.2 Models Step

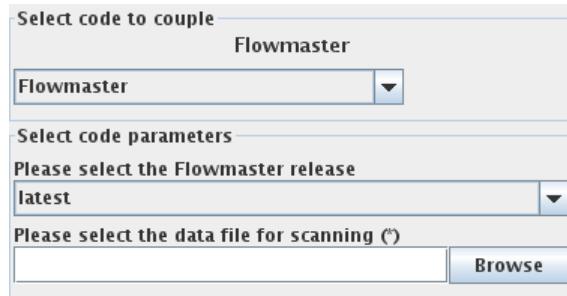


Figure 4: Flowmaster options in the Models Step

In the Models Step, the following options must be chosen:

Flowmaster release Select the release of Flowmaster 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 Flowmaster project.

The MpCCI scanner uses the Flowmaster data file to extract model information. This file has the suffix ".fmlink".

7.2.3 Coupling Step

The Flowmaster adapter only stores quantities directly (“Direct”). Flowmaster supports the following quantities for coupling:

Quantity	Dim.	Default Value	Integration Type	Coupling Dimension	Location	Send Option	Receive Option
DeltaTime	Scalar	1.0 s	g-min	Global	global	Direct	Direct
MassFlowRate	Scalar	0.0 kg/s	flux integral	Point	Code	Direct	Direct
MassFluxRate	Scalar	0.0 kg/m ² s	flux dens.	Point	Code	Direct	Direct
PhysicalTime	Scalar	0.0 s	g-min	Global	global	Direct	Direct
Temperature	Scalar	300.0 K	field	Point	Code	Direct	Direct
TotalPressure	Scalar	0.0 N/m ²	flux dens.	Point	Code	Direct	Direct
VelocityMagnitude	Scalar	0.0 m/s	field	Point	Code	Direct	Direct

The quantities are calculated as depicted exemplary for the field quantity temperature T and the flux quantity mass flow m . in [Figure 5](#)

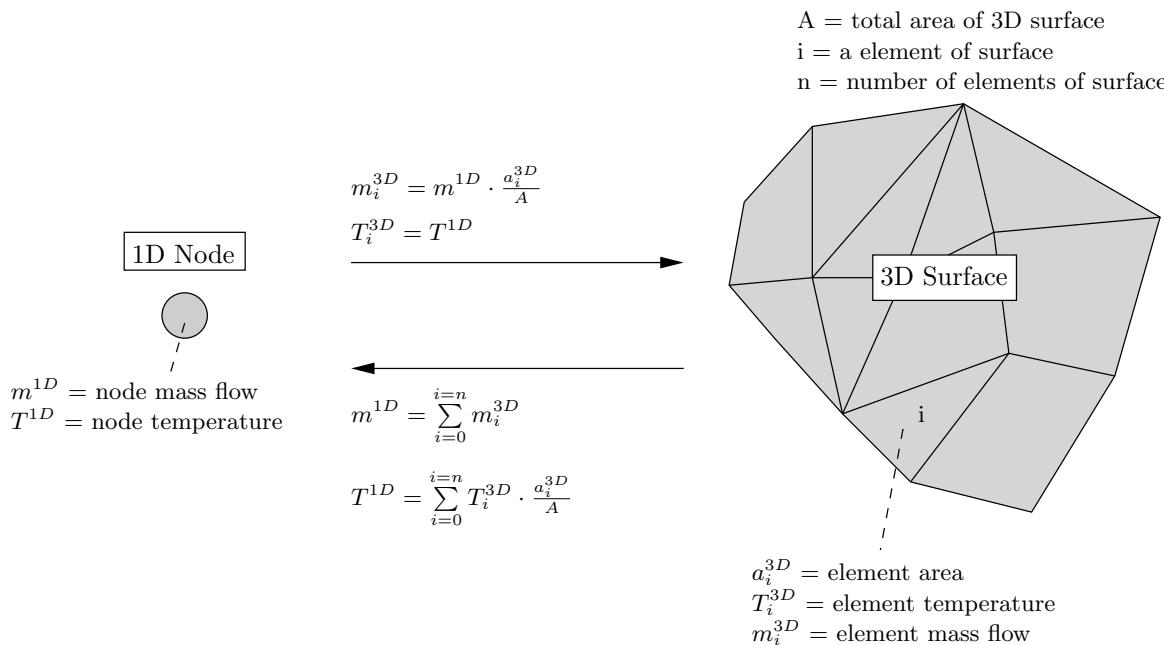


Figure 5: Exemplary calculation of flux quantity mass flow m and field quantity temperature T between 1D and 3D codes

7.2.4 Go Step

Coupling configuration	
User name (*)	admin
User password	<input type="password"/>
Working project name	Flowmaster
Analysis type	ST

Figure 6: Flowmaster options in the Go Step

Flowmaster offers a number of options in the Go panel, see [Figure 6](#).

Coupling configuration See [V-4.8.2 Coupling Configuration Parameters](#).

Define the coupling scheme Flowmaster only supports Explicit-Transient and Explicit-SteadyState coupling schemes.

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:

- Incompressible
 - SS** Steady state simulation.
 - SSH** Steady state simulation with heat transfer.
 - ST** Transient simulation.
 - STH** Transient simulation with heat transfer.
- Compressible
 - CS** Steady state simulation.
 - CT** Transient simulation.

7.2.4.1 Running the Computation

By pressing the **Start** button in the Go Step of the MpCCI GUI, MpCCI starts the Flowmaster application. The output of the Flowmaster simulation is logged in a window and a file "mpcci_<model name>.log".

7.2.4.2 Post-Processing

You may use the Flowmaster interface and analyze the results by querying the database or creating a report.

7.3 Code-Specific MpCCI Commands

The MpCCI subcommands available for Flowmaster are:

Usage:	<code>mpcci Flowmaster [-]option</code>
Synopsis:	
'mpcci Flowmaster' is used to get information about Flowmaster.	
Options:	
-diff <file1> <file2>	Run the scanner on two .fmlink files and print the differences.
-help	This screen.
-info	List verbose information about all Flowmaster releases.
-releases	List all Flowmaster 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 Description

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>/Flowmaster/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 Flowmaster adapter, e.g. "MpCCILink.exe". The connection to MpCCI is established using this executable and the implementation is based on the Flowmaster 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 Flowmaster adapter. After the network analysis is complete the mass flow is read by the boundary component and returned to Flowmaster adapter.

7.5 Trouble Shooting, Open Issues and Known Bugs

Problem:

The Flowmaster 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 Flowmaster 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 Flowmaster installation.

1. Open two file explorers, one pointing to "C:/WINDOWS/assembly" and the other to the Flowmaster 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 Flowmaster 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 Flowmaster support about how to register dlls to the GAC or assembly registry.

8 FLUENT

8.1 Quick Information

Physical Domains	CFD, Fluid, FluidThermal, FluidPlasma
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	▷ VII-2 Vortex-Induced Vibration of a Thin-Walled Structure ◁ ▷ VII-3 Driven Cavity ◁ ▷ VII-4 Elastic Flap in a Duct ◁ ▷ VII-5 Elastic Flap in Water ◁ ▷ VII-6 Exhaust Manifold ◁ ▷ VII-7 Busbar System ◁ ▷ VII-8 Three Phase Transformer ◁ ▷ VII-9 Pipe Nozzle ◁ ▷ VII-10 Cube in a Duct Heater ◁ ▷ VII-11 Y-Junction ◁

8.1.1 Supported Coupling Schemes

FLUENT supports exchange before and after solution. Unidirectional and bidirectional transfer is possible.

Simulation Type	Exchange
Steady state	before solution
Transient	before solution

8.1.2 Supported Platforms and Versions

MpCCI ARCH: Code platform	Supported versions										
	14.5.0	14.5.7	15.0.0	15.0.7	16.0.0	16.1.0	16.2.0	17.0.0	17.1.0	17.2.0	18.0.0
lnx4_x64: lnamdd64	X	X	X	X	X	X	X	X	X	X	X
windows_x64: win64	X		X	X	X	X	X	X	X	X	X

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.

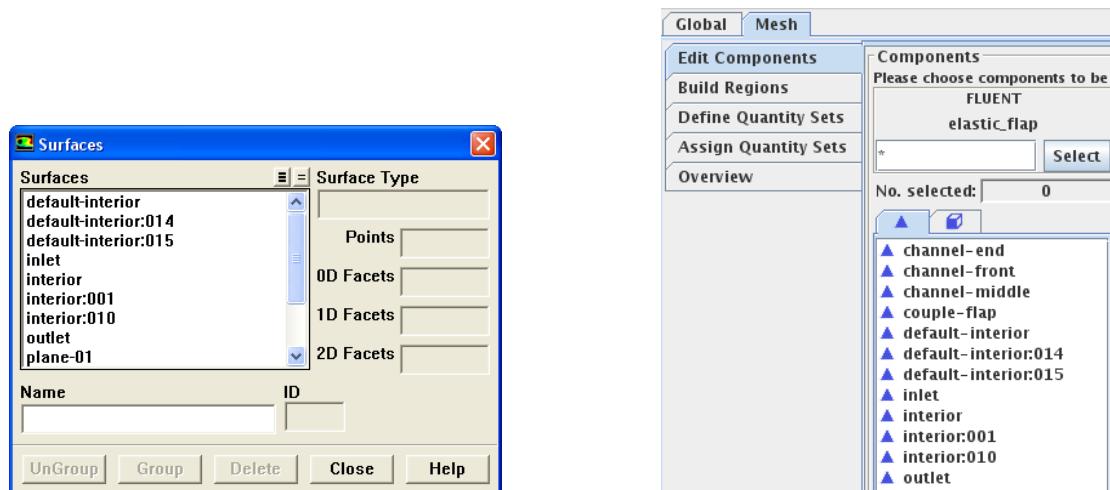


Figure 1: List of surfaces in FLUENT (left) and in the Coupling Step of the MpCCI GUI (right).

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:

- Auto install/make libmpcci for installation of the user-defined library,
- Auto load libmpcci to load it,

- Auto hook functions to hook the library functions, i.e. they will be called at certain stages of the simulation.

See [▷ 8.2.4 Go Step ◄](#) for a complete description of the Go Step options.

In some cases it is preferable not to use these automatic settings. Instead the necessary functions can also be hooked manually. A description of the functions is given in [▷ 8.4 Code Adapter Reference ◄](#).

Some functions can also be called while running the computation via the MpCCI Control Panel in the FLUENT GUI, see [▷ 8.2.5.1 The MpCCI Control Panel ◄](#).

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** ([▷ 8.2.5.2 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

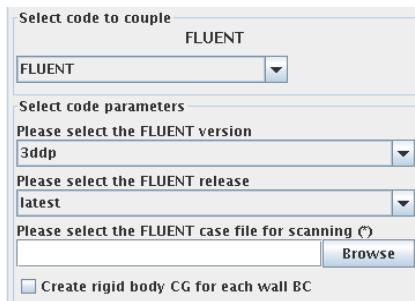


Figure 2: FLUENT options in the Models Step

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.

case file Select the case file of your FLUENT model.

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.

8.2.3 Coupling 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:

Quantity	Dim.	Default Value	Integration Type	Coupling Dimension	Location	Send Option	Receive Option
AbsPressure	Scalar	0.0 N/m ²	flux dens.	Face, Volume	Code	Dir	UDM
AcstPressure	Scalar	0.0 N/m ²	flux dens.	Volume	Code	Dir	UDM
AngularVelocity	Vector	0.0 rad/s	field	Point	Code		Buf
BodyForce	Vector	0.0 N/m ³	flux dens.	Volume	Code	UDM	UDM
CGAngle	Vector	0.0 rad	g-max	Global	global	Dir	Dir

Quantity	Dim.	Default Value	Integration Type	Coupling Dimension	Location	Send Option	Receive Option
CGOmega	Vector	0.0 rad/s	g-max	Global	global	Dir	Dir
CGPosition	Vector	0.0 m	g-max	Global	global	Dir	Dir
CGVelocity	Vector	0.0 m/s	g-max	Global	global	Dir	Dir
ChargeDensity	Scalar	0.0 C/m ³	field	Volume	Code	UDM	UDM
Current1	Scalar	0.0 A	g-max	Global	global	Dir	Dir
Current2	Scalar	0.0 A	g-max	Global	global	Dir	Dir
Current3	Scalar	0.0 A	g-max	Global	global	Dir	Dir
Current4	Scalar	0.0 A	g-max	Global	global	Dir	Dir
CurrentDensity	Vector	0.0 A/m ²	flux dens.	Face, Volume	Code	UDM, UDS	UDM
DeltaTime	Scalar	1.0 s	g-min	Global	global	Dir	Dir
Density	Scalar	1.0 kg/m ³	field	Volume	Code	Dir	UDM
DynPressure	Scalar	0.0 N/m ²	flux dens.	Face	Code	Dir	UDM
ElectrCond1	Scalar	0.0 S/m	field	Face, Volume	Code	UDM	UDM
ElectrCond3	Vector	0.0 S/m	field	Face, Volume	Code	UDM	UDM
ElectrCondX	Scalar	0.0 S/m	field	Volume	Code	UDM	UDM
ElectrCondY	Scalar	0.0 S/m	field	Volume	Code	UDM	UDM
ElectrCondZ	Scalar	0.0 S/m	field	Volume	Code	UDM	UDM
ElectricField	Vector	0.0 V/m	field	Volume	Code	UDM, UDS	UDM
ElectricFlux	Vector	0.0 C/m ²	flux dens.	Face, Volume	Code	UDM, UDS	UDM
ElectrRes1	Scalar	0.0 ohm m	field	Face, Volume	Code	UDM	UDM
ElectrRes3	Vector	0.0 ohm m	field	Face, Volume	Code	UDM	UDM
ElectrResX	Scalar	0.0 ohm m	field	Volume	Code	UDM	UDM
ElectrResY	Scalar	0.0 ohm m	field	Volume	Code	UDM	UDM
ElectrResZ	Scalar	0.0 ohm m	field	Volume	Code	UDM	UDM
Enthalpy	Scalar	0.0 W/m ³	flux dens.	Volume	Code	Dir	UDM
FilmTemp	Scalar	300.0 K	field	Face	Code	Dir	UDM
Force	Vector	0.0 N	flux integral	Face	Code	Dir	UDM
HeatFlux	Vector	0.0 W/m ²	flux dens.	Volume	Code	UDM	UDM
HeatRate	Scalar	0.0 W	field	Face	Code	Dir	UDM
HeatSource	Scalar	0.0 W/m ³	flux dens.	Volume	Code	UDM	UDM
IntFlag	Scalar	0	g-max	Global	global	Dir	Dir
IterationNo	Scalar	0	g-max	Global	global	Dir	Dir
JouleHeat	Scalar	0.0 W/m ³	flux dens.	Volume	Code	UDM	UDM
JouleHeatLin	Scalar	0.0 W/m ³ K	flux dens.	Volume	Code	UDM	UDM
LorentzForce	Vector	0.0 N/m ³	flux dens.	Volume	Code	UDM	UDM
MagneticField	Vector	0.0 A/m	field	Volume	Code	UDM, UDS	UDM
MagneticFlux	Vector	0.0 T	flux dens.	Face, Volume	Code	UDM, UDS	UDM
MassFlowRate	Scalar	0.0 kg/s	flux integral	Face	Code	Dir	UDM
MassFluxRate	Scalar	0.0 kg/m ² s	flux dens.	Face	Code	Dir	UDM
NPosition	Vector	0.0 m	mesh coordinate	Face, Volume	Code	Dir	Buf

Quantity	Dim.	Default Value	Integration Type	Coupling Dimension	Location	Send Option	Receive Option
OverPressure	Scalar	0.0 N/m ²	flux dens.	Face, Volume	Code	Dir	UDM
PhysicalTime	Scalar	0.0 s	g-min	Global	global	Dir	Dir
PorePressure	Scalar	0.0 N/m ²	flux dens.	Volume	Code	Dir	UDM
RealFlag	Scalar	0.0	g-max	Global	global	Dir	Dir
RefPressure	Scalar	1.12e5 N/m ²	g-max	Global	global	Dir	Dir
RelWallForce	Vector	0.0 N	flux integral	Face	Code	Dir	UDM
Residual	Scalar	0.0	g-max	Global	global	Dir	Dir
SpecificHeat	Scalar	1.0 J/kg K	field	Volume	Code	Dir	UDM
Temperature	Scalar	300.0 K	field	Face, Volume	Code	Dir	UDM
ThermCond1	Scalar	0.0 W/m K	field	Face, Volume	Code	Dir	UDM
ThermCond3	Vector	0.0 W/m K	field	Face, Volume	Code	Dir	UDM
ThermCondX	Scalar	0.0 W/m K	field	Volume	Code	Dir	UDM
ThermCondY	Scalar	0.0 W/m K	field	Volume	Code	Dir	UDM
ThermCondZ	Scalar	0.0 W/m K	field	Volume	Code	Dir	UDM
TimeStepNo	Scalar	0	g-max	Global	global	Dir	Dir
TotalPressure	Scalar	0.0 N/m ²	flux dens.	Face	Code	Dir	UDM
Velocity	Vector	0.0 m/s	field	Point, Face, Volume	Code	Dir	UDM, Buf
VelocityMagnitude	Scalar	0.0 m/s	field	Face	Code	Dir	UDM
Voltage1	Scalar	0.0 V	g-max	Global	global	Dir	Dir
Voltage2	Scalar	0.0 V	g-max	Global	global	Dir	Dir
Voltage3	Scalar	0.0 V	g-max	Global	global	Dir	Dir
Voltage4	Scalar	0.0 V	g-max	Global	global	Dir	Dir
WallForce	Vector	0.0 N	flux integral	Face	Code	Dir	UDM
WallHeatFlux	Scalar	0.0 W/m ²	flux dens.	Face	Code	Dir	UDM
WallHTCoeff	Scalar	0.0 W/m ² K	field	Face	Code	Dir	UDM
WallTemp	Scalar	300.0 K	field	Face	Code	Dir	UDM

8.2.4 Go Step

FLUENT offers a number of options in the Go panel, see [Figure 3](#).

Coupling configuration See [V-4.8.2 Coupling Configuration Parameters](#).

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 [8.2.5.4 Batch Execution](#) for details.

Use vglrun control (Linux 64 only) The selection of this option is only considered on Linux 64 bit architecture. If selected, FLUENT will run under the control of VirtualGL.

Project name Name of the FLUENT project, which is also used as name for the output files. If no name is given here, the name of the case file is used.

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

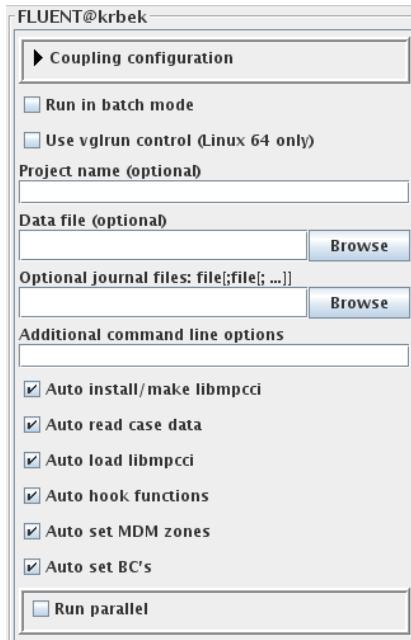


Figure 3: FLUENT options in the Go Step

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.

Auto load libmpcci Load the code adapter library. This option 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 [8.2.5.3 Parallel Execution](#).

8.2.5 Running the Computation

Coupled simulations can be run using the FLUENT window. For running a simulation without a graphical interface, see [8.2.5.4 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
```

(output is followed by a list of all functions)

- 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.

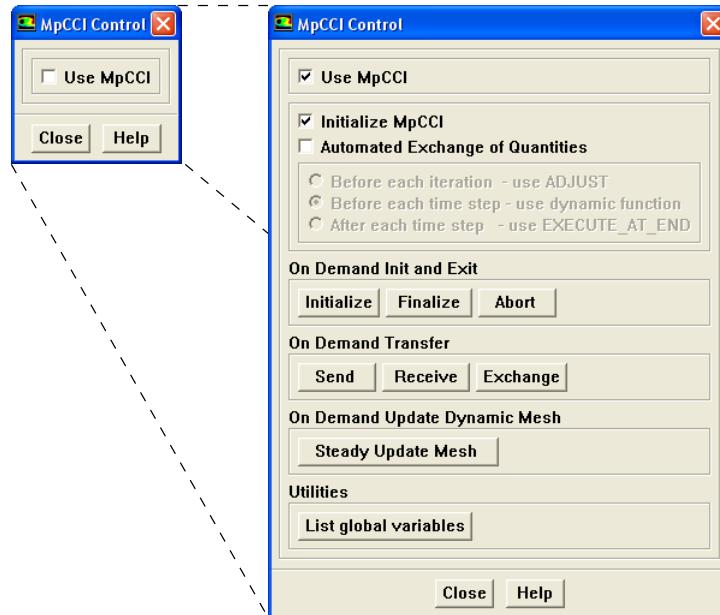


Figure 4: MpCCI Control Panel in FLUENT

8.2.5.1 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** initializes 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 Coupling 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.

8.2.5.2 The MpCCI Run FSI

The MpCCI Run FSI panel is an extra panel in the FLUENT GUI, which is available from the FLUENT menu **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 subcycling 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:

```
> (mpcci-solve n)
```

where n represents the number of time steps or iterations to be calculated.

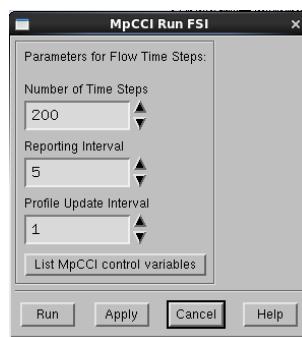


Figure 5: MpCCI Run FSI panel in FLUENT for transient implicit solution

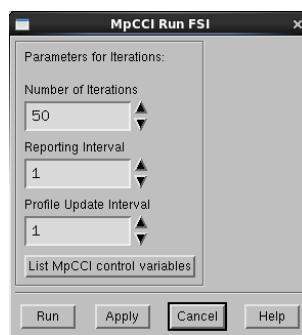


Figure 6: MpCCI Run FSI panel in FLUENT for steady state solution

8.2.5.3 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](#).

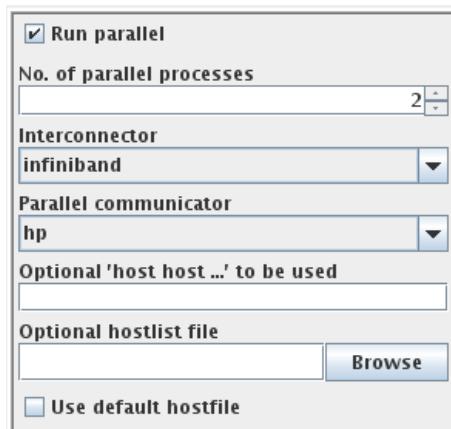


Figure 7: FLUENT options for a parallel run

(i) Please read also chapter “31. Parallel Processing” of the FLUENT 6.3 User’s Guide.

No. 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.

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](#).

8.2.5.4 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:

```
; ; Initialize flow and connect to MpCCI
/solve initialize initialize-flow
; ; Perform unsteady iterations for a specified number of time steps
/solve dual-time-iterate 5 20
; ; Exit simulation
/exit
```

If you want to do subcycling (iterations without exchange), you should open the Coupling configuration and select the Use subcycling option. Then you can provide a No. of steps without coupling.

The journal file might contain the usual commands to start the iterations:

```
; ; Define autosave of case and data files
```

```
/file/autosave/case-frequency 10
/file/autosave/data-frequency 10
/file/autosave/overwrite-existing-files yes
;; Initialize flow
/solve/initialize initialize
;; Iterate
/solve/iterate 1000
;; Write case and data file
wcd file
;; 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 ▷.

8.2.5.5 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-initact` 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:	<code>mpcci FLUENT [-]option</code>
Synopsis:	Use 'mpcci FLUENT' to get information about FLUENT and to build/install your private adapter ...
Options:	<ul style="list-style-type: none"> -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 ...
```

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 92](#).

 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 92](#).

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 Coupling 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.

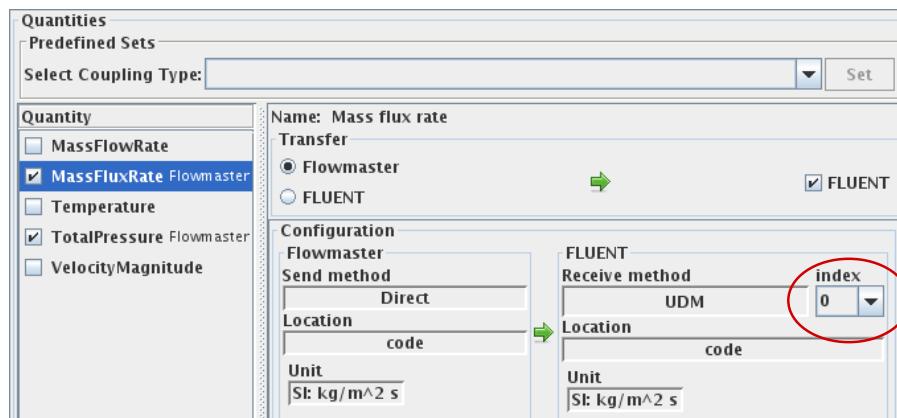


Figure 8: UDM storage index for quantities received by FLUENT.

In addition to this minimal subset of user-function calls, certain situations 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→Run Calculation...). MpCCI will set the Time Stepping Method to Adaptive and select UDF_Deltat::libmpcci as user-defined time step.

UDF Hook	MpCCI Function	Purpose / Description
INIT	UDF_Initialize	Initialize the connection to MpCCI.
EXECUTE_AT_END	UDF_At_end	MpCCI data transfer.
EXECUTE_AT_EXIT	UDF_At_exit	Shutdown the MpCCI connection.
ADJUST	UDF_Adjust	MpCCI data transfer.
ADJUST	UDF_Adjust_Once	MpCCI data transfer.
ADJUST	UDF_Adjust_implicit	MpCCI data transfer.
DELTAT	UDF_Deltat	Set the time step size if it is received by FLUENT.
ON_DEMAND	UDF_List_globals UDF_Dynamic_exchange UDF_Dynamic_exchange_implicit UDF_Dynamic_exchange_Update_Time UDF_Exit_MpCCI UDF_Abort_MpCCI UDF_Init_MpCCI UDF_Send_on_demand UDF_Recv_on_demand UDF_Xchg_on_demand UDF_Remesh_total_on_demand UDF_Remesh_move_on_demand	List global variables. MpCCI data transfer in explicit coupling. MpCCI data transfer in iterative coupling. MpCCI update time information transfer. Shutdown the MpCCI connection. Abort the MpCCI connection. Initialize the connection to MpCCI. MpCCI data transfer: Send only. MpCCI data transfer: Receive only. MpCCI data transfer. MpCCI receives a full remeshing information. MpCCI receives a deformation information.
RW_FILE	UDF_Read_globals UDF_Write_globals	Read global data from " *.dat " file. Write global data into " *.dat " file.
CG_MOTION	UDF_CG_motion	Set the rigid body motion if it is received by FLUENT.
GRID_MOTION	UDF_Grid_position UDF_Grid_position_implicit UDF_Grid_Motion_All UDF_Grid_Motion_All_implicit	Set nodal coordinates to the received position. MpCCI data transfer and set nodal coordinates to the received position in iterative coupling. MpCCI data transfer and set nodal coordinates to the received position in explicit coupling. MpCCI data transfer and set nodal coordinates to the received position in iterative coupling.
PROFILE	UDM00_Profile : UDM10_Profile	Set boundary profile values stored in UDM 0 - 10.
SOURCE	UDM00_Source : UDM10_Source	Set a cell source value stored in UDM 0 - 10.
PROPERTY	UDM00_Property : UDM10_Property	Set a cell property value stored in UDM 0 - 10.

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 the MpCCI lib is hooked for temperature value ([▷ 8.2.1.1 Setting 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:
`/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 JMAG

9.1 Quick Information

Physical Domains	Electromagnetism Thermal
Company Name	JSOL Corporation
Company Homepage	www.jmag-international.com
Support	jmag-support@sci.jsol.co.jp
Tutorials	▷ VII-7 Busbar System ◁ ▷ VII-8 Three Phase Transformer ◁

9.1.1 Supported Coupling Schemes

MpCCI supports coupling with JMAG. JMAG supports exchange before solution. Unidirectional and bidirectional transfer is possible.

9.1.2 Supported Platforms and Versions

The following versions of JMAG are supported by MpCCI:

MPCCI_ARCH: Code platform	Supported versions					
	13.0	13.1	14.0	14.1	15.0	15.1
lnx4_x64: linux64	X	X	X	X	X	X
windows_x64: x64	X	X	X	X	X	X

9.1.3 References

JMAG Documentation is part of the JMAG distribution.

9.1.4 Adapter Description

The JMAG-MpCCI multiphysics coupling is enabled via the JCoupled module. The JMAG-MpCCI code adapter is developed by Fraunhofer SCAI in cooperation with JSOL Corporation. It is delivered as shared library implementing the JMAG Co-Simulation API.

9.1.5 Prerequisites for a Coupled Simulation

To run a coupled simulation you need the following:

- Ordinary JMAG installation.
- JW_MPCCI token license for JMAG-MpCCI multiphysics capability.
- A JMAG 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).

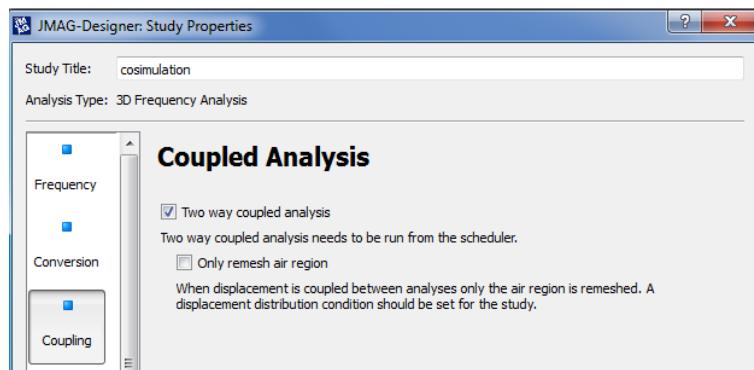


Figure 1: JMAG coupled analysis

Enabling the output of quantities For the current list of supported quantities ([▷ 9.2.3 Coupling 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

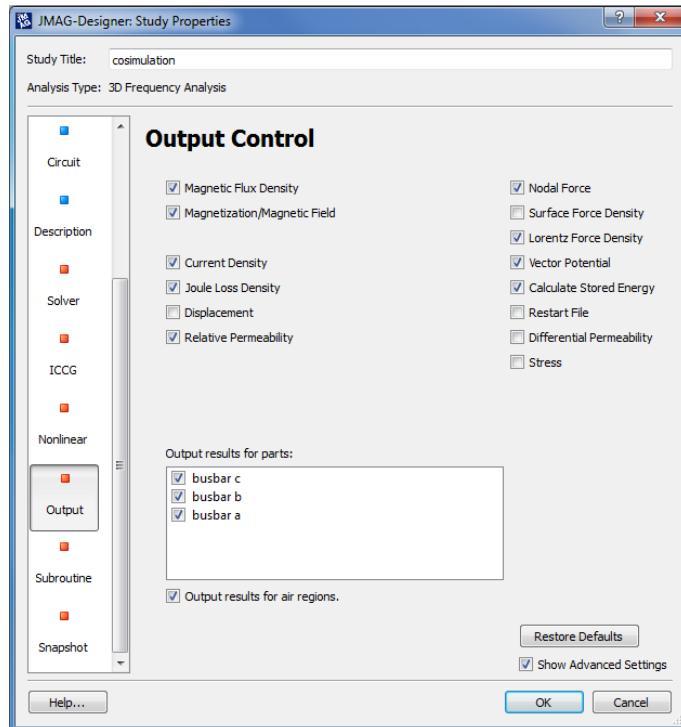


Figure 2: JMAG output control

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.

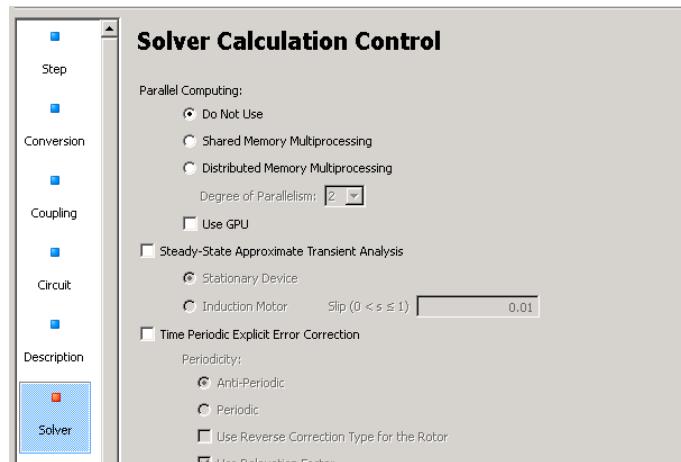


Figure 3: JMAG solver option

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-

ure 4) and graph of type Conductivity vs Temperature should be created.

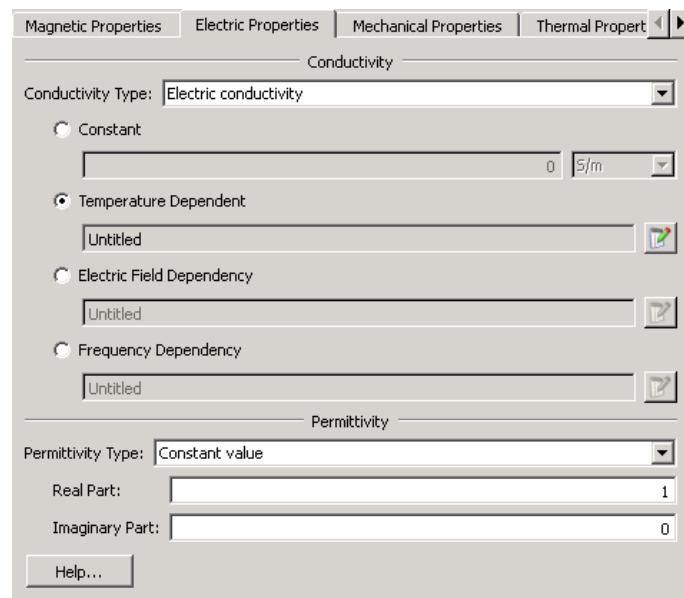


Figure 4: JMAG material properties

9.2.2 Models Step

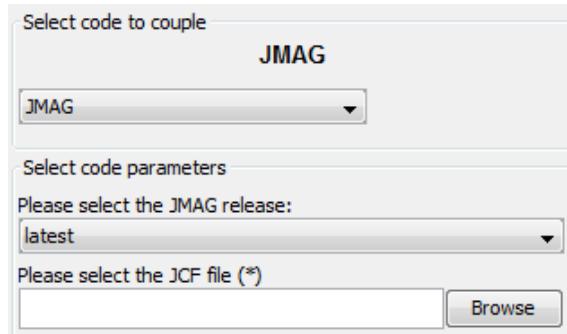


Figure 5: JMAG options in the Models Step

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 Coupling Step

The JMAG adapter stores the quantities using a direct memory access to the JMAG solver (“Direct”). JMAG supports the following quantities for coupling:

Quantity	Dim.	Default Value	Integration Type	Coupling Dimension	Location	Send Option	Receive Option
CurrentDensity	Vector	0.0 A/m ²	flux dens.	Volume	Element	Direct	
DeltaTime	Scalar	1.0 s	g-min	Global	global	Direct	
ElectrCond1	Scalar	0.0 S/m	field	Volume	Node		Direct
ElectrCond3	Vector	0.0 S/m	field	Volume	Node		Direct
JouleHeat	Scalar	0.0 W/m ³	flux dens.	Volume	Element	Direct	
LorentzForce	Vector	0.0 N/m ³	flux dens.	Volume	Element	Direct	
NPosition	Vector	0.0 m	mesh coordinate	Face, Volume	Node	Direct	Direct
Temperature	Scalar	300.0 K	field	Face, Volume	Node		Direct

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.

9.2.4 Go Step

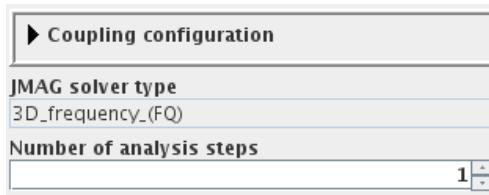


Figure 6: JMAG options in the Go Step

JMAG offers a number of options in the Go panel, see [Figure 6](#).

Coupling configuration See [V-4.8.2 Coupling Configuration Parameters](#)

Define the coupling scheme JMAG only supports Explicit-Transient and Explicit-SteadyState coupling schemes.

JMAG solver type This shows the current solver used for the model. You will see the following possible solver types:

- 3D_transient_(TR)
- 3D_frequency_(FQ)

Number of analysis step In case of a 3D_frequency_(FQ) this field is activated. You should provide the number of steps you want JMAG to perform. For each step a co-simulation will happen.

! 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.

9.2.4.1 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.

9.2.4.2 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:	
-bgcolor	Display the JMAG Designer background color.
-diff <JCF1> <JCF2>	Run the scanner on two .jcf files and print the differences.
-help	This screen.
-info	List verbose information about all JMAG releases.
-releases	List all JMAG releases which MpCCI can find.
-scan <JCF file>	Run the scanner and create a scanner output file.
-solver <JCF file>	Display the JMAG solver used from the JCF file.

The subcommands `diff`, `info`, `releases` and `scan` are described in [▷ 1.1 Common MpCCI Subcommands for Simulation Codes ▷](#).

`mpcci jmag -bgcolor`

The `bgcolor` subcommand gets the JMAG Designer background color and prints it to stdout.

`mpcci jmag -solver <JCF file>`

The `solver` subcommand gets the used solver from the specified JCF file and prints it to stdout.

9.4 Code Adapter Description

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/JMAG/adapters*".

This subdirectory is further divided by several release subdirectories, e.g. "11.0". The version directories are further divided by several architectures (e.g. "linux32", "linux64", "win32", "x64"). There you find the library files of the JMAG adapter (e.g. "libjmagmpcci.[so|dll]"). The connection to MpCCI is established using these shared libraries which are loaded by the JCoupled application automatically.

10 MATLAB

10.1 Quick Information

Physical Domains	System simulation, Controller
Company Name	MathWorks
Company Homepage	www.mathworks.com
Support	www.mathworks.com/support/
Tutorials	▷ VII-12 Spring Mass System ▲

10.1.1 Supported Coupling Schemes

With MATLAB the user can create his own algorithms for the solution process. It is easy to program a special solver for particular problems.

MATLAB is run via an m-file script, therefore all coupling algorithms are possible.

10.1.2 Supported Platforms and Versions

The following versions of MATLAB are supported by MpCCI:

MPCCI_ARCH: Code platform	Supported versions			
	R2013b	R2014b	R2015a	R2016b
lnx4_x64: glnxa64	X	X	X	X
windows_x64: win64	X	X	X	X

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:

```
%%$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 ▷ 10.2.4 Coupling Step ▷ for the list of supported global quantities)
- `ipoint`: Ipoint variables represent an integration point information. These 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](#).

```
%$mpcci cpoint displacement,force
%$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 `$mpcci init`. This call is therefore optional. This call requires that the iteration counter, time are correctly initialized too.
- During the iteration step or time stepping the user should choose where the MpCCI call should be placed: at the begin or at the end of the solving step (see [► V-3.4 Coupling Algorithms](#) for the difference in the coupling algorithm). Use the command `$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`.

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:

```
%$mpcci global var1 var2
```

Declare global variables list.

var1, var2 are the name of the variables.

 The variables can be separated by a space or a comma.

```
%$mpcci ipoint var1 var2
```

Declare ipoint variables list.

var1, var2 are the name of the variables.

 The variables can be separated by a space or a comma.

```
%$mpcci cpoint var1 var2
```

Declare cpoint variables list.

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:

```
%$mpcci cpoint (var2 var2_coord)
```

Basically you should pair the cpoint variable with its coordinates variable with parentheses.

```
%$mpcci surface (var1 var1_elem) (var2 var2_elem)
```

Declare surface variables list.

var1, var2 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,y1,z1;id2,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_elems = [id0,n0,n1,n2,-1,-1,-1,-1,-1;id1,n1,n4,n6,n8,-1,-1,-1,-1;-1;...]
```

To store the quantities exchanged by the co-simulation based on this mesh, MpCCI mex-function is using and expecting the declaration of following variable respecting this syntax:

```
<mesh_prefix>_<quantityname>
```

- The variable name is composed of the name of the mesh (var1) concatenated with an underscore and the name of the MpCCI quantity 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 WallForce and RelWallForce the force vector field may be computed automatically by MpCCI if MATLAB provides the pressure variable value using the naming convention: `<mesh_prefix>_pressure`. Otherwise MpCCI expects to find the variable `<mesh_prefix>_wallforce` or `<mesh_prefix>_relwallforce`.

%\$mpcci time var1

Declare var1 as 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.

%\$mpcci dt var1

Declare 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.

%\$mpcci iter var1

Declare 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 (Go Step), this value will be automatically ignored if Explicit-Transient is selected.

%\$mpcci conv var1

Declare 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 MATLAB solution has converged.
- 4: notify the continue status. MATLAB wants to continue the iteration.

If the variable is not declared MpCCI will set the value 0 (invalid state) for the data exchange call.

Co-simulation control statement:

%\$mpcci init

Declare MATLAB connection with MpCCI.

%\$mpcci sync

Declare a data exchange (send and receive) at this point.

%\$mpcci send

Declare a data exchange (send only) at this point.

%\$mpcci recv

Declare a data exchange (receive only) at this point.

%\$mpcci exit

Disconnect from the MpCCI server and stop MATLAB.

%\$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:

```
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
        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
end

%$mpcci exit

```

In the first comment we can see the definition of the coupling variables `displacement` and `force` as `cpoint`. The next four lines specify the auxiliary variables for the MpCCI server. This "`*.m`"-script demonstrates the iterative transient coupling. Therefore we can see two `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 MpCCI server from the corrector iteration. The second synchronization call sets the convergence flag to 3, which means that the iterative calculation in MATLAB is finished and the last communication for the current time is requested. The comment in the last line stops the communication with the MpCCI server.

Before MATLAB starts MpCCI will create a new MATLAB script file and replace each MpCCI control statement with the real mex function call. Below is an example of the patched MATLAB script.

```

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
        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

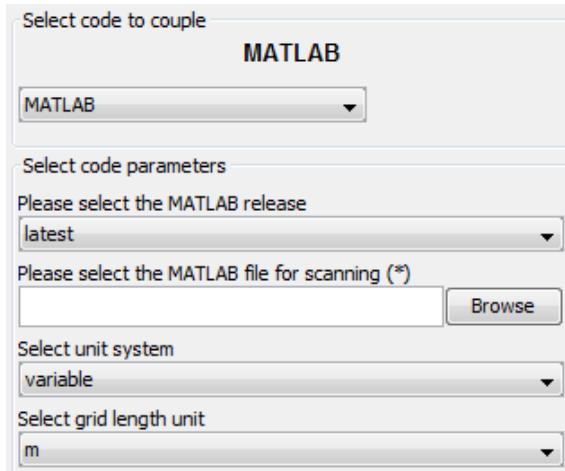


Figure 1: MATLAB options in the Models Step

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.

Select unit system Select the unit system which was used in MATLAB (see ▶1.2 Unit Systems◀).

10.2.4 Coupling 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:

Quantity	Dim.	Default Value	Integration Type	Coupling Dimension	Location	Send Option	Receive Option
AbsPressure	Scalar	0.0 N/m ²	flux dens.	Face	Code, Element	Direct	Direct
Acceleration	Vector	0.0 m/s ²	field	Point	Code	Direct	Direct
AngularAcceleration	Vector	0.0 rad/s ²	field	Point	Code	Direct	Direct
AngularCoordinate	Quaternion	0.0	mesh coordinate	Point	Code	Direct	Direct
AngularVelocity	Vector	0.0 rad/s	field	Point	Code	Direct	Direct
BodyForce	Vector	0.0 N/m ³	flux dens.	Volume	Node, Element	Direct	Direct
CurrentDensity	Vector	0.0 A/m ²	flux dens.	Volume	Node, Element	Direct	Direct
DeltaTime	Scalar	1.0 s	g-min	Global	global	Direct	Direct
ElectrCond1	Scalar	0.0 S/m	field	Volume	Node, Element	Direct	Direct

Quantity	Dim.	Default Value	Integration Type	Coupling Dimension	Location	Send Option	Receive Option
ElectrCond3	Vector	0.0 S/m	field	Volume	Node, Element	Direct	Direct
ElectrCondX	Scalar	0.0 S/m	field	Volume	Node, Element	Direct	Direct
ElectrCondY	Scalar	0.0 S/m	field	Volume	Node, Element	Direct	Direct
ElectrCondZ	Scalar	0.0 S/m	field	Volume	Node, Element	Direct	Direct
ElectricPot	Scalar	0.0 V	field	Volume	Node, Element	Direct	Direct
ElectrRes1	Scalar	0.0 ohm m	field	Volume	Node, Element	Direct	Direct
ElectrRes3	Vector	0.0 ohm m	field	Volume	Node, Element	Direct	Direct
ElectrResX	Scalar	0.0 ohm m	field	Volume	Node, Element	Direct	Direct
ElectrResY	Scalar	0.0 ohm m	field	Volume	Node, Element	Direct	Direct
ElectrResZ	Scalar	0.0 ohm m	field	Volume	Node, Element	Direct	Direct
FilmTemp	Scalar	300.0 K	field	Face	Code, Element	Direct	Direct
Force	Vector	0.0 N	flux integral	Point	Code	Direct	Direct
JouleHeat	Scalar	0.0 W/m ³	flux dens.	Volume	Node, Element	Direct	Direct
LorentzForce	Vector	0.0 N/m ³	flux dens.	Volume	Node, Element	Direct	Direct
MagneticFlux	Vector	0.0 T	flux dens.	Volume	Node, Element	Direct	Direct
MassFlowRate	Scalar	0.0 kg/s	flux integral	Point	Code	Direct	Direct
MassFluxRate	Scalar	0.0 kg/m ² s	flux dens.	Point	Code	Direct	Direct
NPosition	Vector	0.0 m	mesh coordinate	Face, Volume	Code	Direct	Direct
PointPosition	Vector	0.0 m	mesh coordinate	Point	Code	Direct	Direct
RealFlag	Scalar	0.0	g-max	Global	global	Direct	Direct
SpecificHeat	Scalar	1.0 J/kg K	field	Volume	Node, Element	Direct	Direct
Temperature	Scalar	300.0 K	field	Point, Volume	Code, Element	Direct	Direct
ThermCond1	Scalar	0.0 W/m K	field	Volume	Node, Element	Direct	Direct
ThermCond3	Vector	0.0 W/m K	field	Volume	Node, Element	Direct	Direct
ThermCondX	Scalar	0.0 W/m K	field	Volume	Node, Element	Direct	Direct
ThermCondY	Scalar	0.0 W/m K	field	Volume	Node, Element	Direct	Direct
ThermCondZ	Scalar	0.0 W/m K	field	Volume	Node, Element	Direct	Direct
Torque	Vector	0.0 N m	flux integral	Point	Code	Direct	Direct

Quantity	Dim.	Default Value	Integration Type	Coupling Dimension	Location	Send Option	Receive Option
TotalPressure	Scalar	0.0 N/m ²	flux dens.	Point	Code	Direct	Direct
Velocity	Vector	0.0 m/s	field	Point	Code	Direct	Direct
VelocityMagnitude	Scalar	0.0 m/s	field	Point	Code	Direct	Direct
WallForce	Vector	0.0 N	flux integral	Face	Code, Element	Direct	Direct
WallHTCoeff	Scalar	0.0 W/m ² K	field	Face	Code, Element	Direct	Direct
WallTemp	Scalar	300.0 K	field	Face	Code, Element	Direct	Direct

10.2.5 Go Step



Figure 2: MATLAB options in the Go Step

MATLAB offers nothing but the common coupling configuration in the Go Step (see [Figure 2](#)).

Coupling configuration See [▷ V-4.8.2 Coupling Configuration Parameters <](#).

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.

10.2.5.1 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 `info`, `releases` and `scan` are described in [▷1.1 Common MpCCI Subcommands for Simulation Codes](#).

mpcci matlab -join <main.m>

The `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 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/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. "glnxa64"). There you find the library files of the MATLAB adapter (e. g. "mpcci.[mexa64]").

11 MSC Adams

11.1 Quick Information

Physical Domains	Multibody dynamics
Company Name	MSC Software Corporation
Company Homepage	www.msccoftware.com/product/adams
Support	simcompanion.msccoftware.com
Tutorials	▷ VII-12 Spring Mass System ▷

11.1.1 Supported Coupling Schemes

Following coupling schemes are supported by MSC Adams for the dynamic analysis:

MSC Adams simulation mode	Transfer
Explicit	at the end of the time step
Implicit	before each iteration and at the end of the time step

MSC Adams supports iterative coupling, unidirectional and bidirectional transfer is possible.

11.1.2 Supported Platforms and Versions

MSC Adams is supported on the following platforms:

MPCCI_ARCH: Code platform	Supported versions							
	2013	2013.1	2013.2	2014	2015	2015.1	2016	2017
lnx4_x64: linux64	X	X	X	X	X	X	X	X
windows_x64: win64	X	X	X	X	X	X	X	X

For details on the platforms supported by MSC Adams see also the Platform Support page at www.msccoftware.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 Coupling 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 Coupling 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 [11.2.9 Go 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 [11.2.9 Go 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 (e.g. MSC Adams and Abaqus) 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-Transient is selected as the coupling scheme, MpCCI will only exchange data once at the beginning and once at the end of all static analyses and begin co-simulation in the dynamic or kinematic steps, see also [▷ 11.2.4 Multiple Statements ◁](#).

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 [▷ 11.2.2 Dynamic or Kinematic Simulation ◁](#)). All the parameters which you can use for the dynamic mode are also available.

For a static simulation on the MSC Adams side you have to select `Explicit-SteadyState` as the coupling scheme (see [▷ 11.2.9 Go Step ◁](#)). In this case only the iteration number (not the time) is used to tag the exchanged quantities. Please verify that the partner code also runs in an appropriate mode.

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-SteadyState` is selected as the coupling scheme, MpCCI will only exchange data during the static analyses and quit the co-simulation before the dynamic or kinematic steps start, see also [▷ 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 Iterative Coupling

MSC Adams supports iterative coupling. To set an implicit dynamic co-simulation, select `Implicit-Transient`. The same `Coupling Time Step Size` must be chosen for MSC Adams and the partner code. MSC Adams uses a Newton-type solver and exchanges data in every iteration until the `Maximum number of coupling 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 time 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 steps.

11.2.6 MSC Adams Template Products

11.2.6.1 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 [▷ 11.2.8 Coupling 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.

11.2.6.2 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 Coupling 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.

11.2.7 Models Step

In the Models Step, the following options must be chosen:

Please select the 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.

Please select the 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**.

Please select the 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.

Select 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.

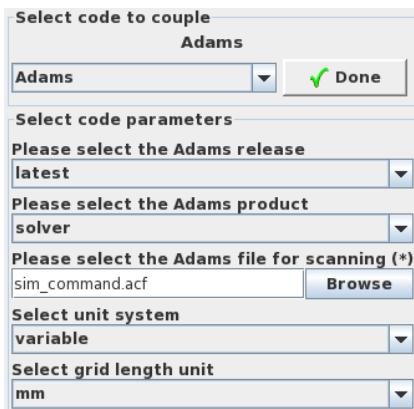


Figure 1: MSC Adams options in the Models Step

11.2.8 Coupling Step

MSC Adams supports the following quantities for coupling:

Quantity	Dim.	Default Value	Integration Type	Coupling Dimension	Location	Send Option	Receive Option
Acceleration	Vector	0.0 m/s ²	field	Point	Node	Direct	Usermemory
AngularAcceleration	Vector	0.0 rad/s ²	field	Point	Node	Direct	Usermemory
AngularCoordinate	Quaternion	0.0	mesh coordinate	Point	Node	Direct	Usermemory
AngularVelocity	Vector	0.0 rad/s	field	Point	Node	Direct	Usermemory
DeltaTime	Scalar	1.0 s	g-min	Global	global	Direct	Usermemory
Force	Vector	0.0 N	flux integral	Point	Node	Direct	Usermemory
PointPosition	Vector	0.0 m	mesh coordinate	Point	Node	Direct	Usermemory
RealFlag	Scalar	0.0	g-max	Global	global	Direct	Usermemory
Torque	Vector	0.0 N m	flux integral	Point	Node	Direct	Usermemory
Velocity	Vector	0.0 m/s	field	Point	Node	Direct	Usermemory

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:

Element	Receive	Send
GFORCE	Force, Torque	PointPosition, Velocity, Acceleration, Angular-Coordinates, -Velocity, -Acceleration
MARKER		PointPosition, Velocity, Acceleration, Angular-Coordinates, -Velocity, -Acceleration
MOTION	Acceleration, AngularAcceleration	
VARIABLE	Value of the Variable (one-dimensional!)	Value for the Variable (one-dimensional!)

In the case of **GFORCE** element the outgoing quantities are requested from the **I-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 [11.2.9 Go 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 [►11.2.9 Go 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-12 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 [►11.2.9 Go 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-12 Spring Mass System](#) for more details.

11.2.9 Go Step

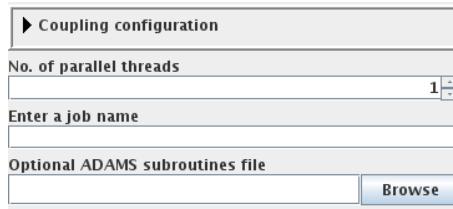


Figure 2: General MSC Adams options in the Go Step

In the Go Step following options are general and can be set independent of the selected coupling scheme (cf. [Figure 2](#)):

Coupling configuration See [V-4.8.2 Coupling Configuration Parameters](#).

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 ").

For Explicit-SteadyState coupling scheme following options can additionally be set (cf. [Figure 3](#) on the left):

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

For Explicit-Transient coupling scheme following options can additionally be set (cf. [Figure 3](#) on the right):

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

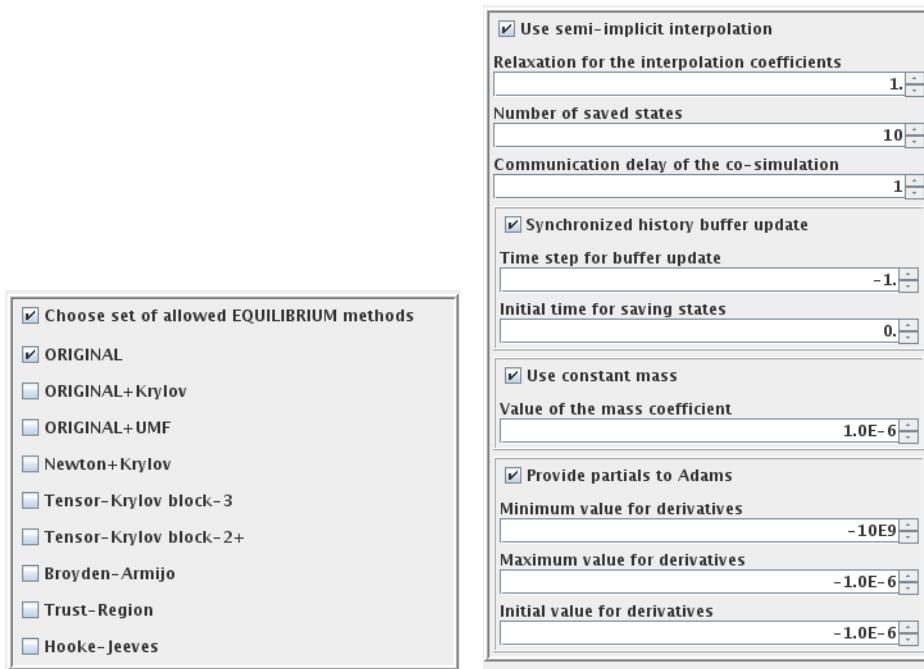


Figure 3: Additional MSC Adams options in the Go Step: for Explicit-SteadyState coupling scheme on the left and for Explicit-Transient coupling scheme on the right

Adams can update the received quantities for each iteration utilizing the available approximation. See [▷11.2.8.1 Semi-implicit Mode](#) and [▷VII-12 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](#)

Coupling Step ⌘).

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-12 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-12 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-12 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.

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 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:

```
!      adams_view_name='ActFrontLeft'
GFORCE/6
, I = 1023
, JFLOAT = 1025
, RM = 1024
,FUNCTION = 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:

```
CONTROL/FUNCTION = USER(2,3,5E-3,1E-3,1e-09,1E-3,1E-5,0.0) \
ROUTINE=/.../libadamsmpcci.so::mpcci_c_consue
```

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-Transient` is selected.

12 MSC Marc

12.1 Quick Information

Physical Domains	SolidStructure, SolidThermal
Company Name	MSC Software Corporation
Company Homepage	www.msccsoftware.com/product/marc
Support	simcompanion.msccsoftware.com
Tutorials	▷ VII-2 Vortex-Induced Vibration of a Thin-Walled Structure ◁ ▷ VII-4 Elastic Flap in a Duct ◁

12.1.1 Supported Coupling Schemes

MSC Marc currently uses exchange before solution.

12.1.2 Supported Platforms and Versions

MSC Marc is supported on the following platforms:

MPCCI_ARCH: Code platform	Supported versions						
	2013	2013.1	2014	2014.1	2014.2	2015	2016
lnx4_x64: linux64	X	X	X	X	X	X	X
windows_x64: win64	X	X	X	X	X	X	X

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 MpCCI 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 Coupling 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

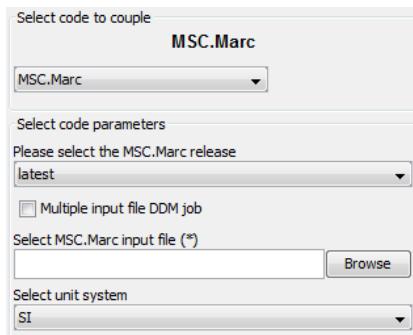


Figure 1: MSC Marc options in the Models Step

In the Models Step the following options must be chosen:

Select 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.

Select MSC.Marc input file Select the MSC Marc input file for the analysis.

Select 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 Coupling Step

The following quantities are supported:

Quantity	Dim.	Default Value	Integration Type	Coupling Dimension	Location	Send Option	Receive Option
AbsPressure	Scalar	0.0 N/m ²	flux dens.	Line, Face, Volume	Element		Direct
DeltaTime	Scalar	1.0 s	g-min	Global	global	Direct	Direct
FilmTemp	Scalar	300.0 K	field	Line, Face	Element		Direct
NPosition	Vector	0.0 m	mesh coordinate	Line, Face, Volume	Node	Direct	
OverPressure	Scalar	0.0 N/m ²	flux dens.	Line, Face, Volume	Element		Direct
RelWallForce	Vector	0.0 N	flux integral	Line, Face	Node		Direct
WallForce	Vector	0.0 N	flux integral	Line, Face	Node		Direct
WallHeatFlux	Scalar	0.0 W/m ²	flux dens.	Line, Face	Element		Direct
WallHTCoeff	Scalar	0.0 W/m ² K	field	Line, Face	Element		Direct
WallTemp	Scalar	300.0 K	field	Line, Face	Node	Direct	

 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.

12.2.4 Go Step

In the Go Step the following options can be chosen:

Coupling configuration See [V-4.8.2 Coupling Configuration Parameters](#).

Define the coupling scheme MSC Marc only supports Explicit-Transient and Explicit-SteadyState coupling schemes.

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 [12.2.5.1 Parallel Execution](#) below.

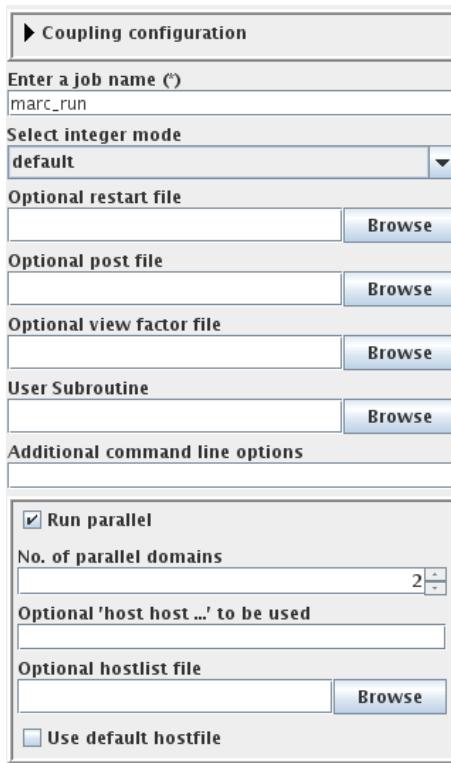


Figure 2: MSC Marc options in the Go Step

12.2.5 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**.

12.2.5.1 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 ◁](#).

12.2.6 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:

Usage:

```
mpcci MSC.Marc [-]option
```

Synopsis:

'mpcci MSC.Marc' is used to get information about MSC.Marc.

Options:

- align <ARGS> 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.
- help This screen.
- info List verbose information about all MSC.Marc releases.
- releases List all MSC.Marc releases which MpCCI can find.
- scan [-ddm] <input-file>
Run the scanner and create a scanner output file.
Specify -ddm if you have multiple input files.

The subcommands `align`, `info`, `releases` and `scan` are described in [▷ 1.1 Common MpCCI Subcommands for Simulation Codes](#) ▷.

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 -lmpifort -lrt ${OPENMP} -threads -lpthread -shared-intel"
```

to

```
SYSLIBS="-L${MPI_ROOT}/lib64 -lmpi_mt -lmpifort -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

Physical Domains	SolidStructure
Company Name	MSC Software Corporation
Company Homepage	www.mscsoftware.com/product/msc-nastran
Support	simcompanion.msccsoftware.com
Tutorials	▷ VII-2 Vortex-Induced Vibration of a Thin-Walled Structure ◁ ▷ VII-4 Elastic Flap in a Duct ◁ ▷ VII-5 Elastic Flap in Water ◁ ▷ VII-9 Pipe Nozzle ◁

13.1.1 Supported Coupling Schemes

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.

13.1.2 Supported Platforms and Versions

MSC NASTRAN is supported on the following platforms:

MpCCI_ARCH: Code platform	Supported versions						
	2013.0	2013.1	2014.0	2014.1	2016.0	2016.1	2017.0
lnx4_x64: lx8664	X	X	X	X	X	X	X
windows_x64: win8664		X	X	X	X	X	X

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 an 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').

13.2 Coupling Process

Please read also the corresponding [IV-2 Setting up a Coupled Simulation](#).

13.2.1 Model Preparation

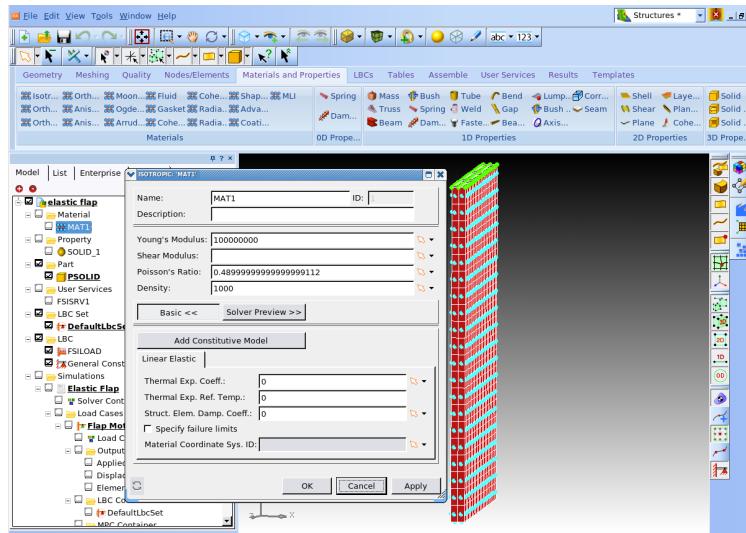


Figure 1: Model preparation under SimXpert.

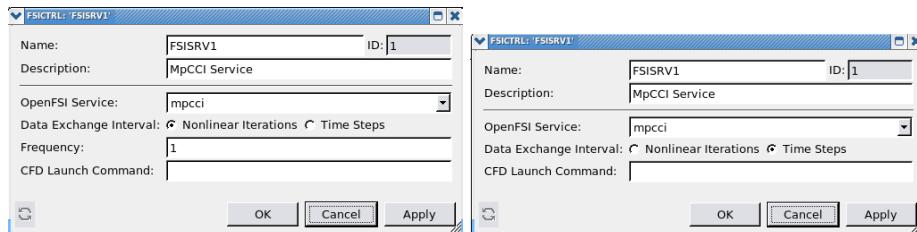


Figure 2: OpenFSI Data Exchange Interval under SimXpert: on the left implicit and on the right explicit setting

1. Create Structural model using SimXpert ([Figure 1](#)).
 - Define appropriate properties for the structure.
2. Define Service for OpenFSI.
 - Define CFD service under User Services OpenFSI.
 - Provide a dummy name. This will be updated by MpCCI before starting the computation.
 - Choose whether coupling should occur at each (Data Exchange Interval [Figure 2](#)):
 - solution time step (explicit): Time Steps.
 - within the iteration loop (implicit): Nonlinear Iterations.

You can define the Frequency of the data exchange within the time step.

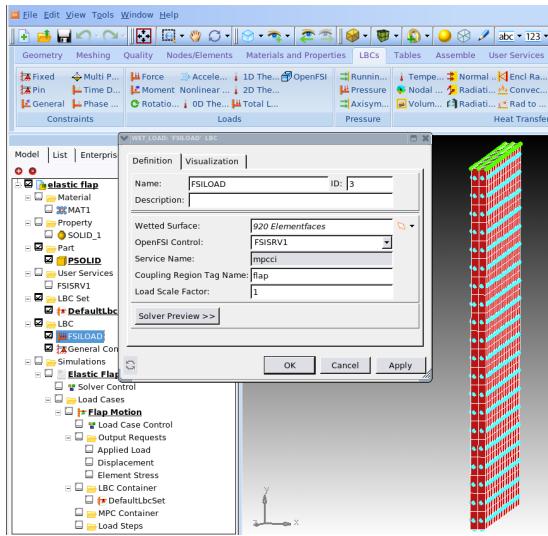


Figure 3: OpenFSI Loads Definition under SimXpert.

3. Define 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

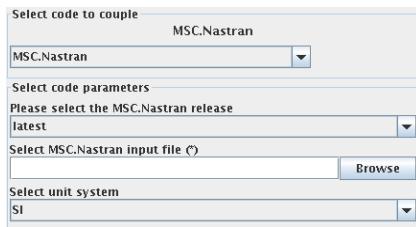


Figure 4: MSC NASTRAN options in the Models Step

In the Models Step the following options must be chosen:

Please select the MSC NASTRAN release Select the MSC NASTRAN release you want to use. latest (default) selects the latest release which is installed on your system.

Select MSC NASTRAN input file Select the MSC NASTRAN input file for the analysis (".**.bdf**" or "**.dat**" file)

Select unit system Select the unit system which was used to define the MSC NASTRAN model (see [▷ 1.2 Unit Systems](#)).

13.2.3 Coupling Step

The following quantities are supported:

Quantity	Dim.	Default Value	Integration Type	Coupling Dimension	Location	Send Option	Receive Option
DeltaTime	Scalar	1.0 s	g-min	Global	global	Direct	Direct
NPosition	Vector	0.0 m	mesh coordinate	Line, Face	Code	Direct	
RelWallForce	Vector	0.0 N	flux integral	Line, Face	Code		Direct
Velocity	Vector	0.0 m/s	field	Line, Face	Code	Direct	
WallForce	Vector	0.0 N	flux integral	Line, Face	Code		Direct

13.2.4 Go Step

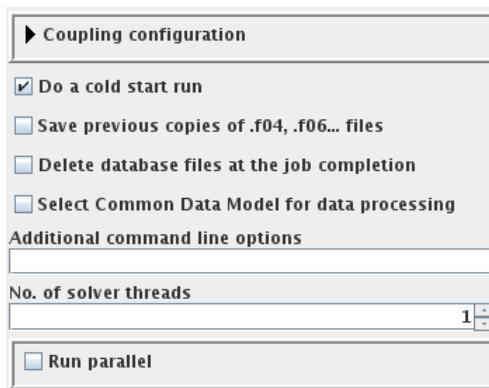


Figure 5: MSC NASTRAN options in the Go Step

In the Go Step the following options can be chosen:

Coupling configuration See [▷V-4.8.2 Coupling Configuration Parameters](#).

If the coupling scheme Implicit-Transient is selected, you should set up the iteration control in order to configure the FSICTRL command. ([▷V-3.4.4.4 Settings for Iterative Coupling in the MpCCI GUI and Supported Codes](#)). Information such as time step size, iterations number without coupling and maximal number of iterations must be provided.

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 [▷13.2.5.1 Parallel Execution](#) below.

13.2.5 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-Transient is selected, the FSICTRL from the ".bdf" file will be automatically set to IMPLICIT.

If the coupling scheme Explicit-Transient or Explicit-SteadyState is selected, the FSICTRL from the ".bdf" file will be automatically set to EXPLICIT.

13.2.5.1 Parallel Execution

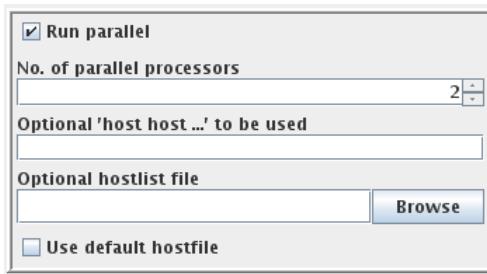


Figure 6: MSC NASTRAN options for parallel execution

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.

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 ▶ [V-3.5.2 Hostlist File](#) ◄.

13.2.5.2 Batch Execution

MSC NASTRAN is always run as a batch process, therefore no special settings are necessary for batch execution.

13.2.6 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:

Usage:

```
mpcci MSC.Nastran [-]option
```

Synopsis:

```
'mpcci MSC.Nastran' is used to get information about MSC.Nastran.
```

Options:

```
-align <ARGS>
      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.

-bdinfo [file file ...]
      Extract and display coupling information on BDF files.

-help
      This screen.

-info
      List verbose information about all MSC.Nastran releases.

-releases
      List all MSC.Nastran releases which MpCCI can find.

-scan <input-file>
      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 ◃](#).

mpcci msc.nastran -bdinfo [file file ...]

The `bdinfo` 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.

13.5 Trouble Shooting, Open Issues and Known Bugs

Feature:

Face Coupling

Version:

2010.1

Problem:

MSC NASTRAN could not load a model with QUAD8 elements for the wetted surface.

Workaround:

Replace the hexahedral element CHEXA20 to CHEXA8 which will create QUAD4 elements for the coupling.

14 OpenFOAM

14.1 Quick Information

Physical Domains	CFD, Fluid, FluidThermal
Company Name	OpenCFD Limited
Company Homepage	www.openfoam.com
Support	www.openfoam.com/support/
Tutorials	▷ VII-4 Elastic Flap in a Duct ◁ ▷ VII-5 Elastic Flap in Water ◁ ▷ VII-6 Exhaust Manifold ◁ ▷ VII-11 Y-Junction ◁

14.1.1 Supported Coupling Schemes

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. In case of surface-coupling bidirectional transfer is possible. Volumetric data can only be transferred unidirectionally – namely from OpenFOAM to another code.

14.1.2 Supported Platforms and Versions

MpCCI_ARCH: Code platform	Supported versions								
	1.7	2.0	2.1	2.2	2.3	2.4	3.0.1	1606+	1612+
lnx4_x64: linux64	X	X	X	X	X	X	X	X	X

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.

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.
 - "`<$FOAM_INST_DIR>/OpenFOAM-1.7.1`"
 - "`<$FOAM_INST_DIR>/OpenFOAM-2.2.0`"
- 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. ([▷ 14.2.4 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.

Sample from the "`0/T`":

```
internalField uniform $temperature;

boundaryField
{
    couple-flap
    {
        type          fixedValue;
        value         uniform $temperature;
    }
    ....
}
```

All further options required for coupling can be set in the MpCCI GUI.

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

```

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;
    }
)
// ****

```

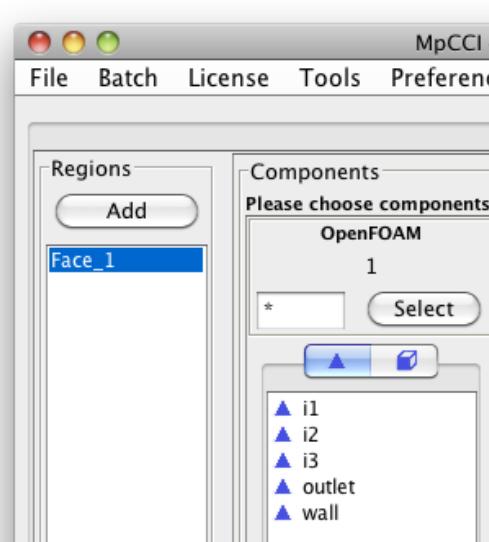


Table 1: List of surfaces in the Coupling Step of the MpCCI GUI (right) and in OpenFOAM (left) defined in "*<case directory>/constant/polymesh/boundary*".

the appropriate init- and transfer-functions.

```

libs ("libfoammpcci.so");

functions
(
    MpCCI_functionObject_adapter_for_OpenFOAM
    {
        type mpcci; // Type of functionObject
    }
);

```

See ▷ 14.2.4 Go Step ◁ for a complete description of the Go Step options.

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. This is necessary even if the bare OpenFOAM-case itself has a static mesh. 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. 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

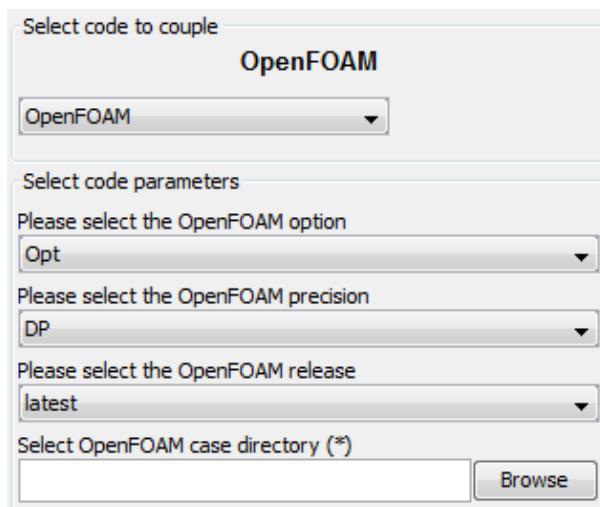


Figure 1: OpenFOAM options in the Models Step

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.

case directory Select the case directory of your OpenFOAM model.

14.2.3 Coupling Step

OpenFOAM supports the following quantities for coupling:

Quantity	Dim.	Default Value	Integration Type	Coupling Dimension	Location	Send Option	Receive Option
AbsPressure	Scalar	0.0 N/m ²	flux dens.	Line, Face, Volume	Code	Dir	
DeltaTime	Scalar	1.0 s	g-min	Global	global	Dir	Dir
FilmTemp	Scalar	300.0 K	field	Line, Face	Code	Dir	
HeatRate	Scalar	0.0 W	field	Face	Code	Dir	
MassFlowRate	Scalar	0.0 kg/s	flux integral	Face	Code	Dir	Dir
MassFluxRate	Scalar	0.0 kg/m ² s	flux dens.	Face	Code	Dir	Dir
NPosition	Vector	0.0 m	mesh coordinate	Line, Face, Volume	Code		Dir
OverPressure	Scalar	0.0 N/m ²	flux dens.	Line, Face, Volume	Code	Dir	
RelWallForce	Vector	0.0 N	flux integral	Line, Face	Code	Dir	
Temperature	Scalar	300.0 K	field	Line, Face, Volume	Code	Dir	Dir
TotalPressure	Scalar	0.0 N/m ²	flux dens.	Line, Face, Volume	Code	Dir	Dir
Velocity	Vector	0.0 m/s	field	Line, Face, Volume	Code	Dir	Dir
WallForce	Vector	0.0 N	flux integral	Line, Face	Code	Dir	
WallHeatFlux	Scalar	0.0 W/m ²	flux dens.	Line, Face	Code	Dir	
WallHTCoeff	Scalar	0.0 W/m ² K	field	Line, Face	Code	Dir	
WallTemp	Scalar	300.0 K	field	Line, Face	Code	Dir	Dir

14.2.4 Go Step

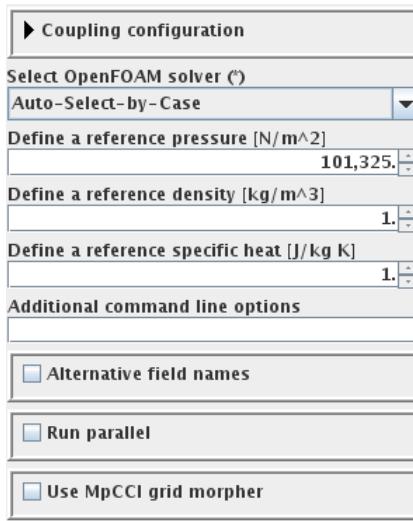


Figure 2: OpenFOAM options in the Go Step

OpenFOAM offers a number of options in the Go panel, see [Figure 2](#).

Coupling configuration See [▷V-4.8.2 Coupling Configuration Parameters◀](#).

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/m²] 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/m³] 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 $\frac{p}{\rho}$ 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 $\frac{p}{\rho}$ 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 define alternative names here. 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 [▷14.2.5.1 Parallel Execution◀](#).

Use MpCCI Grid Morpher Activate the MpCCI Grid Morpher and access to the MpCCI Grid Morpher configuration, see [▷14.3 Grid Morphing◀](#).

14.2.5 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.

14.2.5.1 Parallel Execution

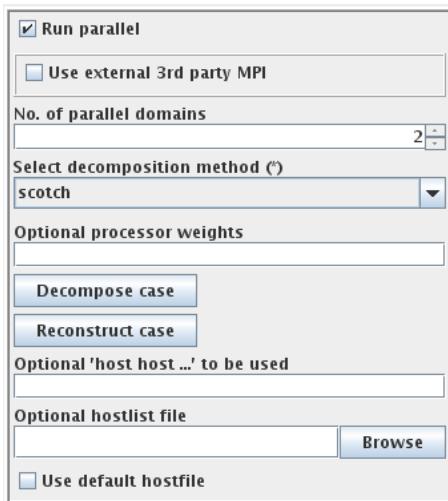


Figure 3: OpenFOAM options for a parallel run

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 [▷ 14.2.5.2 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` or `metis`.

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` or `metis` 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 ◁](#).

14.2.5.2 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 foamExecMpcc 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 "`foamExecMpcc`" to start the simulation. This script can be provided by the user or generated by MpCCI at runtime.

Generate foamExecMpcc MpCCI generates the script "`foamExecMpcc`" 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 "`envMpcc`" under linux, "`envMpcc.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.

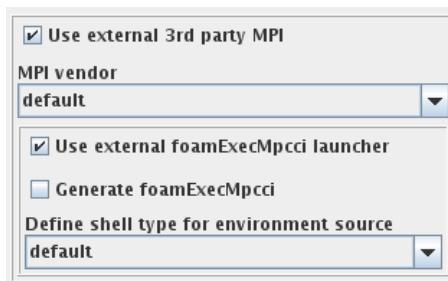


Figure 4: OpenFOAM options for third party MPI

14.2.5.3 Batch Execution

OpenFOAM is always run as a batch process, therefore no special settings are necessary for batch execution.

14.2.6 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-7 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](#).

<input checked="" type="checkbox"/> Use MpCCI grid morpher	
Select morpher method	
Morpher Standard	
Optional morpher hostname	
Morpher port no.	47,020
No. of parallel morpher threads.	1
Morpher options file	<input type="button" value="Browse"/>
Please select the output level	
default	
<input checked="" type="checkbox"/> Check edges	
<input checked="" type="checkbox"/> Check faces	
<input checked="" type="checkbox"/> Check cells	
List of cell zone names (strings or ALL)	
ALL	
► Standard Parameters	
Max. no. of iterations	30
Convergence tolerance	1.0E-3
Smoothing steps	0
Fixed nodes on fixed boundaries	0
Fixed nodes on deformed boundaries	1
Optional list of floating boundary regions	
Optional list of fixed boundary regions	

▼ Standard Parameters

Min. relative edge length change	0.1
Max. relative edge length change	10.
Min. change of face area	0.01
Max. change of face area	100.
Max. face aspect ratio	10.
Min. angle allowed in faces and cells	10.
Morphing relaxation factor	1

Figure 5: OpenFOAM Options for MpCCI Grid Morpher

```
/*-----*\
| ====== |
| \ \ / F ield | OpenFOAM: The Open Source CFD Toolbox |
| \ \ / O peration | Version: 1.7.1 |
| \ \ / A nd | Web: www.OpenFOAM.com |
| \ \ \ M anipulation |
\*-----*/
Build : 1.7.1-03e7e056c215
Exec  : /home/test/OpenFOAM/test-1.7.1/applications/bin/linux64GccDP0pt/rhoPimple-
DyMFoamMpCCI
Date  : Mar 05 2012
Time  : 19:45:10
Host   : zeus
PID    : 21311
Case   : /home/test/coupling/ElasticFlap/OpenFoam/1.7
nProcs: 1
SigFpe: Enabling floating point exception trapping (FOAM_SIGFPE).

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * // 
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.

-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 -decompose [OPTIONS]

The `decompose` subcommand decomposes the serial OpenFOAM case into a parallel case. Use `-force` option to overwrite an existing decomposition.

Usage:

```
mpcci OpenFOAM decompose [-]options
```

Synopsis:

'`mpcci OpenFOAM decompose`' is used to decompose an OpenFOAM case.

Options:

<code>-case</code>	<code><casedir></code>	Defines the case direcory.
<code>-force</code>		Force a new decomposition.
<code>-help</code>		This screen.
<code>-method</code>	<code><simple scotch></code>	Defines the decomposition method.
<code>-nprocs</code>	<code><nprocesses></code>	Defines the no. of processes.
<code>-pweights</code>	<code><weights></code>	Defines the processor weights.
<code>-release</code>	<code><release></code>	Define the release to be used.
<code>-splits</code>	<code><x y z></code>	Defines the split in each direction.

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.

mpcci openfoam -envmpcci [RELEASE]

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.

mpcci openfoam -gmdmake [CASEDIR]

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.

mpcci openfoam -magic [CASEDIR]

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.

mpCCI openfoam -reconstruct [CASEDIR] [RELEASE]

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

- The MpCCI code adapter for OpenFOAM uses so called function objects to manage the data transfer. 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:
`"./<$WM_PROJECT_VERSION>/<$WM_OPTIONS>/libfoammppci.so"`.

There's also an auxiliary library used in case of dynamic mesh motion; this is
`"./<$WM_PROJECT_VERSION>/<$WM_OPTIONS>/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 and later.

- 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:
`"./<$WM_PROJECT_VERSION>/<$WM_OPTIONS>/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 RadTherm/TAITherm

15.1 Quick Information

Physical Domains	Radiation
Company Name	ThermoAnalytics
Company Homepage	www.thermoanalytics.com
Support	www.thermoanalytics.com/support
Tutorials	▷ VII-10 Cube in a Duct Heater ◁

15.1.1 Supported Coupling Schemes

RadTherm may only exchange data at the time step or iteration start hook, resp. for transient, steady state simulation.

RadTherm supports the following Coupling Type [▷ V-4.5 Coupling Step ◁](#):

- Steady state radiative heat transfer (Explicit-SteadyState)
- Transient radiative heat transfer (Explicit-Transient)

15.1.2 Supported Platforms and Versions

The following versions of RadTherm are supported by MpCCI:

MpCCI_ARCH: Code platform	Supported versions										
	11.0.0	11.1.0	11.1.1	11.2.0	11.3.0	11.3.2	12.0.0	12.0.3	12.1.0	12.1.1	12.2.0
lnx4_x64: x86_64	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

15.1.3 References

RadTherm Documentation is part of the RadTherm distribution.

15.1.4 Adapter Description

The code adapter for RadTherm is based on a shared library "libradthermmppci.so|dll" which is loaded by RadTherm and it includes the necessary interface functions.

15.1.5 Prerequisites for a Coupled Simulation

To run a coupled simulation you need the following:

- Ordinary RadTherm installation.

15.2 Coupling Process

Please read also [▷ IV-2 Setting up a Coupled Simulation ◁](#).

15.2.1 Model Preparation

Models can be prepared as usually.

The view factors file may be already computed by RadTherm before starting the coupled simulation because it may take some time until the view factors file is created.

You can execute the following RadTherm command to run the view factors computation separately:

```
radtherm -runviewfactors <file.tdf>
```

If the view factors have not been pre-computed before the coupled simulation start, RadTherm will perform this calculation as first step before solving the thermal problem.

The coupling components must be defined as separate parts.

15.2.2 Models Step

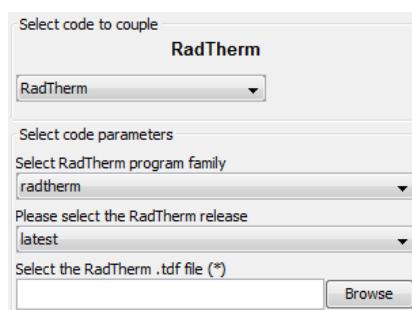


Figure 1: RadTherm options in the Models Step

In the Models Step, the following options must be chosen:

RadTherm program family Select the RadTherm program family from

- muses,
- musespro,
- radtherm, (default)
- radthermir,
- radthermr.

RadTherm release Select the release of RadTherm you want to use. latest (default) will select the latest version which is installed on your system.

.tdf file Select the ".tdf" file of your RadTherm model.

The MpCCI RadTherm scanner reads the ".tdf" file and extracts all the parts by checking the following rules:

- the RadTherm 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.
- RadTherm 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.

15.2.3 Coupling Step

The RadTherm adapter only stores quantities directly (“Direct”).

RadTherm supports the following quantities for coupling:

Quantity	Dim.	Default Value	Integration Type	Coupling Dimension	Location	Send Option	Receive Option
DeltaTime	Scalar	1.0 s	g-min	Global	global	Direct	
FilmTemp	Scalar	300.0 K	field	Face	Code		Direct
IterationNo	Scalar	0	g-max	Global	global	Direct	
PhysicalTime	Scalar	0.0 s	g-min	Global	global	Direct	
Residual	Scalar	0.0	g-max	Global	global	Direct	
TimeStepNo	Scalar	0	g-max	Global	global	Direct	
WallHeatFlux	Scalar	0.0 W/m ²	flux dens.	Face	Code		Direct
WallHTCoeff	Scalar	0.0 W/m ² K	field	Face	Code		Direct
WallTemp	Scalar	300.0 K	field	Face	Code	Direct	

15.2.4 Go Step

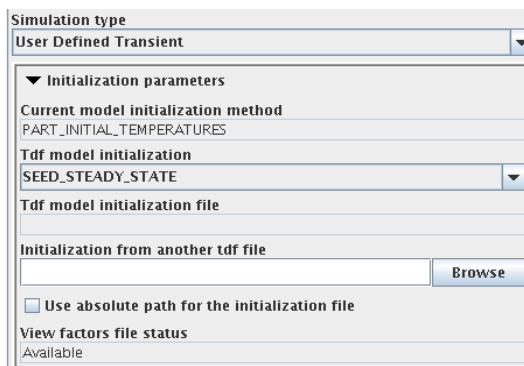


Figure 2: RadTherm options for Simulation type and Initialization parameters in the Go Step.

In the Go Step the following options can be chosen:

Simulation type Defines the type of the simulation (cf. [Figure 2](#)) and presets the Coupling Scheme and Initial quantities transfer options with appropriate values. Choosable simulation types are:

Stabilized Case Leads to an Explicit-SteadyState coupling scheme and sets the Initial quantities transfer to exchange.

Thermal Transient Case Leads to an Explicit-Transient coupling scheme and sets the Initial quantities transfer to exchange.

Full Transient Case Leads to an Explicit-Transient coupling scheme and sets the Initial quantities

transfer to send.

User Defined Steady State Leads to an Explicit-SteadyState coupling scheme. The Initial quantities transfer is selectable as described in [▷ V-4.8.2 Coupling Configuration Parameters ◁](#).

User Defined Transient Leads to an Explicit-Transient coupling scheme. The Initial quantities transfer is selectable as described in [▷ V-4.8.2 Coupling Configuration Parameters ◁](#).

Initialization parameters Groups the options which can be set for initialization (cf. [Figure 2](#)).

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 RadTherm 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.

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.

Coupling configuration Groups the coupling configuration options which differ slightly from those described in [▷ V-4.8.2 Coupling Configuration Parameters ◁](#) to the effect that for RadTherm some additional options exist.

Define the coupling scheme RadTherm only supports Explicit-Transient and Explicit-SteadyState coupling schemes. It will be set automatically depending on the chosen Simulation type.

Initial quantities transfer Preset by the chosen Simulation type.

Convergence management Groups the subcycling options which are described in detail in [▷ V-3.4.6.1 Subcycling Setting in the MpCCI GUI ◁](#).

Convergence criteria The convergence criteria define the stopping criteria to apply.

The criteria control the number of iterations within a subcycling step in case of an Explicit-SteadyState scheme or the number of iterations within a time step for an Explicit-Transient scheme. Two options are available:

- Fixed number of loops
- Tolerance slope

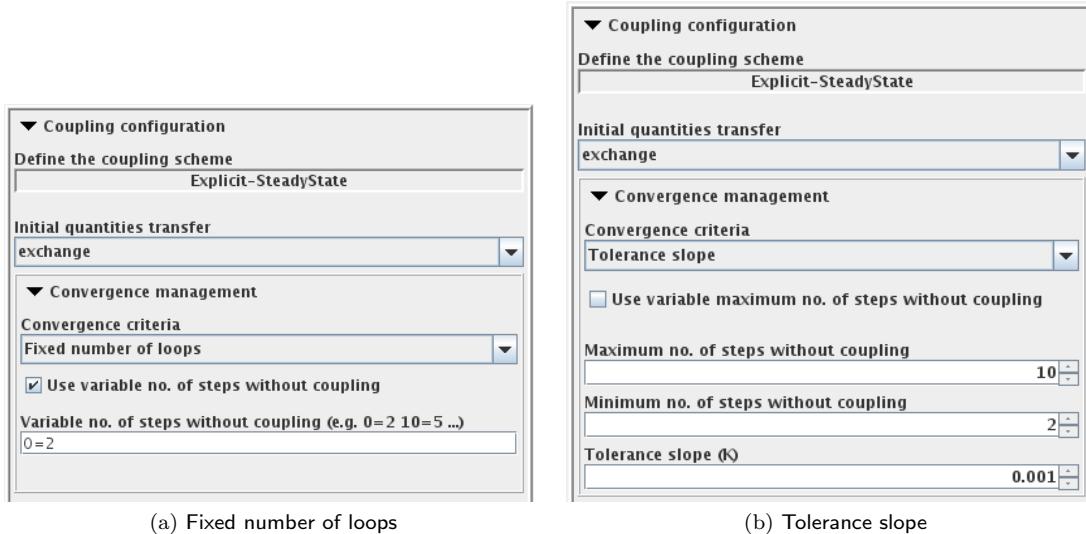


Figure 3: RadTherm convergence management options for Explicit-SteadyState coupling scheme.

For the Explicit-SteadyState coupling scheme, different options will be proposed to be set up according to the selected Convergence criteria. The options will define the subcycling behaviour (cf. Figure 3).

- When the Fixed number of loops is selected, following options are available (cf. Figure 3a):

Use variable no. of steps without coupling Enables the definition of a variable subcycling function depending on the coupling step number.

Variable no. of steps without coupling (e.g. 0=2 10=5 ...) Specifies a list of couplingStepNo=nofSteps pairs each defining the coupling step number (couplingStepNo) where the number of iterations without coupling will be set to the given value nofSteps.

Available if Use variable no. of steps without coupling is selected.

No. of steps without coupling Defines the number of iterations without coupling.

Available if Use variable no. of steps without coupling is not selected.

- When the Tolerance slope is selected, the options listed below are available (cf. Figure 3b). They allow you to use the tolerance slope as additional criterion to the maximum number of steps to define the subcycling stopping criteria. The Tolerance slope is evaluated at each solver iteration step in the iteration interval defined by Minimum no. of steps (i_{min}) and Maximum no. of steps (i_{max}) (cf. Figure 4).

Use variable maximum no. of steps without coupling Enables the definition of a variable subcycling function depending on the coupling step number.

Variable maximum no. of steps without coupling (e.g. 0=2 10=5 ...) Specifies a list of couplingStepNo=nofSteps pairs each defining the coupling step number (couplingStepNo) from where the maximum number of steps without coupling will be set to the given value nofSteps.

 Available if Use variable maximum no. of steps without coupling is selected.

Maximum no. of steps without coupling Defines the maximum number of iterations without coupling.

 Available if Use variable maximum no. of steps without coupling is not selected.

Minimum no. of steps without coupling Defines the minimum number of iterations without coupling to perform at least before activating the selected convergence criterion.

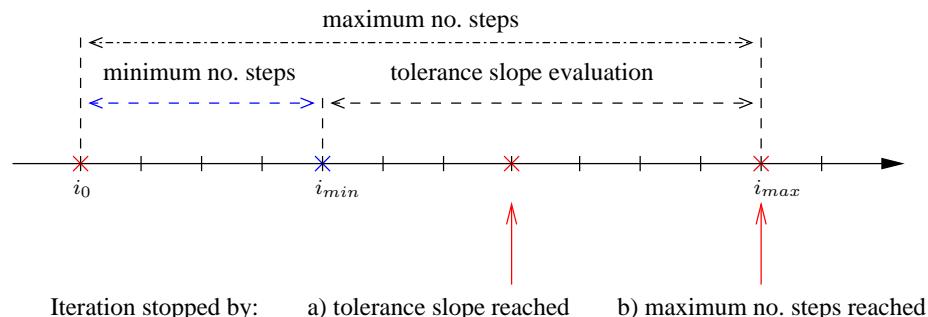


Figure 4: Subcycling behaviour with tolerance slope.

For the Explicit-Transient coupling scheme, different options will be proposed to be set up according to the selected Convergence criteria. The options will define the subcycling behaviour (cf. Figure 5).

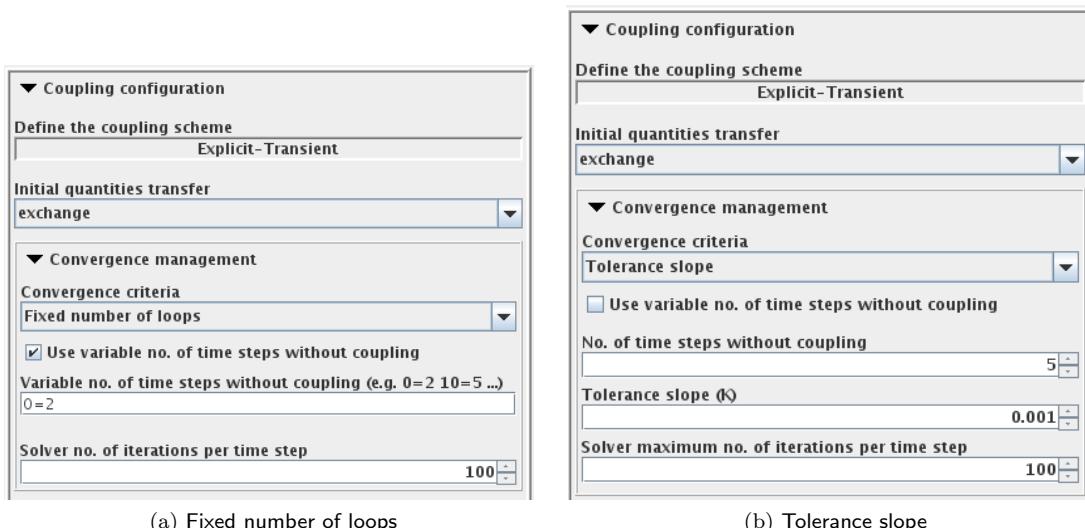


Figure 5: RadTherm convergence management options for Explicit-Transient coupling scheme.

- When the Fixed number of loops is selected, each time step is forced to perform the same amount of iterations defined in Solver no. of iterations per time step. The following options define the subcycling behaviour (cf. Figure 5a):

Use variable no. of time steps without coupling Enables the definition of a variable subcycling function depending on the coupling step number.

Variable no. of time steps without coupling (e.g. 0=2 10=5 ...) Specifies a list of couplingStepNo=nofSteps pairs each defining the coupling step number (couplingStepNo) where the number of time steps without coupling will be set to the given value nofSteps.

(i) Available if Use variable no. of time steps without coupling is selected.

No. of time steps without coupling Defines the number of time steps without coupling.

(i) Available if Use variable no. of time steps without coupling is not selected.

Solver no. of iterations per time step Defines the number of iterations per time step.

- When the Tolerance slope is selected, each time step may be stopped by the maximum number of iterations defined in Solver maximum no. of iterations per time step or the tolerance slope defined in Tolerance slope (K). The following options are available to configure the subcycling (cf. Figure 5b):

Use variable no. of time steps without coupling Enables the definition of a variable subcycling function depending on the coupling step number.

Variable no. of time steps without coupling (e.g. 0=2 10=5 ...) Specifies a list of couplingStepNo=nofSteps pairs each defining the coupling step number (couplingStepNo) from where the number of time steps without coupling will be set to the given value nofSteps.

(i) Available if Use variable no. of time steps without coupling is selected.

No. of time steps without coupling Defines the number of time steps without coupling.

(i) Available if Use variable no. of time steps without coupling is not selected.

Tolerance slope (K) 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.

Solver maximum no. of iterations per time step Defines the maximum number of iterations per time step.

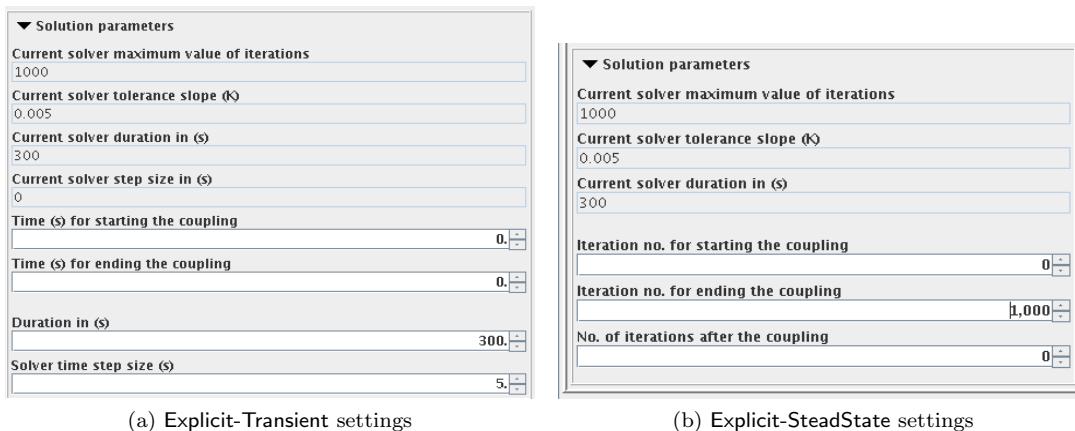


Figure 6: RadTherm coupling configuration options Solution parameters.

Solution parameters Groups the options for starting and ending the coupling (cf. Figure 6). RadTherm provides some additional options to those already described as duration control in ▷ V-3.4.6.1 Subcycling Setting in the MpCCI GUI <. The additional options for RadTherm are:

Current solver maximum value of iterations Displays the current solver setting used by the current model file for the maximum value of iterations.

Current solver tolerance slope Displays the current solver setting used by the current model

file for the tolerance slope.

Current solver duration in (s) Displays the current solver setting used by the current model file for the duration.

Current solver step size in (s) Displays the current solver setting used by the current model file for the step size.

Solver time step size (s) Define the solver time step size to use.

- (!) To keep the setting from the current model file, leave this field value at zero.
This allows you to use a curve definition for the time step size.

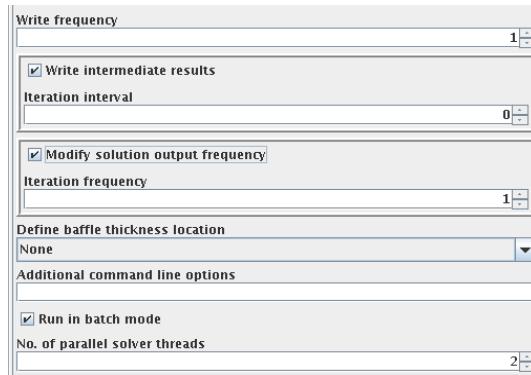


Figure 7: Output and start settings.

Write frequency Defines the frequency (in time step) to store the results (cf. [Figure 7](#)). For steady state simulation the results are saved at the end of the calculation.

- (i) 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 [Figure 30 \(part V\)](#) is activated, this option Define baffle thickness location will be displayed. The user should provide the thickness information about how the RadTherm 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 RadTherm can be given here, they will directly be used when RadTherm is started.

Run in batch mode If you want to run RadTherm in batch mode, select this option.

No. of parallel solver threads Specifies the number of threads for a parallel run. See [▷ 15.2.6.2 Parallel Execution, Multi-Threaded ◄](#) below.

15.2.5 Checking the Computation

RadTherm provides a configuration checker as described in [▷ V-4.8.3 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.
 - 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.

15.2.6 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 [▷ 15.2.6.1 Hook Functions ◄](#)).
- does a local copy of the MpCCI RadTherm adapter library before the code starts. It facilitates the port of the RadTherm "tdf" file.

The RadTherm computation is started, when using the GUI mode (non-batch mode), by clicking the **Run** button in section **Analyse** of the RadTherm GUI, otherwise RadTherm runs in batch mode.

When running a steady state simulation, RadTherm 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 (cf. Figure 6b). 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 (cf. Figure 3b).

If RadTherm gets disconnected from the coupling server, the RadTherm solution will continue until the defined convergence criterion is reached.

When running a transient simulation, RadTherm saves the data at specified Write frequency value.

If RadTherm gets disconnected from the coupling server, the RadTherm solution will abort the current time step where the incident occurred and the results should be saved by RadTherm.

Using the MpCCI GUI **Stop** button executes a stopper script which sends a stop signal to RadTherm. Receiving this signal, RadTherm is able to terminate and save the tdf results.

15.2.6.1 Hook Functions

By starting the RadTherm 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 RadTherm hook functions. The MpCCI RadTherm adapter decides which hook function to use according to the simulation type (steady state or transient).
- RadTherm model is automatically checked: the part convection type will be adjusted to RadTherm HandTfluid property for the co-simulation with the following default values:
 - $0 \frac{W}{m^2 * K}$ for heat transfer coefficient.
 - 273 K for the film temperature.

Running RadTherm with GUI allows you to see the changes applied to the model.

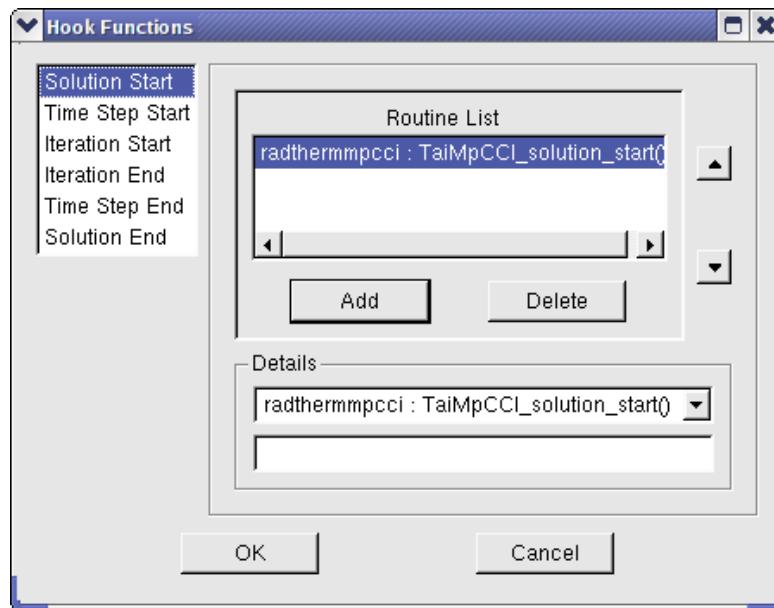


Figure 8: RadTherm Hook Functions Setup.

You can check the hooks setting by selecting the **Analyse → Params** section in the RadTherm GUI in order to access the hook functions setup [Figure 8](#). By clicking on the button **HookFunctions...** the setup window will appear. You have to set up these functions for the following available RadTherm hooks:

Solution Start The routine `radthermmppci::TaiMpCCI_solution_start()` has been automatically added.
This is the function to initialize RadTherm with MpCCI.

Time Step Start The routine `radthermmppci::TaiMpCCI_timestep_start()` has been automatically added.
This is the function to exchange data before a new computation for transient simulation.

Iteration Start The routine `radthermmppci::TaiMpCCI_iteration_start()` has been automatically added.
This is the function to exchange data before a new computation for steady state simulation.

IterationEnd The routine `radthermmppci::TaiMpCCI_iteration_end()` has been automatically added.
This is the function to check the solution for the steady state simulation.

Solution End The routine `radthermmppci::TaiMpCCI_solution_end()` has been automatically added.
This is the function to terminate the coupled simulation.

15.2.6.2 Parallel Execution, Multi-Threaded

There are two ways to launch RadTherm in parallel:

In batch mode By activating the batch mode you may specify the number of parallel solver threads.

In RadTherm GUI Before starting the RadTherm computation you need to specify the number of processors to use in the **Edit→Preferences→Application** [Figure 9](#).

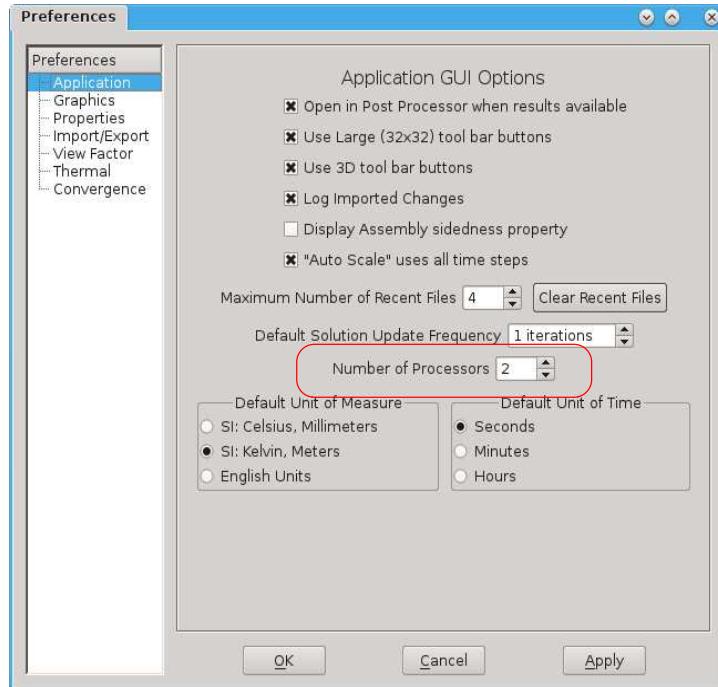


Figure 9: Number of processors setting in RadTherm GUI v11.0.

15.2.7 Post-Processing

After a coupled simulation has finished the results computed on the RadTherm side may be visualized by using the RadTherm post-processing tool.

15.3 Code-Specific MpCCI Commands

The MpCCI subcommands available for RadTherm are:

Usage:

```
mpcci RadTherm [-]option
```

Synopsis:

'mpcci RadTherm' is used to get information about RadTherm.

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.

-info
List verbose information about all RadTherm releases.

-releases
List all RadTherm 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 radtherm -clean [-r release]<tdf file>
```

The `clean` subcommand cleans up the MpCCI hooks from the given tdf file. If no release is specified, the latest one will be taken.

15.4 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/RadTherm/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 RadTherm adapter (e.g. "libradthermmppci.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 RadTherm TDFIO library to process the model file. The files are located within the subdirectory "<MpCCI_home>/codes/RadTherm/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 RadTherm utility tool "mpcci_radthermutil.exe".

15.4.1 Quantity handling

Following paragraph describes the default rules applied by the code adapter for the following quantities prescribed to RadTherm:

- 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 inversed.

16 SIMPACK

16.1 Quick Information

Physical Domains	Multibody dynamics
Company Name	SIMPACK AG
Company Homepage	www.simpack.com
Support	www.simpack.com/mbs-software-training-support.html
Tutorials	▷ VII-12 Spring Mass System ◁

16.1.1 Supported Coupling Schemes

Following coupling schemes are supported by SIMPACK for the dynamic analysis:

SIMPACK simulation mode	Transfer
Explicit	at the end of the time step

Unidirectional and bidirectional transfer is possible for SIMPACK.

16.1.2 Supported Platforms and Versions

SIMPACK is supported on the following platforms:

MPCCI_ARCH: Code platform	Supported versions	
	9.7	9.8.1
lnx4_x64: linux64	X	X
windows_x64: win64	X	X

For details on the platforms supported by SIMPACK see also the Platform Support page at www.simpack.com/platforms.html.

16.1.3 References

SIMPACK Documentation Your SIMPACK installation contains the SIMPACK Assistent.

16.1.4 Adapter Description

The code adapter for SIMPACK is developed and distributed by the Fraunhofer SCAI.

16.1.5 Prerequisites for a Coupled Simulation

To run a coupled simulation you need the following:

- Ordinary SIMPACK installation.
- Enough license tokens.

16.2 Coupling Process

Please read also [IV-2 Setting up a Coupled Simulation](#).

16.2.1 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 Coupling 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 Coupling 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.

16.2.2 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-4.7.2 Job](#) and [V-3.4.7 Non-matching Time Steps](#)).

16.2.3 Models Step

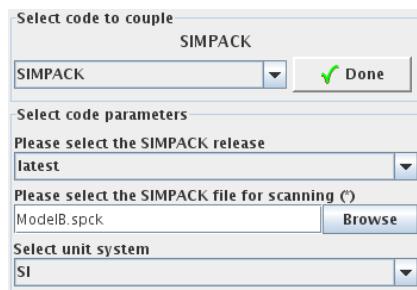


Figure 1: SIMPACK options in the Models Step

In the Models Step, the following options must be chosen:

Please select the 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.

Please select the SIMPACK file for scanning Select the SIMPACK input file "`*.spck`" to provide the simpack model definition to MpCCI.

Select 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.

16.2.4 Coupling Step

SIMPACK supports the following quantities for coupling:

Quantity	Dim.	Default Value	Integration Type	Coupling Dimension	Location	Send Option	Receive Option
AngularCoordinate	Quaternion	0.0	mesh coordinate	Point	Node	Direct	
AngularVelocity	Vector	0.0 rad/s	field	Point	Node	Direct	
DeltaTime	Scalar	1.0 s	g-min	Global	global		Direct
Force	Vector	0.0 N	flux integral	Point	Node	Direct	Direct
PointPosition	Vector	0.0 m	mesh coordinate	Point	Node	Direct	
RealFlag	Scalar	0.0	g-max	Global	global		Direct
Torque	Vector	0.0 Nm	flux integral	Point	Node	Direct	Direct
Velocity	Vector	0.0 m/s	field	Point	Node	Direct	

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 [16.2.5 Go Step](#)). If the user requires time step size to be sent by SIMPACK, the operation is ignored.

16.2.4.1 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 [16.2.5 Go Step](#)).

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 [16.2.5 Go Step](#)).

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.

16.2.5 Go Step

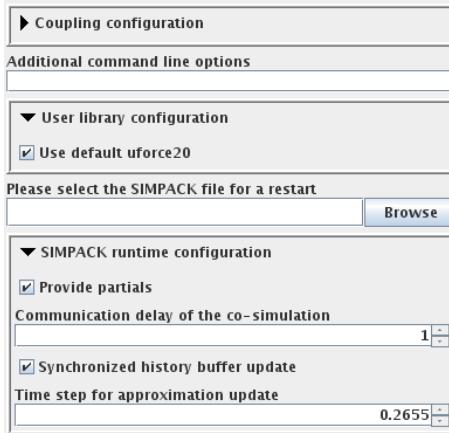


Figure 2: SIMPACK options in the Go Step

In the Go Step, the following options can be selected:

Coupling configuration See [V-4.8.2 Coupling Configuration Parameters](#).

Define the coupling scheme SIMPACK only supports Explicit-Transient and Explicit-SteadyState coupling schemes.

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.

SIMPACK runtime configuration Expand this section to access the additional SIMPACK configuration. In this section the user can set the options which are specific for SIMPACK.

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 [16.2.4.1 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 [▷ 16.2.4 Coupling 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.

16.2.6 Running the Computation

With the **Start** 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 **Stop** button is pressed SIMPACK receives a **STOP** 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.

16.2.6.1 Batch Execution

SIMPACK always runs as a batch process, therefore no special settings are necessary for batch execution.

16.2.7 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.

16.3 Code-Specific MpCCI Commands

The MpCCI subcommands available for SIMPACK are:

Usage:

```
mpcci SIMPACK [-]option
```

Synopsis:

```
'mpcci SIMPACK' is used to get information about SIMPACK.
```

Options:

-help

```
This screen.
```

-info

```
List verbose information about all SIMPACK releases.
```

-releases

```
List all SIMPACK releases which MpCCI can find.
```

-scan [-r release] <spck file>

```
Run the scanner and create a scanner output file.
```

The subcommands `[info]`, `[releases]` and `[scan]` are described in [▷ 1.1 Common MpCCI Subcommands for Simulation Codes ◁](#).

17 STAR-CCM+

17.1 Quick Information

Physical Domains	CFD, Fluid, FluidThermal
Company Name	CD-adapco
Company Homepage	www.cd-adapco.com/products/star_ccm_plus
Support	www.cd-adapco.com/support
Tutorials	▷ VII-2 Vortex-Induced Vibration of a Thin-Walled Structure ◁ ▷ VII-4 Elastic Flap in a Duct ◁ ▷ VII-6 Exhaust Manifold ◁ ▷ VII-9 Pipe Nozzle ◁ ▷ VII-10 Cube in a Duct Heater ◁ ▷ VII-11 Y-Junction ◁

17.1.1 Supported Coupling Schemes

STAR-CCM+ is run via a Java macro script.

The initial transfer settings determine the coupling algorithm [▷ V-3.4 Coupling Algorithms ◁](#).

Unidirectional and bidirectional transfer is possible.

Depending on the STAR-CCM+ analysis type the following scheme should be applied:

- Steady-State Analysis: STAR-CCM+ exchanges before or after solution ([Explicit-SteadyState](#)).
- Transient: STAR-CCM+ exchanges before or after solution ([Explicit-Transient](#)).

The data exchange action depends on how the Java macro file is constructed.

(!) For FSI simulation it is recommended to place the data exchange before the solver begins the time step. The nodal coordinates need to be updated before the morpher runs.

(!) Volume coupling is currently not supported.

17.1.2 Supported Platforms and Versions

MpCCI_ARCH: Code platform (windows lnx4)_x64: lib	Supported versions								
	9.02	9.04	9.06	10.02	10.04	10.06	11.02	11.04	11.06
	X	X	X	X	X	X	X	X	X

17.1.3 References

STAR-CCM+ Documentation is part of the STAR-CCM+ distribution (e.g. STAR-CCM+ User Guide).

17.1.4 Adapter Description

For STAR-CCM+ the MpCCI plugin library "libstarccmmpcci.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 ([▷ 17.5 Code Adapter Reference ◁](#)).

17.1.5 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.

17.2 Coupling Process

Please read also [IV-2 Setting up a Coupled Simulation](#).

17.2.1 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 ([17.4 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.

 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.

17.2.2 Models Step

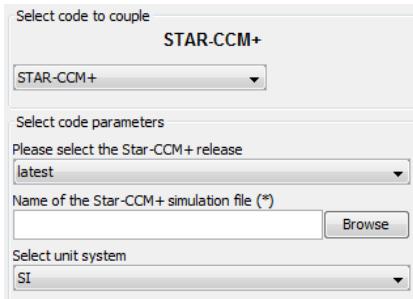


Figure 1: STAR-CCM+ options in the Models Step

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.

Name of the Star-CCM+ simulation file Select the model file (".**.sim**") of your STAR-CCM+ model.

Select unit system Select the unit system which was used in STAR-CCM+ (see [▷ 1.2 Unit Systems](#)).

17.2.3 Coupling Step

STAR-CCM+ supports the following quantities for coupling:

Quantity	Dim.	Default Value	Integration Type	Coupling Dimension	Location	Send Option	Receive Option
AbsPressure	Scalar	0.0 N/m ²	flux dens.	Face	Code	Direct	
AngularVelocity	Vector	0.0 rad/s	field	Point	Code		Direct
DeltaTime	Scalar	1.0 s	g-min	Global	global	Direct	Direct
FilmTemp	Scalar	300.0 K	field	Face	Code	Direct	
Force	Vector	0.0 N	flux integral	Face	Code	Direct	
HeatRate	Scalar	0.0 W	field	Face	Code	Direct	
IterationNo	Scalar	0	g-max	Global	global	Direct	Direct
MassFlowRate	Scalar	0.0 kg/s	flux integral	Face	Code	Direct	Direct
MassFluxRate	Scalar	0.0 kg/m ² s	flux dens.	Face	Code	Direct	Direct
NPosition	Vector	0.0 m	mesh coordinate	Face	Code		Direct
OverPressure	Scalar	0.0 N/m ²	flux dens.	Face	Code	Direct	
PhysicalTime	Scalar	0.0 s	g-min	Global	global	Direct	Direct
RefPressure	Scalar	1.12e5 N/m ²	g-max	Global	global	Direct	Direct
RelWallForce	Vector	0.0 N	flux integral	Face	Code	Direct	Direct
StaticPressure	Scalar	0.0 N/m ²	flux dens.	Face	Code	Direct	Direct
Temperature	Scalar	300.0 K	field	Face	Code	Direct	Direct
TimeStepNo	Scalar	0	g-max	Global	global	Direct	Direct
Torque	Vector	0.0 N m	flux integral	Face	Code	Direct	
TotalPressure	Scalar	0.0 N/m ²	flux dens.	Face	Code	Direct	Direct
Velocity	Vector	0.0 m/s	field	Point, Face	Code	Direct	Direct
WallForce	Vector	0.0 N	flux integral	Face	Code	Direct	Direct

Quantity	Dim.	Default Value	Integration Type	Coupling Dimension	Location	Send Option	Receive Option
WallHeatFlux	Scalar	0.0 W/m ²	flux dens.	Face	Code	Direct	Direct
WallHTCoeff	Scalar	0.0 W/m ² K	field	Face	Code	Direct	Direct
WallTemp	Scalar	300.0 K	field	Face	Code	Direct	Direct

17.2.4 Go Step

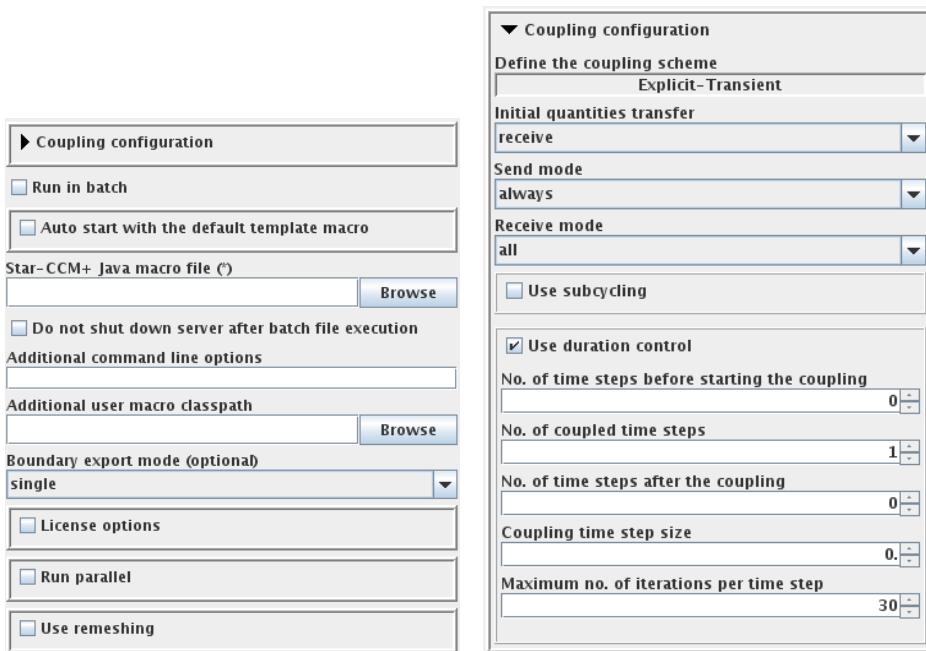


Figure 2: STAR-CCM+ options in the Go Step: in general on the left and special duration control options for Explicit-Transient coupling scheme on the right

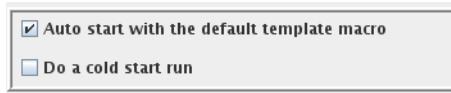


Figure 3: STAR-CCM+ options using default template macro

In the Go Step, the following options can be selected:

Coupling configuration See [V-4.8.2 Coupling Configuration Parameters](#) for more information.

Define the coupling scheme STAR-CCM+ only supports Explicit-Transient and Explicit-SteadyState coupling schemes. The coupling scheme should be chosen in agreement with the solution type of the ".sim" file.

Use subcycling and Use duration control The standard configuration options for subcycling and duration control can be looked up in [V-3.4.6.1 Subcycling Setting in the MpCCI GUI](#). They are used by the default Java macro template provided by MpCCI (see [17.5 Code Adapter Reference](#)). For the Explicit-Transient coupling scheme in STAR-CCM+ special options for the

duration control configuration are provided which are shown in [Figure 2](#) on the right and which are described in the following:

No. of time steps before starting the coupling Provide the number of time steps to perform before beginning the coupling.

No. of coupled time steps Provide the number of coupled time steps. This corresponds to the number of data exchanges.

No. of time steps after the coupling Provide the number of time steps to perform after the coupling ends.

Coupling time step Provide the time step size to use for the unsteady implicit STAR-CCM+ model. This field is optional.

The provided value overwrites the value set in the solver setting in the ".sim" file, otherwise the value from the ".sim" file is used if this value is not coupled.

 In case of an Explicit Model used in STAR-CCM+ this field is not shown.

Maximum no. of iterations per time step Set the maximum number of iterations for the implicit solver.

 In case of an Explicit Model used in STAR-CCM+ this field is not shown.

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 [17.2.4.1 License Setting Details](#).

Run parallel Run STAR-CCM+ in parallel mode, see [17.2.5.1 Parallel Execution](#)

Use remeshing Activate the MpCCI remeshing module, see [17.2.5.2 MpCCI Remeshing Module](#)

17.2.4.1 License Setting Details

License options Define the STAR-CCM+ license options (see [Figure 4](#)).

- Power Session: use power session license model.

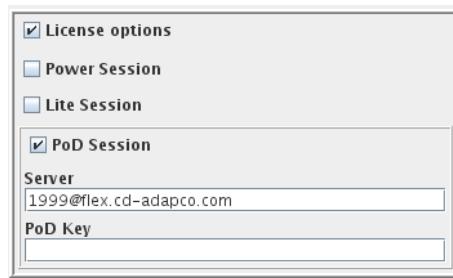


Figure 4: STAR-CCM+ options for license settings

- **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+.

17.2.5 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+.

17.2.5.1 Parallel Execution

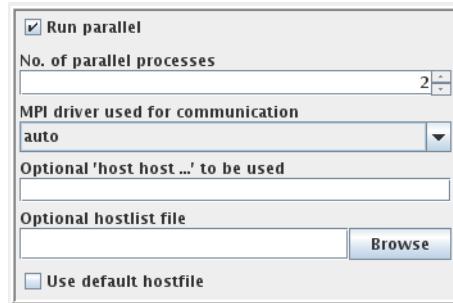


Figure 5: STAR-CCM+ options for a parallel run

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 [▷ V-3.5.2 Hostlist File ◁](#).

(i) 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).

17.2.5.2 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.

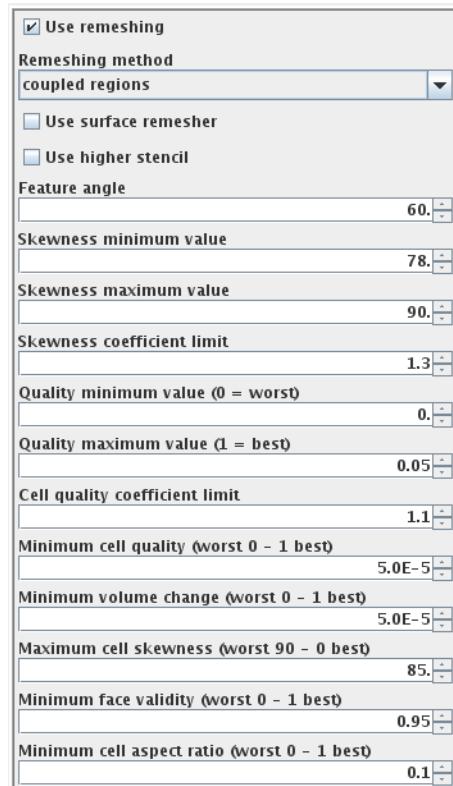


Figure 6: STAR-CCM+ options for remeshing

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.

 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 ration 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.

17.2.5.3 Batch Execution

Activate the option Run in batch (see [Figure 2](#) on the left) to run STAR-CCM+ as a batch process.

17.2.6 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.

17.3 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.

17.4 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

17.5 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 ([▷17.2.4 Go Step◁](#)) and when STAR-CCM+ connects to the MpCCI server.

17.5.1 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.

```
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.

17.5.1.1 Available Commands

The command for MpCCI calls within STAR-CCM+ is available after having instantiated the MpCCI adapter `new Adapter(simulation)`.

```

Adapter mpcci = new Adapter(simulation);
mpcci.initCouplingInfo();
JMPCCIClient.MPCCI_TINFO tinfo = mpcci.getTinfo();
mpcci.mpcciTinfoHasDuration();
mpcci.mpcciTinfoRemoveDurationCtl();
mpcci.mpcciTinfoHasSubcycle();
mpcci.mpcciTinfoRemoveSubcycleCtl();
mpcci.init()
mpcci.exchange()
mpcci.exit();

```

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:

- **String initTimeStep = System.getenv("_MPCCI_STARCCM_TIMENBEG")**: Number of time steps before starting the coupling. The value should be converted into integer in order to process it.
- **String nbTimeStep = System.getenv("_MPCCI_STARCCM_TIMENSTEP")**: Number of coupled time steps. The value should be converted into integer in order to process it.
- **String postTimeStep = System.getenv("_MPCCI_STARCCM_TIMEPEND")**: Number of time steps after the coupling. The value should be converted into integer in order to process it.
- **String dt = System.getenv("_MPCCI_STARCCM_TIMEDT")**: Coupling time step. The value should be converted into double in order to process it.
- **String maxIter = System.getenv("_MPCCI_STARCCM_TIMEITER")**: Maximum number of iterations per time step. The value should be converted into integer in order to process it.

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+.

17.5.1.2 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.
 - nblIteration: 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.

```
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() && tinfo.sub_step > 0)
    {
        nbIteration = tinfo.sub_step;
        mpcci.mpcciTinfoRemoveSubcycleCtl();
    }
}

```

```

if (coldStart != null)
{
    removeInitialSolution = Boolean.parseBoolean(coldStart);
}

nbCoupling = (lastIteration - initIteration) / nbIteration;

int currentIteration = simulation.getSimulationIterator().getCurrentIteration();

// total number of iterations for the solver
int totalIteration = initIteration + nbIteration * nbCoupling + postIteration;
if (removeInitialSolution)
{
    totalIteration += currentIteration;
}

simulation.println("!!! MpCCI Co-simulation setup for steady state analysis:");
simulation.println("!!! Number of data exchange      : " + nbCoupling);
simulation.println("!!! Number of pre-iteration     : " + initIteration);
simulation.println("!!! Number of sub-iteration     : " + nbIteration);
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 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())
{
    // perform nbCoupling data exchange
    for (int i=0; i<nbCoupling; i++)
    {
        // Exchange the data to MpCCI
        mpcci.exchange();
        simulation.println("Starting sub-iterations cycle " + (i+1) + "/"
                           + nbCoupling + "...");
        // iterate nbIteration before exchange
        simulation.getSimulationIterator().step(nbIteration);

        abort = abortFile.exists();
        if (abort)
        {
            simulation.println("Aborting sub-iterations...");
            break;
        }
    }
    // 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();
    }
}
}

```

17.5.1.3 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:

- **nbCouplingSteps**: defines the number of data exchanges.
- **initNbTimeSteps**: defines the number of time steps before the coupling begins.
- **maxInnerIterations**: defines the number of iterations for a time step.
- **endNbTimeSteps**: defines the number of time steps 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).

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 simulation will be initialized and the pre time steps executed.
4. 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 **nbCoupling-Steps** times.
 - The connection to the MpCCI server is closed.
5. The simulation will terminate with some post time steps.

```
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() {
        // number of data exchanges
        int nbCouplingSteps = 1;
        // number of initial time steps before the coupling begins
        int initNbTimeSteps = 0;
        // number of time steps after the coupling
        int endNbTimeSteps = 0;
        // number of time step 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))
{
    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 initTimeStep = System.getenv("_MPCCI_STARCCM_TIMENBEG");
String nbTimeStep = System.getenv("_MPCCI_STARCCM_TIMENSTEP");
String postTimeStep = 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 (nbTimeStep != null)
{
    nbCouplingSteps = Integer.parseInt(nbTimeStep);
}
else
{
    simulation.println("!!! No number of time step has been specified
                        in the MpCCI coupling configuration");
    simulation.println("!!! Simulation will exits after one coupled time step.");
}

if (mpcci.mpcciTinfoHasDuration())
{
    tinfo.time_cend = -1.0;
    if(initTimeStep != null)
    {
        initNbTimeSteps = Integer.parseInt(initTimeStep);
        tinfo.time_cbeg = -1.0;
    }
    if (postTimeStep != null)
    {
        endNbTimeSteps = Integer.parseInt(postTimeStep);
    }
}

```

```

        }
        mpcci.mpcciTinfoRemoveDurationCtl();
    }
    if (mpcci.mpcciTinfoHasSubcycle() && tinfo.sub_step > 0)
    {
        nbStep = tinfo.sub_step;
        mpcci.mpcciTinfoRemoveSubcycleCtl();
    }
    if (maxIter != null)
    {
        maxInnerIterations = Integer.parseInt(maxIter);
    }
    if (dt != null && Double.parseDouble(dt) > 0)
    {
        changeTimeStepSizeSetting = true;
        timeStepSize = Double.parseDouble(dt);
    }
    if (coldStart != null)
    {
        removeInitialSolution = Boolean.parseBoolean(coldStart);
    }

    int currentTimeStep = simulation.getSimulationIterator().getCurrentTimeLevel();

    // total maximum number of time steps
    int maxTimeSteps = initNbTimeSteps + nbCouplingSteps + endNbTimeSteps;
    if (removeInitialSolution)
    {
        maxTimeSteps += currentTimeStep;
    }

    simulation.println("/** MpCCI Co-simulation setup for transient analysis:");
    simulation.println("/** Number of data exchange: " + nbCouplingSteps);
    simulation.println("/** Number of pre-time step: " + initNbTimeSteps);
    simulation.println("/** Number of time steps without coupling: " + nbStep);
    simulation.println("/** Number of inner iteration/time step: " + maxInnerIterations);
    simulation.println("/** Number of post-time step: " + endNbTimeSteps);
    simulation.println("/** Number of existing time steps: " + currentTimeStep);
    simulation.println("/** Total number of time steps: " + maxTimeSteps);
    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().getValue();
        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 steps 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 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;
    } else
    if (stopingCriterion instanceof InnerIterationStoppingCriterion)
    {
        innerIterationStoppingCriterion =
            (InnerIterationStoppingCriterion) stopingCriterion;
    } else
    if (stopingCriterion instanceof PhysicalTimeStoppingCriterion)
    {
        physicalTimeStoppingCriterion =
            (PhysicalTimeStoppingCriterion) stopingCriterion;
    }
}

if (stepStoppingCriterion != null) // reset the maximum steps for the model
{
    stepStoppingCriterion.setMaximumNumberSteps(maxTimeSteps);
}
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");
    }

    // set the maximum number of inner iterations for the time step
    if (innerIterationStoppingCriterion != null)
    {
        innerIterationStoppingCriterion.setIsUsed(true);
        innerIterationStoppingCriterion.setMaximumNumberInnerIterations(maxInnerIterations);
    }
    else
    {
        simulation.println("!!! No inner iteration criterion has been found!");
    }

    // disable maximum physical time stop criteria
    if (physicalTimeStoppingCriterion != null)
    {
        physicalTimeStoppingCriterion.setIsUsed(false);
    }

    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 && initNbTimeSteps > 0)
    {
        // Start a pre-computation without coupling
        simulation.println("Starting pre time steps... ");
        simulation.getSimulationIterator().step(initNbTimeSteps);
        abort = abortFile.exists();
    }

    // Initialize MpCCI co-simulation
    if (!abort && mpcci.init())
    {
        // perform nbCouplingSteps data exchanges
        for (int i=0; i<nbCouplingSteps; i++)
        {
            // Exchange the data to MpCCI
            mpcci.exchange();
        }
    }
}

```

```
simulation.println("Starting time step cycle "+ (i+1) + "/"  
+ nbCouplingSteps + "...");  
// do 1 time step before exchange  
simulation.getSimulationIterator().step(nbStep);  
abort = abortFile.exists();  
if (abort)  
{  
    simulation.println("Aborting time step cycle...");  
    break;  
}  
}  
// Finalize the co-simulation: close the communication with the server  
mpcci.exit();  
}  
  
// Terminate the CFD solution by performing some timesteps: endNbTimeSteps  
// sim is automatically saved  
if (!abort)  
{  
    if (endNbTimeSteps > 0)  
    {  
        simulation.println("Starting post time steps");  
        simulation.getSimulationIterator().step(endNbTimeSteps);  
    }  
}  
else  
{  
    simulation.println("Solution has been aborted.");  
    abortFile.delete();  
}  
}  
}  
}
```

17.6 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.

18 STAR-CD

18.1 Quick Information

Physical Domains	CFD, Fluid, FluidThermal
Company Name	CD-adapco
Company Homepage	www.cd-adapco.com/products/star_cd
Support	www.cd-adapco.com/support
Tutorials	▷ VII-4 Elastic Flap in a Duct ◁ ▷ VII-6 Exhaust Manifold ◁ ▷ VII-7 Busbar System ◁ ▷ VII-10 Cube in a Duct Heater ◁ ▷ VII-11 Y-Junction ◁

 STAR-CD 4.x the STAR-CD plugin concept is used.

Open issues and known bugs are documented in [▷ 18.6 Trouble Shooting, Open Issues and Known Bugs ◁](#)

18.1.1 Supported Coupling Schemes

STAR-CD supports exchange before solution. The initial transfer settings determine the coupling algorithm [▷ V-3.4 Coupling Algorithms ◁](#). Unidirectional and bidirectional transfer is possible.

STAR-CD 4.x

In the plugin version of the adapter at the end of a simulation run a final transfer will be accomplished which is inverted to the initial exchange setting.

Initial Transfer	Final Transfer
receive	send
send	receive
skip	-
exchange	-

 In a simulation where the solution algorithm was set to transient SIMPLE or PISO, after the last iteration step another transfer of data is carried out. Unfortunately at this last exchange received node movements are not accounted for in STAR-CD. This might be important for a restart as the partner code might start from a different state of nodal positions.

18.1.2 Supported Platforms and Versions

The following versions of STAR-CD are supported by MpCCI:

MpCCI_ARCH: Code platform	Supported versions			
	4.16.034	4.18.044	4.20.041	4.22.005
lnx4_x64: linux64_2.6-ifort_11.1-glibc_2.3.4-dso linux64_2.6-ifort_12.1-glibc_2.3.4-dso linux64_2.6-ifort_13.1-glibc_2.5	X	X	X	X
windows_x64: windows64_6.1-ifort_11.1-dso windows64_6.1-ifort_12.1-dso windows64_6.1-ifort_13.1-link_11	X	X	X	X

18.1.3 References

STAR-CD Documentation is part of the STAR-CD distribution (e.g. STAR-CD User Guide).

18.1.4 Adapter Description

The code adapter is a DSO (Dynamic Shared Object) library.

STAR-CD 4.x

For STAR-CD 4.x the MpCCI plugin library "libstarcdmpcci.so" of the desired precision is retrieved from MpCCI installation and dynamically loaded by STAR-CD at runtime and all data transfer and management is carried out over plug-in routines.

18.1.5 Prerequisites for a Coupled Simulation

To run a coupled simulation you need the following:

- Ordinary STAR-CD installation.
- The command `star` should be set in the PATH.
- The environment variable STARDIR should be set to the current installation.
- A MpCCI Grid Morpher license token if it will be used.

18.2 Coupling Process

Please read also [IV-2 Setting up a Coupled Simulation](#).

18.2.1 Model Preparation

The STAR-CD model can be prepared with pro-STAR. Please consider the following approach for model preparation:

- The STAR-CD solver internally operates only in SI units. The geometrical dimensions should best be defined in meters. However, it is possible to scale the mesh dimensions by a scaling factor during the preparation of the geometry files (".`geom`").

- The STAR-CD model must contain a definition of the coupling domains and boundary regions (e.g. couple-wall). Boundary faces of coupled boundaries may not lie on trimmed cells. Those boundary faces must be removed from coupling region.
 - When receiving wall temperature the particular boundary's Wall Heat option must be set to Fixed, receiving heat flux requires option Flux.
 - 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 can be carried out by MpCCI automatically. For special configurations one might be obliged to accomplish manual rework, which is described in [▷18.5 Code Adapter Reference◁](#).
 - Restarting a STAR-CD case will work as usual by definition in pro-STAR (see command `RDATA`). In the MpCCI Go Step the restart options of the problem file may be overwritten by selecting option **Restart prepared case** ([▷18.2.4 Go Step◁](#)), which will invoke STAR-CD with option `-restart`. This option will cause a **Standard Restart**.
-  It is recommended to create a complete STAR-CD 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.

18.2.2 Models Step

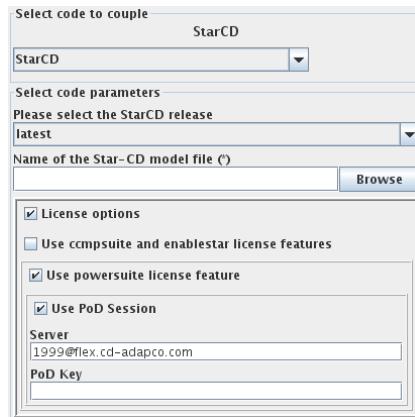


Figure 1: STAR-CD options in the Models Step

In the Models Step, the following options must be selected: ([Figure 1](#)):

STAR-CD release Select the release of STAR-CD you would like to use, latest (default) will select the latest installed STAR-CD version for which an MpCCI adapter is also installed. Please make sure that the model file was not created with a STAR-CD version newer than the STAR-CD version used with MpCCI.

Name of the Star-CD modelfile Select the model file (".**.mdl**") of your STAR-CD model.

License options Define the STAR-CD license options.

- Use ccmpsute and enablestar license features.
- Use powersuite license feature.
 - Use PoD Session.

18.2.3 Coupling Step

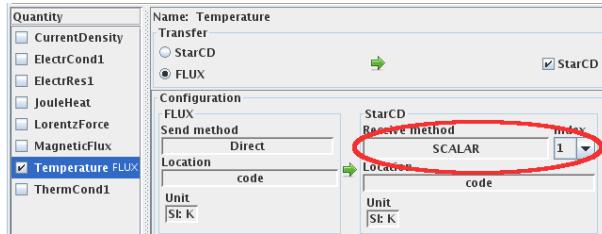


Figure 2: Coupling Step: Specification of the storage index for additional scalar arrays in STAR-CD

In STAR-CD, the exchange of all quantities is handled by plug-ins with STAR-CD Version (STAR-CD 4.x). Transfer data is read and written directly or over user scalars (see STAR-CD documentation “Additional Scalars”). The scalar array is solely used for storing the received or to be sent data. STAR-CD offers scalar arrays and the user needs to decide on which scalar array the data will be stored. The array index is defined in the Quantities section of the corresponding quantity to be exchanged as depicted in Figure 2. The storage index is automatically increased in the MpCCI GUI whenever new quantities are added. However, the user is responsible to choose a free storage index if some scalar arrays are already reserved for settings going beyond MpCCI (e.g. multi-component models).

STAR-CD supports the following quantities for coupling:

Quantity	Dim.	Default Value	Integration Type	Coupling Dimension	Location	Send Option	Receive Option
AbsPressure	Scalar	0.0 N/m ²	flux dens.	Face, Volume	Code	Direct	SCALAR
AcstPressure	Scalar	0.0 N/m ²	flux dens.	Volume	Code	Direct	SCALAR
BodyForce	Vector	0.0 N/m ³	flux dens.	Volume	Code	SCALAR	SCALAR
CGAngle	Vector	0.0 rad	g-max	Global	global	Direct	Direct
CGOmega	Vector	0.0 rad/s	g-max	Global	global	Direct	Direct
CGPosition	Vector	0.0 m	g-max	Global	global	Direct	Direct
CGVelocity	Vector	0.0 m/s	g-max	Global	global	Direct	Direct
ChargeDensity	Scalar	0.0 C/m ³	field	Volume	Code	SCALAR	SCALAR
Current1	Scalar	0.0 A	g-max	Global	global	Direct	Direct
Current2	Scalar	0.0 A	g-max	Global	global	Direct	Direct
Current3	Scalar	0.0 A	g-max	Global	global	Direct	Direct
Current4	Scalar	0.0 A	g-max	Global	global	Direct	Direct
CurrentDensity	Vector	0.0 A/m ²	flux dens.	Face, Volume	Code	SCALAR	SCALAR
DeltaTime	Scalar	1.0 s	g-min	Global	global	Direct	Direct
Density	Scalar	1.0 kg/m ³	field	Volume	Code	Direct	SCALAR
DynPressure	Scalar	0.0 N/m ²	flux dens.	Face	Code	Direct	
ElectrCond1	Scalar	0.0 S/m	field	Face, Volume	Code	SCALAR	SCALAR
ElectrCond3	Vector	0.0 S/m	field	Face, Volume	Code	SCALAR	SCALAR
ElectrCondX	Scalar	0.0 S/m	field	Volume	Code	SCALAR	SCALAR
ElectrCondY	Scalar	0.0 S/m	field	Volume	Code	SCALAR	SCALAR
ElectrCondZ	Scalar	0.0 S/m	field	Volume	Code	SCALAR	SCALAR
ElectricFlux	Vector	0.0 C/m ²	flux dens.	Face, Volume	Code	SCALAR	SCALAR

Quantity	Dim.	Default Value	Integration Type	Coupling Dimension	Location	Send Option	Receive Option
ElectrRes1	Scalar	0.0 ohm m	field	Face, Volume	Code	SCALAR	SCALAR
ElectrRes3	Vector	0.0 ohm m	field	Face, Volume	Code	SCALAR	SCALAR
ElectrResX	Scalar	0.0 ohm m	field	Volume	Code	SCALAR	SCALAR
ElectrResY	Scalar	0.0 ohm m	field	Volume	Code	SCALAR	SCALAR
ElectrResZ	Scalar	0.0 ohm m	field	Volume	Code	SCALAR	SCALAR
Enthalpy	Scalar	0.0 W/m ³	flux dens.	Volume	Code	Direct	SCALAR
FilmTemp	Scalar	300.0 K	field	Face	Code	Direct	SCALAR
HeatFlux	Vector	0.0 W/m ²	flux dens.	Volume	Code	Direct	SCALAR
HeatSource	Scalar	0.0 W/m ³	flux dens.	Volume	Code	SCALAR	SCALAR
IntFlag	Scalar	0	g-max	Global	global	Direct	Direct
IterationNo	Scalar	0	g-max	Global	global	Direct	Direct
JouleHeat	Scalar	0.0 W/m ³	flux dens.	Volume	Code	SCALAR	SCALAR
LorentzForce	Vector	0.0 N/m ³	flux dens.	Volume	Code	SCALAR	SCALAR
MagneticFlux	Vector	0.0 T	flux dens.	Face, Volume	Code	SCALAR	SCALAR
MassFlowRate	Scalar	0.0 kg/s	flux integral	Face	Code	Direct	SCALAR
MassFluxRate	Scalar	0.0 kg/m ² s	flux dens.	Face	Code	Direct	SCALAR
NPosition	Vector	0.0 m	mesh coordinate	Face, Volume	Code		Buffered
OverPressure	Scalar	0.0 N/m ²	flux dens.	Face, Volume	Code	Direct	SCALAR
PhysicalTime	Scalar	0.0 s	g-min	Global	global	Direct	Direct
PorePressure	Scalar	0.0 N/m ²	flux dens.	Volume	Code	Direct	SCALAR
RealFlag	Scalar	0.0	g-max	Global	global	Direct	Direct
RefPressure	Scalar	1.12e5 N/m ²	g-max	Global	global	Direct	Direct
RelWallForce	Vector	0.0 N	flux integral	Face	Code	Direct	SCALAR
Residual	Scalar	0.0	g-max	Global	global	Direct	Direct
SpecificHeat	Scalar	1.0 J/kg K	field	Volume	Code	Direct	SCALAR
Temperature	Scalar	300.0 K	field	Volume	Code	Direct	SCALAR
ThermCond1	Scalar	0.0 W/m K	field	Face, Volume	Code	Direct	SCALAR
ThermCond3	Vector	0.0 W/m K	field	Face, Volume	Code	Direct	SCALAR
ThermCondX	Scalar	0.0 W/m K	field	Volume	Code	Direct	SCALAR
ThermCondY	Scalar	0.0 W/m K	field	Volume	Code	Direct	SCALAR
ThermCondZ	Scalar	0.0 W/m K	field	Volume	Code	Direct	SCALAR
TimeStepNo	Scalar	0	g-max	Global	global	Direct	Direct
TotalPressure	Scalar	0.0 N/m ²	flux dens.	Face	Code	Direct	SCALAR
Velocity	Vector	0.0 m/s	field	Face, Volume	Code	Direct	SCALAR
Voltage1	Scalar	0.0 V	g-max	Global	global	Direct	Direct
Voltage2	Scalar	0.0 V	g-max	Global	global	Direct	Direct
Voltage3	Scalar	0.0 V	g-max	Global	global	Direct	Direct
Voltage4	Scalar	0.0 V	g-max	Global	global	Direct	Direct
WallForce	Vector	0.0 N	flux integral	Face	Code	Direct	SCALAR
WallHeatFlux	Scalar	0.0 W/m ²	flux dens.	Face	Code	Direct	SCALAR
WallHTCoeff	Scalar	0.0 W/m ² K	field	Face	Code	Direct	SCALAR
WallTemp	Scalar	300.0 K	field	Face	Code	Direct	SCALAR

18.2.4 Go Step

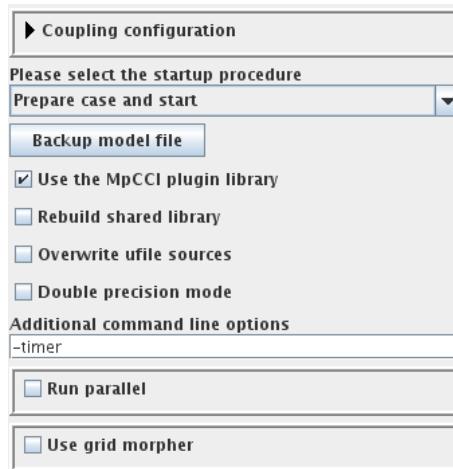


Figure 3: STAR-CD options in the Go Step

STAR-CD offers a number of options in the Go Step, see [Figure 3](#).

Coupling configuration See [▷ V-4.8.2 Coupling Configuration Parameters ◁](#).

Define the coupling scheme STAR-CD only supports Explicit-Transient and Explicit-SteadyState coupling schemes.

(i) For a steady state simulation (e.g. thermal simulation) it might be necessary to let STAR-CD do subcycling in order to reach a stable solution. After each coupling step STAR-CD will carry out the specified number of iterations before the next coupling step is accomplished. ([▷ VII-6 Exhaust Manifold ◁](#))

Initial quantities transfer In STAR-CD the first exchange is invoked prior to the first iteration step (e.g. a time step in a transient simulation).

Please select the startup procedure Define what startup procedure is desired:

Prepare case and start MpCCI modifies your model file ("<casename>.mdl") and generates all STAR-CD input files (e.g. "<casename>.prob"). A geometry file "<casename>.geom" is written if nonexistent. For STAR-CD 4.x necessary hooks are activated and if Use MpCCI plug-ins is selected the plugin is activated in the STAR-CD model file.

Start prepared case MpCCI assumes that all input files (e.g. "<casename>.prob") for the simulation run with the necessary modifications (hooked user subroutines etc.) are already existent. Modifications could have been accomplished by MpCCI through Prepare case and start or manually in pro-STAR.

Restart prepared case This will lead to the same assumption as for Start prepared case, but will also start the STAR-CD simulation with the option -restart, which will overwrite restart settings in problem file to launch a Standard Restart. An Initial Field Restart must be configured over pro-STAR (see command [\[RDATA\]](#)).

A detailed description of the preparation procedure for advanced users is given in [▷ 18.5 Code Adapter Reference ◁](#).

Backup model file In the start-up procedure, depending on the settings above, the model file ("<casename>.mdl") might be modified by MpCCI. Pushing this button will create a backup of your model file.

Use the MpCCI plugin library Enabling this button will cause MpCCI to use the plug-in library in STAR-CD. It can only be used with STAR-CD 4.04 or a higher version. For an activation of the plug-in in the STAR-CD model the option **Prepare case and start** must be selected. See [18.5.2 Automatic Model Preparation STAR-CD 4.x](#) for a manual activation of the plugin in pro-STAR.

Rebuild shared library If this option is selected, STAR-CD will rebuild the DSO library (DSO - dynamic shared objects) with the user subroutines in "<starcdworkdir>/ufile/". An alternative procedure would be to rebuild those libraries with `mpcci starcd -dsomake` before the simulation is started ([18.3 Code-Specific MpCCI Commands](#) and [18.5 Code Adapter Reference](#)).

Overwrite ufile sources Existing sources are automatically copied into backup files. Additional activation of Rebuild shared library is necessary. This procedure corresponds to calling `mpcci starcd -ufile` prior to the simulation run ([18.3 Code-Specific MpCCI Commands](#) and [18.5 Code Adapter Reference](#)).

Double precision mode Use this option to select the double precision mode for STAR-CD computations.

Additional command line options It is possible to invoke STAR-CD with additional command line options (Star Run Options). Please refer to the STAR-CD User Guide for possible options.

Run parallel Run STAR-CD in parallel mode, see [18.2.5.1 Parallel Execution](#)

Use grid morpher Activate the MpCCI Grid Morpher to adapt the mesh to moved or deformed boundaries which are caused e.g. by structural deformations. Depending on the degree of deformation, deactivation of the grid morpher will probably result in corrupted grids. The morpher is described in detail in [18.4.1 MpCCI Grid Morpher](#). Alternatively the pro-STAR Grid Morpher might be applied [18.4.3 pro-STAR Grid Morpher](#).

18.2.5 Running the Computation

When the **Start** button is pressed, STAR-CD is started with the options given in the Go Step. If the **Stop** button is pressed, a stop-file is created and STAR-CD will stop the next time it checks the presence of a stop file.

18.2.5.1 Parallel Execution

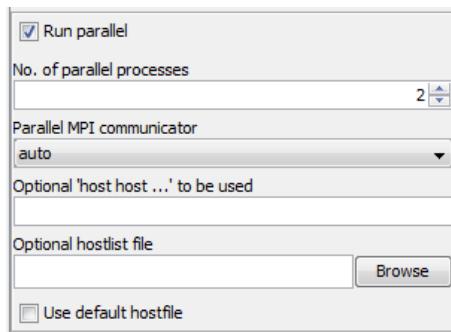


Figure 4: STAR-CD options for a parallel run

If the Run parallel check button is activated the following options are raised ([Figure 4](#)):

No. of parallel processes Select the number of parallel STAR-CD processes to be launched.

Parallel MPI communicator Select the MPI implementation. Further details may be taken from the STAR-CD User Guide under “star run options”.

Optional 'host host...' to be used Enter host names for parallel execution of STAR-CD.

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 [▷V-3.5.2 Hostlist File◀](#).

- (i)** 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-CD. 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).

18.2.5.2 Batch Execution

STAR-CD is always run as a batch process, therefore no special settings are necessary for batch execution.

18.2.6 Post-Processing

Post-processing for the STAR-CD part of the results can be performed as in ordinary computations, e.g. with pro-STAR. The results "*<job name>.pst/ccmp*" or "*<job name>.pstt/ccmt*" file can be found in the same directory as the input file.

To visualize the data from user scalar you should select the data type Cell & Wall/Bound (Smooth) option in pro-STAR.

18.3 Code-Specific MpCCI Commands

The MpCCI subcommands available for STAR-CD are:

Usage:

```
mpcci StarCD [-]option
```

Synopsis:

Use 'mpcci StarCD' to get information about StarCD, to build/install your private PNP DSO adapter 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 <case1> <case2>
Run the scanner on two cases and print the differences.

-dsomake [-dp] [release]
Install the StarCD PNP DSO library for MpCCI in your local "ufile" directory.

If you have your own sources the new PNP DSO will contain both, MpCCI and your own code.

If you wish to modify the sources then please first copy the MpCCI versions of POSDAT and NEWXYZ using the option -ufile.

[-dp] is for linking DOUBLE PRECISION code.

-gmdmake <case>
Make an MpCCI grid morpher data file from a StarCD case file.

-help
This screen.

-info
List verbose information about all StarCD releases.

-magic [case]
Print the magic tokens of the MpCCI grid morpher data file <case>.gmd or all gmd files in the current directory.

-probcheck [case]
Print the MpCCI relevant switches from the problem file <case>.prob or all problem files in the current directory.

-releases

List all StarCD releases which MpCCI can find.

```
-scan <case>
    Run the scanner and create a scanner output file.

-ufile [release]
    Copy the MpCCI versions of POSDAT and NEWXYZ into
    your local "ufile" directory and do some checks.
    Existing sources are automatically copied into backup
    files before.

    You may then edit the sources, add your own code to
    POSDAT and NEWXYZ and make the PNP DSO (see -dsomake).
```

The subcommands `diff`, `info`, `releases` and `scan` are described in [▷ 1.1 Common MpCCI Subcommands for Simulation Codes](#).

`mpcci starcd -dsomake [-dp] [release]`

The `dsomake` subcommand installs the STAR-CD PNP DSO library for MpCCI in your local "ufile" directory. If you have your own sources, the new PNP DSO will contain both, MpCCI and your own code. If you wish to modify the sources then please first copy the MpCCI versions of POSDAT and NEWXYZ using the subcommand `ufile`.

If no release is specified the latest installed release will be taken.

Specify `-dp` for linking double precision code.

`mpcci starcd -gmdmake <case>`

The `gmdmake` subcommand makes an MpCCI grid morpher data file from the given STAR-CD case file.

`mpcci starcd -magic [case]`

The `magic` subcommand gets the magic tokens of the MpCCI grid morpher data gmd file "`<case>.gmd`" or of all gmd files in the current directory if `case` is not specified and prints them to stdout.

`mpcci starcd -probcheck [case]`

The `probcheck` subcommand gets the MpCCI relevant switches either from all problem files in the current directory or from the problem file "`<case>.prob`" if option `case` is specified and prints them to stdout.

`mpcci starcd -ufile [release]`

The `ufile` subcommand copies the MpCCI versions of POSDAT and NEWXYZ into your local "ufile" directory and does some checks. Existing sources are automatically copied into backup files before.

You may then edit the sources, add your own code to POSDAT and NEWXYZ and make the PNP DSO using the subcommand `dsomake`.

18.4 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 has the choice between the MpCCI Grid Morpher ([▷ 18.4.1 MpCCI Grid Morpher](#)) and the pro-STAR Grid Morpher ([▷ 18.4.3 pro-STAR Grid Morpher](#)). Restarting a previously morphed analysis with MpCCI Grid Morpher is described in [▷ 18.4.2 Restart with MpCCI Grid Morpher](#)

18.4.1 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 STAR-CD working directory ("mpcci_morpher_<model name>.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-7 MpCCI Grid Morpher ◁](#).

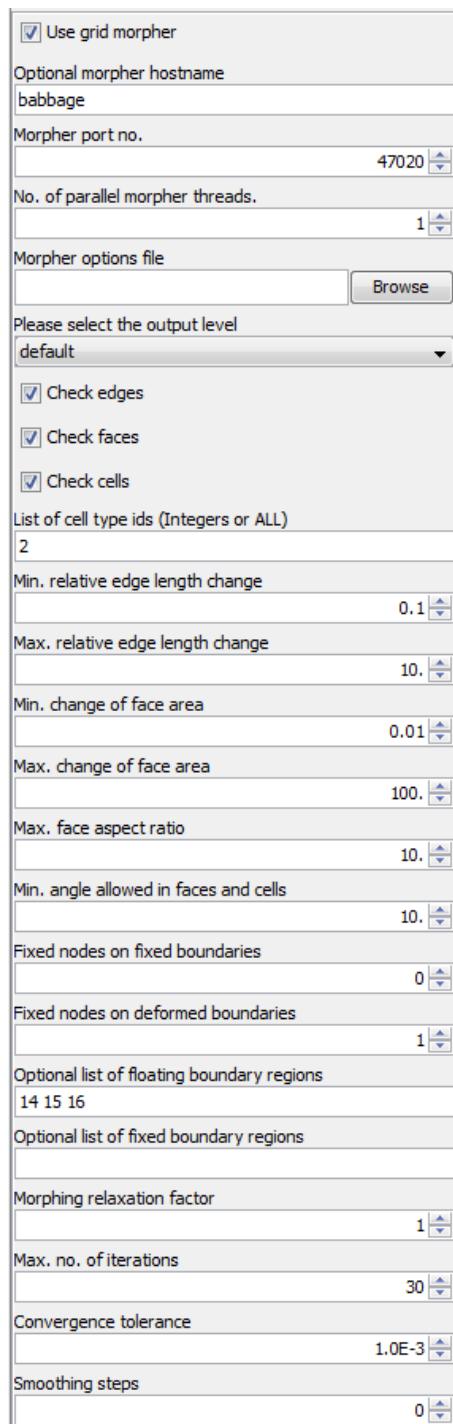


Figure 5: STAR-CD Options for MpCCI Grid Morpher

18.4.2 Restart with MpCCI Grid Morpher

There are two ways to carry out a restart with previously morphed data:

- Take the original model file and read the physical quantities including moved grid coordinates from restart file ("<casename>.pst/<casename>.ccmp"). The MpCCI Grid Morpher reads the grid coordinates from original model file and then receives the restart coordinates in one step. For large displacements the morpher might collapse.
- For a restart run one can build up a new model file with the already moved coordinates:

1. Start pro-STAR
2. Resume original model "<case_org>.mdl"
3. Read previous result "<case_org>.ccmp/.pst" which contains moved coordinates
4. Get the moved coordinates with **GETV COOR**, pro-STAR will confirm *VERTEX COORDINATES STORED*
5. Define restart from "<case_org>.ccmp/.pst"
6. Write problem file "<case_restart>.prob"
7. Write geom file "<case_restart>.geom"
8. Save model as "<case_restart>.mdl", confirm

**WARNING - THE COORDINATES CURRENTLY STORED WERE READ FROM POST DATA.
IF YOU CONTINUE YOU WILL OVERWRITE YOUR ORIGINAL GEOMETRY. CONTINUE (N/Y)?

- with Y
9. You have now stored a model with moved coordinates which can be retrieved by the morpher. Use <case_restart> for further simulation. Be careful not overwrite your original model.

18.4.3 pro-STAR Grid Morpher

The pro-STAR Grid Morpher is available since STAR-CD 4.04 and supplied and supported by CD-adapco. There is a pro-STAR Grid Morpher documentation in the STAR-CD supplementary notes. Further documentation can be retrieved in pro-STAR help regarding the command pro-STAR **morp**. The pro-STAR Grid Morpher is applied by calling pro-STAR in batch mode during the simulation. The nodal deformations are received through MpCCI Code Adapter and handed over to pro-STAR.

In order to use the pro-STAR Grid Morpher the following steps must be accomplished. As the procedure might change in future it is advisable to contact CD-adapco for recent definitions.

1. Open model file "<casename>.mdl" in pro-STAR.
2. Add the following lines to the extended data block:

```
BEGIN MOVINGMESH
IGNORE_LAHEAD_REQUESTS
NXYZ_BEFORE
NO_NXYZ_AFTER
MOVE_ALL_PROCS
END MOVINGMESH
```

3. Create a file "morph.cgrd" containing the line:

```
IFILE starmorph.MAC
```

4. Create an event file "<casename>.evn" in pro-STAR:

```

MEMO MAXEVE 10
MEMO WRITE
MVGR ON EVENT PROSTAR
EVFI INITIAL <casename>.evn
EVSTEP 1 TIME 0.0
EGRID READ morph.cgrd
EVSAVE 1

```

5. Provide a macro "starmorph.MAC" which is executed by pro-STAR. The following commands are exemplary:

```

!scale of boundary point length scale of influence (m)
*set scl1 0.0
*set scl2 0.01
*set scl3 0.1
!
!-set increment frequency for updating reference mesh
!- incrm = 0 reference grid is always initial grid
!- incrm = 1 reference grid is always previous step
!- incrm = n reference grid is updated every nth step
*set incrm 10
!
!copy the moved and all other vertices coordinates into
!vertex register 1
vreg copy,, 0,1
!
! write out the reference grid at beginning
*if time eq 0.
  vwrit PROSTAR_VERT1.vrt,,,bloc
*else
  ! read in the reference grid during run
  vread PROSTAR_VERT1.vrt,,,bloc
*endif
!
!reset the morpher
morp clear
!
!grab the vertices on the fsi interface
!(here e.g. boundary region 5)
bset news rlist 5
vset news bset
!
!If morphing in 2d in xy-plane is desired
morp dime xy
!
!limit number of control points on the fsi surface
!(here e.g. to 200)
morp tout 200
!
!set the control points on the fsi surface for morphing
morp vreg 1,vset,,, scl1
!
```

```

!fix the vertices on boundary planes
!( here e.g. region 1
morp gpla regi 1,, scl2
!
!allow sliding on boundary planes
!(here e.g. region 2)
morp gsym regi 2,, scl3
!
!grab the vertices on other boundaries to fix vertices
!(here e.g. region 8,9 and 10)
bset news rlist 8,9,10
vset news bset
!maybe thin out to get less control points on other boundaries
!(here e.g. to 100)
morp tout 100
!
!set control points of other boundaries for morphing
morp vreg 1,vset,,, scl2
!
!excute the morpher
morp exec
!
!update the reference grid at requested increments
bset news rlist 5
vset news bset
vreg copy vset 1 0
!
*if incr m gt 0
*calc modu,,mod,ITER,,incr m
*if modu eq 0
  vwrit PROSTAR_VERT1.vrt,,,bloc
*endif
*endif

```

6. Activate necessary hooks as described in [18.5.2 Automatic Model Preparation STAR-CD 4.x](#).
7. Use option Start prepared case in Go Step of MpCCI GUI.

When post-processing the results in pro-STAR carry out the following commands:

1. Connect to event file `EVFI,CONN,<casename>.evn`
2. Load result file `TRL0,<casename>.ccmt,MVGRID,, , C`

18.5 Code Adapter Reference

The necessary libraries or plug-ins can be found in "<MpCCI_home>/codes/StarCD/adapters/<StarCDrelease>/"

18.5.1 STAR-CD 4.x

The data transfer and management is handled over plug-ins, which are implemented in the STAR-CD solver and no user subroutines are needed anymore. The activation of the plug-ins are invoked either in pro-STAR ([Table 2](#)) or automatically by MpCCI, when the option **Use the MpCCI plugin library** is selected and **Prepare case and start** is activated ([Figure 3](#)). STAR-CD will verify the adequate (platform, precision) plug-in library "libstarcdmpcci.so" in the MpCCI installation.

As the data transfer is handled over the plug-in, "posdat.f" and the activation of a user defined posdat in pro-STAR is not needed anymore. Nevertheless STAR-CD must know which other plug-ins for data management are needed. Therefore user defined subroutines must be activated automatically or manually (e.g. user bcdefw, see [18.5.2 Automatic Model Preparation STAR-CD 4.x](#)). If the MpCCI plug-in and a user subroutine is activated in the model (e.g. bcdefw), STAR-CD will first call the plug-in bcdefw which is included in the plug-in library and afterward call, if existing, the user bcdefw in the ufile directory.

18.5.2 Automatic Model Preparation STAR-CD 4.x

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. This will be carried out when starting the simulation in the Go Step ([18.2.4 Go Step](#)) when option **Prepare case and start** is selected. On the other hand a user can do the model preparations. The commands which have to be carried out in pro-STAR are depicted in [Table 2](#).

In [Table 2](#) the variables stand for:

- <name> = arbitrary name,
- <region id> = boundary region id in STAR-CD of the coupled surface,
- <case> = STAR-CD casename,
- <scid> = scalar index for data storage,
- <ctype> = cell table number to apply source terms, defined in the Coupling Step,
- \$ = settings which are present in the original model,
- n x \$ = depending on solver settings pro-STAR will prompt for your present settings (e. g. Roughness options).

Feature	STAR-CD assignment	pro-STAR command	Quantity example
Data transfer	activate plugin (only 4.x)	<code>PLUGIN,mpcci,ON</code>	all
MpCCI Morpher	user <i>newxyz</i>	<code>MVGRID,ON,\$</code> <code>EDATA,NEW,1 \$ KEEP_VERTDAT</code> (only 3.26)	NPosition
pro-STAR Morpher	event handling	<code>MVGRID,ON,EVENT,PROSTAR</code> see ▷18.4.3 pro-STAR Grid Morpher◀	NPosition
Receiving time step	user <i>dtstep</i>	<code>DELTIME,USER</code>	DeltaTime
STAR-CD parallel	switch 110	<code>SWITCH,110,ON</code>	
Receive/send	scalar array	<code>SC,<scid>,DEFINE,PASSIVE,UNDEF,1 <name></code>	WallTemp
	scalar post	<code>SC,<scid>,OFF</code> <code>POWA,CW<scid>,Y</code>	
Receive/send Face(2D)	user <i>bcdew</i>	<code>RDEF,<region id>,WALL,USER</code> <code>n x \$</code>	WallTemp
Receive/send Face(2D)	user <i>bcdeti</i>	<code>RDEF,<region id>,INLET,USER</code> <code>n x \$</code>	AbsPressure
Receive/send Volume(3D)	user <i>fluinj</i>	<code>RSOURCE,MASS,CTAB,<ctype>,USER</code>	Density
Receive/send Volume(3D)	user <i>sourmom</i>	<code>RSOURCE,MOME,CTAB,<ctype>,USER</code>	BodyForce
Receive/send Volume(3D)	user <i>sourent</i>	<code>RSOURCE,ENTH,CTAB,<ctype>,USER</code>	JouleHeat

Table 2: Definitions in pro-STAR to enable the model for a coupled simulation

18.6 Trouble Shooting, Open Issues and Known Bugs

<p>Feature: Volume Coupling</p> <p>Version: 4.04</p> <p>Problem: User scalars can not be retrieved from STAR-CD result file. In a restart run STAR-CD retrieves zeros for the first iteration.</p> <p>Workaround: Possible workaround for fluids is to hook the user subroutine "scalfn.f" inserting <code>PHI=SCALAR(<scid>)</code>. There is no workaround for coupled solid volumes.</p> <p>References: ▷ VII-7 Busbar System ◁</p>
<p>Feature: Receiving time step from partner code</p> <p>Version: 4.04</p> <p>Problem: Initial transfer is set to skip or send. Time step is unknown and not retrieved from model file.</p> <p>Workaround: Change your coupling algorithm to have an initial reception of time step. ▷ V-3.4.5 Exchange of Time Step Size ◁, ▷ VII-4 Elastic Flap in a Duct ◁</p>
<p>Feature: Face Coupling</p> <p>Version: 4.04</p> <p>Problem: Boundary face on trimmed cells are not accepted in MpCCI.</p> <p>Workaround: Remove boundary face on trimmed cells from coupling region.</p>
<p>Feature: Transfer after last iteration</p> <p>Version: 4.04</p> <p>Problem: In STAR-CD with transient SIMPLE and PISO algorithm a transfer is called after last iteration. Nodal displacements or moved nodal positions are received but they are not accounted for anymore in STAR-CD.</p> <p>Workaround: no workaround available yet.</p>
<p>Feature: Receiving Wall Temperature</p> <p>Version: 4.04</p> <p>Problem: In STAR-CD receiving WallTemp is configured in MpCCI GUI, but temperature seems not to be accounted for in the solution.</p> <p>Solution: Check the boundary settings for the coupled wall in pro-STAR. Wall Heat must be set to Fixed.</p>

<p>Feature: Polyhedral Meshes</p> <p>Version: 4.04</p> <p>Problem: Cells smoothed with MpCCI Grid Morpher might collapse.</p> <p>Workaround: No workaround available. Further development is in progress.</p>
<p>Feature: Restarting a previously morphed (MpCCI Grid Morpher) simulation by reading moved geometry from result file (.pst/.ccmp).</p> <p>Version: 4.04</p> <p>Problem: At beginning of restart the morpher reads initial geometry which is not moved ▷ 18.4.2 Restart with MpCCI Grid Morpher <4>. The morphing algorithm might collapse.</p> <p>Workaround: Increase iteration number and/or tolerance or build a new model file with moved grid for restart as described in ▷ 18.4.2 Restart with MpCCI Grid Morpher <4>.</p>
<p>Feature: Receiving nodal positions/displacements in STAR-CD</p> <p>Version: 4.04</p> <p>Problem: MpCCI Grid Morpher is not activated, STAR-CD does not get nodal displacements. The plugin for receiving nodal positions/displacements is not called in STAR-CD</p> <p>Workaround: Add lines to Extended Data Block in pro-STAR: BEGIN MOVINGMESH NXYZ_BEFORE NO_NXYZ_AFTER MOVE_ALL_PROCS END MOVINGMESH</p>
<p>Feature: Restart with deformed Coupling Regions</p> <p>Version: 4.04</p> <p>Problem: At the end of the previous run the coupling regions do not match due to time difference. There are orphaned nodes when restarting.</p> <p>Solution: Doing a neighborhood search in MpCCI fails due to mesh differences. Use MpCCI restart files as described in ▷ V-3.4.8 Restarting a Coupled Simulation <4>.</p>

Feature:

Parallel run on remote host

Version:

4.04, 4.06, 4.08

Problem:

STAR-CD exits with error message:

```
PNP: ***ERROR*** The required "mpcci" plug-in is not available in the STAR installation.  
PNP: ==> Please check that the plug-in has been installed properly.
```

Solution:

The hosts configured for parallel run of STAR-CD do not include the host where MpCCI starts STAR-CD. Either include the host on which MpCCI starts STAR-CD in hostlist or provide the environment variable STARPLUGIN_MPCCI=<MpCCI_home>/codes/StarCD to all hosts intended for parallel run.

Feature:

Compiling a ufile with a newxyz.f routine using the function STARMPCCI_NEWXXYZ(VCORN) for using the new displaced nodal coordinates from MpCCI.

The user uses the `mpcci starcd dsomake 4.14.011` to compile the "ufile".

Version:

4.14.011 Windows 32 bit platform

Intel FORTRAN 11.1 environment

Problem:

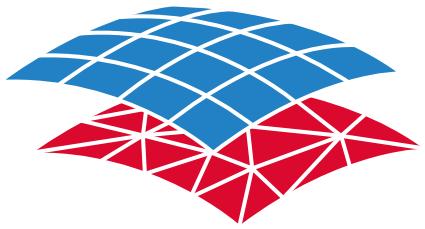
STAR-CD failed to compile the user "ufile" with error message:

```
PNP:           LINK : fatal error LNK1181: Input file "libstarusr.exp" could not be opened.  
PNP: ***ERROR*** Failed to generate user coding dynamic shared object  
"ufile/windows_5.2-ifort_11.1-dso/libstarusr.dll".  
PNP: ==> Please check the dynamic shared object generation error messages.
```

Workaround:

Use STAR-CD 4.14.042 or STAR-CD 4.16.010 release.

This is a STAR-CD issue. Using STAR-CD standalone for compiling a "ufile" directory using the `star -ufile` failed, too.



MpCCI
CouplingEnvironment

Part VII

Tutorial

Version 4.5.0

MpCCI 4.5.0-1 Documentation
Part VII Tutorial
PDF version
March 30, 2017

MpCCI is a registered trademark of Fraunhofer SCAI
www.mpcci.de



Fraunhofer Institute for Algorithms and Scientific Computing SCAI
Schloss Birlinghoven, 53754 Sankt Augustin, Germany

Abaqus and SIMULIA are trademarks or registered trademarks of Dassault Systèmes
ANSYS, FLUENT and ANSYS Icepak are trademarks or registered trademarks of Ansys, Inc.
Elmer is an open source software developed by CSC
FINE/Open and FINE/Turbo are trademarks of NUMECA International
Flowmaster is a registered trademark of Flowmaster Group NV
JMAG is a registered trademark of The JSOL Corporation
MATLAB is a registered trademark of The MathWorks, Inc.
MSC Adams, MSC Marc, MD NASTRAN and MSC NASTRAN are trademarks or registered trademarks of
MSC.Software Corporation
OpenFOAM is a registered trademark of OpenCFD Ltd.
PERMAS is a registered trademark of Intes GmbH
RadTherm, TAITherm is a registered trademark of ThermoAnalytics Inc.
SIMPACK is a registered trademark of SIMPACK AG.
STAR-CCM+ and STAR-CD are registered trademarks of CD adapco Group

ActivePerl has a Community License Copyright of Active State Corp.
FlexNet Publisher is a registered trademark of Flexera Software.
Java is a registered trademark of Oracle and/or its affiliates.
Linux is a registered trademark of Linus Torvalds
Mac OS X is a registered trademark of Apple Inc.
OpenSSH has a copyright by Tatu Ylonen, Espoo, Finland
Perl has a copyright by Larry Wall and others
UNIX is a registered trademark of The Open Group
Windows, Windows XP, Windows Vista and Windows 7 are registered trademarks of Microsoft Corp.

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	Coupling Step	17
2.3.3	Monitors Step	17
2.3.4	Edit Step	18
2.3.5	Go Step	18
2.4	Running the Computation	20
2.5	Discussion of Results	20
3	Driven Cavity	22
3.1	Problem Description	22
3.2	Model Preparation	22
3.2.1	Fluid Model	22
3.2.2	Solid Model	23
3.3	Setting Up the Coupled Simulation with MpCCI GUI	23
3.3.1	Models Step	23
3.3.2	Coupling Step	24
3.3.3	Monitors Step	25
3.3.4	Edit Step	26
3.3.5	Go Step	26
3.4	Running the Computation	27
3.5	Discussion of Results	27
4	Elastic Flap in a Duct	30
4.1	Problem Description	30
4.2	Model Preparation	30
4.2.1	Solid Model	30
4.2.2	Fluid Model	32
4.3	Setting Up the Coupled Simulation with MpCCI GUI	33
4.3.1	Models Step	33
4.3.2	Coupling Step	36

4.3.3	Monitors Step	38
4.3.4	Edit Step	38
4.3.5	Go Step	38
4.4	Running the Computation	41
4.4.1	Starting the Simulation	41
4.4.2	End of the Simulation	42
4.5	Discussion of Results	42
5	Elastic Flap in Water	44
5.1	Problem Description	44
5.2	Model Preparation	44
5.2.1	Solid Model	44
5.2.2	Fluid Model	45
5.3	Setting Up the Coupled Simulation with MpCCI GUI	46
5.3.1	Models Step	46
5.3.2	Coupling Step	48
5.3.3	Monitors Step	49
5.3.4	Edit Step	49
5.3.5	Go Step	49
5.4	Running the Computation	51
5.5	Discussion of Results	52
6	Exhaust Manifold	54
6.1	Problem Description	54
6.2	Model Preparation	54
6.2.1	Solid Model	55
6.2.2	Fluid Model	55
6.2.3	Uncoupled Flow Simulation	57
6.2.4	Prepare Models for Coupled Simulation	57
6.3	Setting Up the Coupled Simulation with MpCCI GUI	58
6.3.1	Models Step	58
6.3.2	Coupling Step	60
6.3.3	Monitors Step	61
6.3.4	Edit Step	61
6.3.5	Go Step	61
6.4	Running the Computation	63
6.5	Post-processing	63
7	Busbar System	65
7.1	Problem Description	65

7.2	Model Preparation	66
7.2.1	Fluid Model	66
7.2.2	Electromagnetic Model	66
7.3	Setting Up the Coupled Simulation with MpCCI GUI	67
7.3.1	Models Step	67
7.3.2	Coupling Step	68
7.3.3	Monitors Step	70
7.3.4	Edit Step	70
7.3.5	Go Step	71
7.4	Running the Computation	72
7.5	Discussion of Results	72
8	Three Phase Transformer	75
8.1	Problem Description	75
8.2	Model Preparation	76
8.2.1	Fluid Model	76
8.2.2	Electromagnetic Model	76
8.3	Setting Up the Coupled Simulation with MpCCI GUI	78
8.3.1	Models Step	79
8.3.2	Coupling Step	79
8.3.3	Monitors Step	80
8.3.4	Edit Step	81
8.3.5	Go Step	81
8.4	Running the Computation	82
8.5	Discussion of Results	82
9	Pipe Nozzle	85
9.1	Problem Description	85
9.2	Model Preparation	85
9.2.1	Fluid Model	86
9.2.2	Solid Model	87
9.3	Setting Up the Coupled Simulation with MpCCI GUI	87
9.3.1	Models Step	88
9.3.2	Coupling Step	89
9.3.3	Monitors Step	90
9.3.4	Edit Step	90
9.3.5	Go Step	90
9.4	Running the Computation	91
9.5	Discussion of Results	92

10 Cube in a Duct Heater	93
10.1 Problem Description	93
10.2 Model Preparation	93
10.2.1 Radiation Model	93
10.2.2 Fluid Model	94
10.3 Setting Up the Coupled Simulation with MpCCI GUI	95
10.3.1 Models Step	96
10.3.2 Coupling Step	97
10.3.3 Monitors Step	98
10.3.4 Edit Step	98
10.3.5 Go Step	98
10.4 Running the Computation	100
10.4.1 Starting the Simulation	100
10.5 Discussion of Results	101
11 Y-Junction	105
11.1 Problem Description	105
11.2 Model Preparation	105
11.2.1 Network Model	106
11.2.2 Fluid Model	109
11.3 Setting Up the Coupled Simulation with MpCCI GUI	111
11.3.1 Models Step	111
11.3.2 Coupling Step	112
11.3.3 Monitors Step	115
11.3.4 Edit Step	115
11.3.5 Go Step	115
11.4 Running the Computation	117
11.4.1 Starting the Simulation	117
11.5 Discussion of Results	118
12 Spring Mass System	122
12.1 Problem Description	122
12.2 Model Preparation	122
12.2.1 Model Description	122
12.2.2 Model A	123
12.2.3 Model B	124
12.3 Setting Up the Coupled Simulation with MpCCI GUI	125
12.3.1 Models Step	125
12.3.2 Coupling Step	129
12.3.3 Monitors Step	132

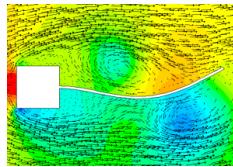
12.3.4 Edit Step	132
12.3.5 Go Step	132
12.4 Running the Computation	138
12.4.1 Starting the Simulation	138
12.5 Discussion of Results	138
12.5.1 Iterative Coupling compared to Explicit Coupling	138
12.5.2 Non-matching Time Step Sizes	139
13 Periodic Rotating Fan Model	141
13.1 Problem Description	141
13.2 Model Preparation	142
13.2.1 Fluid Model	142
13.2.2 Solid Model	143
13.3 Setting Up the Coupled Simulation with MpCCI GUI	144
13.3.1 Models Step	144
13.3.2 Coupling Step	145
13.3.3 Monitors Step	146
13.3.4 Edit Step	146
13.3.5 Go Step	146
13.4 Running the Computation	147
13.5 Discussion of Results	148

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.

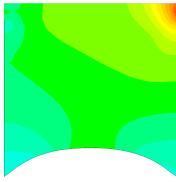
Overview of Tutorials

[▷ 2 Vortex-Induced Vibration of a Thin-Walled Structure ◁](#)



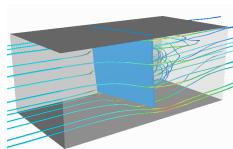
Physical domains:	Fluid-Structure Interaction
Codes:	Fluid Mechanics: FINE/Open, FINE/Turbo, FLUENT, STAR-CCM+ Solid Mechanics: Abaqus, ANSYS, MSC NASTRAN, MSC Marc
Quantities:	NPosition, RelWallForce
Analysis type:	Transient with pre-computed flow field
Special topics:	Different unit systems

[▷ 3 Driven Cavity ◁](#)



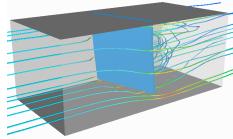
Physical domains:	Fluid-Structure Interaction
Codes:	Fluid Mechanics: FLUENT Solid Mechanics: Abaqus
Quantities:	NPosition, WallForce
Analysis type:	Iterative transient
Special topics:	Quasi-Newton relaxation

[▷ 4 Elastic Flap in a Duct ◁](#)



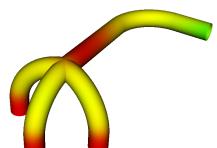
Physical domains:	Fluid-Structure Interaction
Codes:	Solid Mechanics: Abaqus, ANSYS, MSC NASTRAN, MSC Marc Fluid Mechanics: FINE/Open, FINE/Turbo, FLUENT, OpenFOAM, STAR-CCM+, STAR-CD
Quantities:	DeltaTime, NPosition, RelWallForce
Analysis type:	Transient
Special topics:	Exchange of time step size, grid morphing

[▷ 5 Elastic Flap in Water ◁](#)



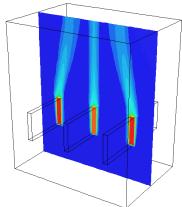
Physical domains:	Fluid-Structure Interaction
Codes:	Solid Mechanics: Abaqus, MSC NASTRAN Fluid Mechanics: FLUENT, OpenFOAM
Quantities:	NPosition, RelWallForce
Analysis type:	Iterative Transient
Special topics:	Usage of the MpCCI Grid Morpher, iterative coupling

[▷ 6 Exhaust Manifold ◁](#)



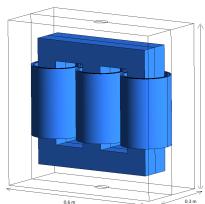
Physical domains:	Thermal Coupling Fluid-Solid
Codes:	Solid Mechanics: Abaqus, ANSYS Fluid Mechanics: FLUENT, OpenFOAM, STAR-CCM+, STAR-CD
Quantities:	FilmTemp, WallHTCoeff, WallTemp
Analysis type:	Steady-state with pre-computed flow field
Special topics:	Subcycling of fluid solver

▷ 7 Busbar System ◁



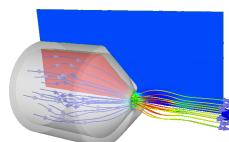
Physical domains: Electrothermal Analysis
 Codes: Fluid Mechanics: FLUENT, STAR-CD
 Electromagnetism: ANSYS, JMAG
 Quantities: JouleHeat, Temperature
 Analysis type: Steady-state flow field calculation and magnetic calculation in frequency domain

▷ 8 Three Phase Transformer ◁



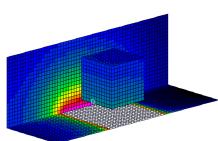
Physical domains: Electrothermal Analysis
 Codes: Fluid Mechanics: FLUENT
 Electromagnetism: JMAG
 Quantities: JouleHeat, Temperature
 Analysis type: Steady-state flow field calculation and magnetic calculation in frequency domain

▷ 9 Pipe Nozzle ◁



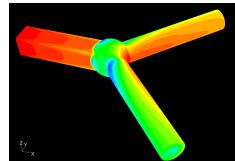
Physical domains: Fluid-Structure Interaction
 Codes: Fluid Mechanics: FLUENT, STAR-CCM+
 Solid Mechanics: ANSYS, MSC NASTRAN
 Quantities: RelWallForce
 Analysis type: Steady-state unidirectional exchange
 Special topics: Axisymmetry, different unit systems

▷ 10 Cube in a Duct Heater ◁



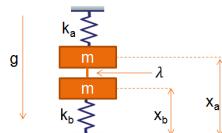
Physical domains: Thermal Coupling Fluid
 Codes: Radiation: RadTherm
 Fluid Mechanics: FLUENT, STAR-CCM+, STAR-CD
 Quantities: FilmTemp, WallHTCoeff, WallTemp
 Analysis type: Steady-state radiative heat transfer
 Special topics: Subcycling

▷ 11 Y-Junction ◁



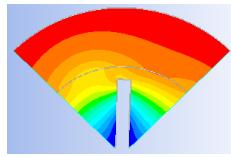
Physical domains: Fluid Mechanics and Fluid Pipe System
 Codes: Network: Flowmaster
 Fluid Mechanics: FLUENT, OpenFOAM, STAR-CCM+, STAR-CD
 Quantities: MassFluxRate, TotalPressure, MassFlowRate
 Analysis type: Steady-state

▷ 12 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

▷ 13 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

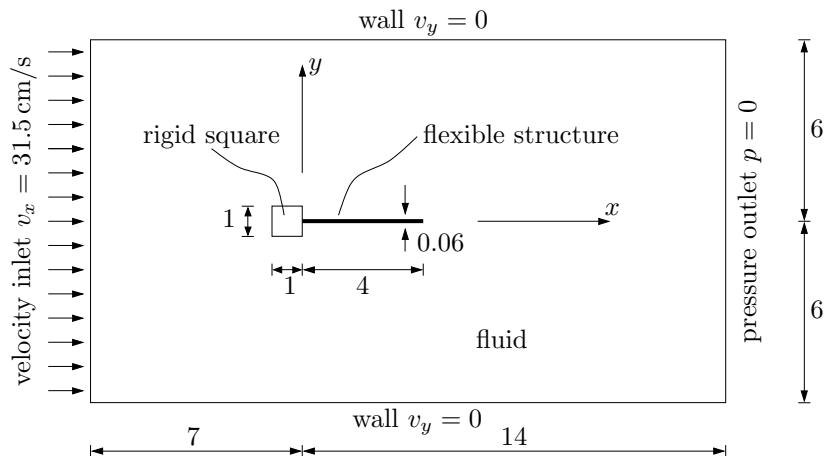


Figure 1: Vortex-induced vibration: Geometry and boundary conditions, the dimensions are given in cm.

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 $\varrho = 1.18 \cdot 10^{-3} \frac{\text{g}}{\text{cm}^3}$ and a viscosity $\mu = 1.82 \cdot 10^{-4} \frac{\text{g}}{\text{cm}\cdot\text{s}}$.

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 reset 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.

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”.

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](#).

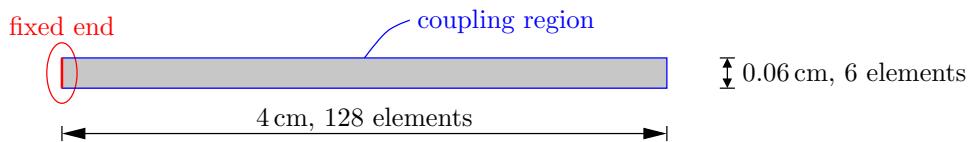


Figure 2: Vortex-induced vibration: flexible solid model and meshing parameters.

The material is linear elastic with density $\varrho = 2.0 \text{ g/cm}^3$, elastic modulus $E = 2.0 \cdot 10^6 \frac{\text{g}}{\text{cm s}^2}$ and Poisson's ratio $\nu = 0.35$.

The mesh consists of 128×6 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 [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`.

2.3.1 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:

FINE/Open

Option	Action
Code to couple	Select FINE/Open.
Release	Should be set to latest or a version supported by MpCCI (see VI-5 FINE/Open).
Model file	Select "vortexVibration_serial.run" in the subdirectory "FINEOpen/vortexVibration_serial/".
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

FINE/Turbo

Option	Action
Code to couple	Select FINE/Turbo.
Release	Should be set to latest or a version supported by MpCCI (see VI-6 FINE/Turbo).
Run file	Select the run file "vortex_cpl_ustd.run" in the subdirectory "FINETurbo/vortex_cpl_ustd".
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

FLUENT

Option	Action
Code to couple	Select FLUENT.
Version	The model is two-dimensional, thus select the FLUENT version 2d.
Release	Should be set to latest or a version supported by MpCCI (see VI-8 FLUENT).
Case file	Select the case file "FLUENT/plate.cas".
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

STAR-CCM+

Option	Action
Code to couple	Select STAR-CCM+.
Release	Should be set to latest or a version supported by MpCCI (see VI-17 STAR-CCM+ ▷).
Simulation file	Select the simulation file "STAR-CCM+/plate2160.sim".
[Start scanner]	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

...

2. Choose your solid mechanics code as second code to couple:

Abaqus

Option	Action
Code to couple	Select Abaqus.
Release	Should be set to latest or a version supported by MpCCI (see VI-2 Abaqus ▷).
Input file	Select input file "Abaqus/plate.inp".
Unit system	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.
[Start scanner]	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

ANSYS

Option	Action
Code to couple	Select ANSYS.
Release	Should be set to latest or a version supported by MpCCI (see VI-3 ANSYS ▷).
Product	You can select any ANSYS product which can be used for structural analysis, e.g. ansys.
Database file	Select database file "ANSYS/plate.db".
Unit system	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.
[Start scanner]	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

MSC NASTRAN

Option	Action
Code to couple	Select MSC NASTRAN.
Release	Should be set to latest or a version supported by MpCCI (see VI-13 MSC NASTRAN).
Input file	Select input file "MSCNastran/vortexvibration.bdf".
Unit system	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.
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

MSC Marc

Option	Action
Code to couple	Select MSC Marc.
Release	Should be set to latest or a version supported by MpCCI (see VI-12 MSC Marc).
Input file	Select input file "MSC.Marc/plate.dat".
Unit system	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.
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

...

Press the **Coupling Step >** button at the bottom of the Models Step to get to the Coupling Step.

2.3.2 Coupling Step

In the Coupling Step build the coupling regions, define quantities to be exchanged and assign the defined quantities to the built regions (cf. ▷IV-2.5 Coupling 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:

For region Region_1	
For fluid mechanics codes	Select coupling component(s)
FINE/Open	▲ VortexVibration_group_6 ▲ VortexVibration_group_7 ▲ VortexVibration_group_8
FINE/Turbo	▲ Beam_1 ▲ Beam_2 ▲ Beam_3
FLUENT	▲ flexible-struct
STAR-CCM+	▲ flexible-struct
For solid mechanics codes	Select coupling component
Abaqus	▲ ASSEMBLY_OUTSIDE
ANSYS	▲ OUTSIDE
MSC NASTRAN	▲ struct
MSC Marc	▲ outside

3. Select Define Quantity Sets tab. As currently no quantities are assigned to “QuantitySet_1” (created by default), a warning symbol is displayed in front of it. Now choose the quantities to be exchanged.

For quantity set	Select quantities	Set sender
QuantitySet_1	NPosition	Solid mechanics code
	RelWallForce	Fluid mechanics code

4. Select Assign Quantity Sets tab and assign the defined quantity set to the built region.

For selected region	Check quantity set
Region_1	QuantitySet_1

Proceed to the Monitors Step by pressing [Monitors Step >].

2.3.3 Monitors Step

No changes are required in the Monitors Step. Proceed to the Edit Step by pressing [Edit Step >].

2.3.4 Edit Step

No changes are required in the Edit Step. Proceed to the Go Step by pressing [Go Step >].

2.3.5 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 Start-Up and Starting Server and Codes](#)).

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:

- Set Coupling configuration options which are the same for all fluid mechanics codes.

Option	Action
Coupling scheme	Select Explicit-Transient.
Initial quantities transfer	Select exchange.

- Set options which are code specific.

FINE/Open

Option	Action
Expert mode	Check to access further options.
Mesh deformation	Select Radial basis functions.

FINE/Turbo

Option	Action
Expert mode	Check to access further options.
Mesh deformation	Select Radial basis functions.
Save inverse matrix of radial basis functions	Select Compute and use inverse matrix.

FLUENT

Option	Action
Data file (optional)	Choose the data file "FLUENT/plate-2160.dat" because FLUENT does a restart from pre-computed flow field.

STAR-CCM+

Option	Action
Use duration control	Check and access further options.
No. of coupled time steps	Set to 10000.
Coupling time step size	Set to 0.001.
Maximum no. of iterations per time step	Set to 30.
Run in batch	Check this option.
Auto start with the default template macro	Check this option.
License options	Set according to your license type (cf. VI-17.2.4 Go Step).

...

2. For your solid mechanics code:

- Set Coupling configuration options which are the same for all solid mechanics codes.

Option	Action
Coupling scheme	Select Explicit-Transient.
Initial quantities transfer	Select receive.

- Set options which are code specific.

Abaqus

Option	Action
Constant coupling time step	Check this option to use a constant time step.
Coupling time step	Set to 0.001, which equals the time step size in the Abaqus input file.
Use subcycling	Deselect this option.

ANSYS

Option	Action
Gui option	Keep -b to start ANSYS in batch mode.
APDL input script	Select the input script "ANSYS/runplate.ans".

MSC NASTRAN

No code specific options need to be set.

MSC Marc

No code specific options need to be set.

...

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 three **Start** buttons in the Go Step and 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→Run Calculation**. 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 maximum number of time steps is read from the MpCCI GUI setting which was defined to 10000.

...

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](#).

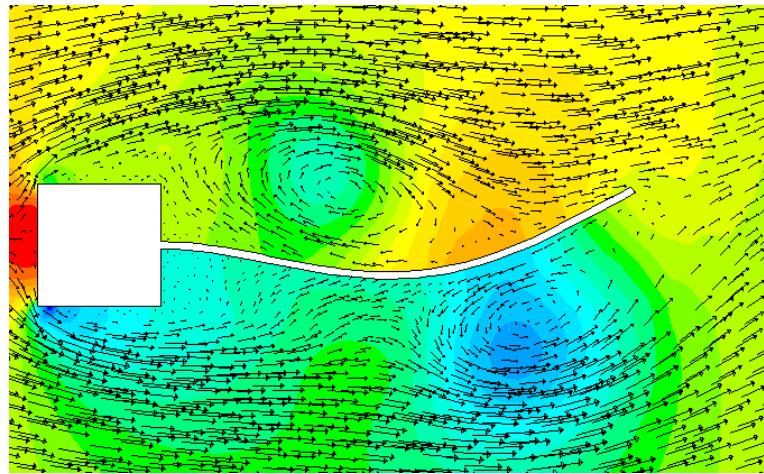


Figure 3: Vortex-induced vibration: Part of the flow field in FLUENT with velocity vectors and pressure distribution (red=high pressure, blue=low pressure).

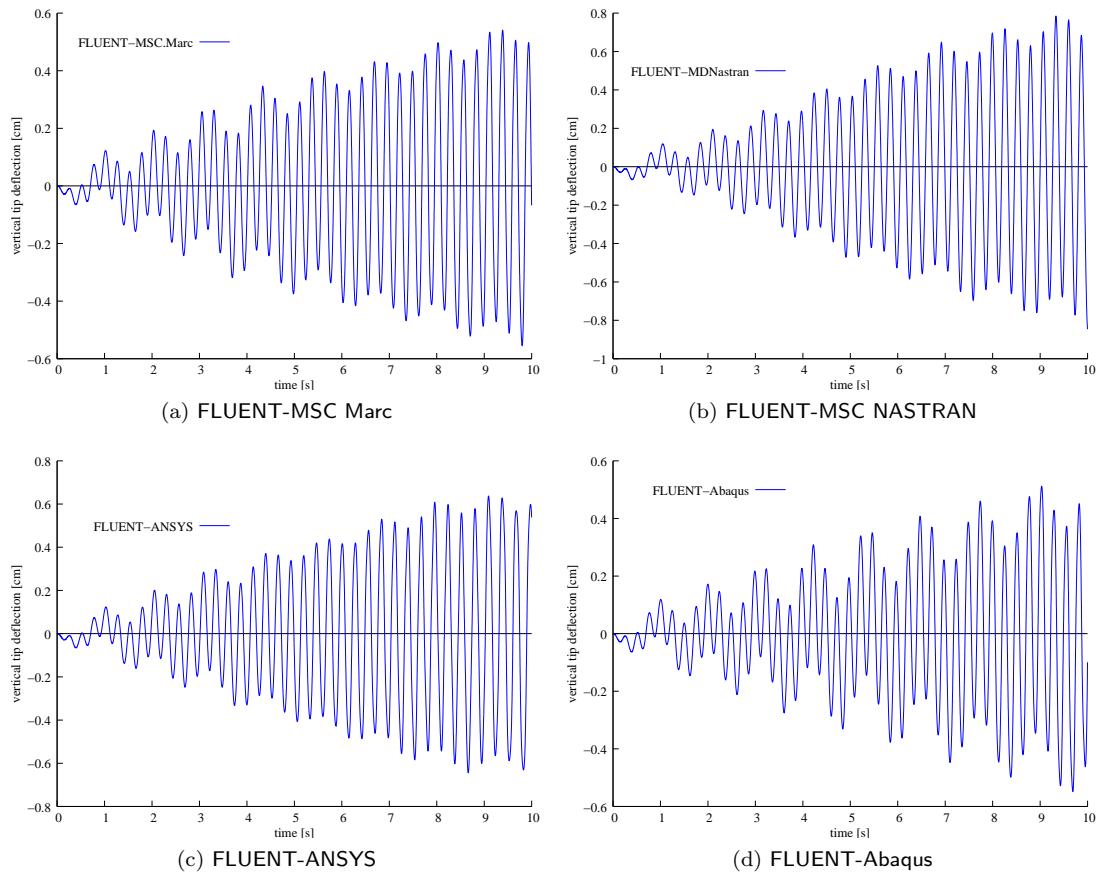


Figure 4: Vortex-induced vibration: Vertical deflection of the tip.

3 Driven Cavity

3.1 Problem Description

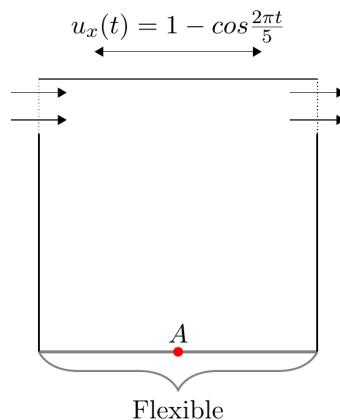


Figure 1: Driven cavity: Geometry of the two-dimensional flowed through cavity with moving top face and flexible bottom.

Topics of this Tutorial

- Fluid-Structure Interaction (FSI)
- Iterative Transient Coupling with Quasi-Newton methods
- 2D-model

Simulation Codes

- Fluid Mechanics: FLUENT
- Solid Mechanics: Abaqus

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.

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 ($1 \text{ m} \times 1 \text{ m}$) shown in [Figure 1](#). 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 $\rho_F = 1 \frac{\text{kg}}{\text{m}^3}$ and a viscosity of $\nu_F = 0.001 \frac{\text{m}^2}{\text{s}}$.

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

An adaptive mesh is used. Overall, the mesh consists of 24×24 quadrilateral 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 $\rho_S = 250 \frac{\text{kg}}{\text{m}^3}$, elastic modulus $E = 250 \frac{\text{kg}}{\text{m} \cdot \text{s}^2}$ 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

In Abaqus, the flexible wall is meshed by three layers of 80 CPS4 elements. In order to define the region, where quantities are exchanged, a surface named “COUPLINGSURF” is built from the top of the flexible structure.

...

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 `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:

FLUENT

Option	Action
Code to couple	Select FLUENT.
Version	The model is two-dimensional, thus select the FLUENT version 2ddp.
Release	Should be set to latest or a version supported by MpCCI (see VI-8 FLUENT).
Case file	Select the case file "FLUENT/cavity.cas".
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

- ...
2. Choose your solid mechanics code as second code to couple:

Abaqus	
Option	Action
Code to couple	Select Abaqus
Release	Should be set to latest or a version supported by MpCCI (see VI-2 Abaqus).
Input file	Select input file "ABAQUS/body.inp".
Unit system	Select the SI unit system (which is the standard).
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

Press the **Coupling Step >** button at the bottom of the Models Step to get to the Coupling Step.

3.3.2 Coupling Step

In the Coupling Step build the coupling regions, define quantities to be exchanged and assign the defined quantities to the built regions (cf. [IV-2.5 Coupling 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:

For region Region_1	
For fluid mechanics code	Select coupling component
FLUENT	▲ bottom
For solid mechanics code	Select coupling component
Abaqus	▲ COUPLINGSURF

3. Select Define Quantity Sets tab. As currently no quantities are assigned to “QuantitySet_1” (created by default), a warning symbol is displayed in front of it. Now choose the quantities to be exchanged.

For quantity set	Select quantities	Set sender
QuantitySet_1	NPosition	Solid mechanics code
	WallForce	Fluid mechanics code

4. Add a relaxation operator to quantity WallForce (cf. [V-4.5.7.4 Applying Operators to Mesh Based Components](#)).

For a stable and accelerated computation, the Quasi-Newton relaxation is used in conjunction with transient iterative coupling. MpCCI offers the following Quasi-Newton methods:

- Anderson mixing in the variants Standard, Inverse and LeastSquares,
- Broyden’s method, and

- Jacobi method in the variants Standard and Inverse.

Anderson mixing and the Jacobi method have high memory and computational cost, which depend quadratically on the degrees of freedoms of the quantity. In exchange, both methods have a good convergence behaviour. The complexity of the Anderson mixing is linear in the quantity's degrees of freedom, but usually converges slower than the other two methods. Thus, for a not too large number of degrees of freedom, i. e. low tens of thousands, the Jacobi method is the best choice. For higher discretized interfaces, it is advisable to use the LeastSquares type of the Anderson mixing method. A more detailed description of Quasi-Newton methods is given in [▷ V-3.3.3.1 Quasi-Newton Methods ◁](#).

Apply the relaxation operator to the WallForce quantity in QuantitySet_1 as follows:

Option	Action
Quantity	Mark the already selected quantity WallForce by clicking on its name.
Operator	Select Relaxation.

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:

Option	Action
Relaxation method	Select Quasi-Newton.
Quasi-Newton method	Select Jacobi.
Type of Jacobi method	Select Inverse.
Omega(ω)	Set to 0.1 which is the default.

5. Add a convergence check operator to quantity WallForce (cf. [▷ V-4.5.7.4 Applying Operators to 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 QuantitySet_1 as follows:

Option	Action
Quantity	Mark the already selected quantity WallForce by clicking on its name.
Operator	Select ConvergenceCheck.

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:

Option	Action
Tolerance value	Set to $1e - 4$ which is appropriate for this simulation.

6. Select Assign Quantity Sets tab and assign the defined quantity set to the built region.

For selected region	Check quantity set
Region_1	QuantitySet_1

Proceed to the Monitors Step by pressing **[Monitors Step >]**.

3.3.3 Monitors Step

No changes are required in the Monitors Step. Proceed to the Edit Step by pressing **[Edit Step >]**.

3.3.4 Edit Step

In the Edit 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:

Parameter	Value
Properties.Monitor.Peaks	Set selected.

Proceed to the Go Step by pressing **[Go Step >]**.

3.3.5 Go Step

In the Go Step configure the application start-up and start the server and the codes (cf. [IV-2.8 Go Step – Configuring the Application Start-Up and Starting Server and Codes](#)).

The current Quasi-Newton methods, described in [3.3.2 Coupling Step](#), rely on serial coupling algorithms, so the **Initial quantities transfer** of the coupled codes have to be chosen as a combination of **receive** and **exchange** or as a combination of **receive** and **skip**.

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:

- Set Coupling configuration options which are the same for all fluid mechanics codes.

Option	Action
Coupling scheme	Select Implicit-Transient .
Initial quantities transfer	Select exchange .

- Set options which are code specific.

FLUENT	
Option	Action
Iteration control	Check and access further options.
Number of iterations without coupling	Set to 25.
Maximum number of coupling iterations	Set to 50.
Coupling time step size	Set to 0.1 s.

2. For your solid mechanics code:

- Set Coupling configuration options which are the same for all solid mechanics codes.

Option	Action
Coupling scheme	Select Implicit-Transient.
Initial quantities transfer	Select receive.

- Set options which are code specific.

Abaqus	
Option	Action
Iteration control	Check and access further options.
Maximum number of coupling iterations	Set to 50.
Coupling time step size	Set to 0.1 s.

...

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 three **Start** buttons in the Go Step and the simulation codes should start.

Some codes require additional actions:

FLUENT

Select **Solve→Run Calculation...** to open the Run Calculation panel.

As the time step size is fixed to 0.1 s, you should select 400 Time Steps, to cover a simulation time of 40.0 s at least.

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 1250 iterations per time step are needed – for 25 iterations between the data exchanges and at most 50 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 **Solve→MpCCI Run FSI**. Set the Number of Time Steps to 400 and press **Run** to start the simulation.

During the simulation, FLUENT will display the residuals.

...

3.5 Discussion of Results

The moving top face induces an oscillation of the flexible wall at the bottom of the cavity, as shown in [Figure 2](#).

Due to the strong added mass effect, the result can only be produced by an iterative coupling: an explicit transient coupling scheme leads to divergence, as shown in [Figure 3](#).

If the iterative coupling is used without any relaxation, the number of necessary coupling iterations until convergence ($tol = 1e - 4$) is on average 30 per time step. This results in a simulation time of 20.5 h. The Quasi-Newton relaxation, which was chosen in this tutorial, leads to a time saving by a factor of 7,

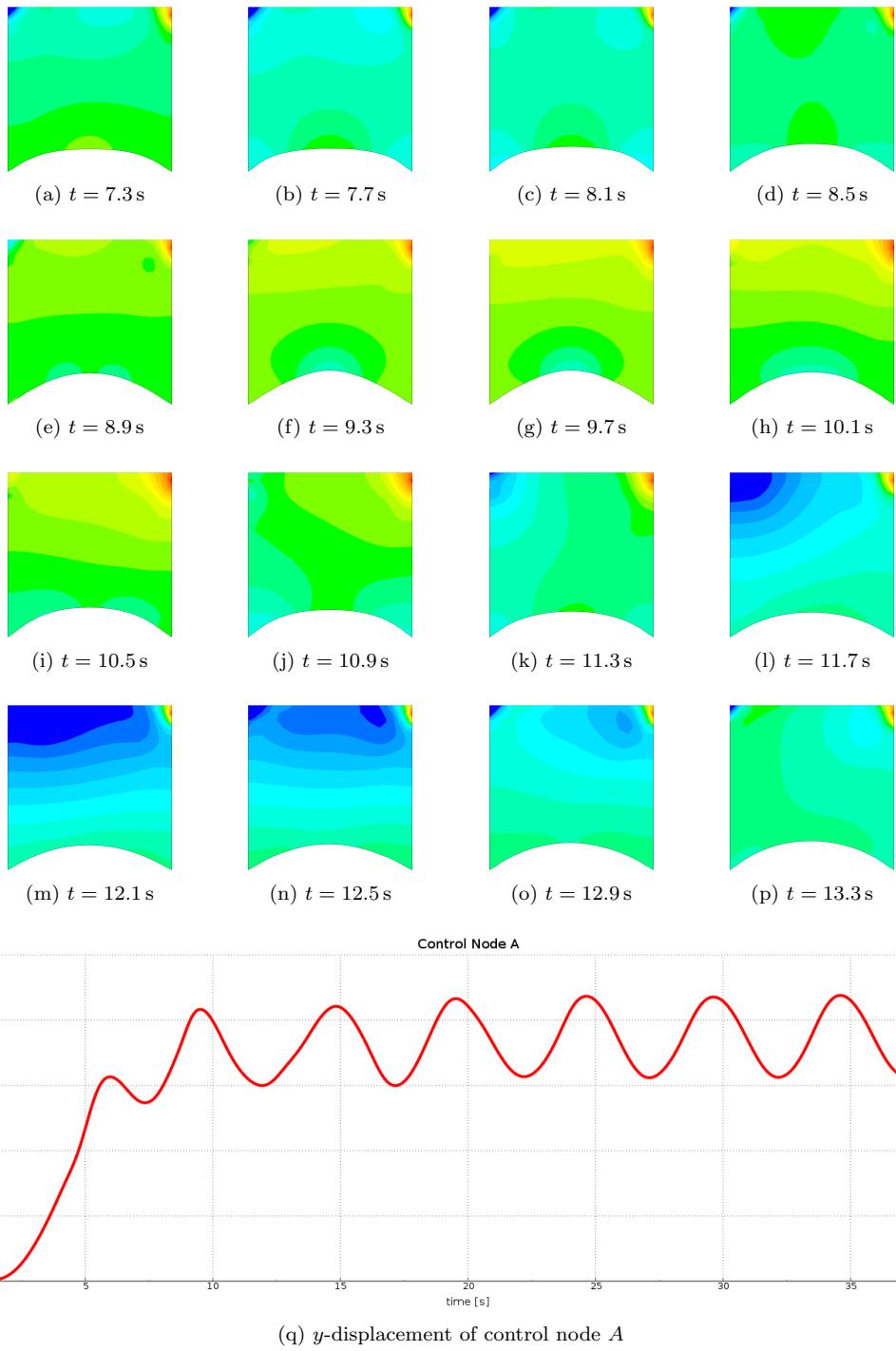


Figure 2: Driven cavity: Simulation results.

while the results of both iterative methods coincide. This reduction is attributed to the small number of necessary inner iterations (average 5), cf. Figure 4a. The oscillation of the iterations is completely

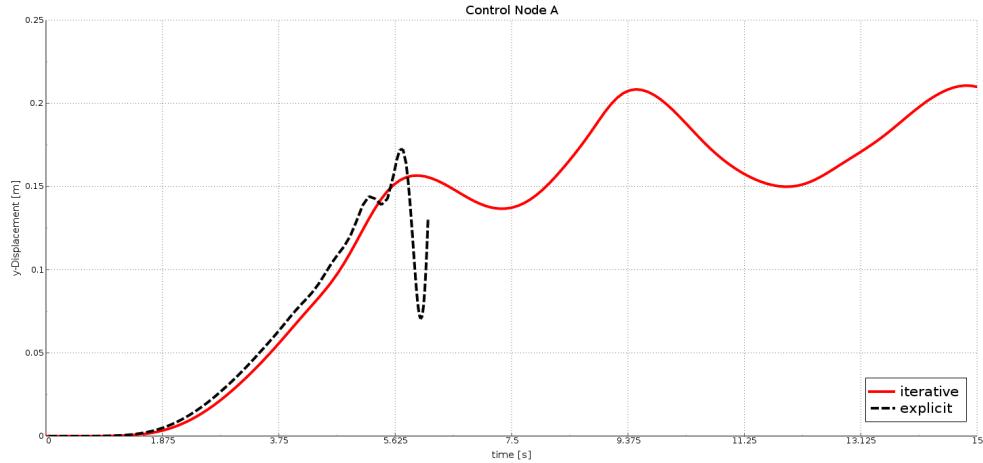


Figure 3: Driven cavity: Computed y -displacement of control node A with an explicit (divergent) and an iterative (convergent) transient coupling scheme.

eliminated by the Quasi-Newton relaxation, cf. Figure 4b.

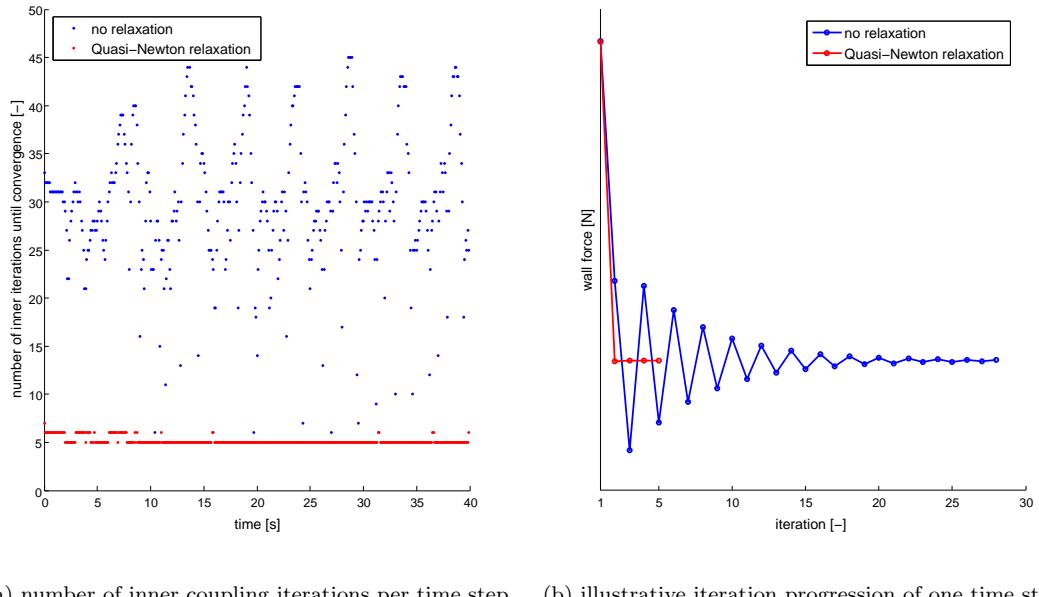


Figure 4: Driven cavity: Comparison of iterative coupling schemes with respect to the number of necessary coupling iterations to converge the time steps.

4 Elastic Flap in a Duct

4.1 Problem Description

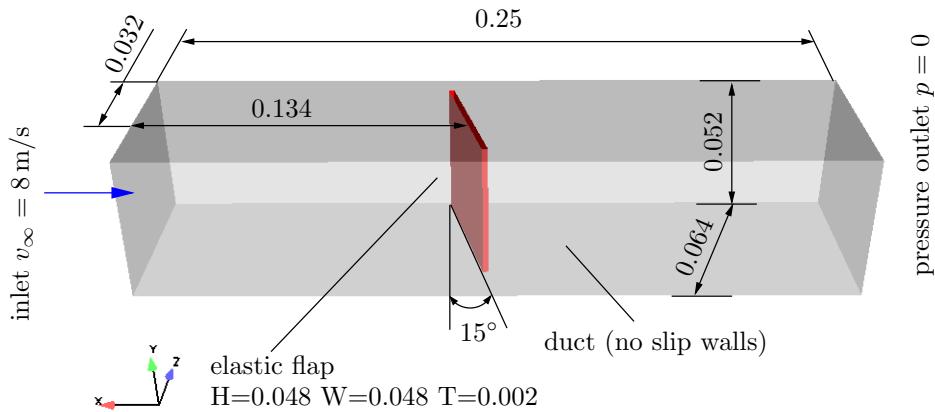


Figure 1: Elastic flap: Geometry [m] and boundary conditions

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 STAR-CD and OpenFOAM
- 3D-model

Simulation Codes

- Solid Mechanics: Abaqus, ANSYS, MSC NASTRAN, MSC Marc
- Fluid Mechanics: FINE/Open, FINE/Turbo, FLUENT, OpenFOAM, STAR-CCM+, STAR-CD

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 $\varrho = 1000 \text{ kg/m}^3$, elastic modulus $E = 1.0 \times 10^8 \text{ Pa}$ 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 $t = 0.2 \text{ s}$.

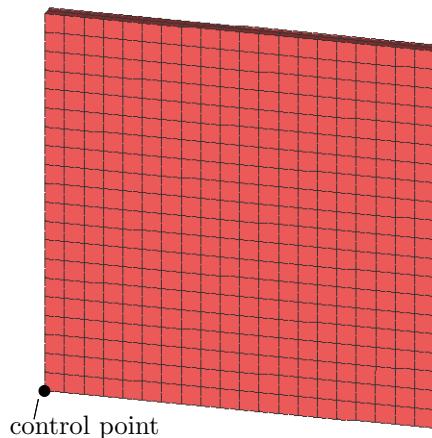


Figure 2: Elastic flap: Mesh of the structural domain

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:

```
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 an 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.

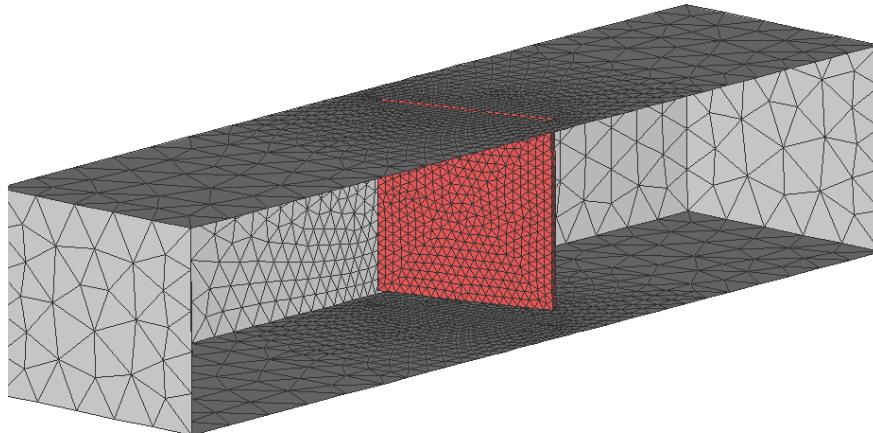


Figure 3: Elastic flap: Partial mesh of the fluid domain

FINE/Open

The mesh was generated with HEXPRESS and consists of hexahedral elements. The surface of the flexible structure is defined as separate solid boundaries.

FINE/Turbo

The mesh was generated with IGG and consists of hexahedral elements. The surface of the flexible structure is defined as separate solid boundaries.

FLUENT

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”.

OpenFOAM

The mesh corresponds to the mesh which is used in FLUENT.

This application requires a solver with mesh motion handling. For OpenFOAM versions 1.7 to 2.1 a customized OpenFOAM solver is provided in order to handle turbulent, compressible fluid flow with mesh motion. Its name is `rhoPimpleDyMFoamMpCCI` and it is derived from `rhoPimpleFoam`. Since source code is not unconditionally compatible with any version of OpenFOAM we provide three versions of our solver – namely one for OpenFOAM 1.7, one for OpenFOAM 2.0 and one for OpenFOAM 2.1. To build either of the solvers

- Go to "`OpenFOAM/<version>/src/rhoPimpleDyMFoamMpCCI`"
e.g. "`OpenFOAM/1.7/src/rhoPimpleDyMFoamMpCCI`" for OpenFOAM 1.7.1.
- Make sure you have already sourced the appropriate environment e.g.
`"$FOAM_INST_DIR/OpenFOAM-<version>/etc/bashrc"`.
- Then type in the commands `wclean` and `wmake` subsequently.

If you are using OpenFOAM 2.2 or higher, the solver `rhoPimpleDyMFoam` is part of the OpenFOAM installation and can be used for this tutorial.

The solver to use is configured in the "controlDict" file from the case directory.

STAR-CCM+

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.

STAR-CD

The mesh corresponds to the mesh which is used in FLUENT. The boundary regions in the middle are subdivided, because they will later be addressed by the MpCCI Grid Morpher as sliding boundaries.

...

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 solid mechanics code as first code to couple:

Abaqus

Option	Action
Code to couple	Select Abaqus.
Release	Should be set to latest or a version supported by MpCCI (see VI-2 Abaqus).
Input file	Select input file "Abaqus/elastic_flap.inp".
Unit system	Select the SI unit system (which is the standard).
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

ANSYS

Option	Action
Code to couple	Select ANSYS.
Release	Should be set to latest or a version supported by MpCCI (see VI-3 ANSYS ▲).
Product	You can select any ANSYS product which can be used for structural analysis, e.g. ansys.
Database file	Select database file "ANSYS/elastic_flap.db".
Unit system	Select the SI unit system (which is the standard).
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

MSC Marc

Option	Action
Code to couple	Select MSC Marc.
Release	Should be set to latest or a version supported by MpCCI (see VI-12 MSC Marc ▲).
Input file	Select input file "MSC.MARC/elastic_flap_job1.dat".
Unit system	Select the SI unit system (which is the standard).
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

MSC NASTRAN

Option	Action
Code to couple	Select MSC NASTRAN.
Release	Should be set to latest or a version supported by MpCCI (see VI-13 MSC NASTRAN ▲).
Input file	Select input file "MSCNastran/elastic_flap.bdf".
Unit system	Select the SI unit system (which is the standard).
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

...

2. Choose your fluid mechanics code as second code to couple:

FINE/Open

Option	Action
Code to couple	Select FINE/Open.
Release	Should be set to latest or a version supported by MpCCI (see VI-5 FINE/Open ▲).
Model file	Select "elasticFlap_serial.run" in the subdirectory "FINEOpen/elasticFlap_serial/".
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

FINE/Turbo

Option	Action
Code to couple	Select FINE/Turbo.
Release	Should be set to latest or a version supported by MpCCI (see VI-6 FINE/Turbo ▲).
Run file	Select the run file "flap_model.run" in the subdirectory "FINETurbo/flap_model".
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

FLUENT

Option	Action
Code to couple	Select FLUENT.
Version	The model is three-dimensional, thus select the FLUENT version 3d.
Release	Should be set to latest or a version supported by MpCCI (see VI-8 FLUENT ▲).
Case file	Select the case file "FLUENT/elastic_flap.cas".
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

OpenFOAM

Option	Action
Code to couple	Select OpenFOAM.
Option	Select the OpenFOAM option Opt or Debug.
Precision	Select the OpenFOAM precision: DP for double or SP for single precision.
Release	Should be set to latest or a version supported by MpCCI (see VI-14 OpenFOAM ▲).
Case directory	Select the case directory "OpenFOAM/<version>" fitting your OpenFOAM version (e.g. "OpenFOAM/3.0" for use with OpenFOAM 3.0.1).
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

STAR-CCM+

Option	Action
Code to couple	Select STAR-CCM+.
Release	Should be set to latest or a version supported by MpCCI (see VI-17 STAR-CCM+ ▲).
Simulation file	Select the simulation file "STAR-CCM+/elasticflap.sim".
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

STAR-CD	
Option	Action
Code to couple	Select STAR-CD.
Release	Should be set to latest or a version supported by MpCCI (see ▷ VI-18 STAR-CD ▷).
Model file	Select the model file "STARCD/elastic_flap.mdl".
License options	Set according to your license type (cf. ▷ VI-18.2.2 Models Step ▷).
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

Press the **Coupling Step >** button at the bottom of the Models Step to get to the Coupling Step.

4.3.2 Coupling Step

In the Coupling Step build the coupling regions, define quantities to be exchanged and assign the defined quantities to the built regions (cf. ▷ IV-2.5 Coupling Step – Defining Coupling Regions and Quantities ▷).

In the following the **fluid mechanics** code will be called CFD code (CFD = Computational Fluid Dynamics) and the **solid mechanics** code CSM code (CSM = Computational Structural Mechanics).

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:

For region Global_1	
For CSM codes	Select global variable
Abaqus	
MSC Marc	Var Time-Step-Size
MSC NASTRAN	
ANSYS	Var DELTATIME
For CFD codes	Select global variable
FINE/Open	
FINE/Turbo	
FLUENT	
OpenFOAM	Var Time-Step-Size
STAR-CCM+	
STAR-CD	

3. Select DeltaTime as quantity to be transferred and configure as follows:

For quantity DeltaTime	
Option	Action
Sender	Select CFD code.
Default value which is set on sender site	Set to 2.5E-4.

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 (\blacktriangle) 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	
For CSM codes	Select coupling component
Abaqus	\blacktriangle ASSEMBLY_WALL_WALL-SURFACE
ANSYS	\blacktriangle WALLSURFACE
MSC Marc	\blacktriangle wallsurface
MSC NASTRAN	\blacktriangle flap
For CFD codes	Select coupling component(s)
FINE/Open	\blacktriangle Elastic_Flap_group_0 \blacktriangle Elastic_Flap_group_1 \blacktriangle Elastic_Flap_group_2 \blacktriangle Elastic_Flap_group_3 \blacktriangle Elastic_Flap_group_4
FINE/Turbo	\blacktriangle flap_front \blacktriangle flap_back \blacktriangle flap_bottom \blacktriangle flap_right \blacktriangle flap_left
FLUENT	
OpenFOAM	
STAR-CCM+	
STAR-CD	

3. Select Define Quantity Sets tab. As currently no quantities are assigned to “QuantitySet_1” (created by default), a warning symbol is displayed in front of it. Now choose the quantities to be exchanged.

For quantity set	Select quantities	Set sender
QuantitySet_1	NPosition	CSM code
	RelWallForce	CFD code

4. Select Assign Quantity Sets tab and assign the defined quantity set to the built region.

For selected region	Check quantity set
Region_1	QuantitySet_1

Proceed to the Monitors Step by pressing **[Monitors Step >]**.

4.3.3 Monitors Step

No changes are required in the Monitors Step, proceed to the Edit Step by pressing **[Edit Step >]**.

4.3.4 Edit Step

No changes are required in the Edit Step, proceed to the Go Step by pressing **[Go Step >]**.

4.3.5 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 Start-Up and Starting Server and Codes](#)).

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:

- Set Coupling configuration options which are the same for all solid mechanics codes.

Option	Action
Coupling scheme	Select Explicit-Transient.
Initial quantities transfer	Select receive.

- Set options which are code specific.

Abaqus

Option	Action
Constant coupling time step	It is important to deselect this option (which is the default) because the time step size is determined by the CFD code.

ANSYS

Option	Action
Gui option	Keep -b to start ANSYS in batch mode.
APDL input script	Select the input script "ANSYS/runflap.ans".

MSC NASTRAN

No code specific options need to be set.

MSC Marc

No code specific options need to be set.

...

2. For your fluid mechanics code:

- Set Coupling configuration options which are the same for all fluid mechanics codes.

Option	Action
Coupling scheme	Select Explicit-Transient.
Initial quantities transfer	Select exchange.

- Set options which are code specific.

FINE/Open

Option	Action
Reference pressure	Set pressure to 101325 N/m ² .
Expert mode	Check and access further options.
Mesh deformation	Select Radial basis functions.

FINE/Turbo

Option	Action
Set the requested memory	Check this option because for the coupling the memory has to be increased.
Number of reals (Mb)	Set to 100.
Number of ints (Mb)	Set to 10.
Expert mode	Check and access further options.
Mesh deformation	Select Radial basis functions.
Save inverse matrix of radial basis functions	Select Compute and use inverse matrix.

FLUENT

No code specific options need to be set.

OpenFOAM

Option	Action
Select OpenFOAM solver	Leave the default Auto-Select-by-Case. The setting from the "controlDict" file will be used.
Reference pressure	Set pressure to 101325 N/m ² .
Use MpCCI grid morpher	Check to use the grid morpher. Additional options become visible.
Optional list of floating boundary regions	Enter 3 to allow sliding of nodes on boundary region channel-middle close to the flap.

Refer to [▷ VI-18.4 Grid Morphing ◁](#) for more information on the MpCCI Grid Morpher options.

-  The numbering of boundary regions corresponds to the numbering in the OpenFOAM "constant/polymesh" subdirectory of your case.

STAR-CCM+

Option	Action
Use duration control	Check and access further options.
No. of coupled time steps	Set to 800.
Coupling time step size	Set to 2.5e-4.
Maximum no. of iterations per time step	Set to 30.
Run in batch	Check this option.
Auto start with the default template macro	Check this option.
License options	Set according to your license type (cf. ▷ VI-17.2.4 Go Step ◁).

STAR-CD

Option	Action
Startup procedure	Select Prepare case and start.
Use the MpCCI plugin library	Check this option.
Use grid morpher	Check to use the grid morpher. Additional options become visible.
List of cell type ids	Enter 2 to activate the morpher only for the fluid part around the flap of cell type 2.
Optional list of floating boundary regions	Enter the boundary region numbers 14, 15, and 16 to allow nodes to slide on boundary regions close to the flap.

Refer to [▷ VI-18.4 Grid Morphing ◁](#) for more information on the MpCCI Grid Morpher options.

-  The numbering of cell types and boundary regions correspond to the numbering in the STAR-CD model file.

...

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 three **Start** buttons in the Go Step and the simulation codes should start. Some codes require additional actions:

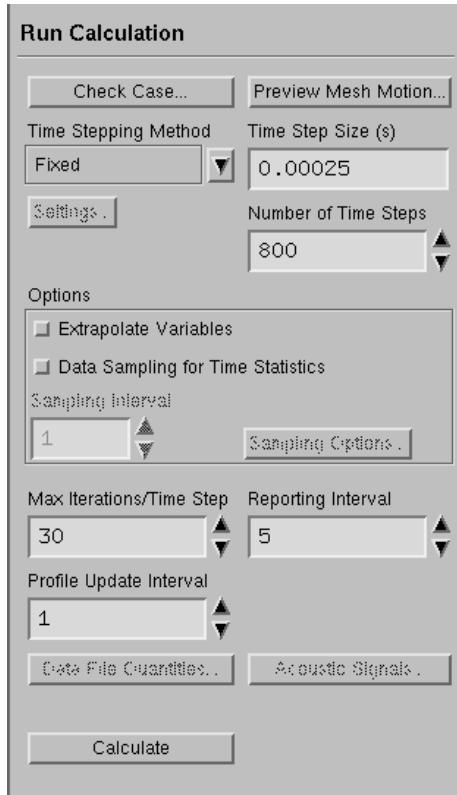
FINE/Open

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.

FLUENT



Initialize the flow field by selecting **Solve→Initialization** to open the initialization panel. And press the button **Initialize**.

Select **Solve→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. 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.

STAR-CD

STAR-CD will run for a simulation time of 0.2 s with a time step size of 0.00025 s. STAR-CD 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 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.

...

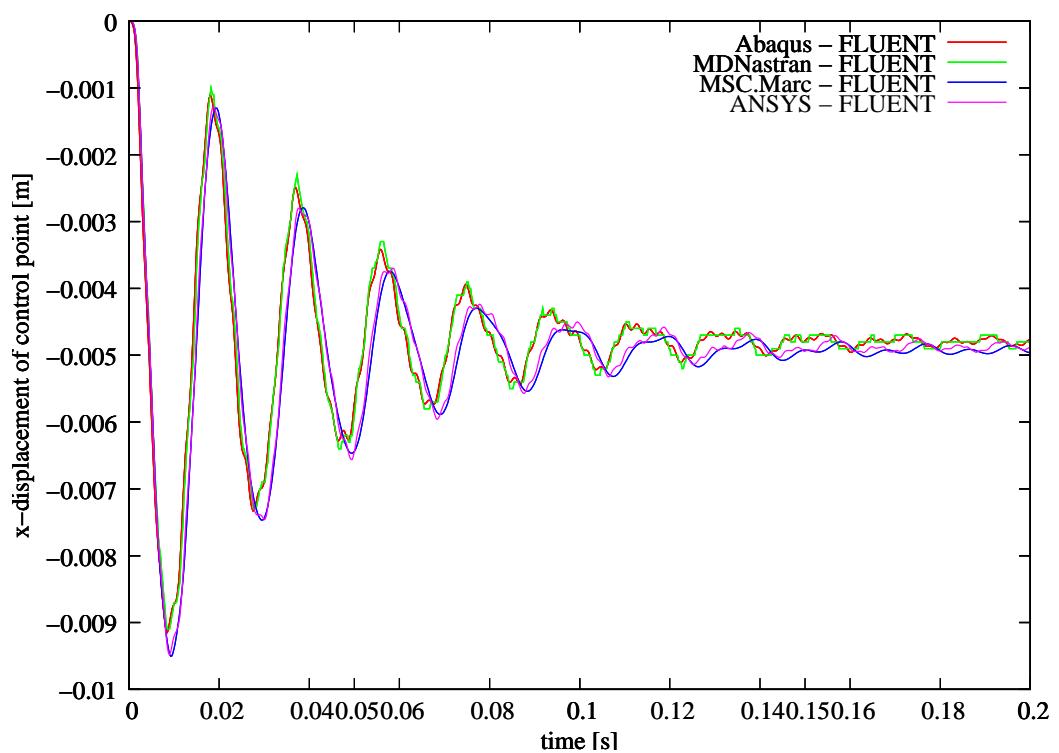


Figure 4: Elastic flap: Displacement of the control point for Abaqus, ANSYS, MSC Marc and MSC NAS-TRAN coupled with FLUENT.

5 Elastic Flap in Water

5.1 Problem Description

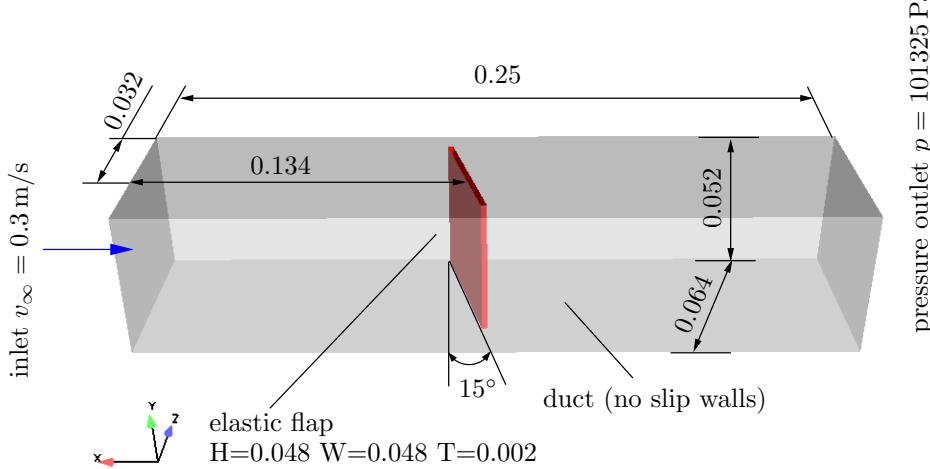


Figure 1: Elastic flap: Geometry [m] and boundary conditions

Topics of this Tutorial

- Iterative Transient Coupling
- Fluid-Structure Interaction (FSI)
- Usage of the MpCCI Grid Morpher for OpenFOAM
- 3D-model

Simulation Codes

- Solid Mechanics: Abaqus, MSC NASTRAN
- Fluid Mechanics: FLUENT, OpenFOAM

5.2 Model Preparation

The simulation couples a solid mechanics model with a fluid mechanics model. The model corresponds to the model from the preceding tutorial (cf. [▷ 4 Elastic Flap in a Duct ◁](#)). The compressible gas in the channel is replaced with water for this example. This leads to a strong coupling between the fluid mechanics and the solid model. Iterative coupling is necessary to get a stable solution.

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/ElasticFlapWater`" which correspond to the simulation codes you want to use.

5.2.1 Solid Model

The solid model is – except for a small change concerning the Young's modulus – the same as in the preceding tutorial [▷ 4 Elastic Flap in a Duct ◁](#). The solid part corresponds to a rectangle ([Figure 2](#)):

The material is linear elastic with density $\rho = 1000 \text{ kg/m}^3$, elastic modulus $E = 0.5 \times 10^8 \text{ Pa}$ and Poisson's ratio $\nu = 0.49$.

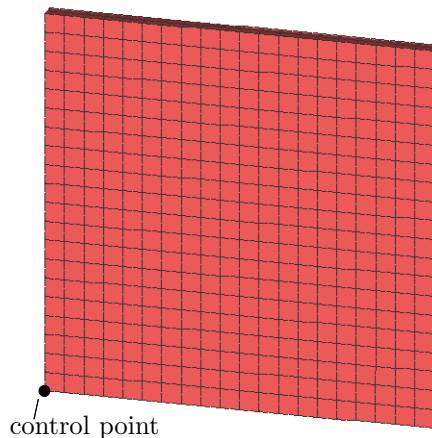


Figure 2: Elastic flap: Mesh of the structural domain

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 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 $\Delta t = 0.001\text{ s}$ and the total time is $t = 0.5\text{ s}$.

A control-point/node is selected in the solid model for evaluation of the results, see [▷ 5.5 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.

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 number of increments for each time step in MSC NASTRAN is set to 300.

...

5.2.2 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 $\Delta t = 0.001\text{ s}$, which equals the time step size of the solid code.

The fluid is assumed to be incompressible with a density of $\rho = 1000\text{ kg/m}^3$ and a dynamic viscosity of $\eta = 0.001\text{ kg/m s}$. The inlet velocity is $v = 0.3\text{ m/s}$; at the outlet atmospheric pressure is applied.

The fluid model directories contain a steady solution for the problem. When solving the transient, implicitly

coupled system, this steady solution should be used as an initial condition.

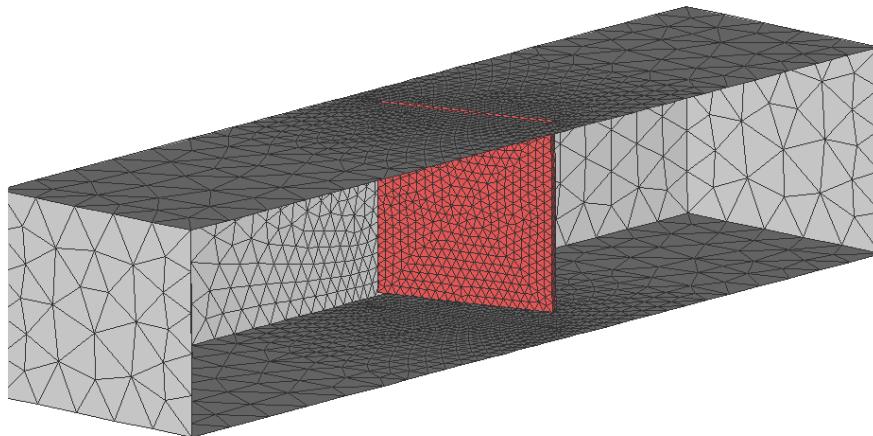


Figure 3: Elastic flap: Partial mesh of the fluid domain

FLUENT

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”.

OpenFOAM

The mesh corresponds to the mesh which is used in FLUENT. The solver “pimpleDyMFoam” – for incompressible fluids on moving meshes – is used.

The initial solution for the velocity field is saved in the respective files in the directory `0` containing the OpenFOAM initial conditions.

...

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

Option	Action
Code to couple	Select Abaqus.
Release	Should be set to latest or a version supported by MpCCI (see ▷ VI-2 Abaqus ◁).
Input file	Select input file "Abaqus/elastic_flap.inp".
Unit system	Select the SI unit system (which is the standard).
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

MSC NASTRAN

Option	Action
Code to couple	Select MSC NASTRAN.
Release	Should be set to latest or a version supported by MpCCI (see ▷ VI-13 MSC NASTRAN ◁).
Input file	Select input file "MSCNastran/elastic_flap.bdf".
Unit system	Select the SI unit system (which is the standard).
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

...

2. Choose your fluid mechanics code as second code to couple:

FLUENT

Option	Action
Code to couple	Select FLUENT.
Version	The model is three-dimensional, thus select the FLUENT version 3d.
Release	Should be set to latest or a version supported by MpCCI (see ▷ VI-8 FLUENT ◁).
Case file	Select the case file "FLUENT/elastic_flap.cas".
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

OpenFOAM

Option	Action
Code to couple	Select OpenFOAM.
Option	Select the OpenFOAM option Opt or Debug.
Precision	Select the OpenFOAM precision: DP for double or SP for single precision.
Release	Should be set to latest or a version supported by MpCCI (see ▷ VI-14 OpenFOAM ◁).
Case directory	Select the case directory "OpenFOAM/<version>" fitting your OpenFOAM version (e.g. "OpenFOAM/3.0" for use with OpenFOAM 3.0.1).
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

...

Press the **Coupling Step >** button at the bottom of the Models Step to get to the Coupling Step.

5.3.2 Coupling Step

In the Coupling Step build the coupling regions, define quantities to be exchanged and assign the defined quantities to the built regions (cf. [IV-2.5 Coupling Step – Defining Coupling Regions and Quantities](#)).

In the following the **fluid mechanics** code will be called CFD code (CFD = Computational Fluid Dynamics) and the **solid mechanics** code CSM code (CSM = Computational Structural Mechanics).

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	
For CSM codes	Select coupling component
Abaqus	ASSEMBLY_WALL_WALL-SURFACE
MSC NASTRAN	flap
For CFD codes	Select coupling component
FLUENT	couple-flap
OpenFOAM	

3. Select Define Quantity Sets tab. As currently no quantities are assigned to “QuantitySet_1” (created by default), a warning symbol is displayed in front of it. Now choose the quantities to be exchanged.

For quantity set	Select quantities	Set sender
QuantitySet_1	NPosition RelWallForce	CSM code CFD code

4. Add operators to both exchanged quantities (cf. [V-4.5.7.4 Applying Operators to Mesh Based Components](#)) to get a stable computation.

Under-relaxation as well as a convergence tolerance for the inner iterations will be needed. For this mark the wanted quantity in **QuantitySet_1** by clicking on its name and configuring the listed operators as follows:

Quantity	Operator	Operator option	Operator action
NPosition	Select Relaxation.	Relaxation method	Select Fixed.
		Relaxation factor	Set to 0.3 for under-relaxation.
RelWallForce	Select Relaxation.	Relaxation method	Select Fixed.
		Relaxation factor	Set to 0.3 for under-relaxation.
	Select ConvergenceCheck.	Tolerance value	Set to 0.001 which is the default value.

The relaxation operator is a post processor which will be applied to the receiver of the quantity. The convergence check operator is a pre processor which will be applied to the sender of the quantity.

5. Select Assign Quantity Sets tab and assign the defined quantity set to the built region.

For selected region	Check quantity set
Region_1	QuantitySet_1

Proceed to the Monitors Step by pressing **[Monitors Step >]**.

5.3.3 Monitors Step

No changes are required in the Monitors Step. Proceed to the Edit Step by pressing **[Edit Step >]**.

5.3.4 Edit Step

In the Edit Step the plots of the relative variation for the inner iterations which are used to check the convergence have to be activated. To achieve this:

Parameter	Value
Properties.Monitor.Inorm	Set selected.

Proceed to the Go Step by pressing **[Go Step >]**.

5.3.5 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 Start-Up and Starting Server and Codes](#)).

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:

- Set Coupling configuration options which are the same for all solid mechanics codes.

Option	Action
Coupling scheme	Select Implicit-Transient.
Initial quantities transfer	Select receive.

- Set options which are code specific.

Abaqus

Option	Action
Iteration control	Expand and access further options.
Maximum number of coupling iterations	Set to 10.
Coupling time step size	Set to 0.001 s.

MSC NASTRAN

Option	Action
Iteration control	Expand and access further options.
Number of iterations without coupling	Set to 30.
Maximum number of coupling iterations	Set to 10.
Coupling time step size	Set to 0.001 s.

...

2. For your fluid mechanics code:

- Set Coupling configuration options which are the same for all fluid mechanics codes.

Option	Action
Coupling scheme	Select Implicit-Transient.
Initial quantities transfer	Select exchange.

- Set options which are code specific.

FLUENT

Option	Action
Iteration control	Expand and access further options.
Number of iterations without coupling	Set to 20.
Maximum number of coupling iterations	Set to 10.
Coupling time step size	Set to 0.001 s.
Data file (optional)	Choose the data file "FLUENT/init.dat" containing the initial steady-state solution.

OpenFOAM	
Option	Action
Iteration control	Expand and access further options.
Maximum number of coupling iterations	Set to 10.
Coupling time step size	Set to 0.001 s.
Select OpenFOAM solver	Leave the default Auto-Select-by-Case.
Reference pressure	Set pressure to 101325 N/m ² .
Reference density	Set density to 1000 kg/m ³ .
Use MpCCI grid morpher	Check to use the grid morpher. Further options will become visible.
Morpher options file	In your case directory "OpenFOAM/<version>/" select file "morpherOptions.gmo" which contains the morpher options.

Refer to [▷ V-7 MpCCI Grid Morpher ↳](#) for more information on the MpCCI Grid Morpher options.

 The numbers of boundary regions correspond to the numbering in the OpenFOAM "constant/polymesh" subdirectory of your case.

...

5.4 Running the Computation

Save the MpCCI project file with name "elastic_flap_water.csp" over the MpCCI GUI menu **File→Save Project As...**.

Press the three **Start** buttons in the Go Step and the simulation codes should start. Some codes require additional actions:

FLUENT

Select **Solve→Run Calculation...** to open the Run Calculation panel.

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 200 iterations per time step are needed – for 20 iterations between the data exchanges and at most 10 coupling steps per time step defined in MpCCI GUI.

Select **MpCCI →MpCCI Run FSI...** to open the MpCCI Run FSI panel.

As the time step size is fixed to 0.001 s, you should select 500 **Number of Time Steps**, to cover a simulation time of 0.5 s at least.

Finally press **Run** to start the simulation. No initialization is required because FLUENT starts with an initial solution.

During the simulation, FLUENT will display the residuals and a section through the flap, where you can see its deformation.

...

5.5 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:

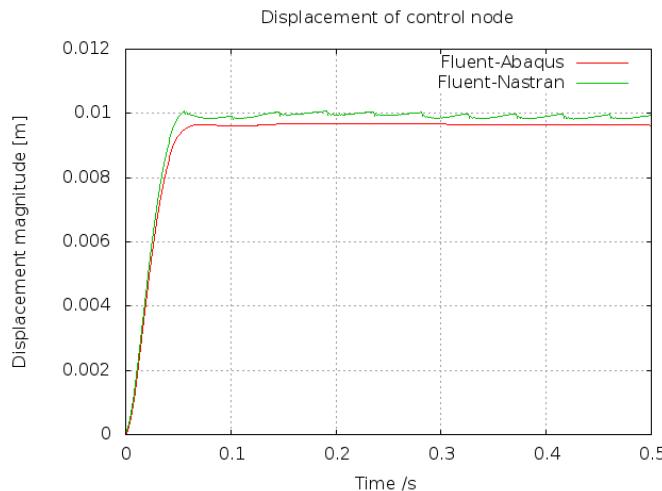


Figure 4: Displacement magnitude of NODE 1 for MSC NASTRAN and control-node for Abaqus.

Abaqus

In Abaqus from the odb file you can get the displacement magnitude for the node-set: control-node.

MSC NASTRAN

In SimXpert or PATRAN attach the results file "mpcci_elastic_flap.xdb". You can create a chart for plotting the displacement magnitude for node 1 for example.

...

[Figure 5](#) shows the convergence of the inner coupling iterations inside the time step. The relative wall force – sent from OpenFOAM to MSC NASTRAN in this example – converges quite nicely after a few inner iterations. Unfortunately, the position sent by MSC NASTRAN does not show this good convergence behavior.

To get faster computation times the convergence tolerance values in the Coupling Step can be adjusted to a value identified in the plots of the relative variation in the MpCCI Visualizer.

The solution for this coupled problem cannot be obtained using an explicit coupling algorithm. To start the explicitly coupled simulation just change the coupling scheme setting from “Implicit-Transient” to “Explicit-Transient”. Ensure that the time step sizes do match (0.001 s), as MpCCI will no longer patch the time step size in the respective input files.

The explicit solution will diverge after some coupling steps. The overestimation of the displacement of the CSM model leads to extremely high pressure values on the CFD side, which again leads to big displacements.

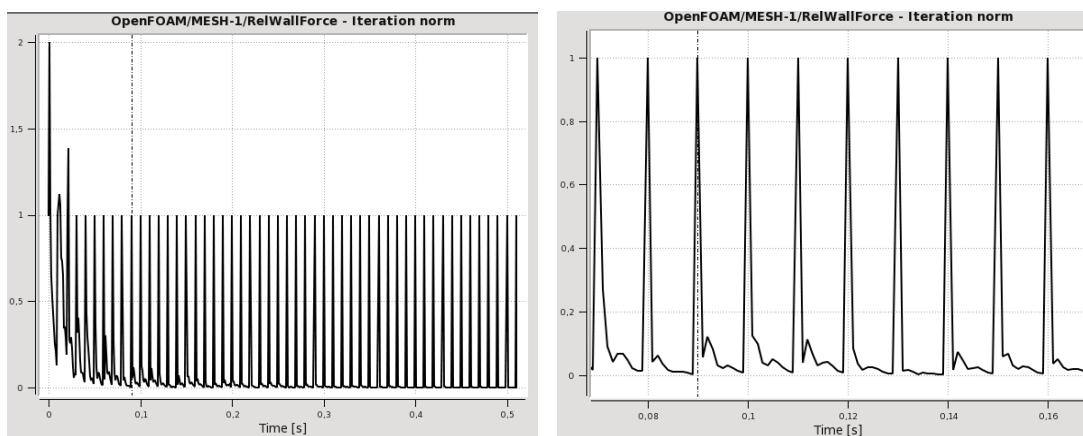


Figure 5: Convergence of inner iterations for the quantity RelativeWallForce.

6 Exhaust Manifold

6.1 Problem Description

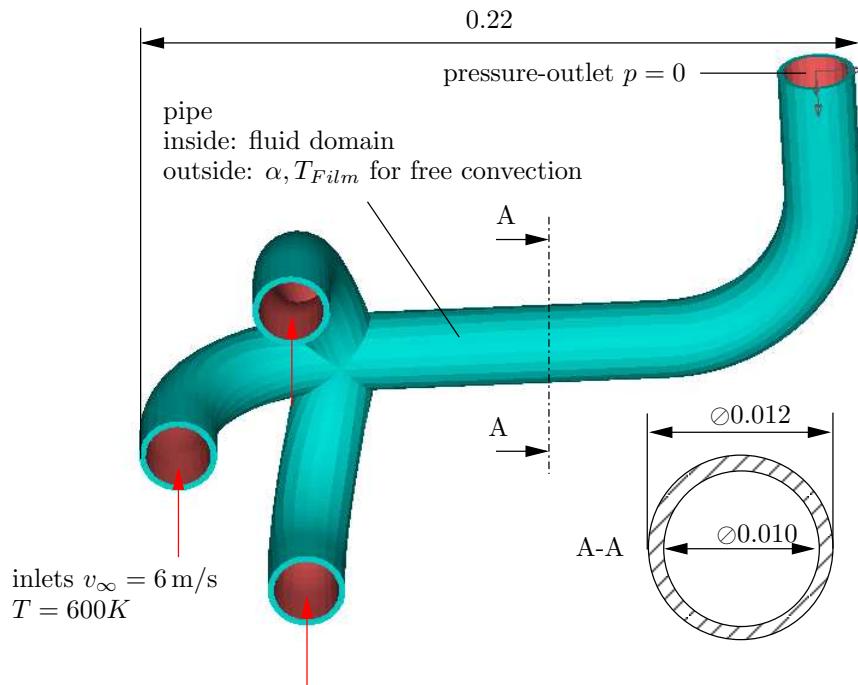


Figure 1: Manifold: Geometry [m] and boundary conditions

Topics of this Tutorial

- Thermal coupling fluid and solid
- Steady state
- Incompressible fluid
- Fluid solver iterations without coupling
- Fluid solver subcycling
- 3D-model

Simulation Codes

- Solid Mechanics: Abaqus, ANSYS
- Fluid Mechanics: FLUENT, OpenFOAM, STAR-CCM+, STAR-CD

6.2 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.

6.2.1 Solid Model

The solid part of the manifold has the following properties ([Figure 2](#)):

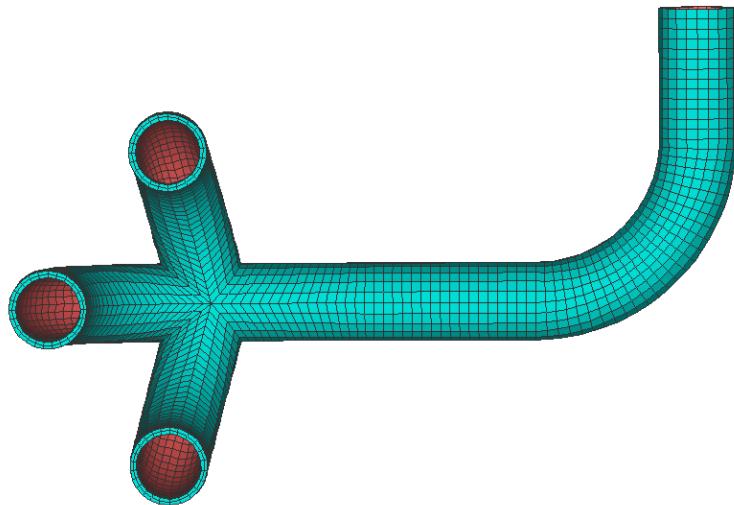


Figure 2: Manifold: Mesh of the solid domain

Conductivity	$55.0 \frac{W}{m \cdot K}$
Film temperature (outside)	$300.0 K$
Heat coefficient (outside)	$50.0 \frac{W}{m^2 \cdot K}$

Table 1: Manifold: Boundary conditions for solid model

The mesh consists of 15244 8-node brick elements.

Abaqus

In Abaqus DC3D8 elements are used.

ANSYS

In ANSYS D3-SOLID70 elements are used.

Additional D3-SHELL57 dummy surface elements are defined for quantity exchange as described in [▷ VI-3.2.1 Model Preparation](#).

...

6.2.2 Fluid Model

The fluid domain comprises the inner part of the pipe ([Figure 3](#)).

The fluid mesh consists of 102236 hexahedral elements.

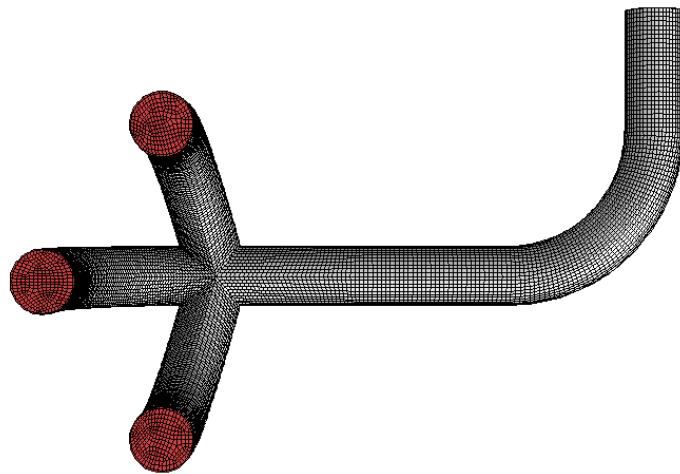


Figure 3: Manifold: Mesh of the fluid domain

Density	$1.225 \frac{kg}{m^3}$
Viscosity	$1.7894E - 05 \frac{kg}{m \cdot s}$
Conductivity	$0.0242 \frac{W}{m \cdot K}$
Specific heat	$1006.43 \frac{J}{kg \cdot K}$
Inlet	
Velocity	$6.0 \frac{m}{s}$
Temperature	$600.0 K$
Turbulent Intensity	0.03
Turbulent Length	$0.003 m$
Outlet	
Relative pressure	0 Pa
Turbulent Intensity	0.03
Turbulent Length	$0.003 m$

Table 2: Manifold: Boundary conditions for fluid model

FLUENT

A standard $k - \epsilon$ model with enhanced wall treatment is applied.

OpenFOAM

A standard $k - \epsilon$ model with standard wall treatment is applied.

STAR-CCM+

A standard $k - \epsilon$ model with enhanced wall treatment is applied.

STAR-CD

A low-Reynolds $k - \epsilon$ model with hybrid wall is applied.

...

6.2.3 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 check mode in the Go 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".

STAR-CD

Start STAR-CD and read the model file "STARCD/exhaust_manifold.mdl". Carry out a flow simulation with 200 iterations. The result is saved in "STARCD/exhaust_manifold.[pst|ccmp]" or depending on the STAR-CD version.

...

6.2.4 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.

STAR-CD

1. Start pro-STAR and resume the model file "exhaust_manifold.mdl".
2. In the STAR GULde go to **Analysis Preparation/Running**→**Set Run Time Controls**. Set Number of iterations to "200".
3. Change to **Analysis Preparation/Running**→**Analysis (Re)Start**. Set Initial Field Restart with options Restart (New Boundary Values) and Continue.
4. Choose the uncoupled result file "exhaust_manifold.[pst|ccmp]" as Restart File depending on the STAR-CD version.
5. Save the model "exhaust_manifold.mdl" and exit pro-STAR.

 If you get such message during the start of STAR-CD:

```
PNP: ***ERROR*** The required "mpCCI" plug-in is not available in the STAR installation.
PNP: ==> Please check that the plug-in has been installed properly.
```

Execute the following procedure:

1. Start pro-STAR and resume the model file "exhaust_manifold.mdl".
2. Enter in the command input field: `plugin mpCCI off`
3. Save the model "exhaust_manifold.mdl" and problem file and exit pro-STAR.

...

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 solid mechanics code as first code to couple:

Abaqus	
Option	Action
Code to couple	Select Abaqus.
Release	Should be set to latest or a version supported by MpCCI (see VI-2 Abaqus).
Input file	Select input file "Abaqus/exhaust_manifold.inp".
Unit system	Select the SI unit system (which is the standard).
<code>Start scanner</code>	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

ANSYS	
Option	Action
Code to couple	Select ANSYS.
Release	Should be set to latest or a version supported by MpCCI (see VI-3 ANSYS).
Product	You can select any ANSYS product which can be used for structural analysis, e.g. <code>ansys</code> .
Database file	Select database file "ANSYS/manifold.db".
Unit system	Select the SI unit system (which is the standard).
<code>Start scanner</code>	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

...

2. Choose your fluid mechanics code as second code to couple:

FLUENT

Option	Action
Code to couple	Select FLUENT.
Version	The model is three-dimensional, thus select the FLUENT version 3ddp.
Release	Should be set to latest or a version supported by MpCCI (see ▶ VI-8 FLUENT ◁).
Case file	Select the case file "FLUENT/exhaust_manifold.cas".
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

OpenFOAM

Option	Action
Code to couple	Select OpenFOAM.
Option	Select the OpenFOAM option Opt or Debug.
Precision	Select the OpenFOAM precision: DP for double or SP for single precision.
Release	Should be set to latest or a version supported by MpCCI (see ▶ VI-14 OpenFOAM ◁).
Case directory	Select the case directory "OpenFOAM/<version>" fitting your OpenFOAM version (e.g. "OpenFOAM/3.0" for use with OpenFOAM 3.0.1).
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

STAR-CCM+

Option	Action
Code to couple	Select STAR-CCM+.
Release	Should be set to latest or a version supported by MpCCI (see ▶ VI-17 STAR-CCM+ ◁).
Simulation file	Select the simulation file "STAR-CCM+/exhaust_manifold.sim".
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

STAR-CD

Option	Action
Code to couple	Select STAR-CD.
Release	Should be set to latest or a version supported by MpCCI (see ▶ VI-18 STAR-CD ◁).
Model file	Select the model file "STARCD/exhaust_manifold.mdl".
License options	Set according to your license type (cf. ▶ VI-18.2.2 Models Step ◁).
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

...

Press the **Coupling Step >** button at the bottom of the Models Step to get to the Coupling Step.

6.3.2 Coupling Step

In the Coupling Step build the coupling regions, define quantities to be exchanged and assign the defined quantities to the built regions (cf. [IV-2.5 Coupling 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	
For solid mechanics code	Select coupling component
Abaqus	ASSEMBLY_MANIFOLD_INNER_SURFACE
ANSYS	INNER_SURFACE
For fluid mechanics code	Select coupling component
FLUENT	
OpenFOAM	
STAR-CCM+	wall
STAR-CD	

3. Select Define Quantity Sets tab. As currently no quantities are assigned to “QuantitySet_1” (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. [V-4.5.8 Assigning Preconfigured Quantity Settings](#)):

For quantity set	Option	Action
QuantitySet_1	Select Coupling Type	Select Steady state surface heat transfer and click Set .

Now the quantities FilmTemp, WallHTCoeff and WallTemp are activated and MpCCI already set the appropriate sender for them.

For co-simulation with Abaqus replace the quantity WallHTCoeff by HeatRate for the conjugated heat transfer analysis.

For coupling with Abaqus	
Quantity	Action
WallHTCoeff	Deselect this quantity.
HeatRate	Select this quantity. Set fluid mechanics code as sender.

4. Select Assign Quantity Sets tab and assign the defined quantity set to the built region.

For selected region	Check quantity set
Region_1	QuantitySet_1

Proceed to the Monitors Step by pressing **Monitors Step >**.

6.3.3 Monitors Step

No changes are required in the Monitors Step. Proceed to the Edit Step by pressing [Edit Step >].

6.3.4 Edit Step

No changes are required in the Edit Step. Proceed to the Go Step by pressing [Go Step >].

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 Start-Up and Starting Server and Codes](#)).

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:

- Set Coupling configuration options which are the same for all solid mechanics codes.

Option	Action
Coupling scheme	Select Explicit-SteadyState.
Initial quantities transfer	Select receive.

- Set options which are code specific.

Abaqus	
Option	Action
Constant coupling time step	Check this option to use a constant time step.
Coupling time step	Set to 1 for one increment size for the steady state analysis.
Use subcycling	Deselect this option as there is no subcycling of Abaqus included.

MpCCI will run the simulation not on the original model file, but rather will use a copy with the prefix "abaqus_run".

When starting the simulation, Abaqus will receive data from the fluid mechanics code and accomplish a steady state heat transfer analysis. After the analysis data is exchanged with the partner code and another analysis is started.

ANSYS	
Option	Action
Gui option	Keep -b to start ANSYS in batch mode.
APDL input script	Select the input script "ANSYS/startjob.ans".

...

2. For your fluid mechanics code:

- Set Coupling configuration options which are the same for all fluid mechanics codes.

Option	Action
Coupling scheme	Select Explicit-SteadyState.
Initial quantities transfer	Select send.

- Set options which are code specific.

FLUENT

Option	Action
Use subcycling	Check and access further options.
No. of steps without coupling	Set to 10.
Auto set MDM zones	Deselect.

OpenFOAM

Option	Action
Use subcycling	Check and access further options.
No. of steps without coupling	Set to 10.
Use duration control	Check and access further options.
Iteration no. for starting the coupling	Set to 200 to activate a pre-calculation of 200 steps.
Reference density	Set density to 1.225 kg/m ³ .
Reference specific heat	Set specific heat to 1006.43 J/kgK.

STAR-CCM+

Option	Action
Use subcycling	Check and access further options.
No. of steps without coupling	Set to 10.
Use duration control	Check and access further options.
Iteration no. for ending the coupling	Set to 200.
Run in batch	Check this option.
Auto start with the default template macro	Check this option.
License options	Set according to your license type (cf. VI-17.2.4 Go Step ▶).

STAR-CD	
Option	Action
Use subcycling	Check and access further options.
No. of steps without coupling	Set to 10.
Startup procedure	Select Prepare case and start to accomplish the necessary modifications in the STAR-CD model for a coupled simulation.
Backup model file	Click this button. As MpCCI will build a new model file, the original model file should be saved.
Use the MpCCI plugin library	Check this option to activate MpCCI plug-in in STAR-CD for data transfer.
Double precision mode	Check this option.

...

6.4 Running the Computation

Save the MpCCI project file with name "exhaust_manifold.csp" via the MpCCI GUI menu **File→Save Project As**.

Press the three **Start** buttons 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→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.

STAR-CD

In order to survey the convergence of the simulation, STAR-CD will write the integral total surface heat flux through the coupling surface on "exhaust_manifold.erd". STAR-CD will run for 200 iterations.

...

6.5 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-2.2.5 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".

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".

STAR-CD

STAR-CD result file "exhaust_manifold.pst" or "exhaust_manifold.ccmp".

...

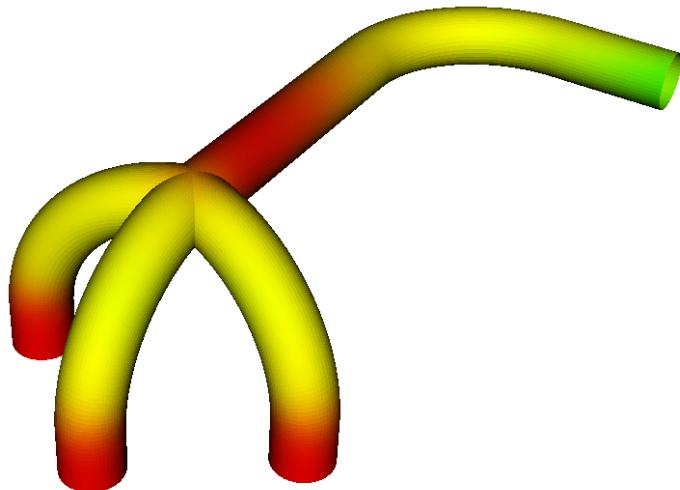


Figure 4: Manifold: Temperature distribution on coupled surface

7 Busbar System

7.1 Problem Description

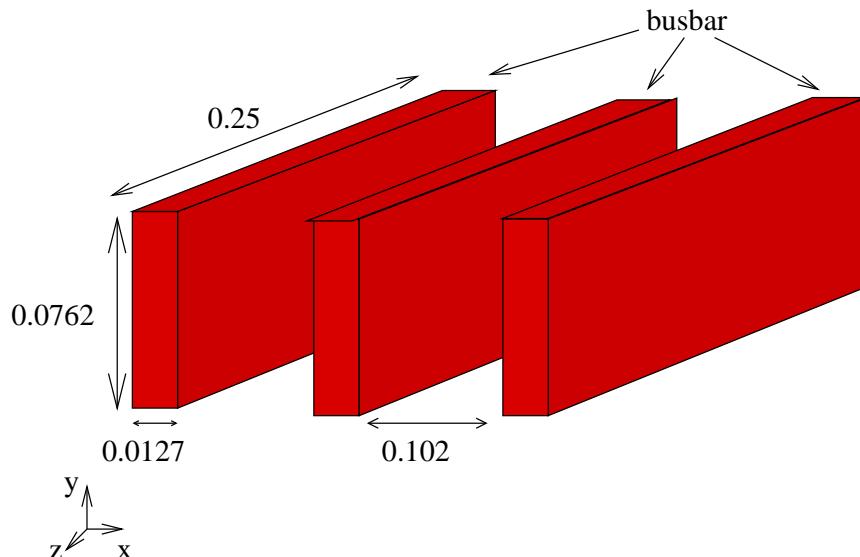


Figure 1: Busbar: Geometry [m].

Topics of this Tutorial

- Magneto-Thermal
- 3D-model
- Volume coupling
- Several coupling regions
- Fluid solver iterations without coupling

Simulation Codes

- Fluid Mechanics: FLUENT, STAR-CD
- 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,
- proximity effect,

- 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 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.

7.2.1 Fluid Model

Both the STAR-CD and FLUENT models are based on the same grid with boundary conditions as similar as possible. 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.

STAR-CD

No modifications have to be done, as in STAR-CD 4.06 and higher MpCCI is treated as plug-in and no user subroutines are needed.

...

7.2.2 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 $\varrho_0 = 1.7241 \cdot 10^{-8}$ Ωm . The temperature coefficient is $\alpha = 0.004 \text{ K}^{-1}$. 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:

- `~mpcci, init, 3D`: Initialize the connection to MpCCI.
- `~mpcci, exchange`: Exchange data from the partner code and wait until the data exchange is done.
- `~mpcci, settag, time, iter`: Set the synchronization tag for the current time / iteration.

Furthermore after the frequency domain solution, the effective power loss density is provided via ANSYS command `powerh` which generates an element table named `plossd`. This element table is just copied to an element table named `mpcci_00` using this command: `smult, mpcci_00, plossd,, 1`. Because the MpCCI adapter can read out of element tables with fulfilling the naming convention `mpcci_<index>`. The `<index>` has to be set in the MpCCI GUI.

JMAG

All necessary model information is prepared, no further action is needed. An electric circuit is defined to set the current loads for the finite element solution. The three-phase current source provides an current amplitude of 2262.7 A.

...

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 `mpcci gui`.

7.3.1 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:

ANSYS

Option	Action
Code to couple	Select ANSYS.
Release	Should be set to latest or a version supported by MpCCI (see VI-3 ANSYS).
Product	You can select any ANSYS product which can be used for electromagnetic analysis, e. g. <code>ane3fl</code> or <code>emag</code> .
Database file	Select database file "ANSYS/busbar.db".
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

JMAG

Option	Action
Code to couple	Select JMAG.
Release	Should be set to latest or a version supported by MpCCI (see VI-9 JMAG).
JCF file	Select file "JMAG/busbar_system.jcf" as input file.
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

...

2. Choose your CFD code (CFD = Computational Fluid Dynamics) as second code to couple:

FLUENT

Option	Action
Code to couple	Select FLUENT.
Version	The model is three-dimensional, thus select the FLUENT version 3ddp.
Release	Should be set to latest or a version supported by MpCCI (see VI-8 FLUENT).
Case file	Select the case file "FLUENT/busbar.cas".
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

STAR-CD

Option	Action
Code to couple	Select STAR-CD.
Release	Should be set to latest or a version supported by MpCCI (see VI-18 STAR-CD).
Model file	Select the model file "STARCD/busbar.mdl".
License options	Set according to your license type (cf. VI-18.2.2 Models Step).
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

...

Press the **Coupling Step >** button at the bottom of the Models Step to get to the Coupling Step.

7.3.2 Coupling Step

In the Coupling Step build the coupling regions, define quantities to be exchanged and assign the defined quantities to the built regions (cf. [IV-2.5 Coupling Step – Defining Coupling Regions and Quantities](#)).

- At first select the Mesh tab to get access to the mesh based element components.
- 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.

For region Region_1	
For EMAG code	Select coupling component
ANSYS	<input checked="" type="checkbox"/> phase-a
JMAG	<input checked="" type="checkbox"/> busbar a
For CFD code	Select coupling component
FLUENT	<input checked="" type="checkbox"/> phase_a-solid
STAR-CD	<input checked="" type="checkbox"/> phase-a

- Click on button **Add** to add new region Region_2.

For region Region_2	
For EMAG code	Select coupling component
ANSYS	<input checked="" type="checkbox"/> phase-b
JMAG	<input checked="" type="checkbox"/> busbar b
For CFD code	Select coupling component
FLUENT	<input checked="" type="checkbox"/> phase_b-solid
STAR-CD	<input checked="" type="checkbox"/> phase-b

- Click on button **Add** to add new region Region_3.

For region Region_3	
For EMAG code	Select coupling component
ANSYS	<input checked="" type="checkbox"/> phase-c
JMAG	<input checked="" type="checkbox"/> busbar c
For CFD code	Select coupling component
FLUENT	<input checked="" type="checkbox"/> phase_c-solid
STAR-CD	<input checked="" type="checkbox"/> phase-c

3. Select Define Quantity Sets tab. As currently no quantities are assigned to “QuantitySet_1” (created by default), a warning symbol is displayed in front of it. Now choose the quantities to be exchanged.

For quantity set QuantitySet_1		
Select quantities	Make quantity settings	
	Option	Action
JouleHeat		
Set sender	Select EMAG code.	
Code specific settings		
ANSYS		
Send method	Select ETAB.	
index	Set to 0 for send method ETAB.	
Location	Select elem.	
JMAG		
Extrapolate orphans	Select as Orphans settings option.	
STAR-CD		
index	Set to 1 for receive method SCALAR.	
Temperature		
Set sender	Select CFD code.	
Code specific settings		
ANSYS		
Location	Select node.	
JMAG		
Orphans settings	Select Extrapolate to extrapolate values into the orphaned regions where this is possible.	

4. Select Assign Quantity Sets tab and assign the defined quantity set to the built regions.

For selected regions	Check quantity set
Region_1	
Region_2	QuantitySet_1
Region_3	

Proceed to the Monitors Step by pressing **Monitors Step >**.

7.3.3 Monitors Step

No changes are required in the Monitors Step. Proceed to the Edit Step by pressing **Edit Step >**.

7.3.4 Edit Step

JMAG

The JMAG model presents some light differences.

For visualizing the orphan information set the following parameter:

Parameter	Value
Properties.Monitor.Orphans	Set selected.

...

No further changes are required in the Edit Step. Proceed to the Go Step by pressing **[Go Step >]**.

7.3.5 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 Start-Up and Starting Server and Codes](#)).

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:

- Set Coupling configuration options which are the same for all EMAG codes.

Option	Action
Coupling scheme	Select Explicit-SteadyState.
Initial quantities transfer	Select receive.

- Set options which are code specific.

ANSYS	
Option	Action
Gui option	Keep -b to start ANSYS in batch mode.
APDL input script	Select the input script "ANSYS/startjob.ans".

JMAG	
Option	Action
Number of analysis steps	Set to 10 for the frequency response analysis simulation.

...

2. For your fluid mechanics code:

- Set Coupling configuration options which are the same for all CFD codes.

Option	Action
Coupling scheme	Select Explicit-SteadyState.
Initial quantities transfer	Select exchange.

- Set options which are code specific.

FLUENT

Option	Action
Use subcycling	Check and access further options.
No. of steps without coupling	Set to 20.

STAR-CD

Option	Action
Use subcycling	Check and access further options.
No. of steps without coupling	Set to 20.
Startup procedure	Select Prepare case and start to accomplish the necessary modifications in the STAR-CD model for a coupled simulation.
Backup model file	Click this button. As MpCCI will build a new model file, the original model file should be saved.
Use the MpCCI plugin library	Check this option to activate MpCCI plug-in in STAR-CD for data transfer.

...

7.4 Running the Computation

Press the **Start** button of the server and save the project as "busbar.csp". Now start the codes by pressing their **Start** buttons and the simulation codes should start. Some codes require additional actions:

ANSYS

ANSYS reads the APDL script file "startjob.ans" and starts the computation.

JMAG

The JMAG reads the model file "busbar_system.jcf" and starts the computation.

FLUENT

1. Open **Solve→Initialization**.
2. Press **Initialize**.
3. Open **Solve→Run Calculation** and enter 200 number of iterations.
4. Press **Calculate**.

STAR-CD

The STAR-CD job will be prepared and started. The data transfer is handled by the MpCCI plug-in calls in STAR-CD.

...

7.5 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](#)).

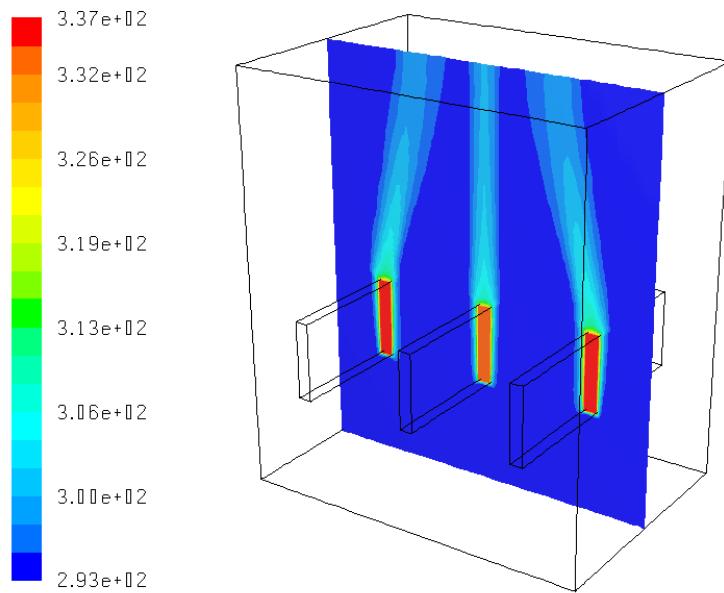


Figure 2: FLUENT results: temperature distribution [K]

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](#)).

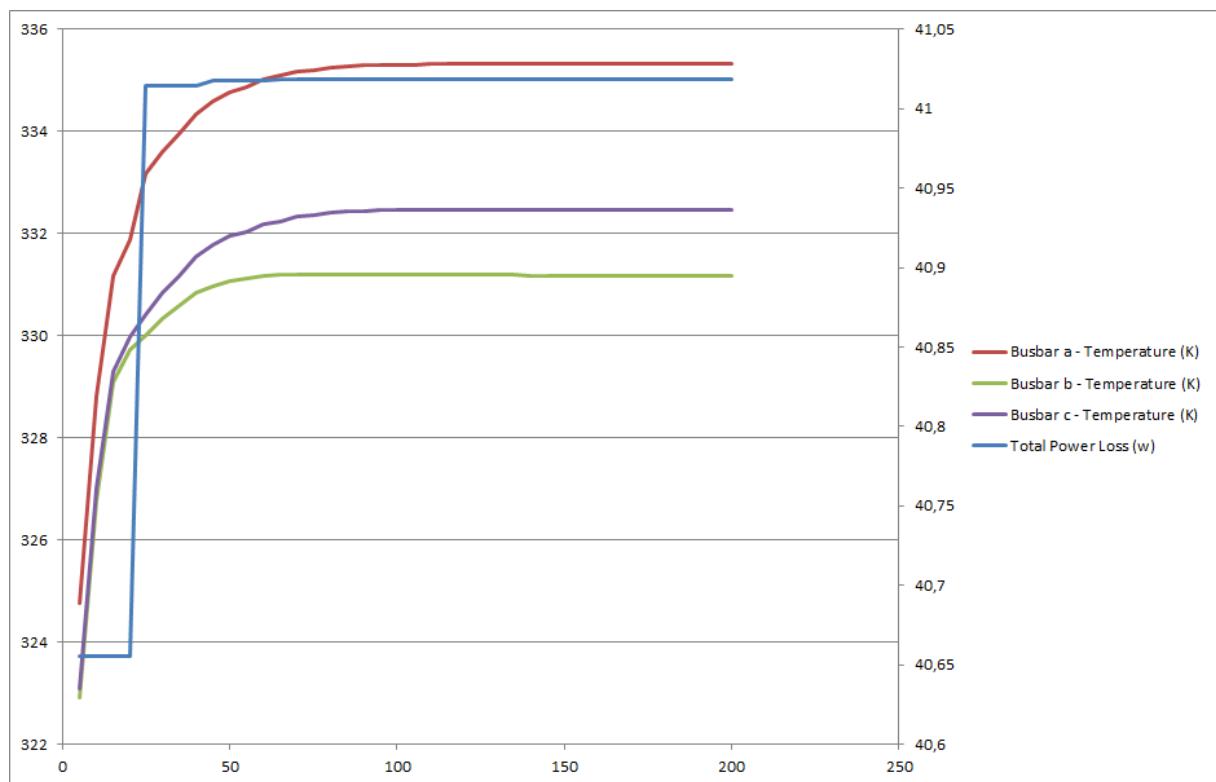


Figure 3: JMAG FLUENT results: temperature average [K] and total power losses [W]

8 Three Phase Transformer

8.1 Problem Description

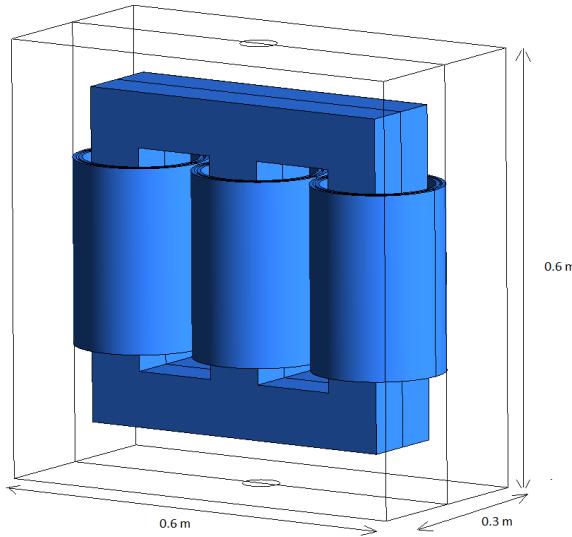


Figure 1: Three phase transformer: Geometry [m].

Topics of this Tutorial

- Magneto-Termal
- 3D-model
- Volume coupling
- Several coupling regions
- Fluid solver iterations without coupling

Simulation Codes

- Fluid Mechanics: FLUENT
- Electromagnetism: JMAG

Description

The transformer geometry model is taken from JMAG 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 $0.031 \Omega/\text{phase}$ and $0.00156 \Omega/\text{phase}$ for secondary winding.
- External load resistance is $0.06 \Omega/\text{phase}$.
- 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.

8.2 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.

8.2.1 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.

...

8.2.2 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 $\varrho_0 = 1.7241 \cdot 10^{-8}$ Ωm . The temperature coefficient is $\alpha = 0.004 \text{ K}^{-1}$. 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:

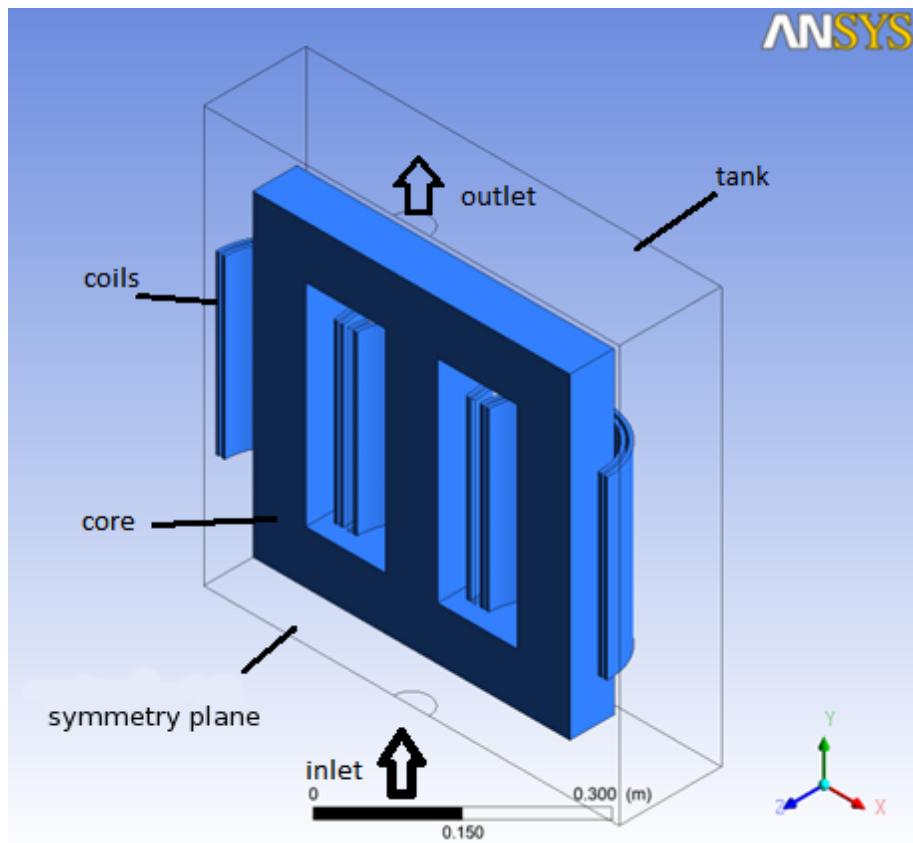


Figure 2: FLUENT boundaries

- 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.

...

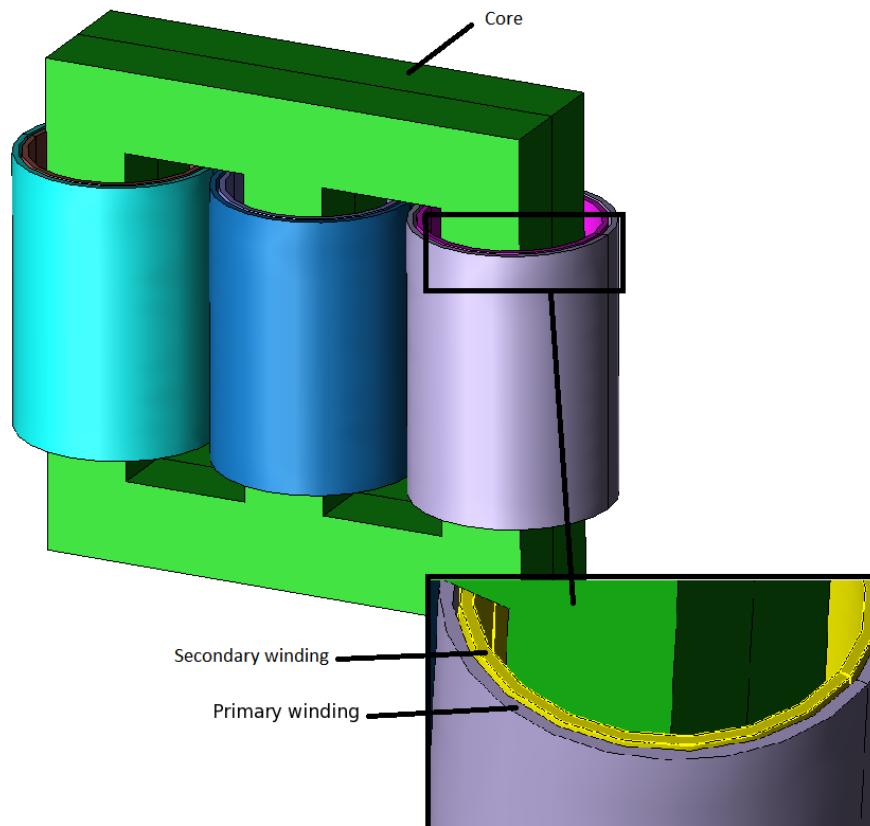


Figure 3: JMAG components

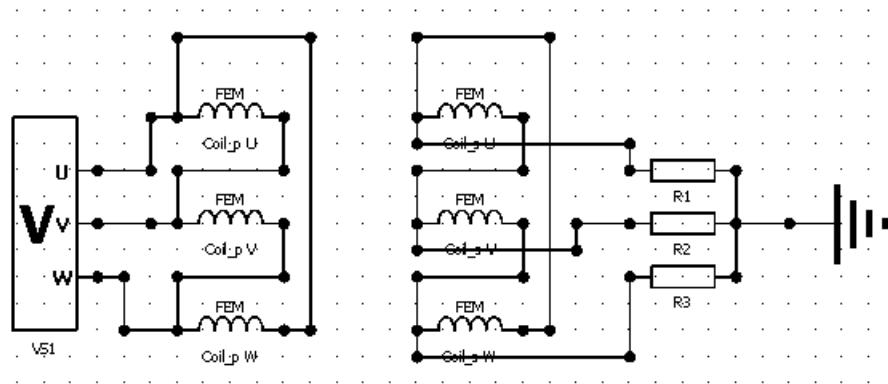


Figure 4: JMAG electric circuit

8.3 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`.

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 EMAG code (EMAG = electromagnetic) as first code to couple:

JMAG	
Option	Action
Code to couple	Select JMAG.
Release	Should be set to latest or a version supported by MpCCI (see VI-9 JMAG).
JCF file	Select file "JMAG/transformer.jcf" as input file.
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

...

2. Choose your CFD code (CFD = Computational Fluid Dynamics) as second code to couple:

FLUENT	
Option	Action
Code to couple	Select FLUENT.
Version	The model is three-dimensional, thus select the FLUENT version 3ddp.
Release	Should be set to latest or a version supported by MpCCI (see VI-8 FLUENT).
Case file	Select the case file "FLUENT/transformer.cas.gz".
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

...

Press the **Coupling Step >** button at the bottom of the Models Step to get to the Coupling Step.

8.3.2 Coupling Step

In the Coupling Step build the coupling regions, define quantities to be exchanged and assign the defined quantities to the built regions (cf. [IV-2.5 Coupling 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 () 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:

	JMAG	FLUENT	Region name
Primary windings	<input checked="" type="checkbox"/> Coil 1_U	<input checked="" type="checkbox"/> p_u	Region_1
	<input checked="" type="checkbox"/> Coil 1_V	<input checked="" type="checkbox"/> p_v	Region_2
	<input checked="" type="checkbox"/> Coil 1_W	<input checked="" type="checkbox"/> p_w	Region_3
Secondary windings	<input checked="" type="checkbox"/> Coil 2_U	<input checked="" type="checkbox"/> s_u	Region_4
	<input checked="" type="checkbox"/> Coil 2_V	<input checked="" type="checkbox"/> s_v	Region_5
	<input checked="" type="checkbox"/> Coil 2_W	<input checked="" type="checkbox"/> s_w	Region_6
Core	<input checked="" type="checkbox"/> core	<input checked="" type="checkbox"/> core	Region_7

- 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 “QuantitySet_1” (created by default), a warning symbol is displayed in front of it. Now choose the quantities to be exchanged.

For quantity set	Select quantities	Set sender
QuantitySet_1	JouleHeat	EMAG code
	Temperature	CFD code

4. Select Assign Quantity Sets tab and assign the defined quantity set to the built regions.

For selected regions	Check quantity set
<all regions> e.g.	
p_u	
p_v	
p_w	
s_u	QuantitySet_1
s_v	
s_w	
core	

Proceed to the Monitors Step by pressing **Monitors Step >**.

8.3.3 Monitors Step

No changes are required in the Monitors Step. Proceed to the Edit Step by pressing **Edit Step >**.

8.3.4 Edit Step

No changes are required in the Edit Step. Proceed to the Go Step by pressing [Go Step >].

8.3.5 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 Start-Up and Starting Server and Codes](#)).

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:

- Set Coupling configuration options which are the same for all EMAG codes.

Option	Action
Coupling scheme	Select Explicit-SteadyState.
Initial quantities transfer	Select receive.

- Set options which are code specific.

JMAG

Option	Action
Number of analysis steps	Set to 20 for the frequency response analysis simulation.

...

2. For your fluid mechanics code:

- Set Coupling configuration options which are the same for all CFD codes.

Option	Action
Coupling scheme	Select Explicit-SteadyState.
Initial quantities transfer	Select exchange.

- Set options which are code specific.

FLUENT

Option	Action
Use subcycling	Check and access further options.
No. of steps without coupling	Set to 20.

...

8.4 Running the Computation

Press the **Start** button of the server and save the project as "transformer.csp". Now start the codes by pressing their **Start** buttons and the simulation codes should start. Some codes require additional actions:

JMAG

JMAG reads the model file "transformer.jcf" and starts the computation automatically.

FLUENT

1. Open **Solve→Initialization**.
2. Press **Initialize**.
3. Open **Solve→Run Calculation** and enter 400 number of iterations.
4. Press **Calculate**.

...

8.5 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](#)).

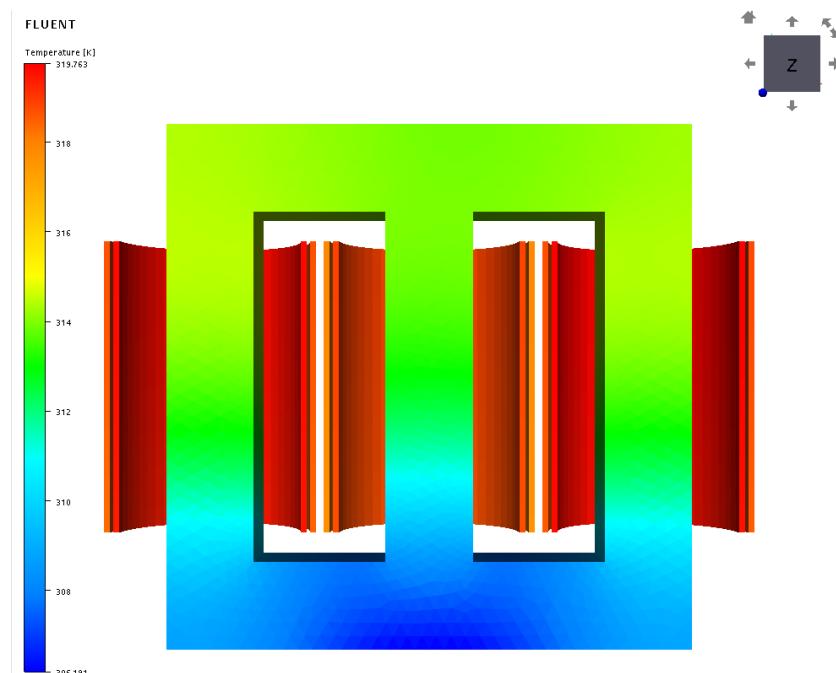


Figure 5: FLUENT results: temperature distribution [K]]

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.

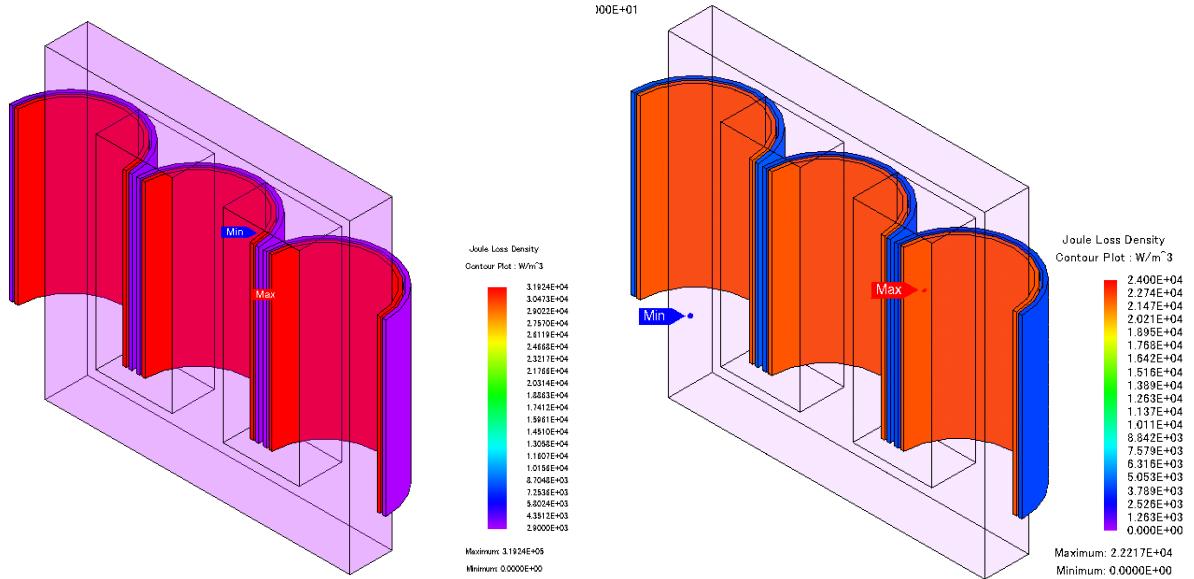


Figure 6: Joule loss density without cooling (left) and with cooling effect (right) [W/m³] (logarithmic view)

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.

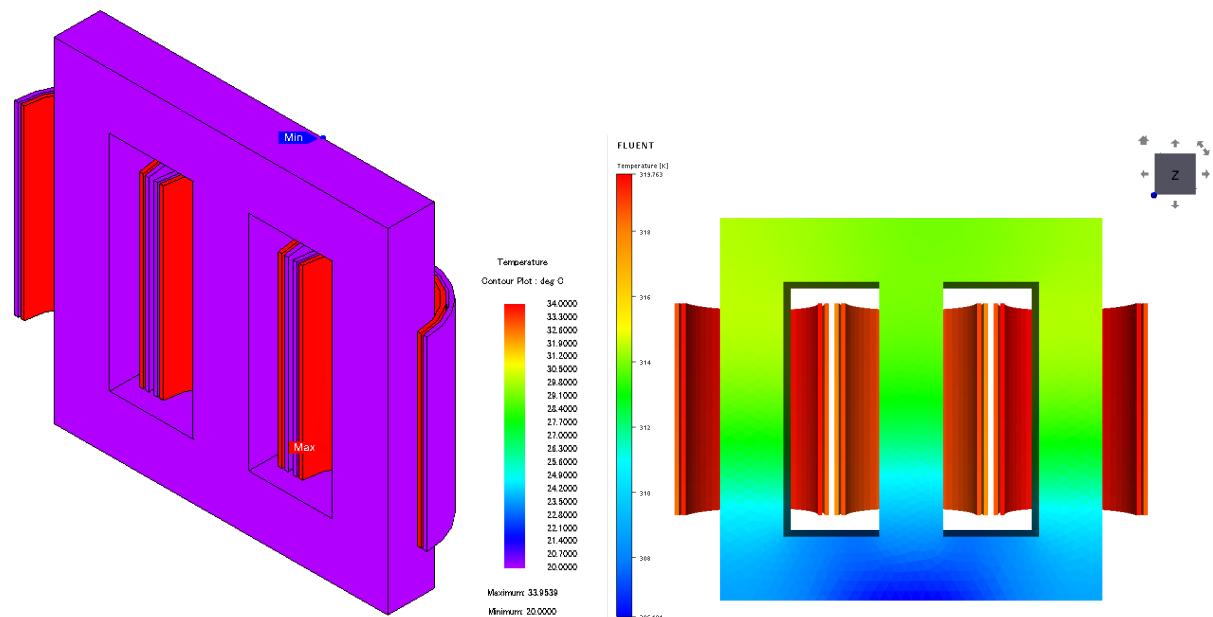
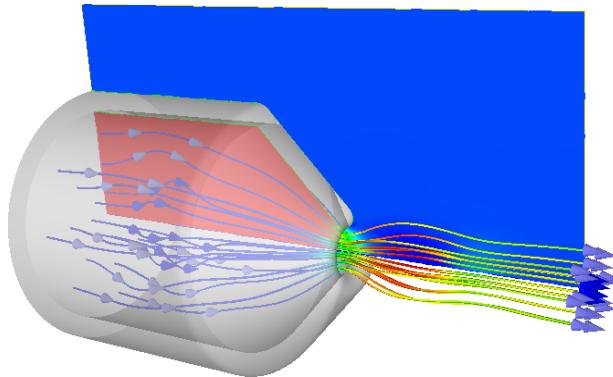


Figure 7: Thermal analysis standalone (left) and with cooling effect (right) [T]

9 Pipe Nozzle

9.1 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

9.2 Model Preparation

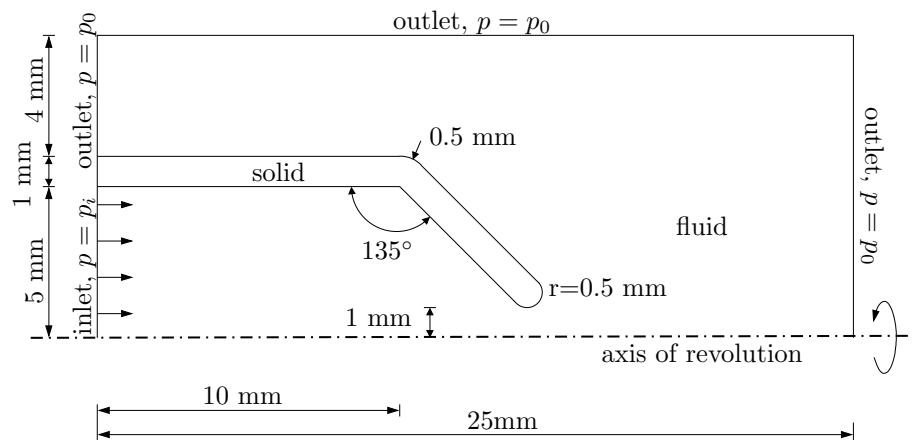


Figure 1: Pipe Nozzle: Section sketch of the axisymmetric model.

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 $E = 2000 \text{ MPa}$ 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 $p_i = 1 \text{ MPa}$.

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.

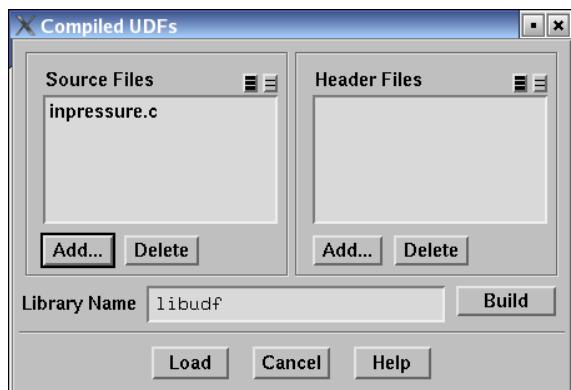
9.2.1 Fluid Model

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 casfile "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_inlet` has been created. This function is hooked to the boundary `fluid:inlet` as Total Pressure.

9.2.2 Solid Model

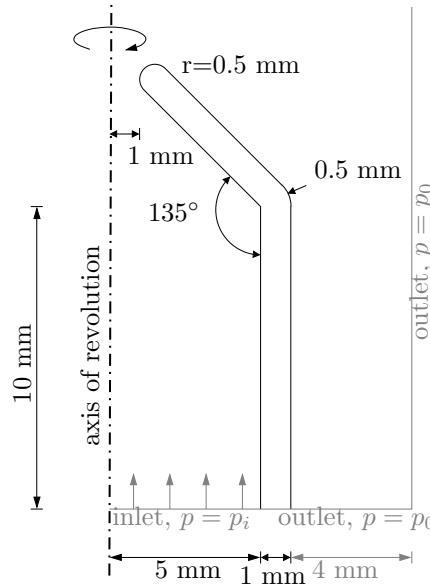


Figure 2: Pipe Nozzle: Section sketch of the axisymmetric model.

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.

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.

! 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`.

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 fluid mechanics code as first code to couple:

FLUENT

Option	Action
Code to couple	Select FLUENT.
Version	The model is axisymmetric, i. e. a 2d model. Thus select the FLUENT version 2d.
Release	Should be set to latest or a version supported by MpCCI (see VI-8 FLUENT).
Case file	Select the case file "FLUENT/nozzle.cas".
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

STAR-CCM+

Option	Action
Code to couple	Select STAR-CCM+.
Release	Should be set to latest or a version supported by MpCCI (see VI-17 STAR-CCM+).
Simulation file	Select the simulation file "STAR-CCM+/nozzle.sim".
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

2. Choose your solid mechanics code as second code to couple:

ANSYS

Option	Action
Code to couple	Select ANSYS.
Release	Should be set to latest or a version supported by MpCCI (see VI-3 ANSYS).
Product	You can select any ANSYS product which can be used for structural analysis, e. g. ansys.
Database file	Select database file "ANSYS/nozzle.db".
Unit system	Select the SI unit system, because the ANSYS model file was created using these units.
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

MSC NASTRAN

Option	Action
Code to couple	Select MSC NASTRAN.
Release	Should be set to latest or a version supported by MpCCI (see VI-13 MSC NASTRAN).
Input file	Select input file "MSCNastran/nozzle.bdf".
Unit system	Select the SI-mm-t-s unit system, because the MSC NASTRAN model was created using these units.
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

...

Press the **Coupling Step >** button at the bottom of the Models Step to get to the Coupling Step.

9.3.2 Coupling Step

In the Coupling Step build the coupling regions, define quantities to be exchanged and assign the defined quantities to the built regions (cf. [IV-2.5 Coupling 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:

For region Region_1	
For fluid mechanics code	Select coupling component
FLUENT	pipesurface
STAR-CCM+	
For solid mechanics code	Select coupling component
ANSYS	PIPESURFACE
MSC NASTRAN	nozzle

3. Select Define Quantity Sets tab. As currently no quantities are assigned to “QuantitySet_1” (created by default), a warning symbol is displayed in front of it. Now choose the quantities to be exchanged.

For quantity set QuantitySet_1		
Select quantities	Make quantity settings	
	Option	Action
RelWallForce	Set sender	Select fluid mechanics code.

4. Select Assign Quantity Sets tab and assign the defined quantity set to the built region.

For selected region	Check quantity set
Region_1	QuantitySet_1

Proceed to the Monitors Step by pressing **Monitors Step >**.

9.3.3 Monitors Step

No changes are required in the Monitors Step. Proceed to the Edit Step by pressing **[Edit Step >]**.

9.3.4 Edit Step

No changes are required in the Edit Step. Proceed to the Go Step by pressing **[Go Step >]**.

9.3.5 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 Start-Up and Starting Server and Codes <](#)).

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:

- Set Coupling configuration options which are the same for all fluid mechanics codes.

Option	Action
Coupling scheme	Select Explicit-SteadyState.
Initial quantities transfer	Select send.

- Set options which are code specific.

FLUENT

Option	Action
Auto hook functions	Deselect to keep FLUENT from sending data in each iteration.

STAR-CCM+

Option	Action
Use duration control	Check and access further options.
Iteration no. for starting the coupling	Set to 120 to activate a pre-calculation of 120 steps.
Iteration no. for ending the coupling	Set to 121.
Run in batch	Check this option.
Auto start with the default template macro	Check this option.
License options	Set according to your license type (cf. VI-17.2.4 Go Step <).

...

2. For your solid mechanics code:

- Set Coupling configuration options which are the same for all solid mechanics codes.

Option	Action
Coupling scheme	Select Explicit-SteadyState.
Initial quantities transfer	Select receive.

- Set options which are code specific.

ANSYS

Option	Action
Gui option	Keep -b to start ANSYS in batch mode.
APDL input script	Select the input script "ANSYS/nozzle.ans".

MSC NASTRAN

No code specific options need to be set.

...

9.4 Running the Computation

Start the MpCCI server by pressing the left **Start** button and save the project as "nozzle.csp". Then start the fluid mechanics and solid mechanics codes by pressing the other **Start** buttons.

FLUENT

- In the FLUENT window, select **Solve→Initialization** and press the **Initialize** button to initialize the FLUENT solution.
- Open the FLUENT Run Calculation panel by selecting **Solve→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:

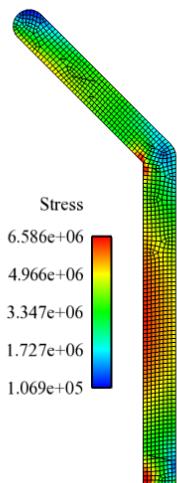
- Initializing the solution.
- Run 120 iterations.
- Connecting STAR-CCM+ to MpCCI server.
- Perform the data exchange, which is only a SEND operation.

- 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.

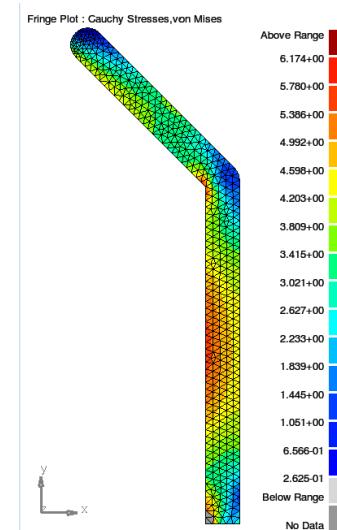
9.5 Discussion of Results

ANSYS



The stress distribution in the ANSYS model is shown on the left.

MSC NASTRAN



The stress distribution in the MSC NASTRAN model is shown on the left.

• • •

10 Cube in a Duct Heater

10.1 Problem Description

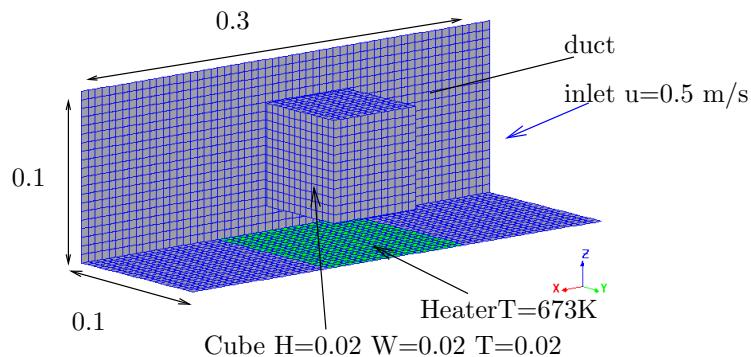


Figure 1: Duct Heater: Geometry [m] and boundary conditions section view

Topics of this Tutorial

- Radiative Heat Transfer
- Steady State
- Forced convection
- 3D-model
- Subcycling

Simulation Codes

- Radiation: RadTherm
- Fluid Mechanics: FLUENT, STAR-CCM+, STAR-CD, OpenFOAM

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.

10.2 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.

10.2.1 Radiation Model

RadTherm

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.

...

10.2.2 Fluid Model

The fluid domain comprises the inner part of the duct heater ([Table 1](#).)

Density	$1.205 \frac{kg}{m^3}$
Conductivity	$0.0237 \frac{W}{m \cdot K}$
Specific heat	$1006 \frac{J}{kg \cdot K}$
Inlet	
Velocity	$0.5 \frac{m}{s}$
Temperature	$293.0 K$
Turbulent Intensity	0.1
Turbulent Length	$0.02 m$
Outlet	standard setting
Cube	
wall heat	fixed
Temperature	$293.0 K$
Duct	
wall heat	fixed
Temperature	$293.0 K$
Heater	
wall heat	fixed
Temperature	$673.0 K$

Table 1: Duct Heater: Boundary conditions for fluid model

The setting used for fluid model are in the following [Table 2](#)

Time Domain	Steady state
Gravity	$-9.81 \frac{m}{s^2}$
Thermal option	
All settings	off
Buoyancy for AIR	On

Table 2: Duct Heater: Parameter setting for fluid model

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 for the thermal coupling, STAR-CCM+ needs an existing solution when STAR-CCM+ should send some data first. A cold start will fail in this case. In the Go Step STAR-CCM+ will be configured to calculate an initial solution before the coupling begins.

STAR-CD

A $k - \epsilon$ high Reynolds turbulence model is applied.

1. Start pro-STAR and resume the model file "`star_model.mdl`".
2. In the STAR GULde go to `Analysis Preparation/Running` → `Set Run Time Controls`.
Set Number of iterations to "500".
3. Save the model "`star_model.mdl`" and exit pro-STAR.

For STAR-CD the data transfer and management is handled via plug-ins and no user subroutines are needed for a coupled simulation.

OpenFOAM

A standard $k - \epsilon$ turbulence model with wall treatment is applied. The solver `buoyantBoussinesqSimpleFoam` is used.

...

10.3 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`.

10.3.1 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:

RadTherm	
Option	Action
Code to couple	Select RadTherm.
Release	Should be set to latest or a version supported by MpCCI (see VI-15 RadTherm/TAITherm).
".tdf" file	Select "radtherm_model.tdf" as input file.
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

2. Choose your fluid mechanics code as second code to couple:

FLUENT	
Option	Action
Code to couple	Select FLUENT.
Version	The model is three-dimensional, thus select the FLUENT version 3ddp.
Release	Should be set to latest or a version supported by MpCCI (see VI-8 FLUENT).
Case file	Select the case file "fluent_model.cas".
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

OpenFOAM	
Option	Action
Code to couple	Select OpenFOAM.
Option	Select the OpenFOAM option Opt or Debug.
Precision	Select the OpenFOAM precision: DP for double or SP for single precision.
Release	Should be set to latest or a version supported by MpCCI (see VI-14 OpenFOAM).
Case directory	Select the case directory "OpenFOAM/<version>" fitting your OpenFOAM version (e.g. "OpenFOAM/3.0" for use with OpenFOAM 3.0.1).
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

STAR-CCM+

Option	Action
Code to couple	Select STAR-CCM+.
Release	Should be set to latest or a version supported by MpCCI (see VI-17 STAR-CCM+ ▲).
Simulation file	Select the simulation file "STAR-CCM+/ductheater.sim".
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

STAR-CD

Option	Action
Code to couple	Select STAR-CD.
Release	Should be set to latest or a version supported by MpCCI (see VI-18 STAR-CD ▲).
Model file	Select the model file "STARCD/star_model.mdl".
License options	Set according to your license type (cf. VI-18.2.2 Models Step ▲).
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

...

Press the **Coupling Step >** button at the bottom of the Models Step to get to the Coupling Step.

10.3.2 Coupling Step

In the Coupling Step build the coupling regions, define quantities to be exchanged and assign the defined quantities to the built regions (cf. [IV-2.5 Coupling Step – Defining Coupling Regions and Quantities ▲](#)).

- At first select the Mesh tab to get access to the mesh based element components.
- 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 equally, you may choose an easier way to select all coupling components by setting following option (cf. [V-4.5.4.1 Simplified Component Selection ▲](#)):

Option	Action
Component Name Matching	Select Exact.

Now with a double-click on each component to be coupled, the same component from the other code will automatically be added to its coupled box, too.

Lookup the components to be coupled in the following table and select by double-click:

For region Region_1	
For all codes	Select coupling components
RadTherm	duct_leftside duct_topside duct_rightside duct_bottomside
FLUENT	cube_backside cube_leftside cube_rightside cube_topside cube_bottomside
OpenFOAM	cube_leftside cube_rightside cube_topside cube_bottomside
STAR-CCM+	cube_leftside cube_rightside cube_topside cube_bottomside
STAR-CD	cube_leftside cube_rightside cube_topside cube_bottomside cube_frontside

3. Select Define Quantity Sets tab. As currently no quantities are assigned to “QuantitySet_1” (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. [IV-4.5.8 Assigning Preconfigured Quantity Settings](#)):

For quantity set	Option	Action
QuantitySet_1	Select Coupling Type	Select Steady state radiative heat transfer and click Set .

Now the quantities FilmTemp, WallHTCoeff and WallTemp are activated and MpCCI already set the appropriate sender for them.

4. Select Assign Quantity Sets tab and assign the defined quantity set to the built region.

For selected region	Check quantity set
Region_1	QuantitySet_1

Proceed to the Monitors Step by pressing **Monitors Step >**.

10.3.3 Monitors Step

No changes are required in the Monitors Step. Proceed to the Edit Step by pressing **Edit Step >**.

10.3.4 Edit Step

No changes are required in the Edit Step. Proceed to the Go Step by pressing **Go Step >**.

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 Start-Up and Starting Server and Codes](#)).

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 radiation code:
Set Coupling configuration and code specific option.

RadTherm	
Option	Action
Simulation type	Select User Defined Steady State.
Initial quantities transfer	Select receive.
Convergence management	Check and access further options.
No. of steps without coupling	Set to 50.
Iteration no. for ending the coupling	Set to 500.
Run in batch	Check this option.

...

2. For your fluid mechanics code:

- Set Coupling configuration options which are mostly the same for all fluid mechanics codes. Only the initial quantities transfer for STAR-CD differs as mentioned in the table.

Option	Action
Coupling scheme	Select Explicit-SteadyState.
Initial quantities transfer	Select send.
	Only for STAR-CD : Select skip.
Use subcycling	Check this option.
No. of steps without coupling	Set to 50.

- Set options which are code specific.

FLUENT	
Run in batch	Check this option.
Optional journal files	Select the journal file "run.jou".

OpenFOAM	
Option	Action
Select OpenFOAM solver	Leave the default Auto-Select-by-Case. The solver setting from the "controlDict" file (buoyantBoussinesqSimpleFoam) will be used.
Reference density	Set density to 1.205 kg/m ³ .
Reference specific heat	Set specific heat to 1006 J/kgK according to the fluid properties from Table 1 .

STAR-CCM+

Option	Action
Use duration control	Check this option.
Iteration no. for starting the coupling	Set to 5. STAR-CCM+ needs a solution before sending the quantities FilmTemp, WallHTCoeff.
Iteration no. for ending the coupling	Set to 505.
Run in batch	Check this option.
Auto start with the default template macro	Check this option.
License options	Set according to your license type (cf. ▷ VI-17.2.4 Go Step ◄).

STAR-CD

Option	Action
Startup procedure	Select Prepare case and start to accomplish the necessary modifications in the STAR-CD model for a coupled simulation.
Backup model file	Click this button. As MpCCI will build a new model file, the original model file should be saved.
Use the MpCCI plugin library	Check this option to activate MpCCI plug-in in STAR-CD for data transfer.
Double precision mode	Check this option.

...

10.4 Running the Computation

10.4.1 Starting the Simulation

Save the MpCCI project file with name "duct_heater.csp" over the MpCCI GUI menu

File→Save Project As.

Press the three **Start** buttons in the Go Step and the simulation codes should start. Some codes require additional actions:

RadTherm

RadTherm 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.

STAR-CD

The STAR-CD calculation is started in batch mode. The maximum number of iterations has been set to 500 in the model file.

OpenFOAM

The OpenFOAM calculation is started in batch mode. The maximum number of iterations has set to 500 in the "controlDict" file.

...

10.5 Discussion of Results

RadTherm

Temperature results may be visualized in RadTherm 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"

STAR-CD

STAR-CD result file "star_model.pst" or "star_model.ccmp" may be post processed with pro-STAR.

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.

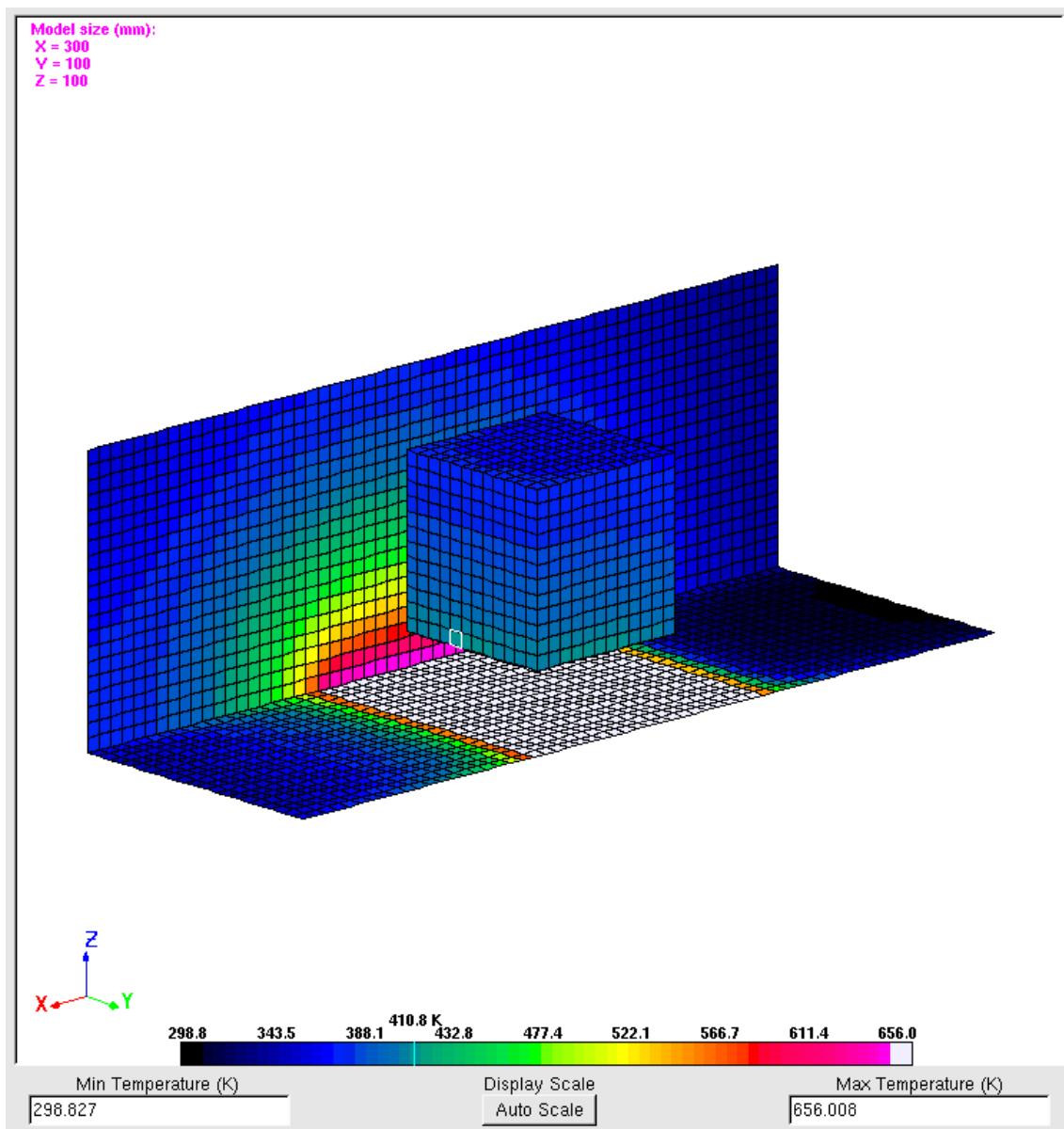


Figure 2: Duct Heater: Temperature section plot

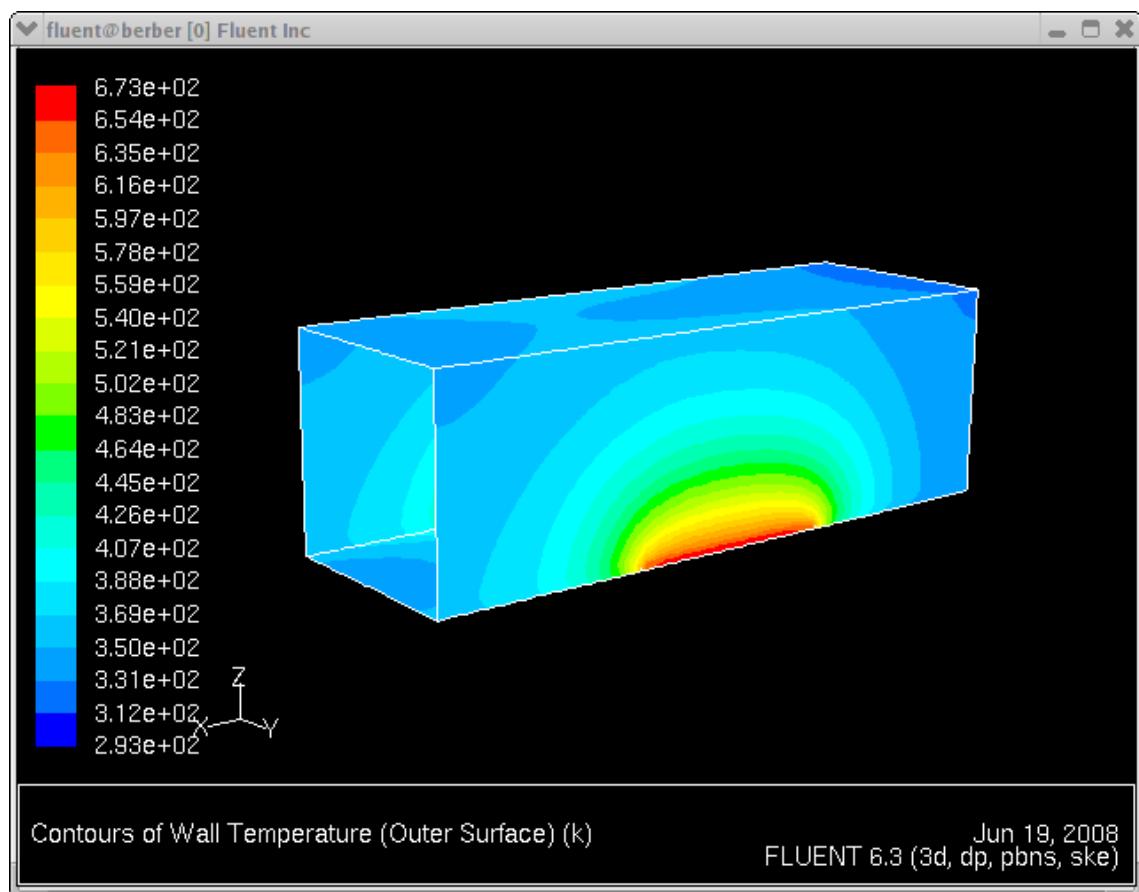


Figure 3: Duct Heater: FLUENT wall temperature contour plot

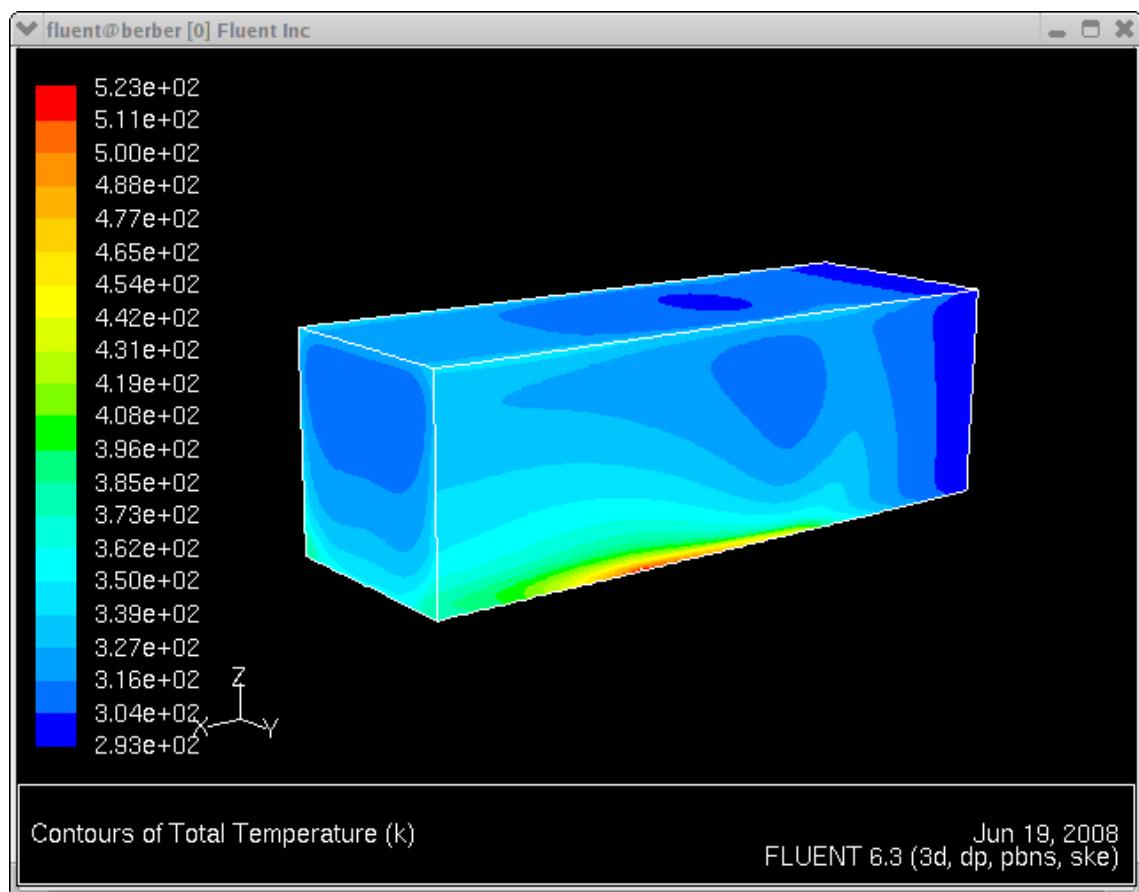


Figure 4: Duct Heater: FLUENT total temperature contour plot

11 Y-Junction

11.1 Problem Description

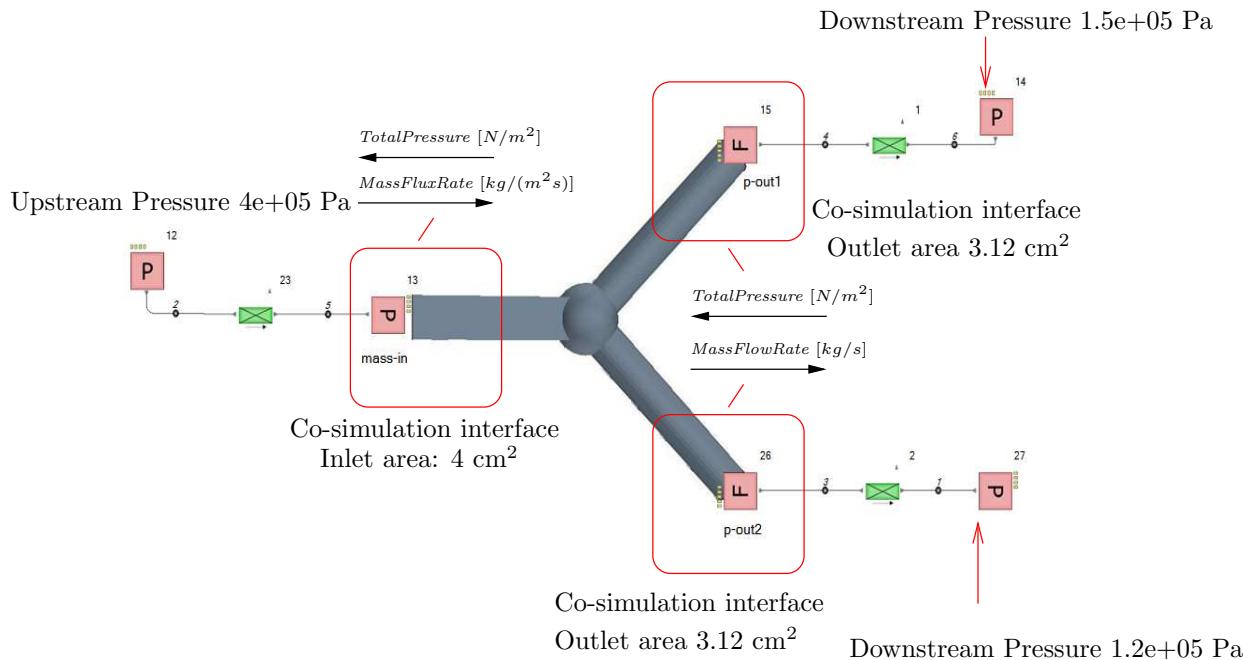


Figure 1: Y-Junction: Fluid geometry and network boundaries

Topics of this Tutorial

- 1D network model
- 3D-model
- Steady State

Simulation Codes

- Network: Flowmaster
- Fluid Mechanics: FLUENT, OpenFOAM, STAR-CCM+, STAR-CD

11.2 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.

11.2.1 Network Model

Flowmaster

The Flowmaster 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 Flowmaster network by a source.

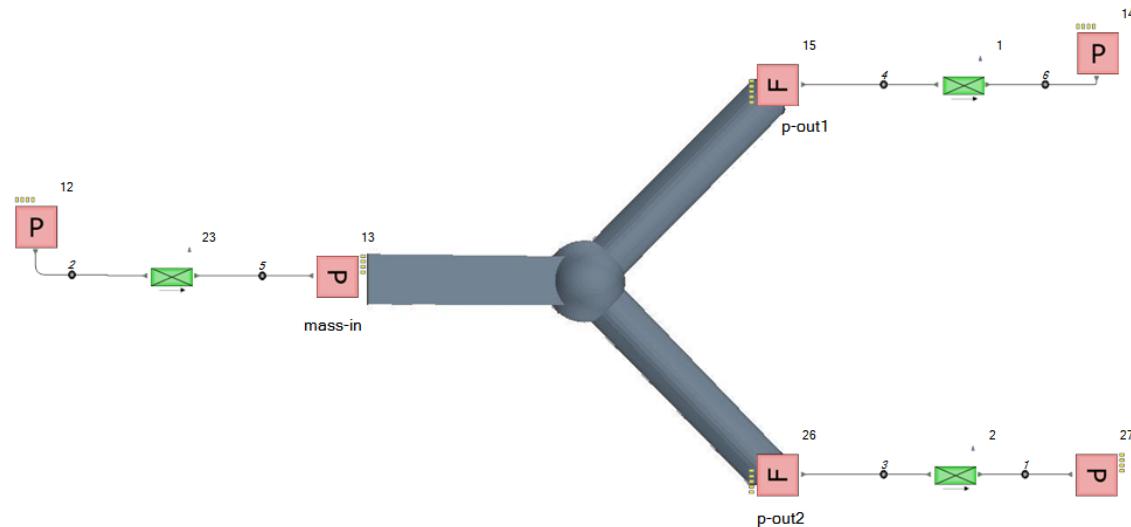
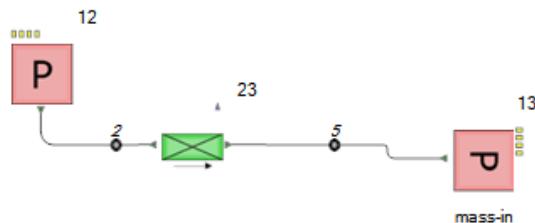


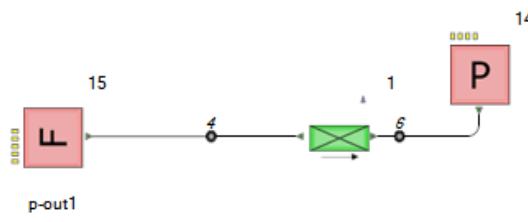
Figure 2: Y-Junction: Flowmaster network

Network Upstream Components Settings



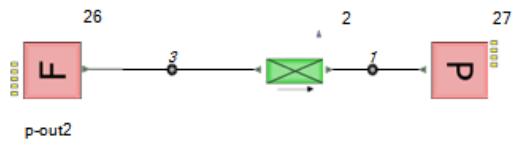
- Liquid type is water.
- The pressure source component “12” has pressure value set to $4e+5$ Pa.
- The pressure source component “13” has no specific pressure value set.

Network First Downstream Branch Settings



- Liquid type is water.
- The pressure source component “14” has pressure value set to $1.5e+5$ Pa.
- The flow source component “15” has an initial volumetric flow rate value set to $0.0005 \frac{m^3}{s}$.

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 $0.0004 \frac{m^3}{s}$.

Extract the Flowmaster Network

1. Start Flowmaster then you need to log on the Flowmaster database.
 2. Unpack the appropriate "Y_Junction.FMpck" pack file from the "*<MpCCI_home>/tutorial/Y-Junction/Flowmaster*" folder.
- !** Unpack the file directly under the root of your project tree.

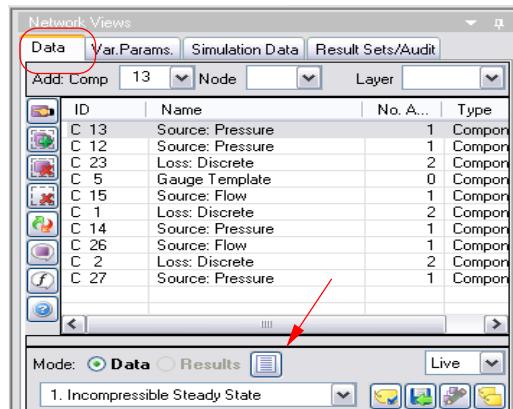
Configure the Flowmaster Network for the Co-Simulation

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:

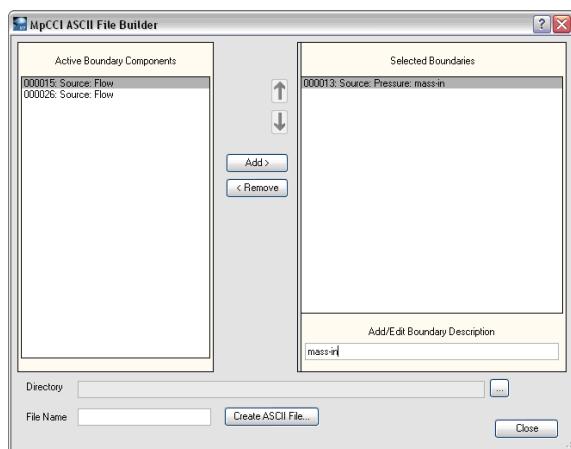
1. Select the boundary component.
2. Edit the boundary with the **Edit** mouse menu context.
3. Select in the property list the External Model Boundary item.
4. Click on the Sub Form... button.
5. Change the property Boundary Active value to Yes.
6. Click on the button **Return**.

Run a standalone incompressible steady state (SS) Flowmaster analysis to test the configuration.

Export the Flowmaster 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**.



- 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/Flowmaster ")
- 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 Flowmaster graphical interface.

! Do not renumber any Network used for co-simulation as component IDs are used to identify boundaries in MpCCI link file.

11.2.2 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 $20 \frac{m}{s}$
- Mass flow inlet boundary has an initial mass flux rate of $24950 \frac{kg}{m^2.s^1}$
- Pressure outlet boundary has an initial pressure of $1.5e^{+5} \text{ Pa}$ and $1.2e^{+5} \text{ Pa}$.
- Reference Pressure is 101325 Pa.

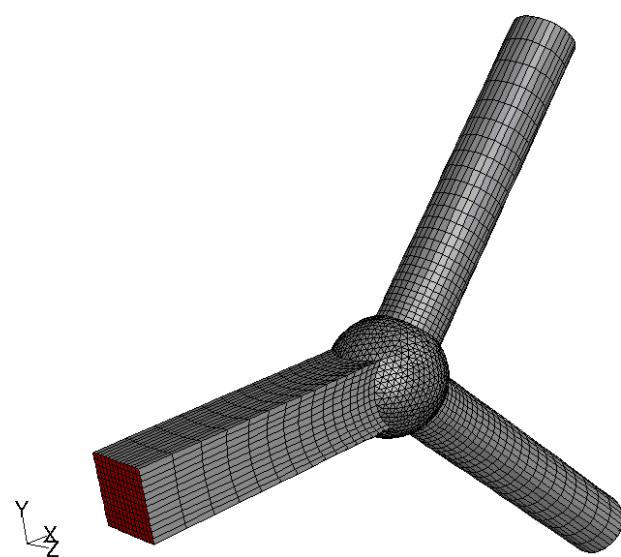


Figure 3: Y-Junction: mesh

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 network code as first code to couple:

Flowmaster	
Option	Action
Code to couple	Select Flowmaster.
Release	Should be set to latest or a version supported by MpCCI (see VI-7 Flowmaster).
Data file	Select data file "Flowmaster/Y_Junction.fmlink".
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.
...	

2. Choose your fluid mechanics code as second code to couple:

FLUENT	
Option	Action
Code to couple	Select FLUENT.
Version	The model is three-dimensional, thus select the FLUENT version 3d.
Release	Should be set to latest or a version supported by MpCCI (see VI-8 FLUENT).
Case file	Select the case file "FLUENT/Y-junction.cas.gz".
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

OpenFOAM

Option	Action
Code to couple	Select OpenFOAM.
Option	Select the OpenFOAM option Opt or Debug.
Precision	Select the OpenFOAM precision: DP for double or SP for single precision.
Release	Should be set to latest or a version supported by MpCCI (see ▷ VI-14 OpenFOAM ▲).
Case directory	Select the case directory "OpenFOAM/<version>" fitting your OpenFOAM version (e.g. "OpenFOAM/3.0" for use with OpenFOAM 3.0.1).
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

STAR-CCM+

Option	Action
Code to couple	Select STAR-CCM+.
Release	Should be set to latest or a version supported by MpCCI (see ▷ VI-17 STAR-CCM+ ▲).
Simulation file	Select the simulation file "STAR-CCM+/yjunction.sim".
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

STAR-CD

Option	Action
Code to couple	Select STAR-CD.
Release	Should be set to latest or a version supported by MpCCI (see ▷ VI-18 STAR-CD ▲).
Model file	Select the model file "STARCD/Y-junction.mdl".
License options	Set according to your license type (cf. ▷ VI-18.2.2 Models Step ▲).
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

A warning window will appear and complains about the model dimensions. You can continue by clicking on **Continue**.

11.3.2 Coupling Step

In the Coupling Step build the coupling regions, define quantities to be exchanged and assign the defined quantities to the built regions (cf. [▷ IV-2.5 Coupling 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.

For region Region_1	
For network code	Select coupling component
Flowmaster	I mass-in
For fluid mechanics code	Select coupling component
FLUENT	
OpenFOAM	
STAR-CCM+	▲ mass-in
STAR-CD	

- Click on button **Add** to add new region Region_2.

For region Region_2	
For network code	Select coupling component
Flowmaster	I p-out1
For fluid mechanics code	Select coupling component
FLUENT	
OpenFOAM	
STAR-CCM+	▲ p-out1
STAR-CD	

- Click on button **Add** to add new region Region_3.

For region Region_3	
For network code	Select coupling component
Flowmaster	I p-out2
For fluid mechanics code	Select coupling component
FLUENT	
OpenFOAM	
STAR-CCM+	▲ p-out2
STAR-CD	

3. Select Define Quantity Sets tab. As currently no quantities are assigned to “QuantitySet_1” (created by default), a warning symbol is displayed in front of it. Now choose the quantities to be exchanged.

For quantity set QuantitySet_1		
Select quantities	Make quantity settings	
	Option	Action
MassFluxRate		
TotalPressure	Set sender	Select network code.
	Code specific settings	
FLUENT		
index	Set to 0 for receive method UDM.	
	STAR-CD	
index	Set to 1 for receive method SCALAR.	
	Set sender	Select fluid mechanics code.

4. Apply an operator to MassFluxRate to get a stable computation (cf. [▷ V-4.5.7.4 Applying Operators to Mesh Based Components ◁](#)).

Under-relaxation will be needed. Therefore mark the wanted quantity in QuantitySet_1 by clicking on its name and configuring the listed operator as follows:

Quantity	Operator	Operator option	Operator action
MassFluxRate	Select Relaxation.	Relaxation method	Select Fixed.
		Relaxation factor	Set to 0.3 for under-relaxation.

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 QuantitySet_2.

For quantity set QuantitySet_2		
Select quantities	Make quantity settings	
	Option	Action
MassFlowRate		Set sender Select fluid mechanics code.
TotalPressure	Set sender	Select network code.
	Code specific settings	
FLUENT		
index	Set to 1 for receive method UDM.	
	STAR-CD	
index	Set to 2 for receive method SCALAR.	

6. Apply an operator to TotalPressure to get a stable computation (cf. [▷ V-4.5.7.4 Applying Operators to Mesh Based Components ◁](#)).

Under-relaxation will be needed. Therefore mark the wanted quantity in QuantitySet_2 by clicking on its name and configuring the listed operator as follows:

Quantity	Operator	Operator option	Operator action
TotalPressure	Select Relaxation.	Relaxation method	Select Fixed.
		Relaxation factor	Set to 0.3 for under-relaxation.

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
Region_1	QuantitySet_1
Region_2	QuantitySet_2
Region_3	

Proceed to the Monitors Step by pressing **[Monitors Step >]**.

11.3.3 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.

For fluid mechanics code	Option	Action
FLUENT	Component Dimension	Select Face ()
OpenFOAM	Monitored	Select component  wall
STAR-CD	Quantity	Check TotalPressure
STAR-CCM+		

Proceed to the Edit Step by pressing **[Edit Step >]**.

11.3.4 Edit Step

No changes are required in the Edit Step. Proceed to the Go Step by pressing **[Go Step >]**.

11.3.5 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 Start-Up and Starting Server and Codes](#)).

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:

- Set Coupling configuration options which are the same for all network codes.

Option	Action
Coupling scheme	Select Explicit-SteadyState.
Initial quantities transfer	Select receive.

- Set options which are code specific.

Flowmaster

Option	Action
User name	Enter the user name to log in the database.
Working project name	Change the name of the working project corresponding to the root of your project list view in Flowmaster if necessary.
Analysis type	Select SS for the analysis type.

...

2. For your fluid mechanics code:

- Set Coupling configuration options which are the same for all fluid mechanics codes.

Option	Action
Coupling scheme	Select Explicit-SteadyState.
Initial quantities transfer	Select exchange.
Use subcycling	Check and access further options.
No. of steps without coupling	Set to 20.

- Set options which are code specific.

FLUENT

Option	Action
Use duration control	Check and access further options.
Iteration no. for starting the coupling	Set to 200.
Optional journal file	Select journal file "FLUENT/job.jou".

OpenFOAM

Option	Action
Use duration control	Check and access further options.
Iteration no. for starting the coupling	Set to 200 to activate a pre-calculation of 200 steps.
Reference pressure	Set pressure to 101325 N/m ² .
Reference density	Set density to 998 kg/m ³ .

STAR-CCM+

Option	Action
Use duration control	Check and access further options.
Iteration no. for starting the coupling	Set to 200 to activate a pre-calculation of 200 steps.
Iteration no. for ending the coupling	Set to 1000.
Run in batch	Check this option.
Auto start with the default template macro	Check this option.
License options	Set according to your license type (cf. ▷ VI-17.2.4 Go Step ◄).

STAR-CD

Option	Action
Use duration control	Check and access further options.
Iteration no. for starting the coupling	Set to 200.
Startup procedure	Select Prepare case and start to accomplish the necessary modifications in the STAR-CD model for a coupled simulation.
Backup model file	Click this button. As MpCCI will build a new model file, the original model file should be saved.

...

11.4 Running the Computation

11.4.1 Starting the Simulation

Save the MpCCI project file with name "y_junction.csp" over the MpCCI GUI menu **File→Save Project As...**.

Press the three **Start** buttons in the Go Step and the simulation codes should start.

Flowmaster

The Flowmaster calculation is started in batch mode.

FLUENT

The FLUENT GUI is started and the calculation starts automatically. The FLUENT journal file "job.jou" defines the following procedures:

```
;; initialize flow
/solve/initialize initialize-flow

;; do 1000 iterations
/solve iterate 1000
```

FLUENT will perform 1000 iterations with 20 iterations without coupling until FLUENT converges.

OpenFOAM

The OpenFOAM calculation is started in batch mode. OpenFOAM will perform 1000 iterations with 20 iterations without coupling.

STAR-CCM+

The STAR-CCM+ calculation is started in batch mode. STAR-CCM+ will perform 1000 iterations with 20 iterations without coupling.

STAR-CD

The STAR-CD calculation is started in batch mode. STAR-CD will perform 1000 iterations with 20 iterations without coupling until convergence criteria or the maximum number of iteration is reached.

...

11.5 Discussion of Results

Flowmaster

You can open the Flowmaster 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.

STAR-CD

STAR-CD result file "Y-junction.ccmp" may be post processed with pro-STAR. In order to visualize the quantity value stored in the user scalar you have to select in pro-STAR the data type Cell & Wall/Bound (Smooth).

...

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 $4e^{+5}$ Pa was prescribed on the network model and on the fluid a total pressure value around $2.3e^{+5}$ Pa is expected.

At the outlet boundary "p-out1" and "p-out2" a total pressure value of $1.5e^{+5}$ Pa respectively $1.2e^{+5}$ 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`

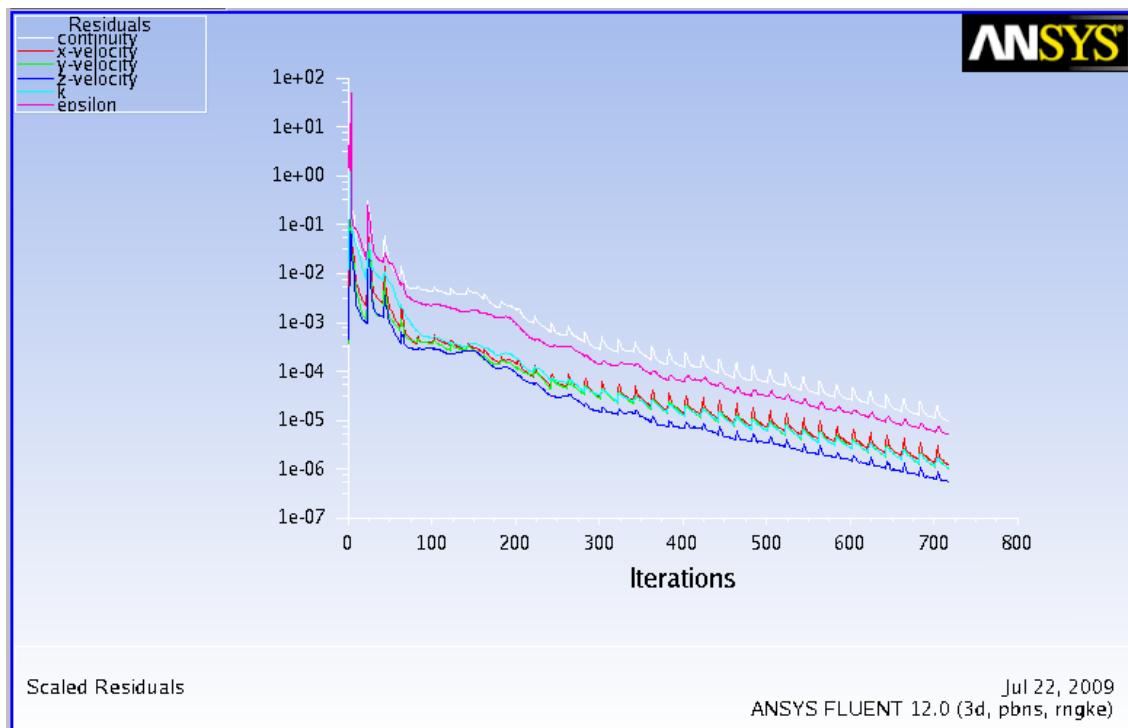


Figure 4: Y-Junction: Fluent Residuals plot

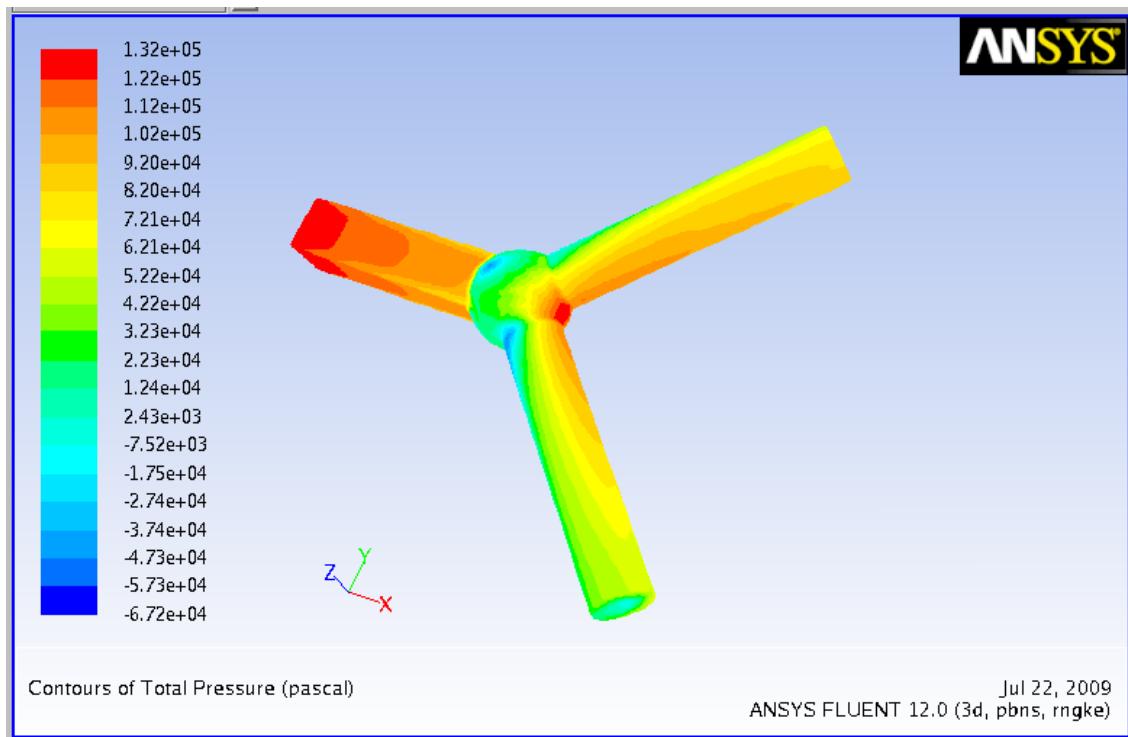


Figure 5: Y-Junction: Fluent Total Pressure plot

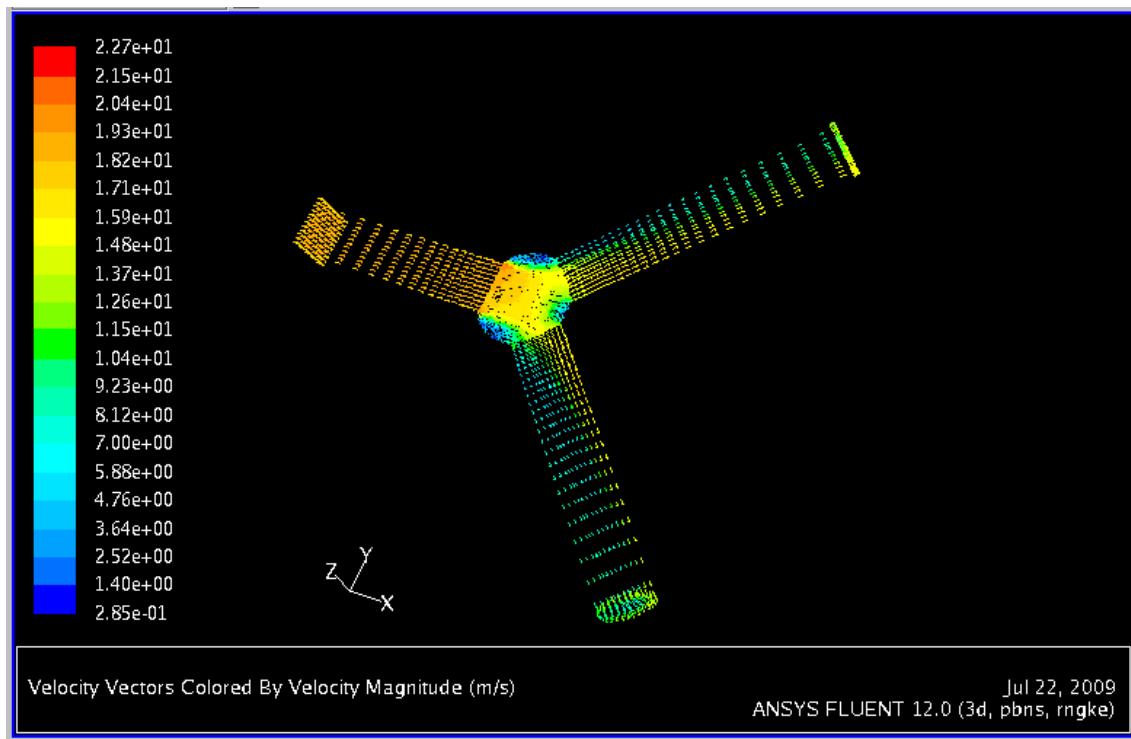


Figure 6: Y-Junction: Fluent Velocity vector section plot

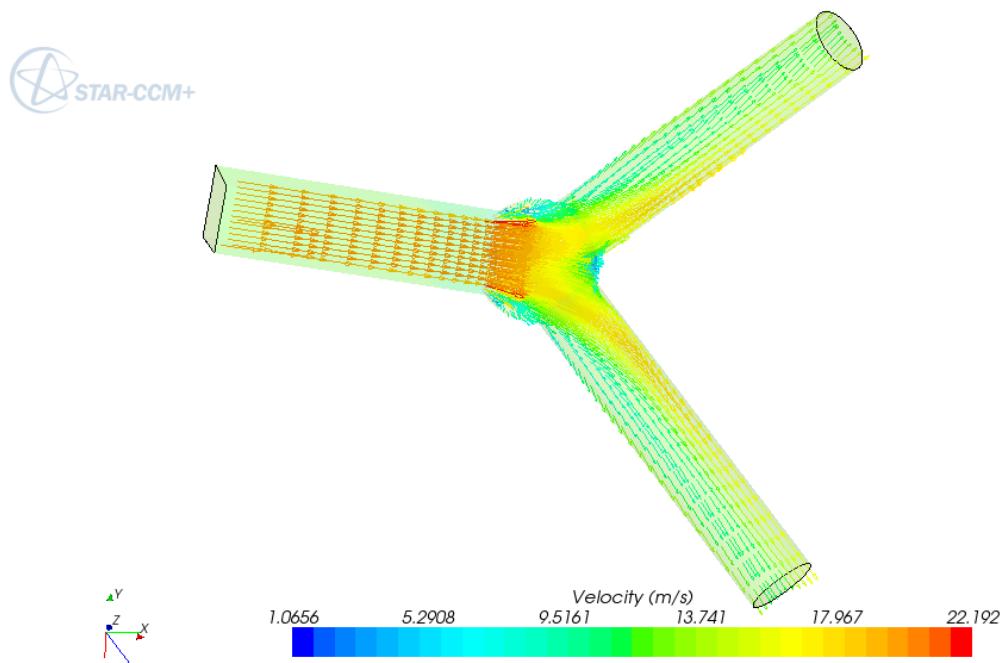


Figure 7: Y-Junction: STAR-CCM+ Velocity vector section plot

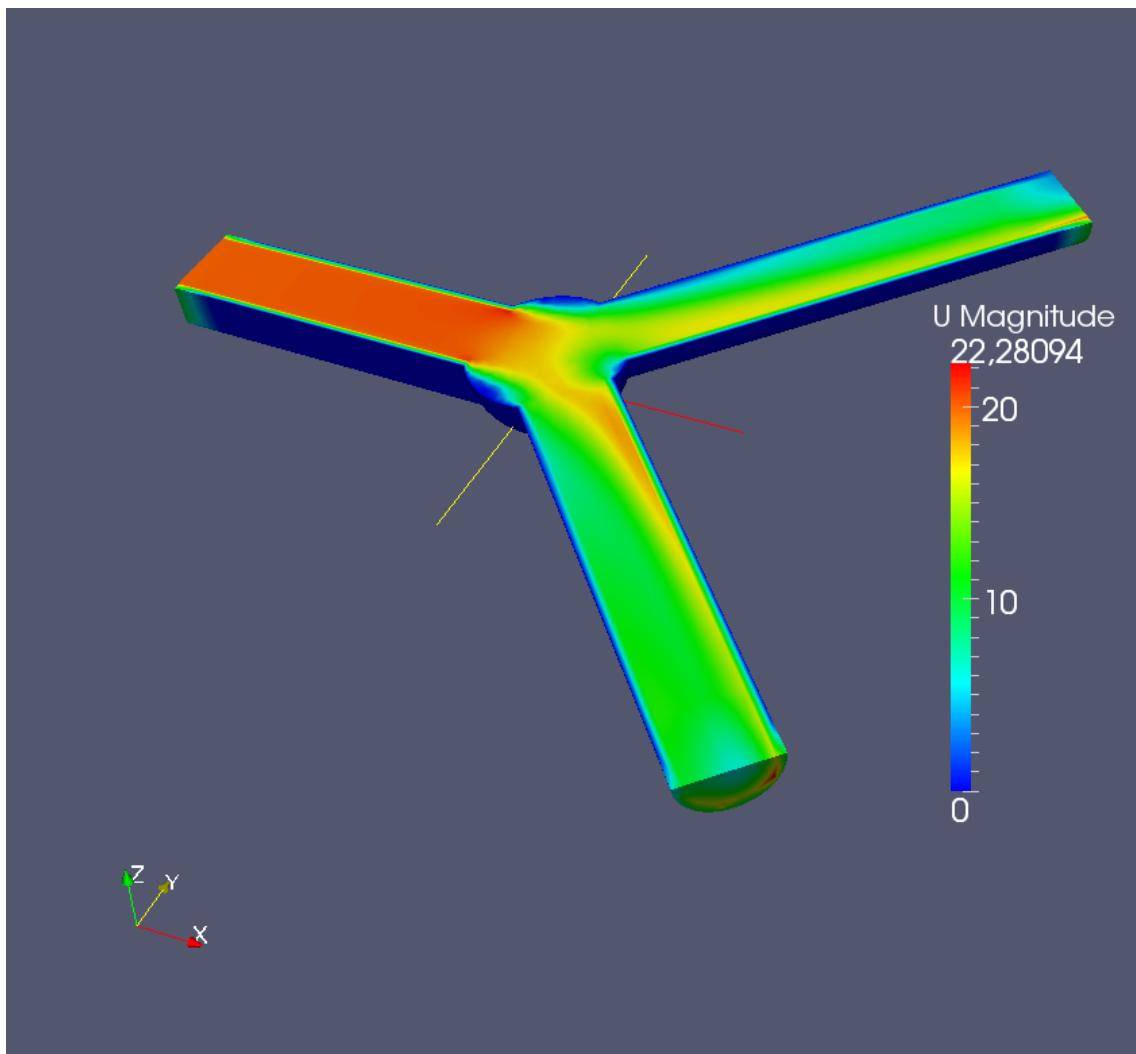


Figure 8: Y-Junction: OpenFOAM Velocity contour section plot

12 Spring Mass System

12.1 Problem Description

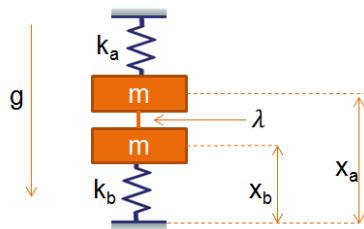


Figure 1: Double mass-spring system

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)

12.2 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`" which correspond to the simulation codes you want to use.

12.2.1 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 $g = 10\text{m}^2/\text{s}$ is imposed at $t = 0$. The displacements are constrained, resulting in both masses moving together. The constraint force between the systems is defined as λ . This results in the following equations for the double mass-spring system:

$$\begin{aligned}
 m_a \ddot{x}_a + k_a x_a &= -m_a g - \lambda \\
 m_b \ddot{x}_b + k_b x_b &= -m_b g - \lambda \\
 x_a &= x_b \\
 \text{at } t = 0 : \quad x_a &= x_b = 0 \quad \text{and} \quad \dot{x}_a = \dot{x}_b = 0
 \end{aligned}$$

For the mass and the stiffness of the spring-mass systems “stiffness factors” α and β are introduced. These factors define the relative distribution of the single system mass and stiffness values to the total system:

$$\begin{aligned}
 k_a &= \alpha k \\
 k_b &= (1 - \alpha)k \\
 m_a &= \beta m \\
 m_b &= (1 - \beta)m
 \end{aligned}$$

Using these factors the previous equations can be written as:

$$\begin{aligned}
 \ddot{x}_a + \frac{\alpha}{\beta} \omega^2 x_a &= -g - \frac{\lambda}{\beta m} \\
 \ddot{x}_b + \frac{1 - \alpha}{1 - \beta} \omega^2 x_b &= -g - \frac{\lambda}{(1 - \beta)m}
 \end{aligned}$$

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.

12.2.2 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 λ_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:

```
% 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:

```
% 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 λ_a :

$$\ddot{x}_a + \frac{\alpha}{\beta} \omega^2 x_a = -g - \frac{\lambda_a}{\beta m}$$

The interface force λ_a is then exported.

...

12.2.3 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.

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.

SimulatorB

The executable for the **SimulatorB** is part of the MpCCI distribution ("<MPCCI_HOME>/codes/SimulatorB/bin/<version>/<MPCCI_ARCH>/Mpcci_SimulatorB.exe").

SimulatorB receives the interface force λ_a (e.g. from **SimulatorA**) and then uses the Newmark time integration method to compute the acceleration, velocity and displacements:

$$\ddot{x}_b + \frac{1-\alpha}{1-\beta}\omega^2 x_b = -g - \frac{\lambda_b}{(1-\beta)m}$$

The displacements are exported to its partner code.

...

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**.

- Choose your code for Model A as first code to couple:

MATLAB	
Option	Action
Code to couple	Select MATLAB.
Release	Should be set to latest or a version supported by MpCCI (see VI-10 MATLAB).
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.
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

SimulatorA	
Option	Action
Code to couple	Select SimulatorA.
Release	Set to latest.
SimulatorA file	Select the SimulatorA file "SimulatorA/model.empty".  This file is empty. The information is generated from the "Scanner" and all settings can be handled in the Go Step.
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

...

- Choose your code for Model B as second code to couple:

Abaqus	
Option	Action
Code to couple	Select Abaqus
Release	Should be set to latest or a version supported by MpCCI (see VI-2 Abaqus).
Input file	Select input file "Abaqus/modelB.inp".
Unit system	Select the SI unit system (which is the standard).
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

MATLAB

Option	Action
Code to couple	Select MATLAB.
Release	Should be set to latest or a version supported by MpCCI (see VI-10 MATLAB).
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.
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

MSC Adams

Option	Action
Code to couple	Select MSC Adams
Release	Should be set to latest or a version supported by MpCCI (see VI-11 MSC Adams).
MSC Adams file	Select MSC Adams file "Adams/sim_command.acf".
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).
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

SIMPACK

Option	Action
Code to couple	Select SIMPACK
Release	Should be set to latest or a version supported by MpCCI (see VI-16 SIMPACK).
SIMPACK file	Select SIMPACK file "Simpack/ModelB.spck".
Unit system	Select the variable unit system (which is the default).
Grid length unit	Select m as unit for the grid length (which is the default).
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

SimulatorB

Option	Action
Code to couple	Select SimulatorB.
Release	Set to latest.
SimulatorB file	Select the SimulatorB file "SimulatorB/model.empty".  This file is empty. The information is generated from the "Scanner" and all settings can be handled in the Go Step.
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

...

Press the **Coupling Step >** button at the bottom of the Models Step to get to the Coupling Step.

12.3.2 Coupling Step

In the Coupling Step build the coupling regions, define quantities to be exchanged and assign the defined quantities to the built regions (cf. [IV-2.5 Coupling 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:

For region Region_1	
For code	Select coupling component
Abaqus	• NQA
MSC Adams	• GFORCE-1::GFORCE_1
SIMPACK	• \$F_Spring
SimulatorA	• BearingPoint
SimulatorB	

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-4.5.2.1 Copying Components](#).

Partner code	Select component for copying	⇒ Copies
MSC Adams	• GFORCE-1::GFORCE_1	• GFORCE-1::GFORCE_1::1 • GFORCE-1::GFORCE_1::2
SIMPACK	• \$F_Spring	• \$F_Spring::1
	(i) Because SIMPACK doesn't support to couple quantity Acceleration only one copied component is needed for coupling with MATLAB.	
SimulatorA	• BearingPoint	• BearingPoint::1
SimulatorB		• BearingPoint::2

(i) 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.

For region Region_1	
For code	Select coupling component
Abaqus	<input checked="" type="radio"/> NQA
MATLAB	<input checked="" type="radio"/> lambda
MSC Adams	<input checked="" type="radio"/> GFORCE-1::GFORCE_1
SIMPACK	<input checked="" type="radio"/> \$F_Spring
SimulatorA	<input checked="" type="radio"/> BearingPoint
SimulatorB	

- Click on button **Add** to add new region Region_2.

For region Region_2	
For code	Select coupling component
Abaqus	<input checked="" type="radio"/> NQA_DUP
MATLAB	<input checked="" type="radio"/> disp1
MSC Adams	<input checked="" type="radio"/> GFORCE-1::GFORCE_1::1
SIMPACK	<input checked="" type="radio"/> \$F_Spring::1
SimulatorA	<input checked="" type="radio"/> BearingPoint::1
SimulatorB	

- 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**.

For region Region_3	
For code	Select coupling component
Abaqus	<input checked="" type="radio"/> NQA_DUP2
MATLAB	<input checked="" type="radio"/> acc1
MSC Adams	<input checked="" type="radio"/> GFORCE-1::GFORCE_1::2
SimulatorA	<input checked="" type="radio"/> BearingPoint::2
SimulatorB	

- 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 "QuantitySet_1" (created by default), a warning symbol is displayed in front of it. Now choose the quantity to be exchanged.

For quantity set QuantitySet_1		
Select quantity	Option	Action
Force	Set sender	Select code for Model A.

For an iterative coupling it is advisable to define a convergence tolerance for the inner iterations. Therefore add a convergence check operator (cf. [V-4.5.7.4 Applying Operators to Mesh Based Components](#)) to the Force quantity. For this example a value of 0.01 is small enough to produce an accurate solution.

The following table lists all quantities coupled in this example. Look up the requested quantity for your quantity set, mark it by clicking on its name and then configure the listed operator:

Quantity	Operator	Operator option	Operator action
Force	Select ConvergenceCheck.	Tolerance value	Set to 0.01.

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.

- Click on button **Add** to add new quantity set **QuantitySet_2**.

For quantity set QuantitySet_2		
Select quantity	Option	Action
PointPosition	Set sender	Select code for Model B.
	Code specific settings	
	MSC Adams	
	Unit	Set to mm.

- For coupling with MSC Adams you need to create a third quantity set for exchanging the Acceleration quantity. For the other codes, but 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.

In order to couple acceleration click on button **Add** to add new quantity set **QuantitySet_3**.

For quantity set QuantitySet_3		
Select quantity	Option	Action
Acceleration	Set sender	Select code for Model B.
	Code specific settings	
	MSC Adams	
	Unit	Set to mm/s ² .

- After all quantity sets have been built, assign them a more understandable name as follows:

- Select all quantity sets.
- Point with the mouse on the quantity sets and do a right click.
- From the popup menu choose Rename automatically. The names of the added quantities and the sender code will be used to rename the quantity sets which leads to quantity sets **Force<-Codename**, **PointPosition<-Codename** and optionally **Acceleration<-Codename**.

- 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<-Codename
	PointPosition<-Codename
	If coupling acceleration
	Acceleration<-Codename

For coupling with MATLAB

For selected regions	Check quantity sets
lambda	Force<-Codename
disp1	PointPosition<-Codename
	If coupling acceleration
acc1	Acceleration<-Codename

Proceed to the Monitors Step by pressing **[Monitors Step >]**.

12.3.3 Monitors Step

No changes are required in the Monitors Step. Proceed to the Edit Step by pressing **[Edit Step >]**.

12.3.4 Edit 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:

Parameter	Value
Properties.Monitor.Inorm	Set selected.

If **explicit coupling with non-matching time step sizes** is used, different interpolation methods may lead to different results. Three interpolation methods are available: Higher order, First order and Zero order interpolation (cf. ▷ 12.5.2 Non-matching Time Step Sizes ▷). For setting the interpolation method set the following parameter:

Parameter	Value
Properties.Job.Interpolation	Select one of Higher order, First order and Zero order.

No further changes are required in the Edit Step. Proceed to the Go Step by pressing **[Go Step >]**.

12.3.5 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 Start-Up and Starting Server and Codes ▷).

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**.

The settings are described in separate sections for iterative and explicit coupling.

Iterative Coupling Note that quantities should have assigned a convergence check operator.

For Abaqus iterative coupling is supported from Abaqus 6.14.

For SIMPACK iterative coupling is not supported.

1. For your Model A code:

- Set Coupling configuration options which are the same for all Model A codes.

Option	Action
Coupling scheme	Select Implicit-Transient.
Initial quantities transfer	E.g. select exchange for a Jacobi-type coupling. Depending on this settings a Jacobi-type or Gauss-Seidel-type iterative coupling is applied (cf. ▷ V-3.4.4 Iterative Coupling).

- Set options which are code specific.

MATLAB

Iteration control	Expand and access further options.
Maximum number of coupling iterations	Set to 10.
Coupling time step size	Set to 2.655E-3 s (The value of the time step for the implicit coupling defined in the m-file by the variable <code>dt</code>).

 The MATLAB file selected in the Models Step must fit to iterative coupling and handling of acceleration quantity.

SimulatorA

Option	Action
Iteration control	Expand and access further options.
Maximum number of coupling iterations	Set to 10.
Coupling time step size	Set to 2.655E-3 s.

...

2. For your Model B code:

- Set Coupling configuration options which are the same for all Model B codes.

Option	Action
Coupling scheme	Select Implicit-Transient.
Initial quantities transfer	E.g. select exchange for a Jacobi-type coupling. Depending on this settings a Jacobi-type or Gauss-Seidel-type iterative coupling is applied (cf. ▷ V-3.4.4 Iterative Coupling).

- Set options which are code specific.

Abaqus

Option	Action
Iteration control	Expand and access further options.
Maximum number of coupling iterations	Set to any value greater than 10.
Coupling time step size	Set to 2.655E-3 s (or to the value of the time step for the implicit coupling).

MATLAB

Iteration control	Expand and access further options.
Maximum number of coupling iterations	Set to 10.
Coupling time step size	Set to 2.655E-3 s (The value of the time step for the implicit coupling defined in the m-file by the variable dt).

 The MATLAB file selected in the Models Step must fit to iterative coupling and handling of acceleration quantity.

MSC Adams

Option	Action
Iteration control	Expand and access further options.
Maximum number of coupling iterations	Set to any value greater than 10. This option has no effect on MSC Adams except that it forces the end of the iterative coupling after the defined maximum number of iterations is reached.
Coupling time step size	Set to 2.655E-3 s (or to the value of the time step for the implicit coupling).
No. of parallel threads	Increase for faster simulation.

SimulatorB

Option	Action
Iteration control	Expand and access further options.
Maximum number of coupling iterations	Set to 10.
Coupling time step size	Set to 2.655E-3 s.

Explicit Coupling

1. For your Model A code:

- Set Coupling configuration options which are the same for all Model A codes.

Option	Action
Coupling scheme	Select Explicit-Transient.
Initial quantities transfer	E.g. select exchange for a Jacobi-type coupling. Depending on this settings a Jacobi-type or Gauss-Seidel-type iterative coupling is applied (cf. ▷ V-3.4.4 Iterative Coupling ◁).

- Set options which are code specific.

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.

SimulatorA

Option	Action
Time step size	Set to e.g. 2.655E-3.
Coupling time step size	Set to 2.655E-3 s. Different time steps can be used for the simulators (cf. ▷ 12.5.2 Non-matching Time Step Sizes ◁).

...

2. For your Model B code:

- Set Coupling configuration options which are the same for all Model B codes.

Option	Action
Coupling scheme	Select Explicit-Transient.
Initial quantities transfer	E.g. select exchange for a Jacobi-type coupling. Depending on this settings a Jacobi-type or Gauss-Seidel-type iterative coupling is applied (cf. ▷ V-3.4.4 Iterative Coupling ◁).

- Set options which are code specific.

Abaqus

Option	Action
Constant coupling time step	Check this option to use a constant time step.
Coupling time step	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. ▷ 12.5.2 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.

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

 See [▷ VI-11.2.9 Go Step ◄](#) for details on the settings in the MSC Adams Go Step.

Option	Action
Iteration control	Expand and access further options.
Maximum number of coupling iterations	Set to any value greater than 10. This option has no effect on MSC Adams except that it forces the end of the iterative coupling after the defined maximum number of iterations is reached.
Coupling time step size	Set to 2.655E-3 s (or to the value of the time step for the implicit coupling).
No. of parallel threads	Increase for faster simulation.
Use semi-implicit interpolation	Check and access further options (see ▷ VI-11.2.8.1 Semi-implicit Mode ◄ for details).
Communication delay of the co-simulation	Set to 1 because we have explicit coupling.
Synchronized history buffer update	Check and access further options.
Time step for buffer update	Set to 2.655E-3.
Use constant mass	Check and access further options.
Value of the mass coefficient	Set to 0.25. This value corresponds to the mass of 2.5 kg used for the simulation multiplied by β and is the mass on the Model B side.
Provide partials to MSC Adams	Check and access further options (see ▷ VI-11.2.8.2 Partial Derivatives in MSC Adams ◄ for details).
Minimum value for derivatives	Set to -100E6.

SIMPACK

Option	Action
User library configuration	Check and access further options.
Use default uforce20	Set selected.
SIMPACK runtime configuration	Check and access further options.
Provide partials	Check and access further options.
Communication delay of the co-simulation	Set to 1 because we have explicit coupling.
Synchronized history buffer update	Check and access further options.
Time step for approximation update	Set to the step size of the co-simulation partner, e.g. 2.655E-3.

SimulatorB	
Option	Action
Time step size	Set to e.g. 2.655E-3.
Coupling time step size	Set to 2.655E-3 s. Different time steps can be used for the simulators (cf. ▷12.5.2 Non-matching Time Step Sizes◀).
...	

12.4 Running the Computation

12.4.1 Starting the Simulation

Save the MpCCI project file with name "springmass.csp" over the MpCCI GUI menu

File→Save Project As.

Press the three Start buttons in the Go Step and the simulation codes should start. No additional actions are required.

12.5 Discussion of Results

12.5.1 Iterative Coupling compared to Explicit Coupling

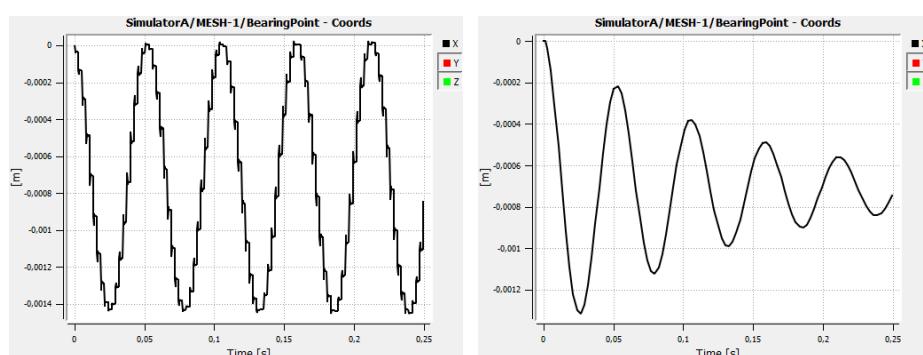


Figure 2: Comparison of implicit (left) and explicit (right) coupling results

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 β and small time steps are fine. Only for larger values of β 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 β 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.

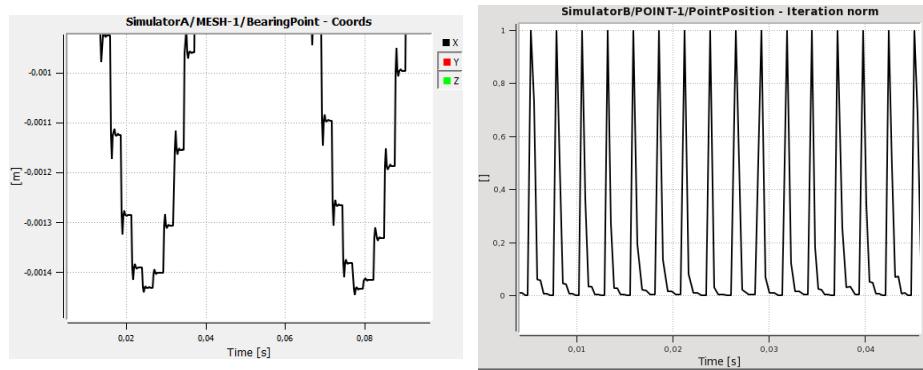


Figure 3: Implicit coupling in more detail: displacement of the bearing point (left) and relative variation of inner iterations (right), used to judge the convergence

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.

12.5.2 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. 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 selection in the Edit 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.

If both time steps are small enough and quite close to each other, a linear interpolation is able to reproduce the oscillations.

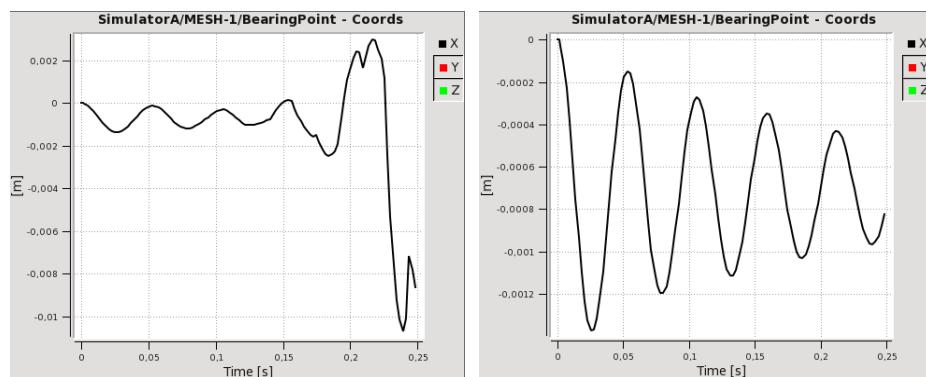


Figure 4: Non-matching time steps (2.3E-3 for SimulatorA and 2.655E-3 for SimulatorB) and different interpolation methods for the spring mass example: displacement for SimulatorA using linear interpolation (left) and higher order interpolation (right).

13 Periodic Rotating Fan Model

13.1 Problem Description

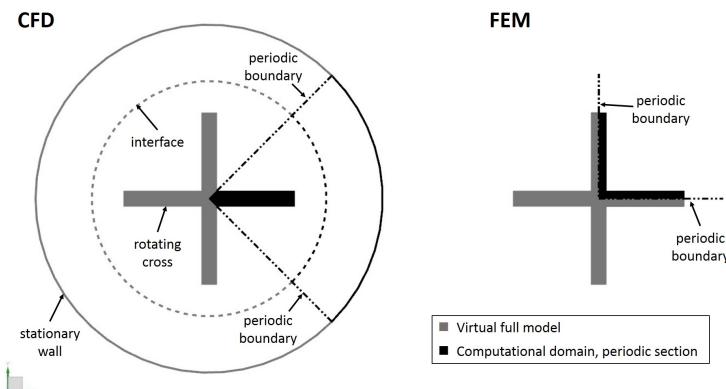


Figure 1: Rotating cross as periodic model

Topics of this Tutorial

- Coupling of periodic models
- Ramping
- Fluid-Structure Interaction (FSI)
- 2D-model

Simulation Codes

- Abaqus 6.12 or higher
- FLUENT 13.0 or higher

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.

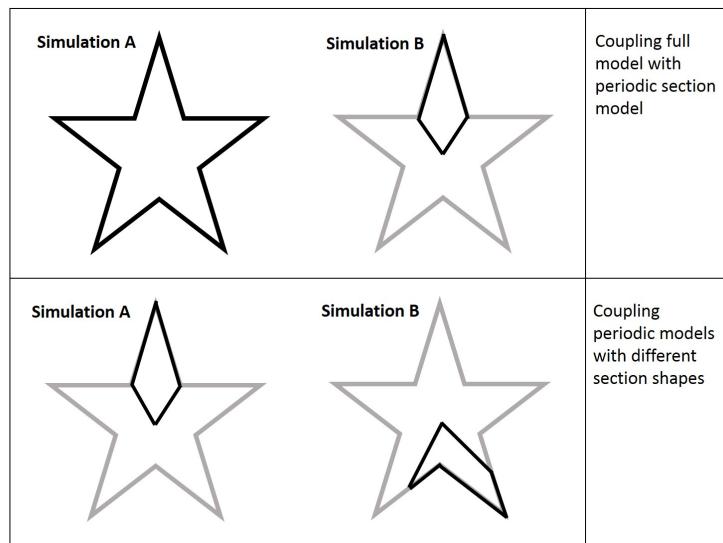


Figure 2: Geometrical coupling schemes for periodic models

13.2 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*".

13.2.1 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 $\mu = 1.7894 \cdot 10^{-5} \frac{\text{kg}}{\text{m s}}$. 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.

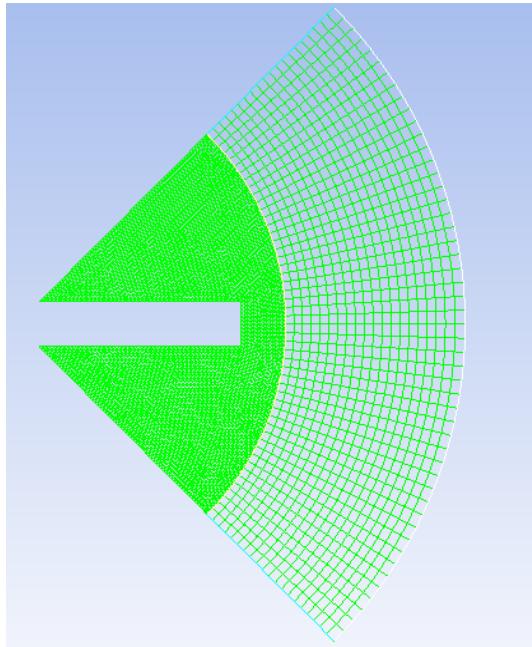


Figure 3: Periodic fluid mesh

13.2.2 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 $\rho = 78 \text{ kg/m}^3$, elastic modulus 0.5 MPa and Poisson's ratio $\nu = 0.3$.

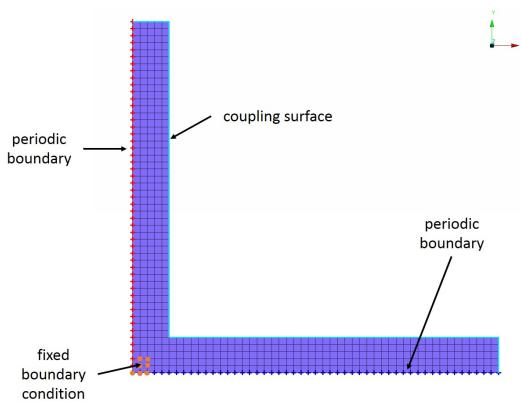


Figure 4: Periodic solid mesh

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.

13.3 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`.

13.3.1 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

Option	Action
Code to couple	Select FLUENT.
Version	The model is two-dimensional, thus select the FLUENT version 2ddp.
Release	Should be set to latest or a version supported by MpCCI (see VI-8 FLUENT).
Case file	Select the case file "FLUENT/fan_period1Arm.cas".
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

...

2. Choose your solid mechanics code as second code to couple:

Abaqus

Option	Action
Code to couple	Select Abaqus.
Release	Should be set to latest or a version supported by MpCCI (see VI-2 Abaqus).
Input file	Select input file "ABAQUS/fan_period2halfArms.inp".
Unit system	Select the SI unit system (which is the standard).
Start scanner	Press the button to scan the model file. A green check mark should appear which means everything is set up correctly.

...

Press the `Coupling Step >` button at the bottom of the Models Step to get to the Coupling Step.

13.3.2 Coupling Step

In the Coupling 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-2.5 Coupling Step – Defining Coupling Regions and Quantities](#)).

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-4.5.2.3 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:

Option	Action
Number of periodicity	Set to 4 which is the number of periodic passages which build the virtual full model.
Define the cyclic symmetry axis by the two following specifications.	
Rotation-axis origin	Set to X=0., Y=0., Z=0..
Rotation-axis direction	Set to X=0., Y=0., Z=1..

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 (Δ) for each components list, select the periodic components and apply the periodicity.

For fluid mechanics code	Select component
FLUENT	Δ turbo
For selected components	
For solid mechanics code	Select component
Abaqus	Δ BLADE-WALL
For selected components	
Option	Action
Apply periodicity to selected components	Press the button to apply the selected periodicity quarter to the selected components.

Note the change of the symbol in front of the selected component names to Δ .

3. Now select Build Regions tab and here the Setup tab. Choose the components to be coupled from the Face tabs (Δ) 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:

For region Region_1	
For fluid mechanics code	Select coupling component
FLUENT	Δ turbo
For selected components	
For solid mechanics code	Select coupling component
Abaqus	Δ BLADE-WALL

4. Select Define Quantity Sets tab. As currently no quantities are assigned to “QuantitySet_1” (created by default), a warning symbol is displayed in front of it. Now choose the quantities to be exchanged.

For quantity set	Select quantities	Set sender
QuantitySet_1	NPosition	Solid mechanics code
	RelWallForce	Fluid mechanics code

5. Add the Ramping operator to the quantity NPosition (cf. [IV-4.5.7.4 Applying Operators to Mesh Based Components](#)) to get an accelerated convergence.

Thus mark NPosition in QuantitySet_1 by clicking on its name and configuring the listed operators as follows:

Quantity	Operator	Operator option	Operator action
NPosition	Select Relaxation.	Relaxation method	Select Ramping.
		Initial ramp value	Set to 0.1 (default).
		Ramp delta	Set to 0.1 (default).

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.

For selected region	Check quantity set
Region_1	QuantitySet_1

Proceed to the Monitors Step by pressing **[Monitors Step >]**.

13.3.3 Monitors Step

No changes are required in the Monitors Step, proceed to the Edit Step by pressing **[Edit Step >]**.

13.3.4 Edit 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.

Parameter	Value
Properties.Monitor.Periodic	Set selected to display the replicated periodic parts and quantities.
Properties.Output.Tracefile.Writers.CCVX.Periodic	Set selected to include the replicated periodic parts and quantities.

Proceed to the Go Step by pressing **[Go Step >]**.

13.3.5 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 Start-Up and Starting Server and Codes](#)).

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:

- Set Coupling configuration options which are the same for all fluid mechanics codes.

Option	Action
Coupling scheme	Select Explicit-SteadyState.
Initial quantities transfer	Select exchange.

- Set options which are code specific.

FLUENT

Option	Action
Use subcycling	Check and access further options.
No. of steps without coupling	Set to 100.

...

2. For your solid mechanics code:

- Set Coupling configuration options which are the same for all solid mechanics codes.

Option	Action
Coupling scheme	Select Explicit-SteadyState.
Initial quantities transfer	Select receive.

- Set options which are code specific.

Abaqus

Option	Action
Constant coupling time step	Check this option to use a constant time step.
Coupling time step	Set to 0.001 for one increment size for the steady state analysis.

MpCCI will run the simulation not on the original model file, but rather will use a copy with the prefix "abaqus_run" which is set as default job name.

...

13.4 Running the Computation

Save the MpCCI project file with name "periodic.csp" over the MpCCI GUI menu **File→Save Project As**. Press the three **Start** buttons 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 **Solve→Initialization** resp. **Solution→Solution Initialization** and
3. Select **Solve→MpCCI Run FSI** resp. **MpCCI→MpCCI Run FSI**. Set the Number of Iterations to 5000

and press **Run** to start the simulation. With 100 subiterations there will be 50 couplings.

13.5 Discussion of Results

Since the parameter **Periodic** was enabled for the monitor in Edit Step, the periodic parts are shown with their virtual replicates, cf. [Figure 5](#).

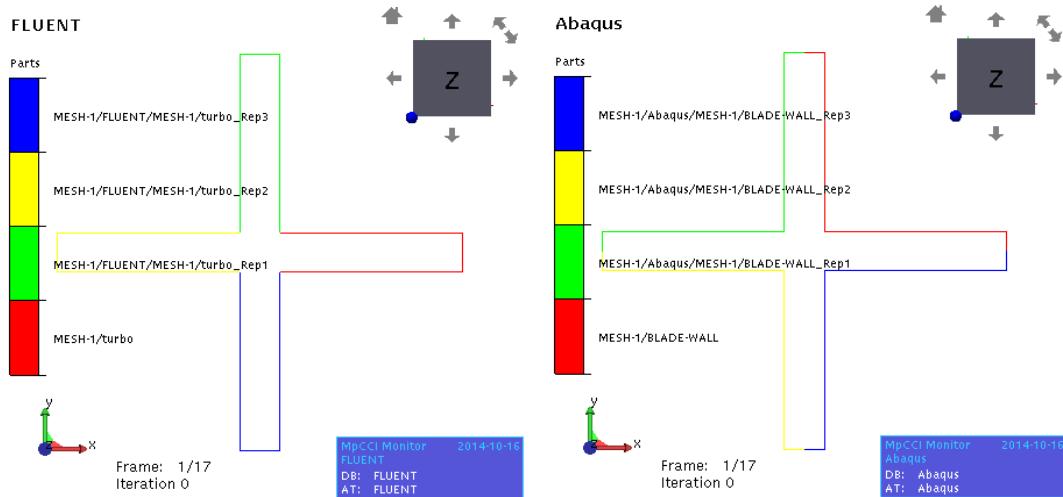


Figure 5: Visualizing the virtual full coupling surfaces in monitor

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".

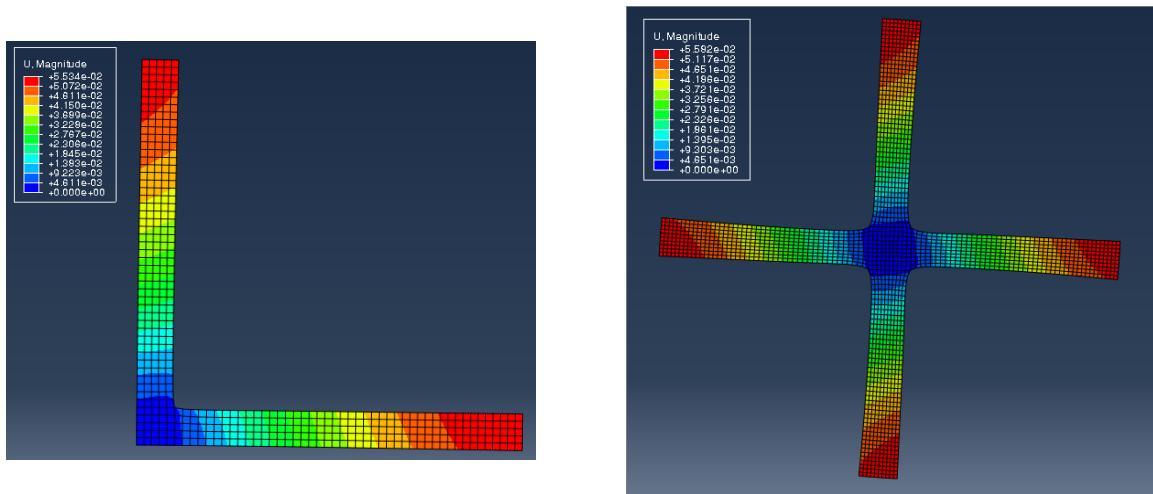


Figure 6: Comparison of coupled deformations on periodic and full Abaqus models

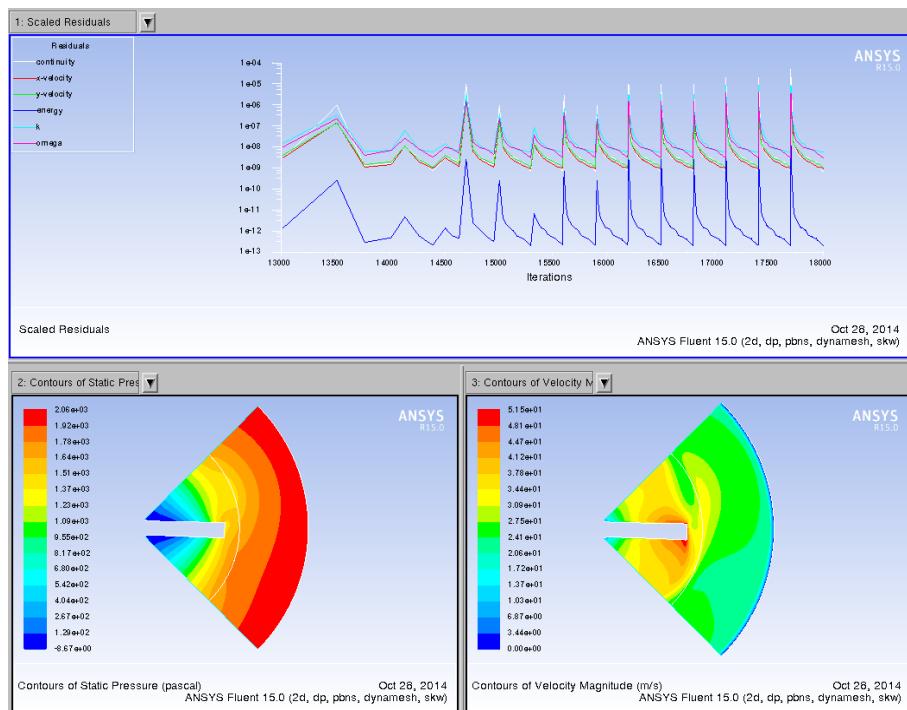


Figure 7: Pressure and velocity as coupling results on periodic FLUENT model with convergence history

FLUENT

The FLUENT result file is "fan_period1Arm.cas".

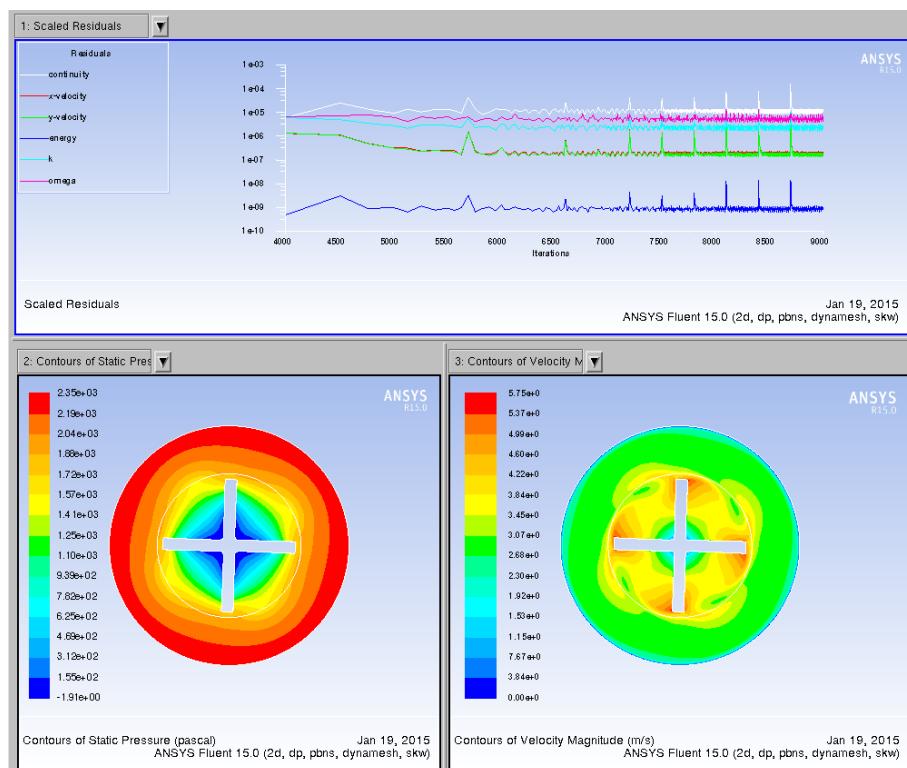
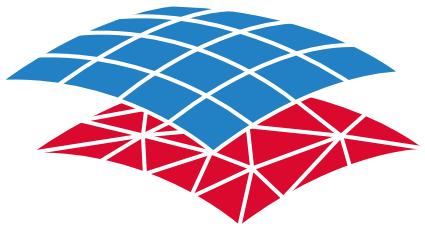


Figure 8: Pressure and velocity as coupling results on full FLUENT model with convergence history



MpCCI
CouplingEnvironment

Part VIII

Programmers Guide

Version 4.5.0

MpCCI 4.5.0-1 Documentation
Part VIII Programmers Guide
PDF version
March 30, 2017

MpCCI is a registered trademark of Fraunhofer SCAI
www.mpcci.de



Fraunhofer Institute for Algorithms and Scientific Computing SCAI
Schloss Birlinghoven, 53754 Sankt Augustin, Germany

Abaqus and SIMULIA are trademarks or registered trademarks of Dassault Systèmes
ANSYS, FLUENT and ANSYS Icepak are trademarks or registered trademarks of Ansys, Inc.
Elmer is an open source software developed by CSC
FINE/Open and FINE/Turbo are trademarks of NUMECA International
Flowmaster is a registered trademark of Flowmaster Group NV
JMAG is a registered trademark of The JSOL Corporation
MATLAB is a registered trademark of The MathWorks, Inc.
MSC Adams, MSC Marc, MD NASTRAN and MSC NASTRAN are trademarks or registered trademarks of
MSC.Software Corporation
OpenFOAM is a registered trademark of OpenCFD Ltd.
PERMAS is a registered trademark of Intes GmbH
RadTherm, TAITherm is a registered trademark of ThermoAnalytics Inc.
SIMPACK is a registered trademark of SIMPACK AG.
STAR-CCM+ and STAR-CD are registered trademarks of CD adapco Group

ActivePerl has a Community License Copyright of Active State Corp.
FlexNet Publisher is a registered trademark of Flexera Software.
Java is a registered trademark of Oracle and/or its affiliates.
Linux is a registered trademark of Linus Torvalds
Mac OS X is a registered trademark of Apple Inc.
OpenSSH has a copyright by Tatu Ylonen, Espoo, Finland
Perl has a copyright by Larry Wall and others
UNIX is a registered trademark of The Open Group
Windows, Windows XP, Windows Vista and Windows 7 are registered trademarks of Microsoft Corp.

VIII Programmers Guide – Contents

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	Go Step: <GoMenuEntries>	26
2.4.7	Environments for Scanner, Checker, Starter, Stopper and Killer	28
2.4.8	General MpCCI GUI Elements	28
2.4.9	Testing gui.xcf	34
2.5	Perl Scripts	35
2.5.1	Using Information from gui.xcf in Scripts	35
2.5.2	Scanner.pm	35
2.5.3	Checker.pm	36
2.5.4	Starter.pm	37
2.5.5	Stopper.pm	37
2.5.6	Killer.pm	38
2.5.7	Info.pm	38
2.5.8	Subcmd.pm	38
2.5.9	Testing the Perl Scripts	39
2.6	MpCCI Adapter Implementation	40
2.6.1	How to Initialize the Code?	40
2.6.2	How to Define the Mesh?	41
2.6.3	How to Deal with Angular Coordinates?	42
2.6.4	How to Transfer Data?	43
2.6.5	How to Terminate the Coupling?	43

2.6.6	How to Notify a Remeshing?	43
2.7	MpCCI Coupling Manager Functions	44
2.7.1	Definition of Output Functions: <code>umpcci_msg_functs</code>	45
2.7.2	Definition of Output Prefix: <code>umpcci_msg_prefix</code>	46
2.7.3	Get Transfer Information: <code>ampcci_tinfo_init</code>	47
2.7.4	Connect and Initialize an MpCCI Server: <code>mpcci_init</code>	48
2.7.5	Configure the Code Adapter: <code>ampcci_config</code>	49
2.7.6	Definition of Part: <code>smpcci_defp</code>	50
2.7.7	Delete a Part: <code>smpcci_delp</code>	52
2.7.8	Definition of Nodes: <code>smpcci_pnod</code>	53
2.7.9	Definition of Elements: <code>smpcci_pels</code>	54
2.7.10	Definition of the Moving Reference Frame: <code>smpcci_pmot</code>	56
2.7.11	Definition of the Baffle thickness: <code>smpcci_pshf</code>	57
2.7.12	Data Exchange: <code>ampcci_transfer</code>	58
2.7.13	Notifying the Remeshing: <code>ampcci_remesh</code>	60
2.7.14	End of Coupled Simulation: <code>mpcci_quit</code> and <code>mpcci_stop</code>	61
2.8	MpCCI Driver Functions	62
2.8.1	Description Values	65
2.8.2	Driver Methods Called before/after some Action	66
2.8.3	Driver Mesh Definition Methods	68
2.8.4	Driver Data Exchange Methods	70
2.9	Data Structures and Predefined Macros	71
2.9.1	Supported Element Types	71
2.9.2	Coordinates System Definition	80
2.9.3	Mesh Dimension Definition	80
2.9.4	Moving Reference Frame Definition	80
2.9.5	Remesh Flag Information	81
2.9.6	Transfer Information: <code>MPCCI_TINFO</code>	81
2.9.7	Code Specific Information: <code>MPCCI_CINFO</code>	87
2.9.8	Coupling Components	89
2.9.9	Quantities	90
2.9.10	Loop Functions	91
2.9.11	Memory Management	93

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

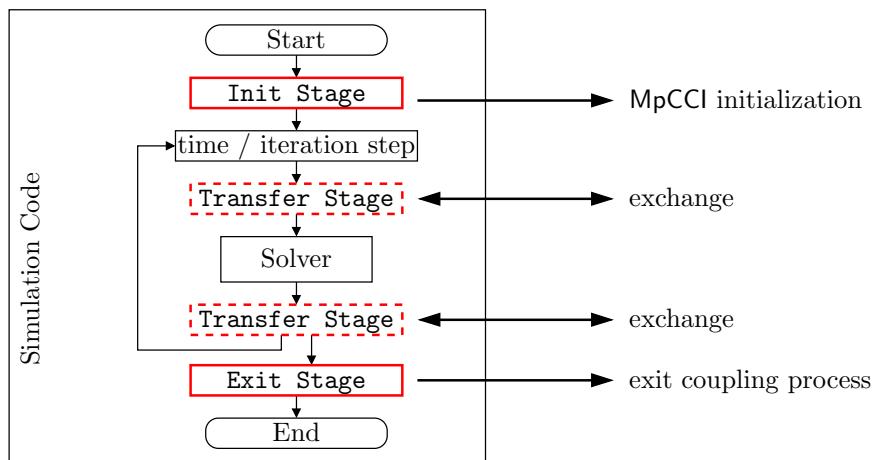


Figure 1: Calls of adapter library routines from the simulation code. The `Transfer Stage` can be inserted before or after the solver.

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

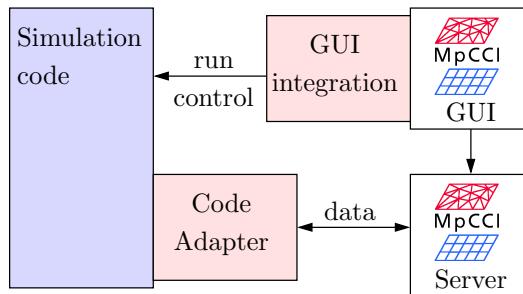


Figure 2: The two basic parts needed for code integration: GUI integration and code adapter

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 21](#).

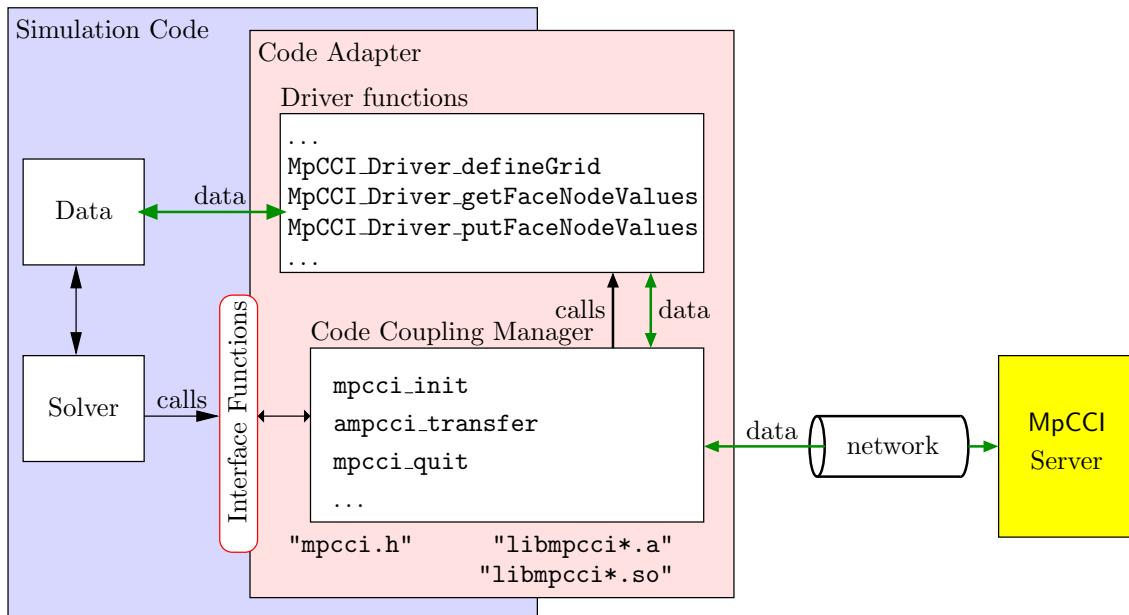


Figure 3: Code adapter structure: The code adapter is a plug-in of the analysis code, which consists of two parts: Driver functions and Code Coupling Manager

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:

Directory	Description
<code>"adapter"</code>	Templates for code adapter
<code>"configuration"</code>	Templates for configuration directory
<code>"example/configuration"</code>	Configuration directory files for the example
<code>"example/C/src"</code>	Source code of the example in C
<code>"example/C/src-nompcci"</code>	Source code of the example without code adapter in C
<code>"example/FORTRAN/src"</code>	Source code of the example in FORTRAN
<code>"example/FORTRAN/src-nompcci"</code>	Source code of the example without code adapter in FORTRAN
<code>"example/test"</code>	Files for testing the example
<code>"example/test-nompcci"</code>	Files for testing the example without code adapter

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/FORTRAN/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".

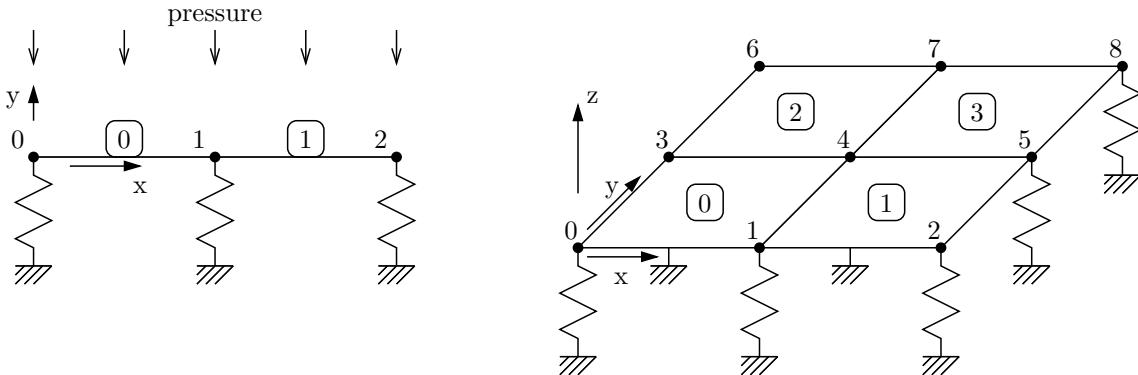


Figure 4: Examples for the simple foundation code.

The structure modeled in "2dexample.fnd" is depicted in [Figure 4](#) on the left, its content is:

```
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:

```
> ./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
```

```

node 1: force=      1      displacement=     -2
node 2: force=      0.5    displacement=     -1

Step 1 of 3
Please enter pressure: 2

results for foundation 'bed1':
node 0: force=      1      displacement=     -2
node 1: force=      2      displacement=     -4
node 2: force=      1      displacement=     -2

```

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 32 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`".

- ➊ 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 creation of a code adapter can be performed on a step-by-step basis:

- Step 1: Preparations.
- Step 2: Create the configuration directory.
- Step 3: Test the configuration.
- Step 4: Create the code adapter.
- Step 5: Test the code adapter.

(i) For each step, the corresponding changes in the example are mentioned as well. To understand the changes you can compare (“diff”) the files with the original versions:

original	new
"configuration/*"	"example/configuration/*"
"example/C/src-nompcci/data.h"	"example/C/src/data.h"
"example/C/src-nompcci/data.c"	"example/C/src/data.c"
"example/C/src-nompcci/main.c"	"example/C/src/main.c"
"example/C/src-nompcci/Makefile"	"example/C/src/Makefile"
"adapter/adapter.h"	"example/C/src/adapter.h"
"adapter/adapter.c"	"example/C/src/adapter.c"

All changes are commented and marked with `CHANGED:`. (For FORTRAN see the corresponding files in "example/FORTRAN".)

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 model files.

 The example “foundation” code computes displacements due to an external pressure. Thus the following quantities will be exchanged:

foundation code (elements) \leftarrow `OverPressure` \leftarrow partner code
foundation code (nodes) \rightarrow `NPosition` \rightarrow 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 will later be passed as a command line argument to the code.

The Scanner only needs to know the model file, the Starter also gets the number of steps, the stopper gets the model file name.

Perl scripts: "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 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 "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.

To run a test, you must provide the model file to scan. The code has a list of available command line option, see `mpcci foundation help`.

```
mpcci foundation scan /home/fred/adapter-test/example.input
```

The results of the scanning process are saved in a scan file "`mpcci_<original filesystem>.scan`", i. e. for the above example the file would be named

`"/home/fred/adapter-test/mpcci_example.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`".

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 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, and press the `[Start Scanner]`. → The scanner should start, otherwise you will receive an error message. If the scanner finished, click on the box which appears and check the return values of the scanner.
5. Select and scan the partner code and proceed to the Coupling Step. → Here, you should be able to select the quantities as defined in "gui.xcf". If you cannot select anything 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 buttons 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 “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.

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 [▷ 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 [▷ 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".
- If your code is compiled and linked, you can proceed with testing the adapter.

 For “foundation” the most important changes are (please see source code files for details):

Driver Functions Only a small set of driver functions,

```
MpCCI_Driver_partUpdate,  
MpCCI_Driver_definePart,  
MpCCI_Driver_getFaceNodeValues  
and 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 Edit 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.

(i) For the foundation code, the code's minimal output (in the yellow window) should look as follows:

```
Starting: foundation elasticwall.fnd 2 1> 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 transfer configuration setting:
MpCCI:   Initial action : receive
MpCCI:   Check mode    : none
MpCCI:   Send mode    : always
MpCCI:   Receive mode : all
MpCCI:   Start time    : -1.000000
MpCCI:   End time     : -1.000000
MpCCI:   Time step size : -1.000000
MpCCI:   Start iteration: -1
MpCCI:   End iteration  : -1
MpCCI:   Iteration step : -1
```

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 for the first exchange the value of `Initial quantities transfer` is used, 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_adapter_foundation: feature test done.
[MpCCI License] mpcci_clients: feature checkout...
[MpCCI License] mpcci_clients: feature checked out.
```

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:

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

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
Starting closePoints search loop...
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.

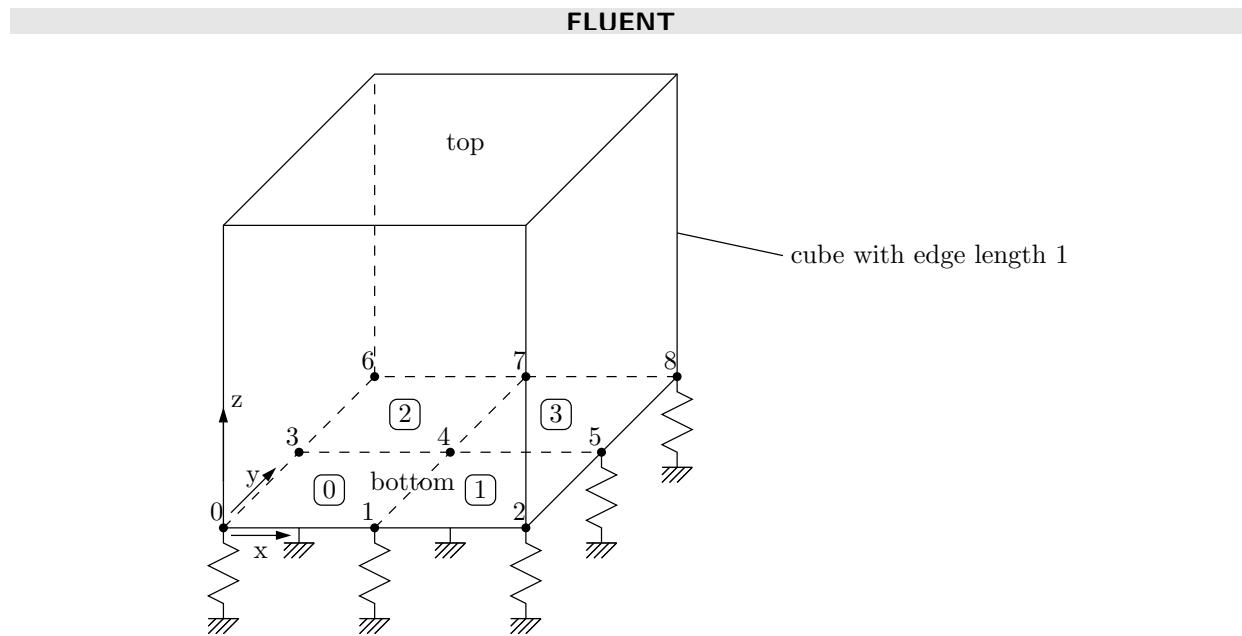


Figure 5: FLUENT – foundation sample problem. A cube filled with an ideal gas is coupled with an elastic foundation which deforms due to the gas pressure.

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 the MpCCI 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)
#
# elasticwall      Face
```

- Select FLUENT as the second code, the FLUENT version 3d, and the latest FLUENT release. Finally choose "FLUENT/cube.cas" as model file.
- Start the Scanner for FLUENT.

- Proceed to the Coupling Step – only the Face (2D)-card should be enabled. Select `elasticwall` and `bottom` as coupling components and `NPosition` and `OverPressure` as quantities.
- Proceed to the Go Step. Select `Run` server processes inside `xterm` for the server. For “foundation” select `receive` for the initial transfer and 3 as number of steps. For `FLUENT` select `exchange` as initial transfer and 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 `Solve`→`Initialize` →`Initialize` and press the `Init` to initialize the `FLUENT` solution.
- Select `Solve`→`Iterate` and set the Number of Time Steps to 3. and press `Iterate`.
- 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.

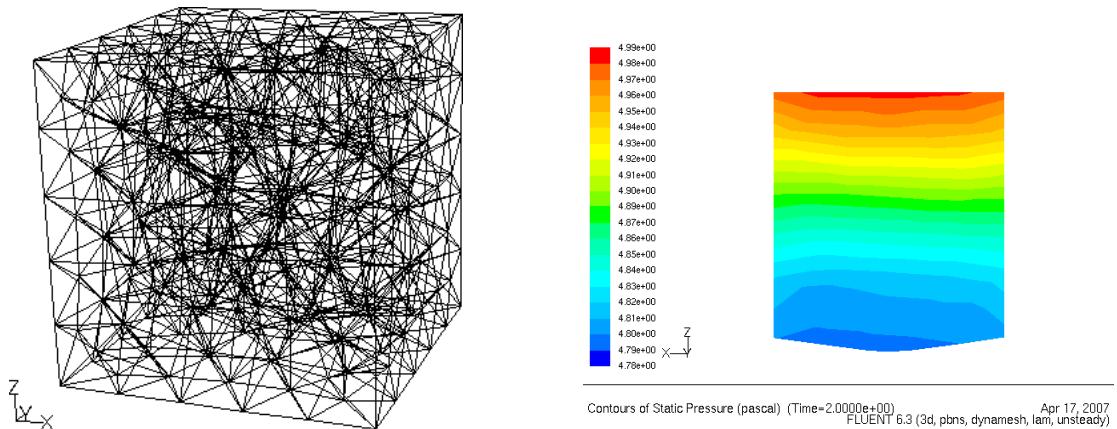


Figure 6: FLUENT mesh and result of the cube example.

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 ◁](#)

"Subcmd.pm" can be used to define MpCCI code specific subcommands.

[▷ 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 [▷ 2.4.8 General MpCCI GUI Elements ◁](#).

2.4.1 Code Information: <CodeInfo>

```
<CodeInfo>
  <Units           type="string" default="SI" />
  <Type            type="string" default="CFD Fluid FluidThermal FluidPlasma" />
  <SelfCoupler     type="bool"   default="true"  />
  <CopyComponents  type="bool"   default="true"  />
</CodeInfo>
```

The code information block contains the basic information of a code:

Units Unit system used by the code

Type Type of code - needed to determine possible coupling types

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)

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.

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.

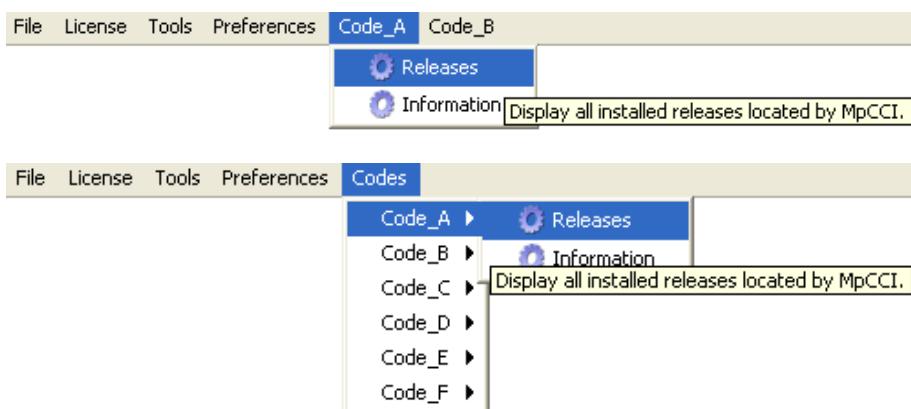


Figure 7: Codes menus for two (Code_A and Code_B) and more than three codes

Each provided command is defined as a menu element

```
<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 [▷ 2.4.8 General MpCCI GUI Elements](#).

The first element in the example is an enumeration element

```
<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:

```
<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](#).

```
<ModelsMenuEntries>
  <Release type="enum" default="latest"
    sort="false" description="Select release">
    <enum value="latest" />
    <enum value="1.1" />
    <enum value="1.0" />
  </Release>

  <ModelFile type="filename" required="true"
    default="" description="Select model file:" >
    <filename suffix=".inp" />
    <filename suffix=".inp.gz" />
  </ModelFile>
</ModelsMenuEntries>
```

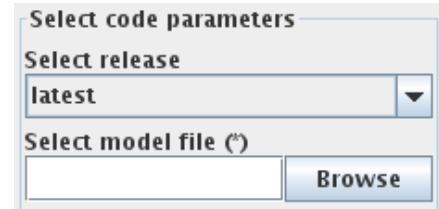


Figure 8: Example of a `<ModelsMenuEntries>` definition in "gui.xcf" and the resulting Models Step.

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.

```
<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. :

```
<Temperature type="quantity" loc="node elem" so="Direct" ro="Direct"
             valid="Point Line Face Volume"/>
```

- 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 ▷ [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="global" so="Direct" ro="Direct" />
  <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 Go Step: <GoMenuEntries>

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>`:

```
<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.

```
<_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`.

```
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.

2.4.7 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. [▷ 2.4.6 Go Step: <GoMenuEntries>](#)).

There are two classes of values:

Required values must be set. These are

Value	Variable name	Required for script
The name of the model file	_MPCCI_MODEL_FILE	Scanner, Checker, Starter, Stopper, Killer
Value for initial exchange (see ▷ V-3.4 Coupling Algorithms)	MPCCI_TINFO	Starter, Checker

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 [▷ 2.4.8 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:

```
<Starter>
  <Environment>
    <MPCCI_TINFO           type="string" default="%({GoMenuEntries.MpCCITinfo.....})"/>
    <_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>
```

2.4.8 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

```
<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

File Selector

```
<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="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 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.

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:

```
<Identifier type="enum" default="SI" description="Select unit system">
    <enum value="British"/>
    <enum value="cgs"/>
    <enum value="variable"/>
</Identifier>
```

`enum value` is one value of the value list.

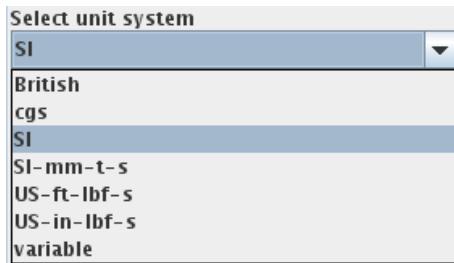


Figure 11: Enumeration representation

2. Source file:

```
<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:

```
<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`.

```
<Identifier type="int" default="1"
            min="1" max="512"
            description="No. of parallel processes"/>

<Identifier type="float" default="0.01"
            min="1e-10" max="1e10"
            description="Coupling time step"/>
```

`default` is the initial default value.

`min` defines the lower limit.

`max` defines the upper limit.

The `min` and `max` options are optional. If one of them lacks the minimum respectively maximum value of the system will be taken. If neither `min` or `max` is given the value is set to the default value and may not be changed.



Figure 12: Range value representation for int and float types

Checkbox

```
<Identifier type="bool" default="true"
            description="Start additional control process"/>
```

default must be initialized with true or false.



Figure 13: Checkbox representation

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.

```
<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.



Figure 14: Command representation

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”.

```
<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.

```
<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.

```
<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. [▷ 2.4.6 Go Step: <GoMenuEntries>](#), [▷ 2.4.7 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:

<code>%(<element>.<subelement>)</code>	value of an element specified within this configuration file.
<code>%(mpcciserver:<element>.<subelement>)</code>	the same as before but the element will be taken from the MpCCI server configuration file.
<code>%(properties:<element>.<subelement>)</code>	value of a property which can be edited in the Edit Step of the MpCCI GUI.
<code>%(#codename)</code>	references the instance name of this code which is also used in the MpCCI GUI to specify this code.
<code>\$(<environment variable name>)</code>	value of the specified environment variable on the local machine.

```
<Identifier type="enum" default="latest" description="Title"
            command="command_for_code %(#example)"
            hostref="%(ModelsMenuEntries.ModelFile)"/>

<BaffleOption type="enum" default="None"
               description="Define baffle thickness location"
               dependsOn="%(properties:BaffleShift.Enable)" dependingValue="true" >

<SampleDir type="filename" dironly="true" default="$(HOME)"
            description="Select directory for %(mpcciserver:GuiMenuEntries.Jobname)">
```

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.

```
<Identifier type="<element type>" default="<default value>" description="Title"
            dependsOn="%(ModelsMenuEntries.Units)" dependingValue="variable"
            dependingCondition="<true | false>"/>
```

<code>dependsOn</code>	is a reference to the element this element depends on.
<code>dependingValue</code>	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 <code> </code> (i.e.: <code>dependingValue="value 1 value 5"</code>)
<code>dependingCondition</code>	provides the condition for the comparison between the depending value and the referenced value. If the condition is <code>true</code> 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 <code>false</code> 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 <code>true</code> is taken when the <code>dependingCondition</code> 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.

```
<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`.

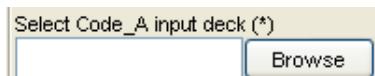


Figure 18: Required representation for a file selector element

Common Option: Visible

Another option which is common to all previously described MpCCI GUI elements is the visible option.

```
<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`.

2.4.9 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 [2.4.7 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 `MpCCICore::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 information will be printed as comment into the scanner output. For example:

```
Global variables => 'Autogenerated by the scanner'
Solver type      => 'Explicit'
```

%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 ▷ 2.4.7 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

`print` for writing it to `stdout` if it is an information messages and use

`warn` for writing it to `stderr` if it is a warning or error messages.

These messages will be gathered by MpCCI GUI and shown in one single dialog including all results from the Check action.

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:

option	default	description
STDOUT	null	What to do with data written to <code>stdout</code> , can be any of <code>xterm</code> (i.e. write to an <code>xterm</code>), <code>mpcci</code> (i.e. output handled by MpCCI), <code>null</code> (ignore output), combined with the keyword <code>file</code> if it shall also be written to a log file.
STDERR	null	Same as <code>STDOUT</code> , but for <code>stderr</code> .
STDLOG	<empty string>	Name of logfile
BACKGR	white	Background color of xterm.
FOREGR	black	Foreground color of xterm
ATSTART	<not set>	Reference to a Perl subroutine which is called before the code starts. You can define this routine in "Starter.pm".
ATEXIT	<not set>	Reference to a Perl subroutine which is called as soon as the code exits. You can define this routine in "Starter.pm".

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.

"Killler.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 ' <code>'STATIC'</code>). |
| 7. Adapter architecture | The adapter architecture token (should be equal to the architecture token of the application). |

Example:

```
$infoHash{$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:

\$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_msg_functs` ([▷ 2.7.1 Definition of Output Functions: `umpcci_msg_functs`](#))
2. `umpcci_msg_prefix` ([▷ 2.7.2 Definition of Output Prefix: `umpcci_msg_prefix`](#))
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. Before calling `mpcci_init` ([▷ 2.7.4 Connect and Initialize an MpCCI Server: `mpcci_init`](#)), an 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. `ampcci_config` ([▷ 2.7.5 Configure the Code Adapter: `ampcci_config`](#)) The list of driver functions must be passed to `ampcci_config` at this point.

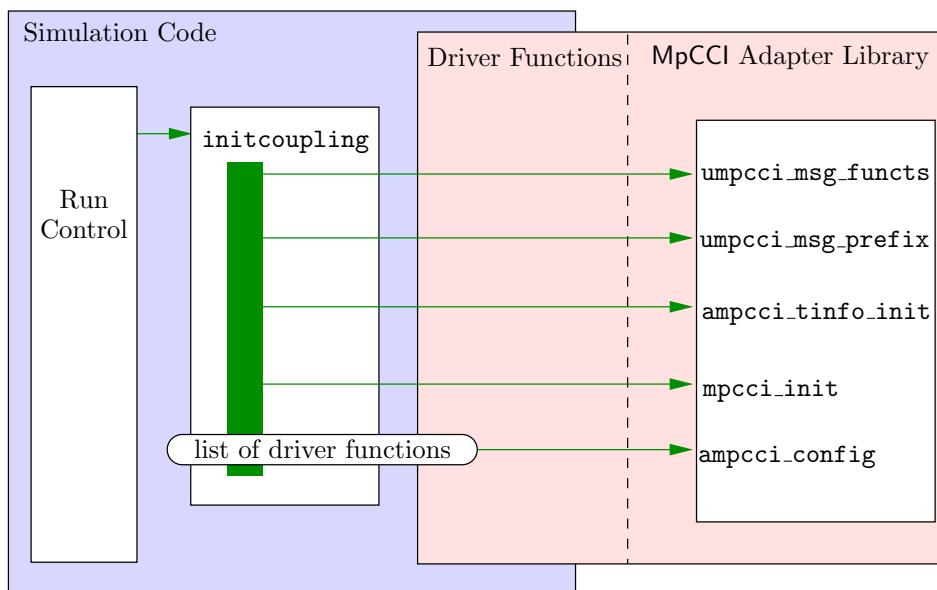


Figure 19: Call sequence for initializing the code with MpCCI

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 [▷ 2.8.3 Driver Mesh Definition Methods ◁](#).

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` ([▷ 2.7.6 Definition of Part: `smpcci_defp` ◁](#))
- `smpcci_delp` ([▷ 2.7.7 Delete a Part: `smpcci_delp` ◁](#))
- `smpcci_pnod` ([▷ 2.7.8 Definition of Nodes: `smpcci_pnod` ◁](#))
- `smpcci_pels` ([▷ 2.7.9 Definition of Elements: `smpcci_pels` ◁](#))
- `smpcci_pmot` ([▷ 2.7.10 Definition of the Moving Reference Frame: `smpcci_pmot` ◁](#))
- `smpcci_pshf` ([▷ 2.7.11 Definition of the Baffle thickness: `smpcci_pshf` ◁](#))

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 20](#)).

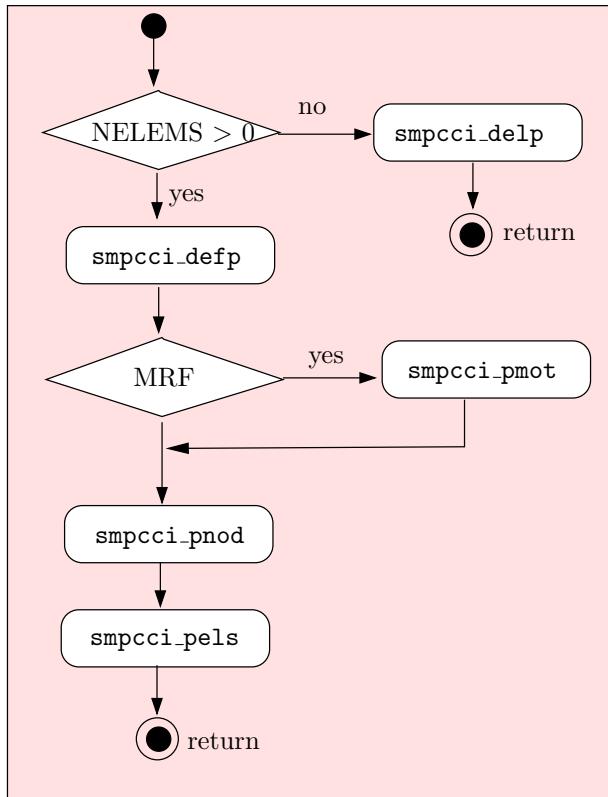


Figure 20: Call sequence for mesh definition using `MpCCI_Driver_definePart`

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.

- **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` ([▷ 2.7.12 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` ([▷ 2.7.14 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` ([▷ 2.7.13 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_functs` ◄ on page 45
- ▷ 2.7.2 Definition of Output Prefix: `umpcci_msg_prefix` ◄ on page 46
- ▷ 2.7.3 Get Transfer Information: `ampcci_tinfo_init` ◄ on page 47
- ▷ 2.7.4 Connect and Initialize an MpCCI Server: `mpcci_init` ◄ on page 48
- ▷ 2.7.5 Configure the Code Adapter: `ampcci_config` ◄ on page 49
- ▷ 2.7.6 Definition of Part: `smpcci_defp` ◄ on page 50
- ▷ 2.7.7 Delete a Part: `smpcci_delp` ◄ on page 52
- ▷ 2.7.8 Definition of Nodes: `smpcci_pnod` ◄ on page 53
- ▷ 2.7.9 Definition of Elements: `smpcci_pels` ◄ on page 54
- ▷ 2.7.10 Definition of the Moving Reference Frame: `smpcci_pmot` ◄ on page 56
- ▷ 2.7.11 Definition of the Baffle thickness: `smpcci_pshf` ◄ on page 57
- ▷ 2.7.12 Data Exchange: `ampcci_transfer` ◄ on page 58
- ▷ 2.7.13 Notifying the Remeshing: `ampcci_remesh` ◄ on page 60
- ▷ 2.7.14 End of Coupled Simulation: `mpcci_quit` and `mpcci_stop` ◄ on page 61

 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_funcs`

```
void umpcci_msg_funcs( 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.

```
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 quantitites 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`.

```
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 [▷ 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 Connect and Initialize an MpCCI Server: `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.

```
MPCCI_JOB mpcci_init(const char *porthost, MPCCI_CINFO *cinfo);
```

The arguments are:

`const char *porthost` The broker location `porthost`, only needed for coupling a code with a multi server configuration, should be `NULL` in other cases.

`MPCCI_CINFO *cinfo` Code information structure. This structure keeps code specific information. (See ▷ 2.9.7 Code Specific Information: `MPCCI_CINFO`).

The return value is an `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 `MPCCI_JOBID` if it is not statically defined by the adapter.
- Check the code id by using the environment `MPCCI_CODEID` if it is not statically defined by the adapter.
- Define the server host by using the environment `MPCCI_SERVER` if no broker is defined.
- Connect to the server.
- Get the type of grid to use for the grid definition. See ▷ 2.9.7 Code Specific Information: `MPCCI_CINFO` for the types.

2.7.5 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.

```
int ampcci_config(MPCCI_JOB *job, const MPCCI_DRIVER *driver);
```

The arguments are:

<code>MPCCI_JOB *job</code>	The MPCCI_JOB value returned after calling the <code>mpcci_init</code> containing the connection information to the MpCCI server.
-----------------------------	---

<code>const MPCCI_DRIVER *driver</code>	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.

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_partSelect` if `MpCCI_Driver_partUpdate` or `MpCCI_Driver_partInfo` is defined.
 - 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 21](#)), 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_getFaceNomals` and `MpCCI_Driver_getElemSize` if defined (see [Figure 22](#)).
- Define the mesh based and global quantities.
- List the information about coupled components and quantities.
- Call the driver function `MpCCIDriver_afterCloseSetup`.
- Return to the calling code.

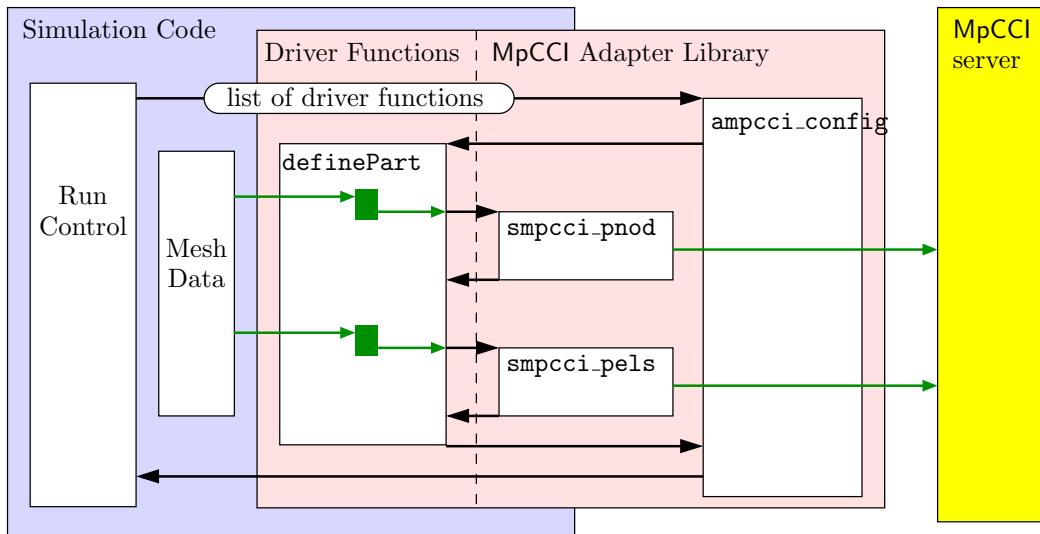


Figure 21: Call of ampcci_config: MpCCI_Driver_definePart is hooked

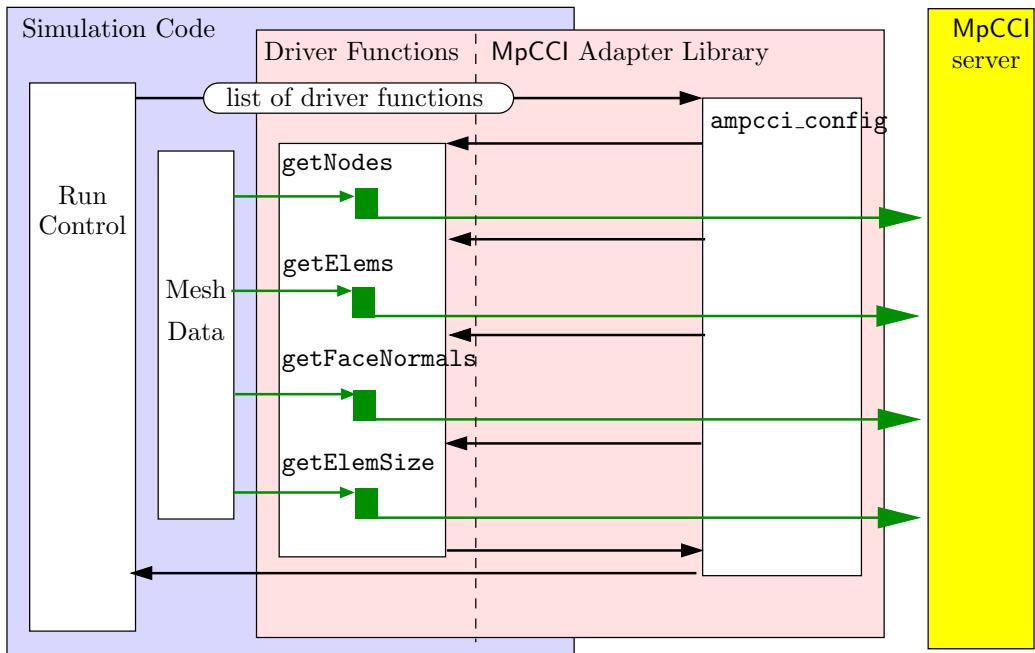


Figure 22: Call of ampcci_config: MpCCI_Driver_getNodes and MpCCI_Driver_getElems are hooked

2.7.6 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_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\(part\)](#).
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.

2.7.7 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:

```
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.

2.7.8 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:

```
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:

<code>MPCCI_SERVER *server</code>	The MpCCI server pointer passed by the <code>MpCCI_Driver_definePart</code> .
<code>int mid</code>	mesh ID, should be set to <code>MPCCI_PART_MESHID(part)</code> . The part is provided by <code>MpCCI_Driver_definePart</code> .
<code>int pid</code>	part ID, should be set to <code>MPCCI_PART_PARTID(part)</code>
<code>unsigned csys</code>	coordinates system of your mesh (see ▷ 2.9.2 Coordinates System Definition), should be set to <code>MPCCI_PART_CSYS(part)</code> if this has been defined in <code>MpCCI_Driver_partInfo</code> .
<code>int nnodes</code>	number of nodes of the component, can be simply set to <code>MPCCI_PART_NNODES(part)</code> .
<code>const void *coords</code>	Array of node coordinates: (x,y) list or (x,y,z) list. This array should be allocated by the code.
<code>unsigned realsize</code>	The size of the floating point value written for the node coordinates.
<code>const int *nodeids</code>	Array of the node IDs. Optional value. This could be set to <code>NULL</code> .
<code>const void *angles</code>	Array of the node angles: (x,y) list or (x,y,z) list. Optional value. This could be set to <code>NULL</code> .

It returns 0 for a successful operation.

2.7.9 Definition of Elements: `smpcci_pels`

The element definition should be called from the driver function `MpCCI_Driver_definePart` after the node definition to define the elements of a coupling component, see also "adapter.c" of the MpCCI API Kit.

! All nodes used in `smpcci_pels` must be already defined with `smpcci_pnod`.

The syntax is:

```
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:

<code>MPCCI_SERVER *server</code>	The MpCCI server pointer passed by the <code>MpCCI_Driver_definePart</code> .
<code>int mid</code>	mesh ID, should be set to <code>MPCCI_PART_MESHID(part)</code>
<code>int pid</code>	part ID, should be set to <code>MPCCI_PART_PARTID(part)</code>
<code>int nelems</code>	number of elements, can be set to <code>MPCCI_PART_NELEMS(part)</code> .
<code>unsigned eltype1</code>	one element type. Define this element type if all elements use the same element type, otherwise the eltypes array should be defined.
<code>const unsigned *eltypes</code>	element types. This can be one of the predefined element types, which are given in 2.9.1 Supported Element Types .
<code>const int *elnodes</code>	array of the node numbers (your node numbers) of the element nodes. See description below.
<code>const int *elemids</code>	array of element IDs. Size should be <code>nelems</code> . Optional value. This could be set to <code>NULL</code> .

It returns 0 for a successful operation.

2.7.9.1 Definition of Element Nodes

The nodes of each element are given in two arrays. The first, `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 `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:

```
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 [2.9.1 Supported Element Types](#).

2.7.9.2 Definition of Element Types

The element types are listed in [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 `eltype1` and `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 `eltypes` should have the same value as `nelems` and the argument `eltype1` should be set to 0.

2.7.10 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 ([▷ 2.7.8 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:

```
int smpcci_pmot(MPCCI_SERVER      *server,
                  int             mid,
                  int             pid,
                  const MPCCI_MOTION *motion);
```

The parameters are:

<code>MPCCI_SERVER *server</code>	The <code>MpCCI</code> server pointer passed by the <code>MpCCI_Driver_definePart</code> .
<code>int mid</code>	mesh ID, should be set to <code>MPCCI_PART_MESHID(part)</code> . The part is provided by <code>MpCCI_Driver_definePart</code> .
<code>int pid</code>	part ID, should be set to <code>MPCCI_PART_PARTID(part)</code> .
<code>const MPCCI_MOTION *motion</code>	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.

2.7.11 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 ([▷ 2.7.9 Definition of Elements: `smpcci_pels` ◃](#)).

The syntax is:

```
int smpcci_pshf(MPCCI_SERVER *server,
                  int mid,
                  int pid,
                  double sval);
```

The parameters are:

<code>MPCCI_SERVER *server</code>	The MpCCI server pointer passed by the <code>MpCCI_Driver_definePart</code> .
<code>int mid</code>	mesh ID, should be set to <code>MPCCI_PART_MESHID(part)</code> . The part is provided by <code>MpCCI_Driver_definePart</code> .
<code>int pid</code>	part ID, should be set to <code>MPCCI_PART_PARTID(part)</code> Usu-
<code>double sval</code>	the thickness of the baffle. The thickness defines how much the current part ID should be shifted on the MpCCI server side.

ally 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.

2.7.12 Data Exchange: `ampcci_transfer`

The transfer function must be called from your code before or after the solver.

```
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.

(i) 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.

2.7.13 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:

```
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.

2.7.14 End of Coupled Simulation: `mpcci_quit` and `mpcci_stop`

The code quits and the MpCCI server continues

```
int mpcci_quit(MPCCI_JOB **job)
```

The arguments are:

`MPCCI_JOB **job` The MpCCI job.

The code stops and the other partners are informed and the MpCCI server stops, too

```
int mpcci_stop(MPCCI_JOB **job, const char *message);
```

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 [▷ 2.7.5 Configure the Code Adapter: ampcci_config](#). The names of the functions can be chosen arbitrarily. Only two functions are *required* functions, 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:

```
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 MPCCCI_TINFO.tact
 ****/
unsigned tact_required;  /* bitmask: MPCCI_TACT_* */

/* ****
 * REQUIRED: define symbolic names for the types of parts in code terminology
 * example:
 *     [0] = "Integration point",
 *     [1] = "Line",
 *     [2] = "SURFACE",
 *     [3] = "Volume domain",
 *     [4] = "Global variable"
 ****/
const char *part_description[5];

/* ****
 * 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);

/*********************************************************************
 * REQUIRED: methods to get information about a part
 *****/
/* variante #1: fast & dirty: returns the MpCCI error code */
int (*definePart)(MPCCI_SERVER *server, MPCCI_PART *part);

/* variante 2: may be slow but better interface */
void (*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);

/*********************************************************************
 * REQUIRED for moving nodes instead of full remeshing: returns the MpCCI error code
 *****/
int (*moveNodes) (MPCCI_SERVER *server, MPCCI_PART *part);

/*********************************************************************
 * OPTIONAL(depends): methods used to get quantities from the application
 *****/
int  (*getPointValues ) (const MPCCI_PART *part, const MPCCI_QUANT *quant, void *values);
int  (*getLineNodeValues)(const MPCCI_PART *part, const MPCCI_QUANT *quant, void *values);
int  (*getLineElemValues)(const MPCCI_PART *part, const MPCCI_QUANT *quant, void *values);
int  (*getFaceNodeValues)(const MPCCI_PART *part, const MPCCI_QUANT *quant, void *values);
int  (*getFaceElemValues)(const MPCCI_PART *part, const MPCCI_QUANT *quant, void *values);
int  (*getVoluNodeValues)(const MPCCI_PART *part, const MPCCI_QUANT *quant, void *values);
int  (*getVoluElemValues)(const MPCCI_PART *part, const MPCCI_QUANT *quant, void *values);
int  (*getGlobValues   )(                           const MPCCI_GLOB *glob , void *values);

/*********************************************************************
 * OPTIONAL(depends): methods used to store quantities into the application
 *****/
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[5]

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 */
  "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.

```
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 ▷ 2.7.10 Definition of the Moving Reference Frame: `smpcci_pmot`).
- Set number of nodes with `smpcci_pnod` (see ▷ 2.7.8 Definition of Nodes: `smpcci_pnod`).
- Set number of elements with `smpcci_pels` (see ▷ 2.7.9 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.

```
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.

```
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`:

- `csys`: coordinates system may be redefined.
- `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. `(int)sizeof(double)`.

```
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 `elemIds` array is not used.
- `0x0001`: the `elemIds` array is valid.

```
int (*moveNodes) (MPCCI_SERVER *server, MPCCI_PART *part)
```

This function is required for moving nodes instead of full remeshing. It is called when the remeshing state is set to `MPCCI_REMESH_STATE_SMOOTHED`. In that case the code should define the moving nodes by using the function `smpcci_pnod` (see ▷ 2.7.8 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:

```
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.

```
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)`.

```
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](#).

```
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

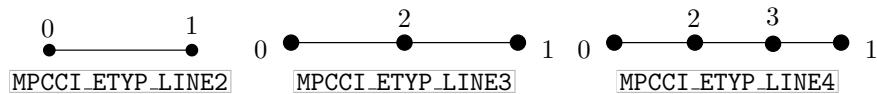


Table 1: Beam element types in the MpCCI code API

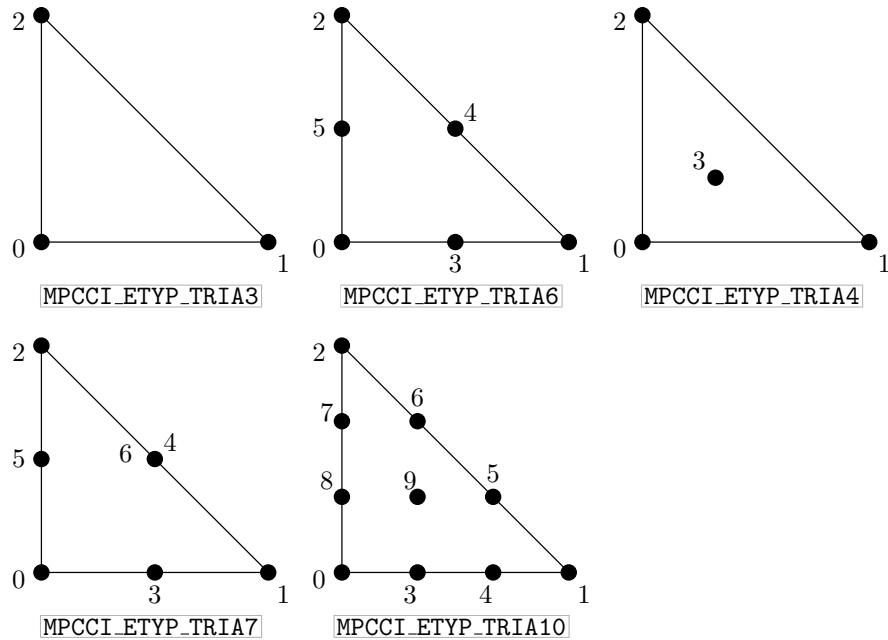
Supported triangle elements

Table 2: Triangle element types in the MpCCI code API

Supported quadrilateral elements

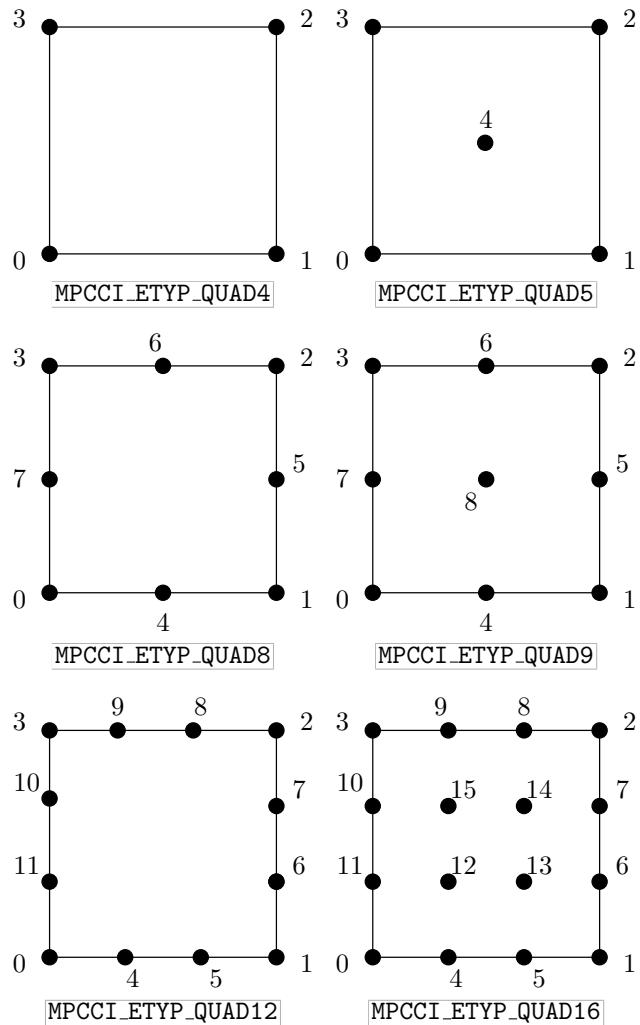


Table 3: Quadrilateral element types in the MpCCI code API

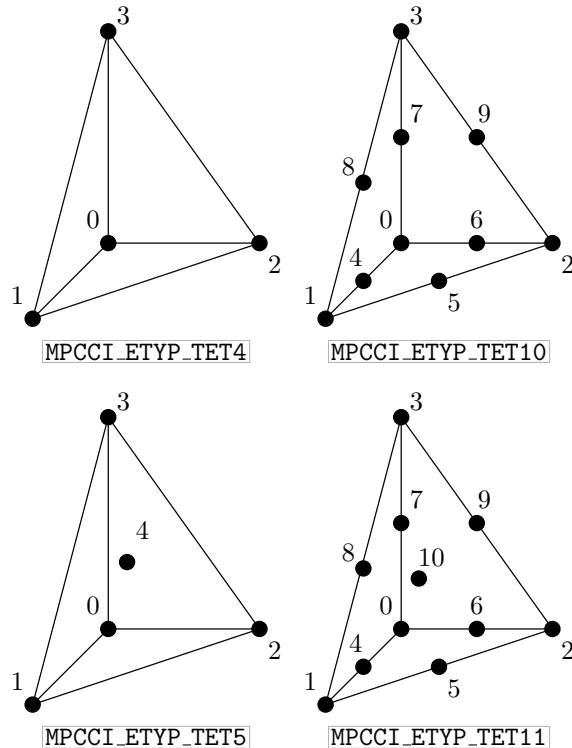
Supported tetrahedral elements

Table 4: Tetrahedral element types in the MpCCI code API

Supported pyramid elements

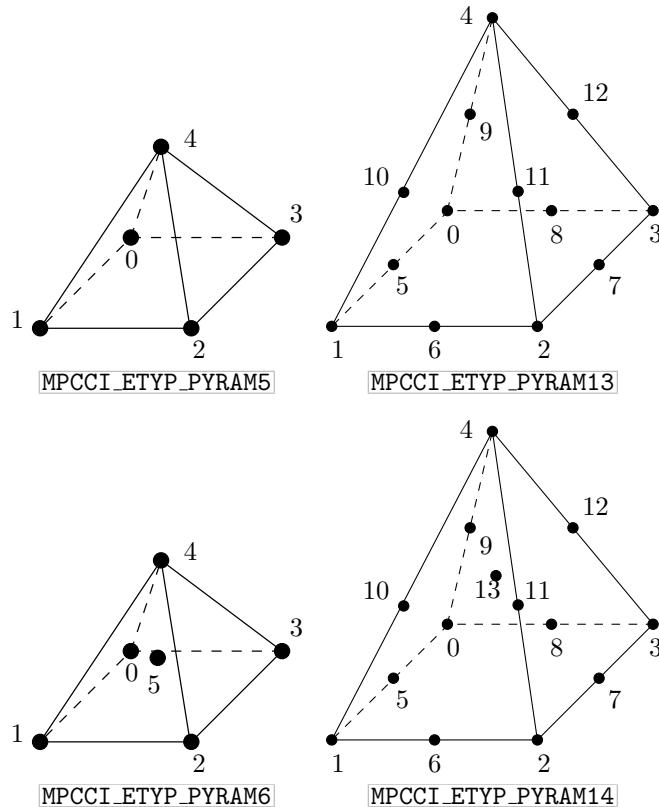


Table 5: Pyramid element types in the MpCCI code API

Supported wedge elements

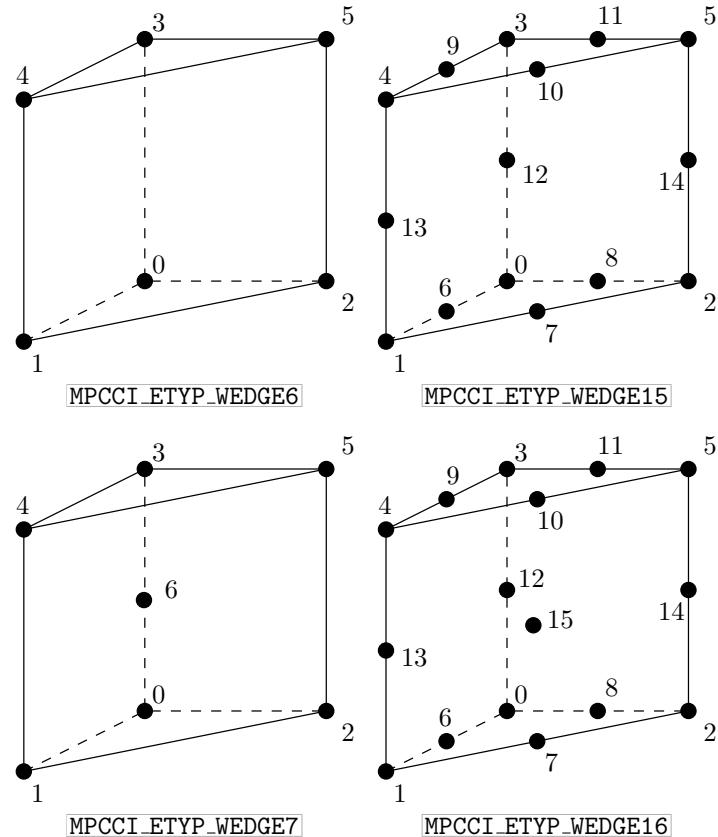


Table 6: Wedge element types in the MpCCI code API

Supported hexahedral elements

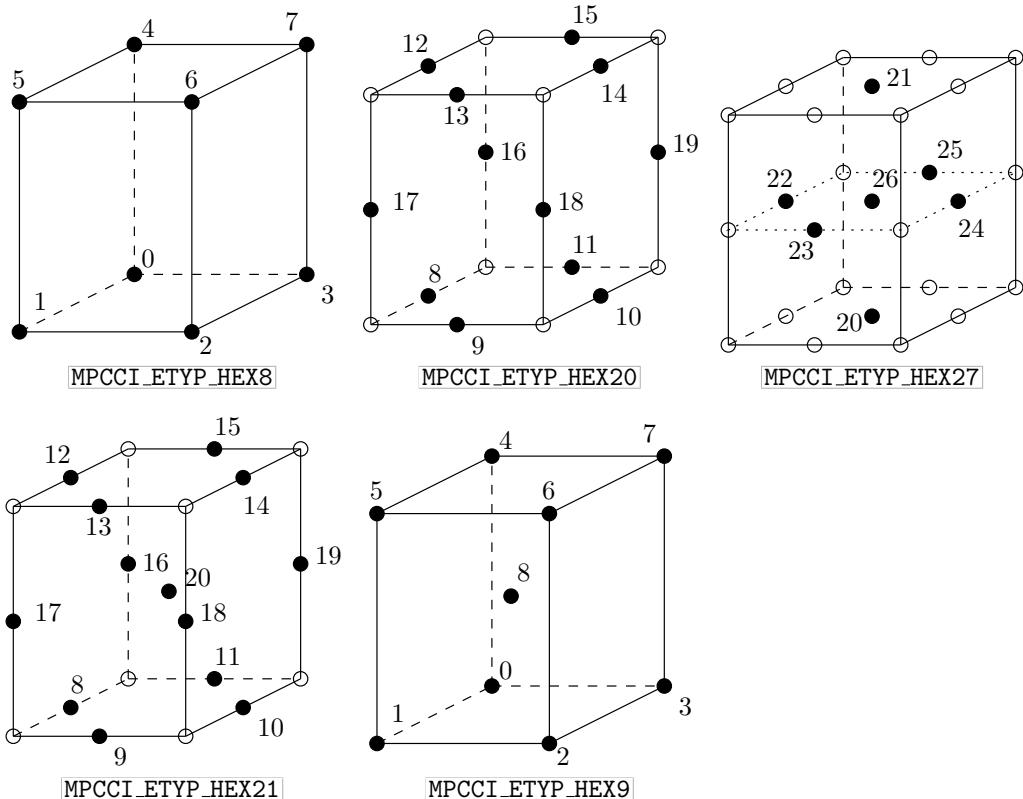


Table 7: Hexahedral element types in the MpCCI code API

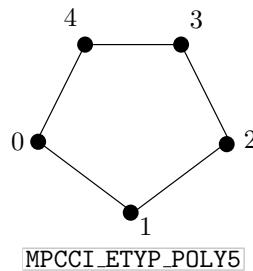
Supported polygonal face elements

To define a polygon face element type

```
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:

```
eltype[3] = MPCCI_ETYP_POLY6;
eltype[4] = MPCCI_ETYP_POLY20;
```



`MPCCI_ETYP_POLY5`

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[1]      start index of first vertex of face[1] in elnodes.  
elnodes[2]      start index of first vertex of face[2] 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  
last face.
```

- `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.
```

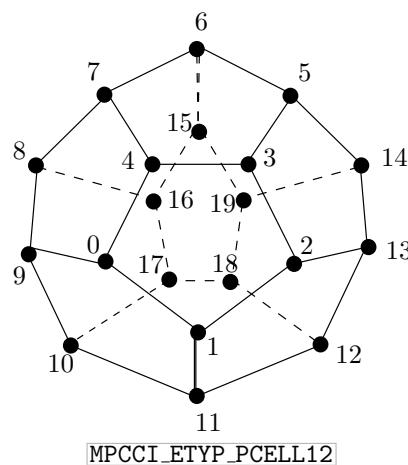
...

```
elnodes[elnodes[1]]    vertex id of first vertex of face 1.
```

```
elnodes[elnodes[1]+1]  vertex id of second vertex of face 1.
```

...

```
elnodes[0]-1          number of faces in PCELL.
```



`MPCCI_ETYP_PCELL12`

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:

```
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 ▶2.7.13 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"):

```
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;
```

```

int      nglob_r;
int      nglob_req;

/*
 * optional convergence status
 */
int      conv_code;
int      conv_job;
int      conv_last;
int      conv_counter;
};

```

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.

You can

`MPCCI_TSOLU_STEADY` : The solution is steady state.

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.

You can

`MPCCI_TMODE_GAUSSSEIDEL` : Gauss-Seidel algorithm.

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` : The code sends if at least one partner is waiting: option `onewait`.

`MPCCI_TMODE_SEND_ALLWAIT` : The code sends if all partners are waiting: option `allwait`.

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`.

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 `ampcci_transfer()` ▷[2.7.12 Data Exchange: 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 `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()`.

int nquant_r

Number of received quantities (sum(mesh * nquant)) within the last `ampcci_transfer()`.

int nquant_req

Number of requested quantities (sum(mesh * nquant)) within the last `ampcci_transfer()`.

int nglob_s

Number of sent global within the last `ampcci_transfer()`.

int nglob_r

Number of received global within the last `ampcci_transfer()`.

int nglob_req

Number of requested global within the last `ampcci_transfer()`.

Optional convergence status:**int conv_code**

Set the convergence state of the code:

<code>MPCCI_CONV_STATE_ERROR</code>	: Error occurred.
<code>MPCCI_CONV_STATE_INVALID</code>	: Initial convergence state set to invalid.
<code>MPCCI_CONV_STATE_STOP</code>	: Simulation is stopped.
<code>MPCCI_CONV_STATE_DIVERGED</code>	: Solution diverged.
<code>MPCCI_CONV_STATE_CONTINUE</code>	: Continue the simulation computation.
<code>MPCCI_CONV_STATE_CONVERGED</code>	: Solution converged.

int conv_job

Set after the transfer and informs about the global convergence state.

int conv_last

Store the last convergence state.

int conv_counter

Store the number of converged transfers in a row.

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:

```
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;
    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.

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

Return value	Macro	Description
bool	MPCCI_CODE_HAS_CURRG(code)	check whether the code has the current grid.
bool	MPCCI_CODE_HAS_ORIGG(code)	check whether the code has the original grid.
bool	MPCCI_CODE_IS_FV(code)	check whether the code is a finite volume code.
bool	MPCCI_CODE_IS_FEA(code)	check whether the code is finite element analysis code.
bool	MPCCI_CODE_IS_NET(code)	check whether the code is a network code.
bool	MPCCI_CODE_IS_RAD(code)	check whether the code is a radiation code.

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.

Return value Macro	Description
<code>char * MPCCI_PART_NAME(part)</code>	name of coupling component
<code>int MPCCI_PART_MDIM(part)</code>	dimension of component. (See 2.9.3 Mesh Dimension Definition)
<code>MPCCI_QUANT * MPCCI_PART_QUANTS(part)</code>	list of quantities to transfer
<code>int MPCCI_PART_NNODES(part)</code>	number of nodes/points in component
<code>int MPCCI_PART_NELEMS(part)</code>	number of faces/elements/cells in component
<code>int MPCCI_PART_NODMAX(part)</code>	max. number of nodes per element in this component
<code>int MPCCI_PART_MESHID(part)</code>	MpCCI mesh id
<code>int MPCCI_PART_PARTID(part)</code>	MpCCI part id
<code>unsigned MPCCI_PART_CSYS(part)</code>	coordinates system
<code>void * MPCCI_PART_USRPTRO(part)</code>	pointer no. 0 to auxiliary optional information defined and used by the application
<code>void * MPCCI_PART_USRPTR1(part)</code>	pointer no. 1 to auxiliary optional information defined and used by the application
<code>void * MPCCI_PART_USRPTR2(part)</code>	pointer no. 2 to auxiliary optional information defined and used by the application
<code>void * MPCCI_PART_USRPTR3(part)</code>	pointer no. 3 to auxiliary optional information defined and used by the application
<code>void * MPCCI_PART_USRPTR4(part)</code>	pointer no. 4 to auxiliary optional information defined and used by the application
<code>void * MPCCI_PART_USRPTR5(part)</code>	pointer no. 5 to auxiliary optional information defined and used by the application
<code>void * MPCCI_PART_USRPTR6(part)</code>	pointer no. 6 to auxiliary optional information defined and used by the application
<code>void * MPCCI_PART_USRPTR7(part)</code>	pointer no. 7 to auxiliary optional information defined and used by the application
<code>void * MPCCI_PART_USRPTR8(part)</code>	pointer no. 8 to auxiliary optional information defined and used by the application
<code>void * MPCCI_PART_USRPTR9(part)</code>	pointer no. 9 to auxiliary optional information defined and used by the application
<code>unsigned MPCCI_PART_USRFLAGS(part)</code>	optional flags information field for the current mesh defined and used by the application
<code>int MPCCI_PART_USRSTATE(part)</code>	optional state information for the current mesh defined and used by the application

Return value macro	description
<code>bool MPCCI_PART_IS_EMPTY(part)</code>	check whether component represents an empty part with zero nodes and zero elements
<code>bool MPCCI_PART_IS_NONE(part)</code>	check whether component is nothing
<code>bool MPCCI_PART_IS_NETP(part)</code>	check whether component represents a network point

Return value macro	description
<code>bool MPCCI_PART_IS_LINE(part)</code>	check whether component is a line
<code>bool MPCCI_PART_IS_FACE(part)</code>	check whether component is a face
<code>bool MPCCI_PART_IS_VOLU(part)</code>	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
<code>const char * MPCCI_QUANT_NAME(quant)</code>	symbolic name e.g. "temp", "JHeat" ...
<code>int MPCCI_QUANT_QID(quant)</code>	ID code of quantity (see also the Quantity Reference in the Appendix)
<code>int MPCCI_QUANT_STATE(quant)</code>	state of transfer <code>MPCCI_QUANT_TSTATE_xxx</code> (used by the manager): <code>MPCCI_QUANT_TSTATE_NONE</code> no transfer was done before <code>MPCCI_QUANT_TSTATE_SEND</code> quantity was sent <code>MPCCI_QUANT_TSTATE_RECV</code> quantity was received contains the direction send/receive bits, the location bits etc. .
<code>int MPCCI_QUANT_FLAGS(quant)</code>	dimension of quantity: 1=scalar, 3=vector etc. .
<code>int MPCCI_QUANT_DIM(quant)</code>	
<code>int MPCCI_QUANT_LOC(quant)</code>	location of value, one of: <code>MPCCI_QFLAG_LOC_CODE</code> <code>MPCCI_QFLAG_LOC_NODE</code> or <code>MPCCI_QFLAG_LOC_VERT</code> <code>MPCCI_QFLAG_LOC_ELEM</code> or <code>MPCCI_QFLAG_LOC_CELL</code> <code>MPCCI_QFLAG_LOC_POINT</code>
<code>int MPCCI_QUANT_PHY(quant)</code>	physical meaning of this quantity, one of <code>MPCCI_QPHY_ANY</code> quantity is not further specified <code>MPCCI_QPHY_MASS</code> a mass sink/source <code>MPCCI_QPHY_MOM</code> a momentum sink/source <code>MPCCI_QPHY_ENTH</code> an energy sink/source <code>MPCCI_QPHY_PROP</code> a property (material) <code>MPCCI_QPHY_BCVAL</code> a boundary condition value <code>MPCCI_QPHY_BCGRAD</code> a boundary condition wall normal gradient <code>MPCCI_QPHY_COORD</code> a grid coordinate/displacement <code>MPCCI_QPHY_MASSFR</code> a chemical component concentration
<code>int MPCCI_QUANT_DIR(quant)</code>	transfer direction, one of <code>MPCCI_QFLAG_DIR_SEND</code> quantity is sent <code>MPCCI_QFLAG_DIR_RECV</code> quantity is received
<code>int MPCCI_QUANT_SMETHOD(quant)</code>	how to store sent/received quantities, one of <code>MPCCI_QSM_UNDEF</code> invalid send method

Return value	Macro	Description
	<code>MPCCI_QSM_DIRECT</code>	directly written into buffer
	<code>MPCCI_QSM_BUFFER</code>	quantity is buffered locally and copied later
	<code>MPCCI_QSM_USRMEM</code>	quantity to store in user's indexed memory
	<code>MPCCI_QSM_SCALAR</code>	quantity to store in user's index scalars
	<code>MPCCI_QSM_SPECIES</code>	quantity to store the chemical species
<code>int MPCCI_QUANT_SINDEX(quant)</code>		index to user supplied memory or user scalars
<code>void * MPCCI_QUANT_SBUFFER(quant)</code>		quantities local transfer buffer
<code>int MPCCI_QUANT_SBFSIZE(quant)</code>		size of allocated local buffer

Return value	macro	description
	<code>bool MPCCI_QUANT_IS_SEND(quant)</code>	check whether the quantity is sent
	<code>bool MPCCI_QUANT_IS_RECV(quant)</code>	check whether the quantity is received
	<code>int MPCCI_QUANT_IS_NODE(quant)</code>	check whether quantity is a nodal quantity
	<code>int MPCCI_QUANT_IS_ELEM(quant)</code>	check whether quantity is an element quantity
	<code>int MPCCI_QUANT_IS_GLOB(quant)</code>	check whether the quantity is a global quantity
	<code>int MPCCI_QUANT_IS_MESH(quant)</code>	check whether the quantity is a mesh based quantity
	<code>int MPCCI_QUANT_IS_COORD(quant)</code>	check whether the quantity is a coordinate/displacement
	<code>int MPCCI_QUANT_IS_FIELD(quant)</code>	check whether the quantity is a field value
	<code>int MPCCI_QUANT_IS_FLUXI(quant)</code>	check whether the quantity is a flux integral value
	<code>int MPCCI_QUANT_IS_FLUXD(quant)</code>	check whether the quantity is a flux density value
	<code>bool MPCCI_QUANT_IS_SCALAR(part)</code>	check whether the quantity is a scalar
	<code>bool MPCCI_QUANT_IS_VECTOR(part)</code>	check whether the quantity is a vector
	<code>bool MPCCI_QUANT_IS_TENSOR(part)</code>	check whether the quantity is a tensor

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:

```
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:

```
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:

```
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:

```
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:

```
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)

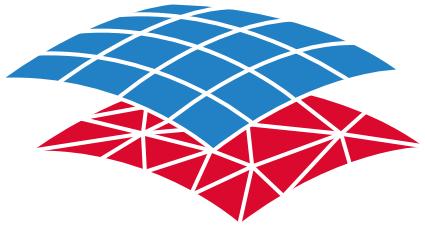
This routine is called for allocating memory, should return a pointer for the memory, a **NULL** pointer if it fails.

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.



MpCCI
CouplingEnvironment

Part IX

How To

Version 4.5.0

MpCCI 4.5.0-1 Documentation

Part IX How To

PDF version

March 30, 2017

MpCCI is a registered trademark of Fraunhofer SCAI

www.mpcci.de



Fraunhofer Institute for Algorithms and Scientific Computing SCAI
Schloss Birlinghoven, 53754 Sankt Augustin, Germany

Abaqus and SIMULIA are trademarks or registered trademarks of Dassault Systèmes

ANSYS, FLUENT and ANSYS Icepak are trademarks or registered trademarks of Ansys, Inc.

Elmer is an open source software developed by CSC

FINE/Open and FINE/Turbo are trademarks of NUMECA International

Flowmaster is a registered trademark of Flowmaster Group NV

JMAG is a registered trademark of The JSOL Corporation

MATLAB is a registered trademark of The MathWorks, Inc.

MSC Adams, MSC Marc, MD NASTRAN and MSC NASTRAN are trademarks or registered trademarks of
MSC.Software Corporation

OpenFOAM is a registered trademark of OpenCFD Ltd.

PERMAS is a registered trademark of Intes GmbH

RadTherm, TAITherm is a registered trademark of ThermoAnalytics Inc.

SIMPACK is a registered trademark of SIMPACK AG.

STAR-CCM+ and STAR-CD are registered trademarks of CD adapco Group

ActivePerl has a Community License Copyright of Active State Corp.

FlexNet Publisher is a registered trademark of Flexera Software.

Java is a registered trademark of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds

Mac OS X is a registered trademark of Apple Inc.

OpenSSH has a copyright by Tatu Ylonen, Espoo, Finland

Perl has a copyright by Larry Wall and others

UNIX is a registered trademark of The Open Group

Windows, Windows XP, Windows Vista and Windows 7 are registered trademarks of Microsoft Corp.

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

Keywords	Full vehicle thermal management, coupled simulation with MpCCI convection, conduction and radiation, steady state and transient
Simulation Codes	STAR-CCM+ FLUENT RadTherm
MpCCI Version	MpCCI 4.2 and higher
Related Tutorials	▷ VII-10 Cube in a Duct Heater ◁

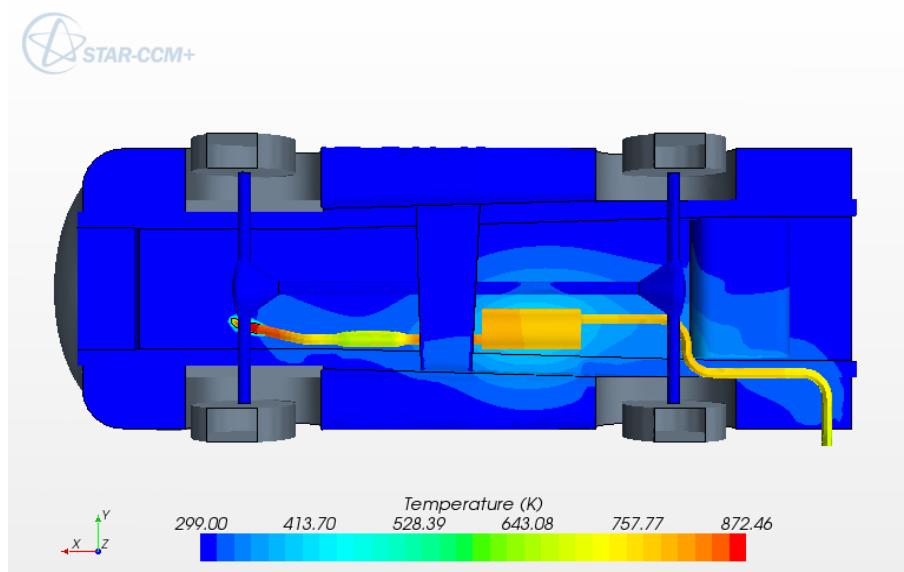


Figure 1: Thermal management

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 RadTherm 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 RadTherm 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.

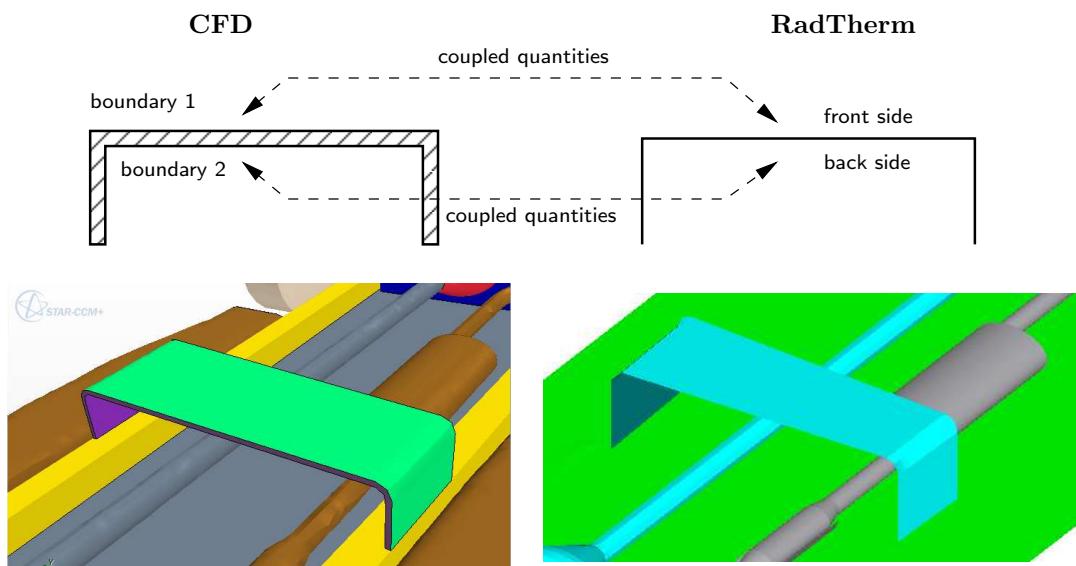


Figure 2: Front and back side assignment for thin parts by separating boundaries

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 RadTherm 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 RadTherm 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 RadTherm 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-4.5.5 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+.

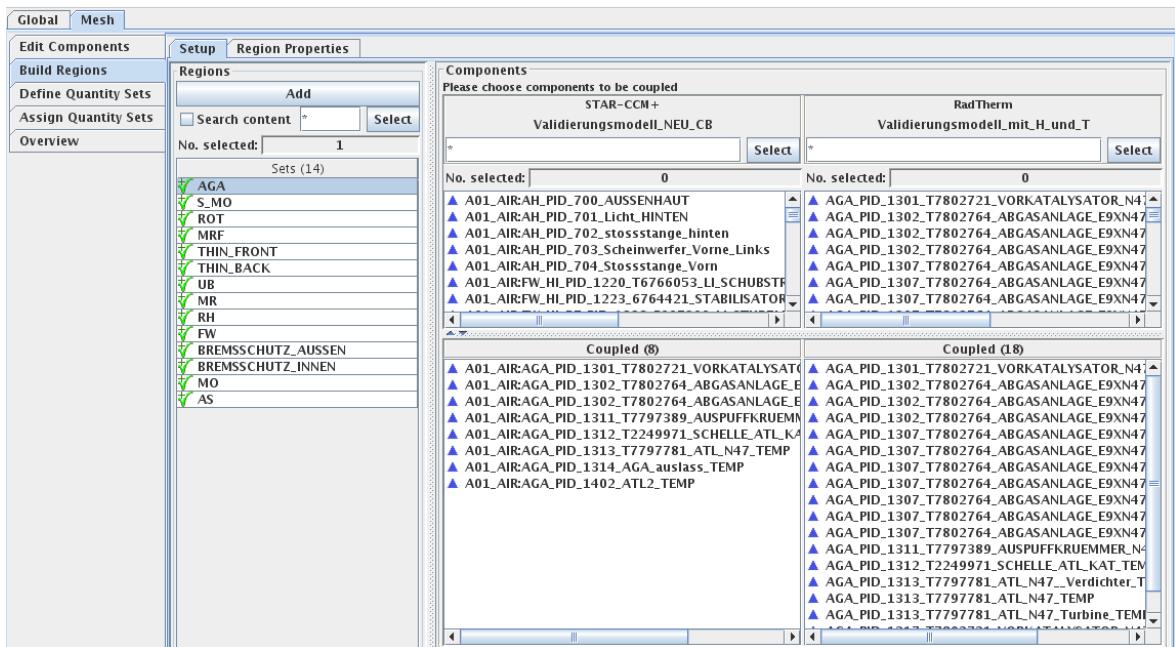


Figure 3: Possible coupling meshes for a full vehicle reflecting part groups

It should be noted that **not insulated back sides** of shell element parts in RadTherm will appear in MpCCI with the string `"-backside"` added to the coupling part name [▷ VI-15.2.3 Coupling 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.

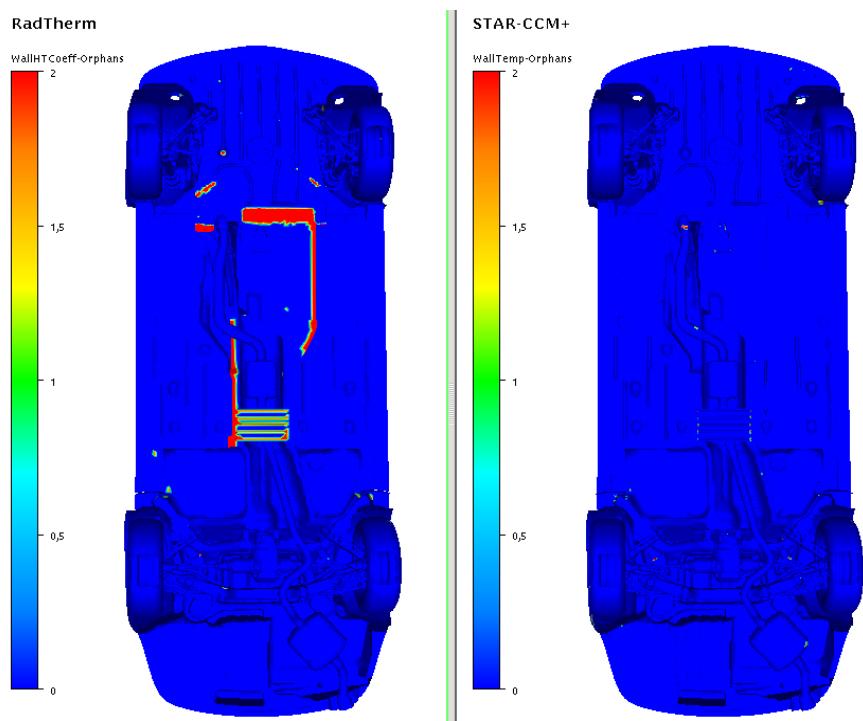
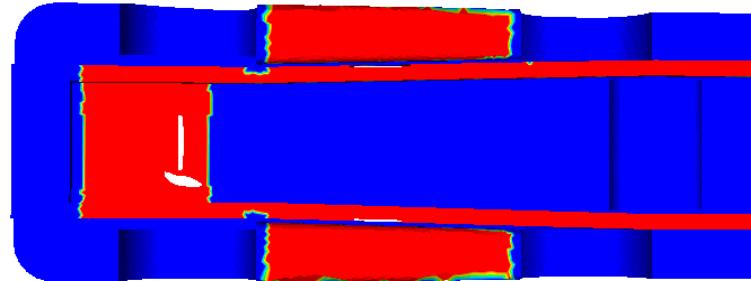
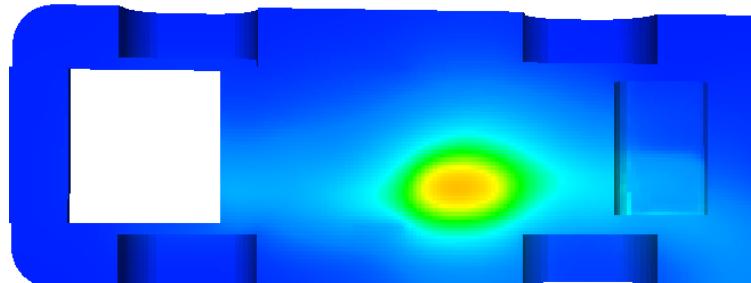


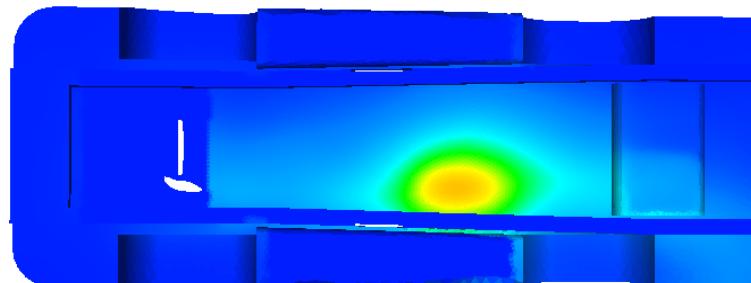
Figure 4: Orphaned regions on bottom of BMW vehicle visualized in the MpCCI visualizer



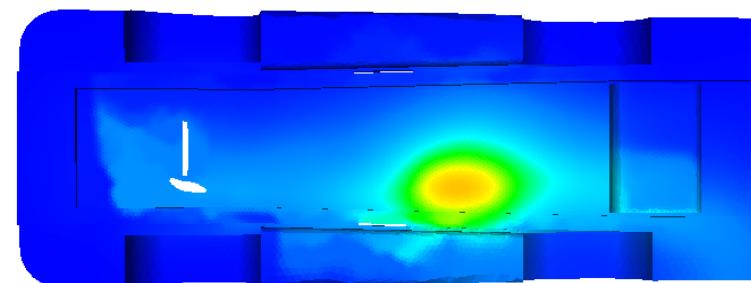
Orphaned regions on sender mesh



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 in MpCCI

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 RadTherm, 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.

RadTherm

RadTherm cannot be steered by a script/macro file. When the necessary MpCCI function hooks are placed, RadTherm 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.

STAR-CCM+	approx. 45 mio. cells: 15 regions, 525 boundaries MRF for fans and wheels Porous regions for heat exchangers and cooling devices Wall rotation of shafts and axles
RadTherm	approx 900.000 cells, 323 shell parts 13 fluid parts for exhaust system
MpCCI	coupled quantities: STAR-CCM+ → RadTherm: film temperature, heat coefficient RadTherm → STAR-CCM+: wall temperature ⇒ approx. 2 mio. faces coupled

Table 1: Model data

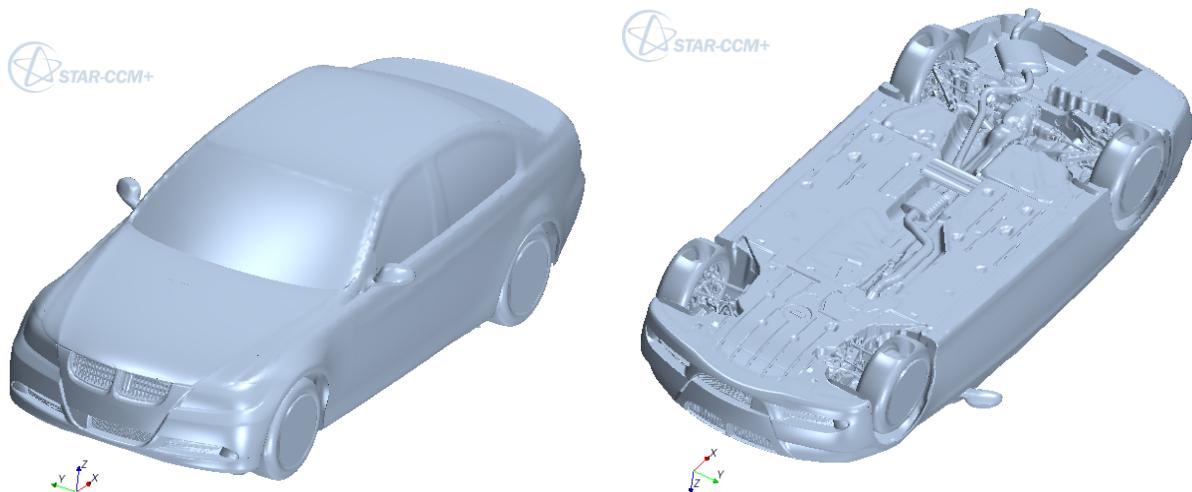


Figure 6: STAR-CCM+ full vehicle model of a BMW top and bottom view

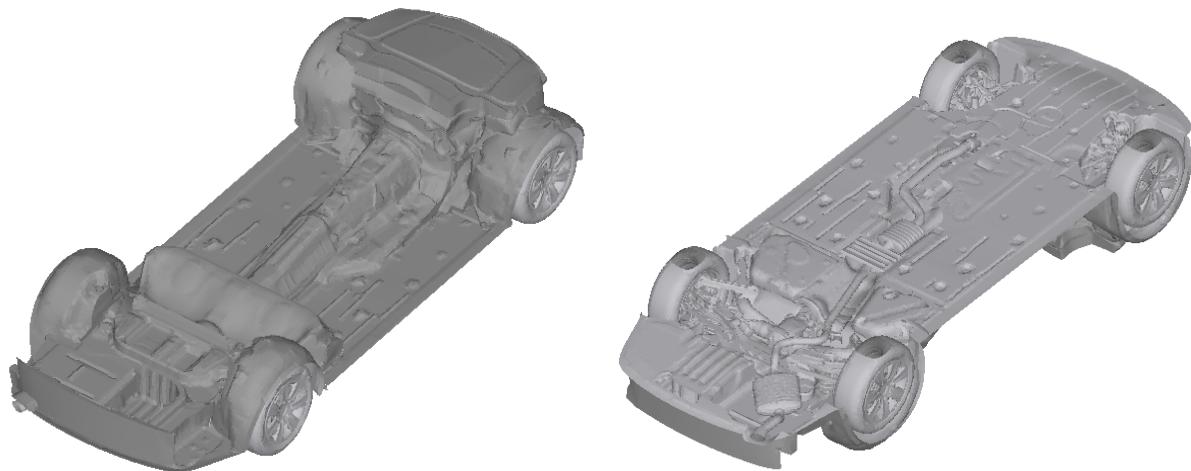


Figure 7: RadTherm underbody model of a BMW top and bottom view

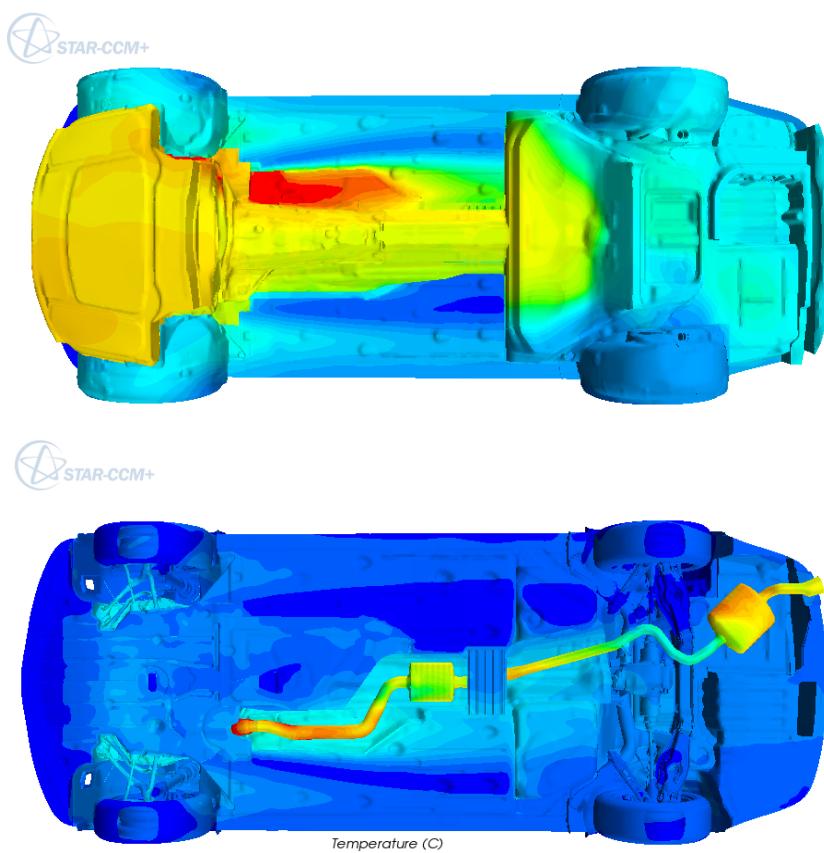


Figure 8: Wall temperature in STAR-CCM+ of BMW vehicle top and bottom view

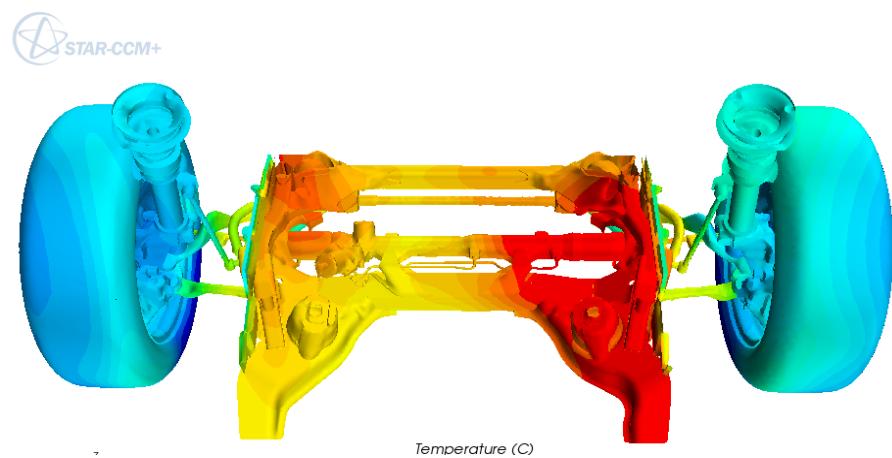


Figure 9: Wall temperature in STAR-CCM+ of BMW vehicle front axle

2 Simulation of Fluid-Structure Interaction for a new Hydraulic Axial Pump

2.1 Quick Information

Keywords	Fluid-structure interaction for compensation chamber in a new pump design coupled simulation with MpCCI
Simulation Codes	STAR-CCM+ FLUENT ANSYS Abaqus
MpCCI Version	MpCCI 4 and higher
Related Tutorials	

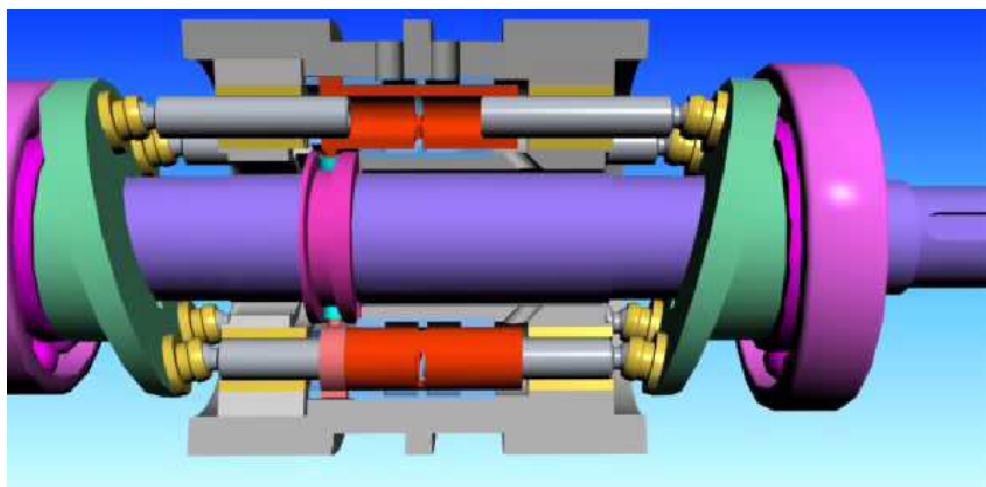


Figure 1: Hydraulic Axial Pump

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.

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

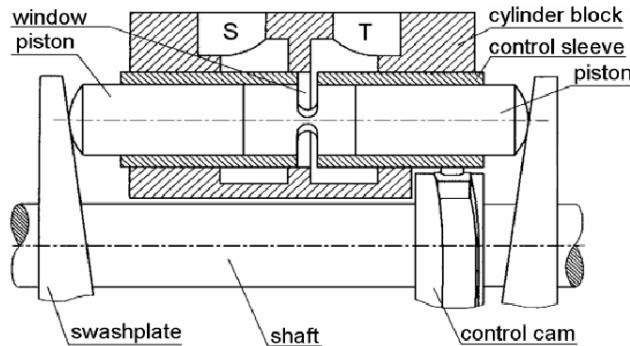


Figure 2: Picture of PWK pumps on the left, main elements of the PWK pump on the right

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 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.

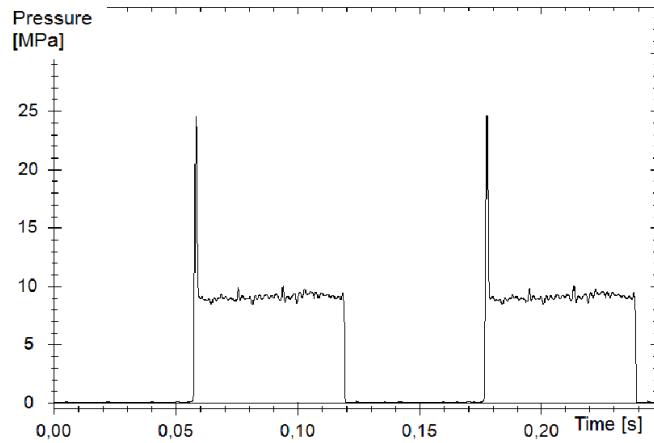


Figure 3: Pressure variation in pump's chamber (rotational speed 500 rpm, oil temperature 35°C, displacement adjustment 22%)

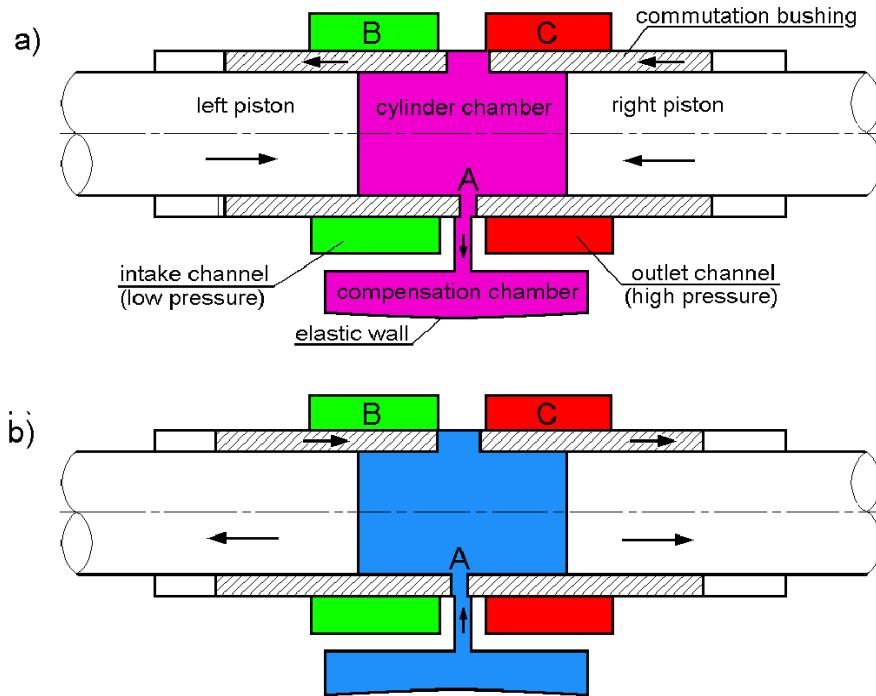


Figure 4: Pressure peak reduction with a compensation chamber with an elastic wall. a) shows high pressure in the cylinder chamber which causes the fluid to flow into the compensation chamber. In b) the pistons are moving outwards and the hydraulic oil flows back into the cylinder chamber.

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

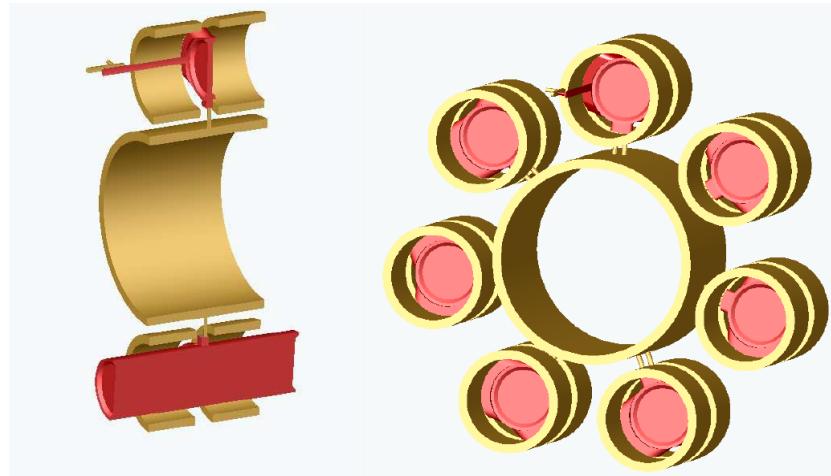


Figure 5: Full CAD model of a pump with seven cylinder chambers (on the right) and a simplified symmetric model of a pump with two chambers (left). Movable parts are shown in red.

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.

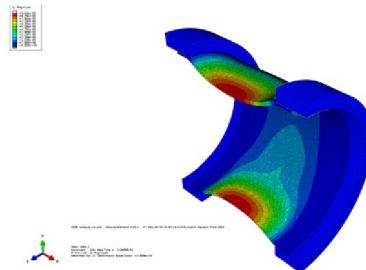


Figure 6: Solid Abaqus model. U displacement contours with a maximum value of 0.0001.

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

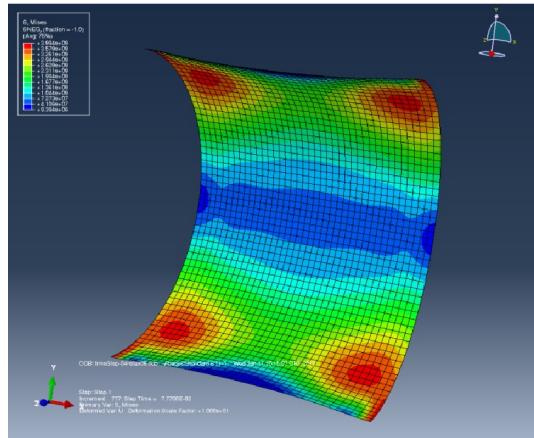


Figure 7: Von-Mises-stress on the (slightly) deformed elastic wall of the compensation chamber.

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.

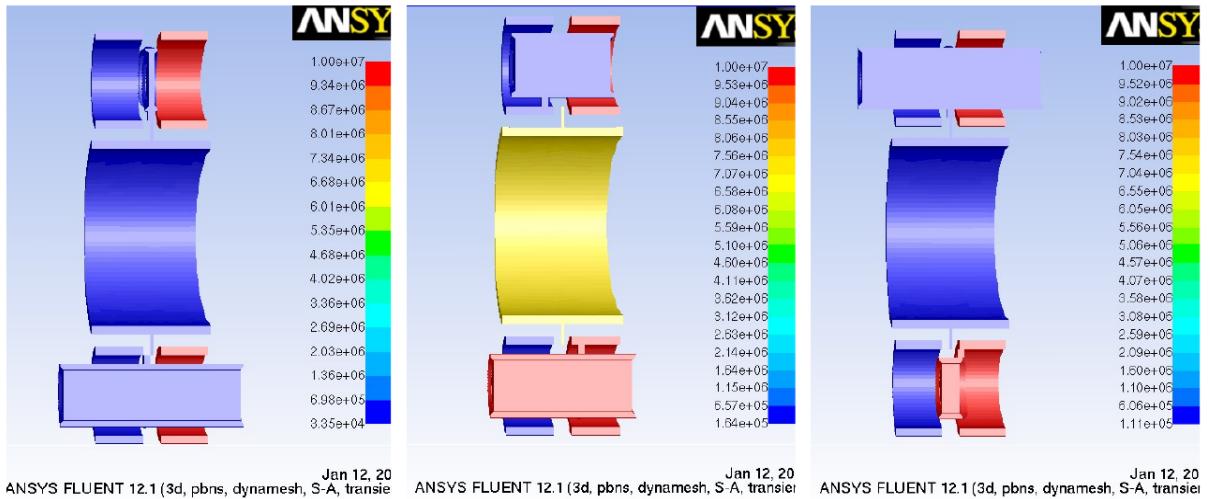


Figure 8: Pressure contours for the PWK pump at different time values: on the left $t = 0.0002\text{s}$, in the middle $t = 0.01\text{s}$ and on the right $t = 0.02\text{s}$.

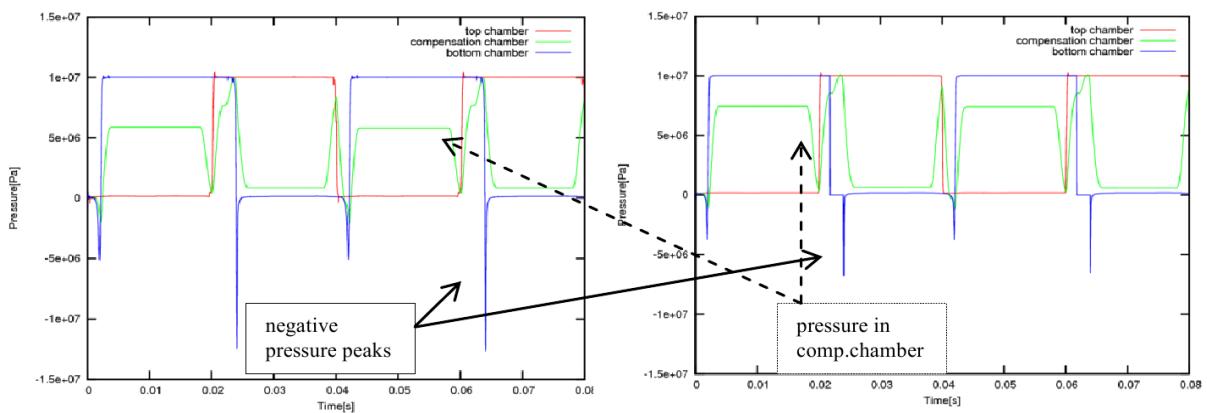
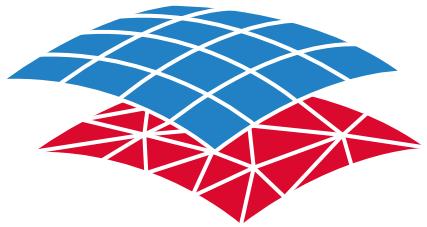


Figure 9: Pressure plots for three discrete points in the pump. CFD stand-alone (FLUENT) results are on the left, FSI results (FLUENT-Abaqus-MpCCI) are on the right. The differences in the pressure peaks and in the pressure value in the compensation chamber can be seen.



MpCCI
FSIMapper

Part X

—

MpCCI FSIMapper

Version 4.5.0

MpCCI 4.5.0-1 Documentation
Part X FSIMapper
PDF version
March 30, 2017

MpCCI is a registered trademark of Fraunhofer SCAI
www.mpcci.de



Fraunhofer Institute for Algorithms and Scientific Computing SCAI
Schloss Birlinghoven, 53754 Sankt Augustin, Germany

Abaqus and SIMULIA are trademarks or registered trademarks of Dassault Systèmes
ANSYS, FLUENT and ANSYS Icepak are trademarks or registered trademarks of Ansys, Inc.
Elmer is an open source software developed by CSC
FINE/Open and FINE/Turbo are trademarks of NUMECA International
Flowmaster is a registered trademark of Flowmaster Group NV
JMAG is a registered trademark of The JSOL Corporation
MATLAB is a registered trademark of The MathWorks, Inc.
MSC Adams, MSC Marc, MD NASTRAN and MSC NASTRAN are trademarks or registered trademarks of
MSC.Software Corporation
OpenFOAM is a registered trademark of OpenCFD Ltd.
PERMAS is a registered trademark of Intes GmbH
RadTherm, TAITherm is a registered trademark of ThermoAnalytics Inc.
SIMPACK is a registered trademark of SIMPACK AG.
STAR-CCM+ and STAR-CD are registered trademarks of CD adapco Group

ActivePerl has a Community License Copyright of Active State Corp.
FlexNet Publisher is a registered trademark of Flexera Software.
Java is a registered trademark of Oracle and/or its affiliates.
Linux is a registered trademark of Linus Torvalds
Mac OS X is a registered trademark of Apple Inc.
OpenSSH has a copyright by Tatu Ylonen, Espoo, Finland
Perl has a copyright by Larry Wall and others
UNIX is a registered trademark of The Open Group
Windows, Windows XP, Windows Vista and Windows 7 are registered trademarks of Microsoft Corp.

X MpCCI FSIMapper – Contents

1 MpCCI FSIMapper Overview	6
2 MpCCI FSIMapper Installation	8
2.1 Part of MpCCI Installation	8
2.2 Standalone Version	8
2.2.1 Local Microsoft Windows Installation	8
2.2.2 Linux or Multi-Platform Installation	8
2.2.3 Perl	9
2.2.4 License	10
2.2.5 Configure a License Manager as UNIX service	11
2.2.6 Configure a License Manager as Windows service	12
3 MpCCI FSIMapper Command	14
4 MpCCI FSIMapper GUI	15
4.1 Starting the MpCCI FSIMapper	15
4.2 The “What to map” Panel	16
4.2.1 Selection of Cases, Parts and Quantities	16
4.2.2 Geometry Compare	18
4.2.3 Quantity Identification for CSM Solver Output	18
4.2.4 Execute a Mapping Process	19
4.3 The “Transformation” Panel	21
4.3.1 Truncation of transient result data	25
4.4 The “How to map” Panel	25
4.4.1 Mapping Algorithms and Neighborhood Parameters	25
4.4.2 Orphan Filling	26
4.4.3 Quantity Location	26
4.4.4 Saving Mapping Configurations	27
4.5 The “Result” Panel	27
4.5.1 Average over Rotation Axis	27
4.5.2 Apply Fourier Transformation	28
4.5.3 MSC NASTRAN Export Options	29
4.6 The “Preferences” Panel	30
4.7 The “Harmonic Wizard” Panel	30
4.8 The “Log” Panel	31
5 Codes and Formats Information	33
5.1 EnSight Gold	33
5.1.1 Supported Quantities	33

5.2	FLUENT	33
5.2.1	Supported Quantities	33
5.2.2	Exporting wallfuncHTC to UDM0 in Data File	33
5.3	FINE/Turbo	34
5.3.1	Supported Quantities	34
5.4	CFX	34
5.4.1	Supported Quantities	35
5.5	FloTHERM	35
5.5.1	Supported Quantities	35
5.6	FloEFD	35
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	Abaqus	36
5.8.1	Limitations	36
5.8.2	Model Preparation	36
5.8.3	Include boundary conditions	37
5.8.4	Elements	37
5.8.5	Supported Quantities	37
5.9	ANSYS	38
5.9.1	Requirements	38
5.9.2	Model Preparation	38
5.9.3	Supported Quantities	40
5.9.4	ANSYS Scanner and Converter	41
5.10	MSC NASTRAN	41
5.10.1	Limitations	41
5.10.2	Include boundary conditions	41
5.10.3	Elements	42
5.10.4	Supported Quantities	43
5.11	LS-Dyna	43
5.11.1	Limitations	43
5.11.2	Elements	43
5.11.3	Supported Quantities	43
5.11.4	Include boundary conditions	43
6	Batch Usage of the MpCCI FSIMapper	44
6.1	File Scanners	44
6.1.1	FLUENT	44

6.1.2	MentorGraphics	46
6.1.3	FloEFD	46
6.1.4	FloTHERM	47
6.1.5	ANSYS	47
6.1.6	Abaqus	48
6.1.7	EnSight Gold Case	49
6.2	Comparing Geometries	52
6.3	Mapping Quantities	53
6.4	Files Written by the MpCCI FSIMapper	53
6.5	Configuration File	53
6.6	Example for a Configuration File	56
7	Numerical Methods	59
7.1	Mapping Algorithms	59
7.2	Parameters for the Mapping	59
7.3	Orphan Filling	60
8	Tutorial	62
8.1	Mapping of Electromagnetic Forces for Transient and Harmonic analyses	62
8.1.1	Problem Description	62
8.1.2	Source Result File	63
8.1.3	Target Mesh File	66
8.1.4	Mapping	66
8.1.5	Target Simulation	72
8.2	Mapping of Harmonic Pressure Excitations in Turbomachinery	73
8.2.1	Using the Harmonic Balance Method of STAR-CCM+	73
8.2.2	Using the Nonlinear Harmonic Method of FINE/Turbo	85

1 MpCCI FSIMapper Overview

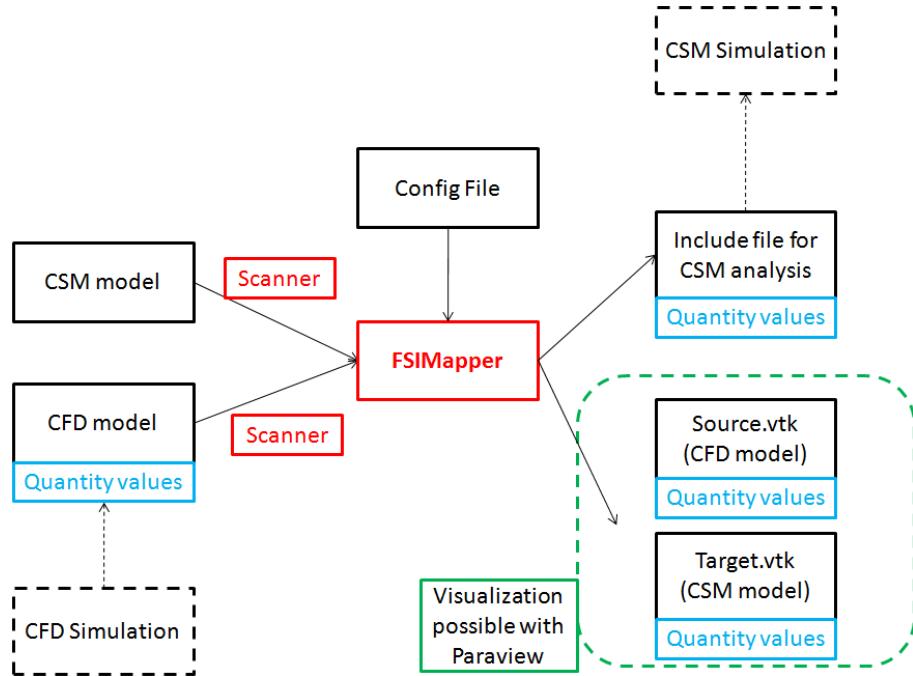


Figure 1: Software architecture of the MpCCI FSIMapper in batch usage (cf. section 6)

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), MentorGraphics FloTHERM XT (.txt), MentorGraphics FloEFD (.efdfea), MagNet (.vtk), FINE/Turbo (.cgns), MSC NASTRAN Bulk (.bdf, .nas, .dat), Abaqus (.inp) 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 [2.2.4 License](#) ▷).

If you did not install Perl or if your Perl installation is too old see [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
`tar zxvf mpccifsimapper-<FSIMapper-version>.tar.gz`
or, if the z option is not available, with the commands
`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: `export PATH=<your install dir>/FSIMapper/bin:$PATH` for korn shell (ksh) and bash resp.
`setenv PATH <your install dir>/FSIMapper/bin:$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 `<your install dir>\FSIMapper\bin`.

3. Windows only: Add firewall rules for MpCCI FSIMapper executables.

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

- "`<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 strawberryperl.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 [2.2.4.2 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, lmgard, 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:	lmgard
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_LOGFILE>
```

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/Sxxxx", etc. Include commands similar to the following.

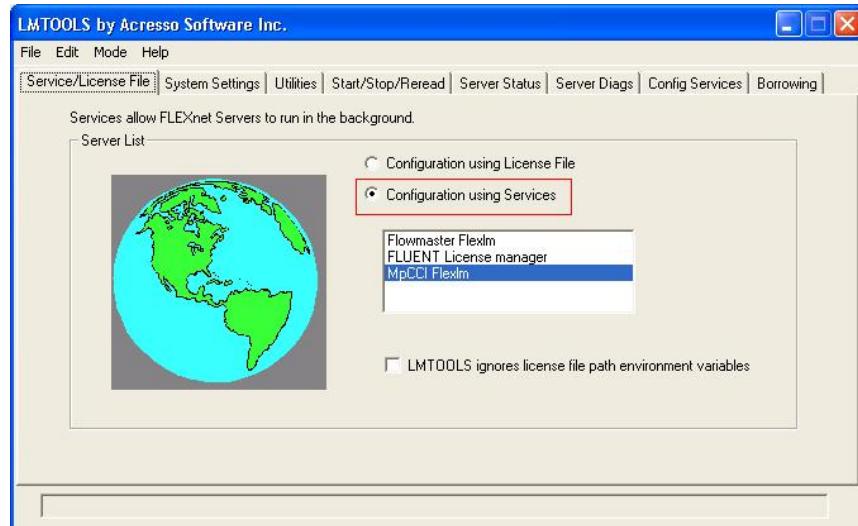
```
<LICENSE_TOOL_INSTALLATION>/bin/lmgrd -c <LICENSE_FILE> -l <LICENSE_LOG_FILE>
```

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

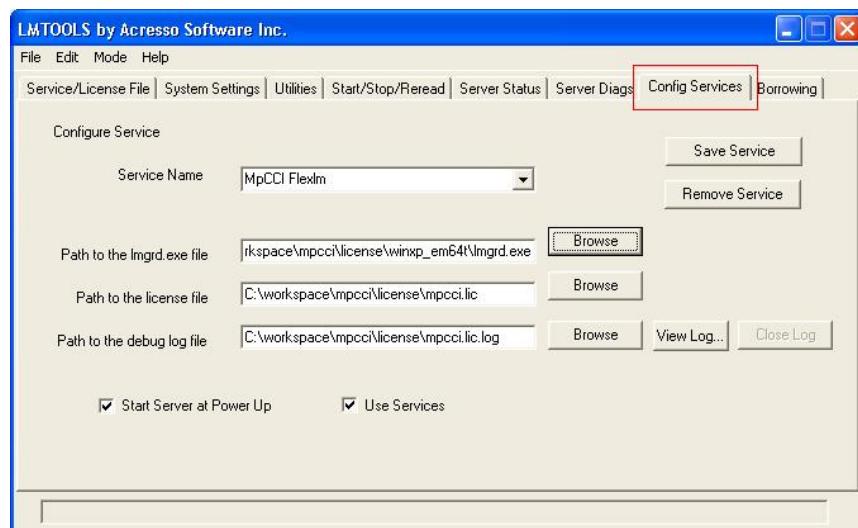
the directory.

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"

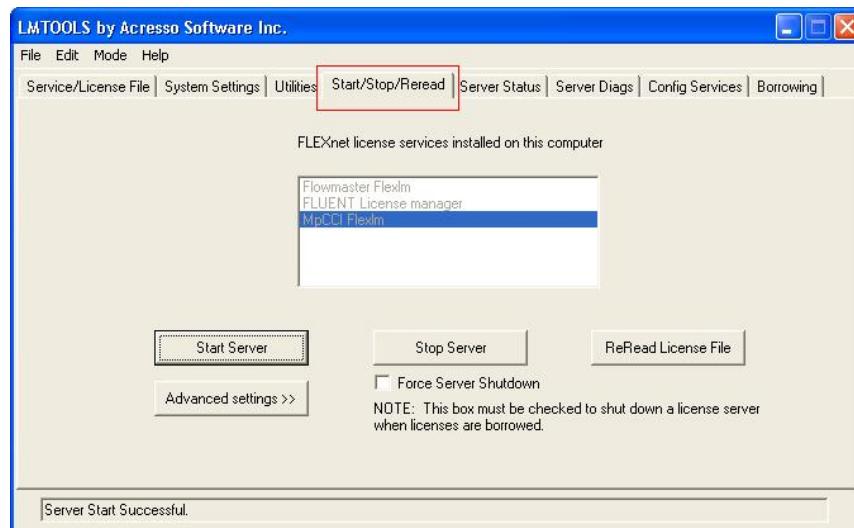


- Select in the Service/License File tab section the option Configuration using Services.
- Click the Config Services tab section.



- 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.



- 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, FLOTHERMMAPLIB, FLOEFDMAPIB, FLOTHERMXT, FINETURBO 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.

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 [▷ 5.9.4 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 ▷ 7 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  after the MpCCI Visualizer has started.

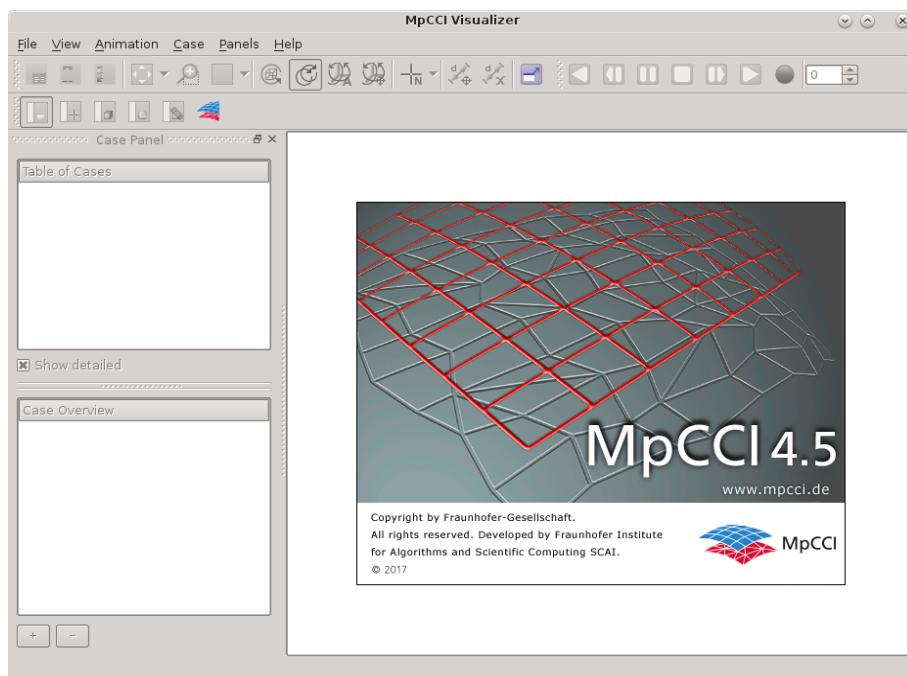


Figure 2: The MpCCI FSIMapper plugin location in the MpCCI Visualizer

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

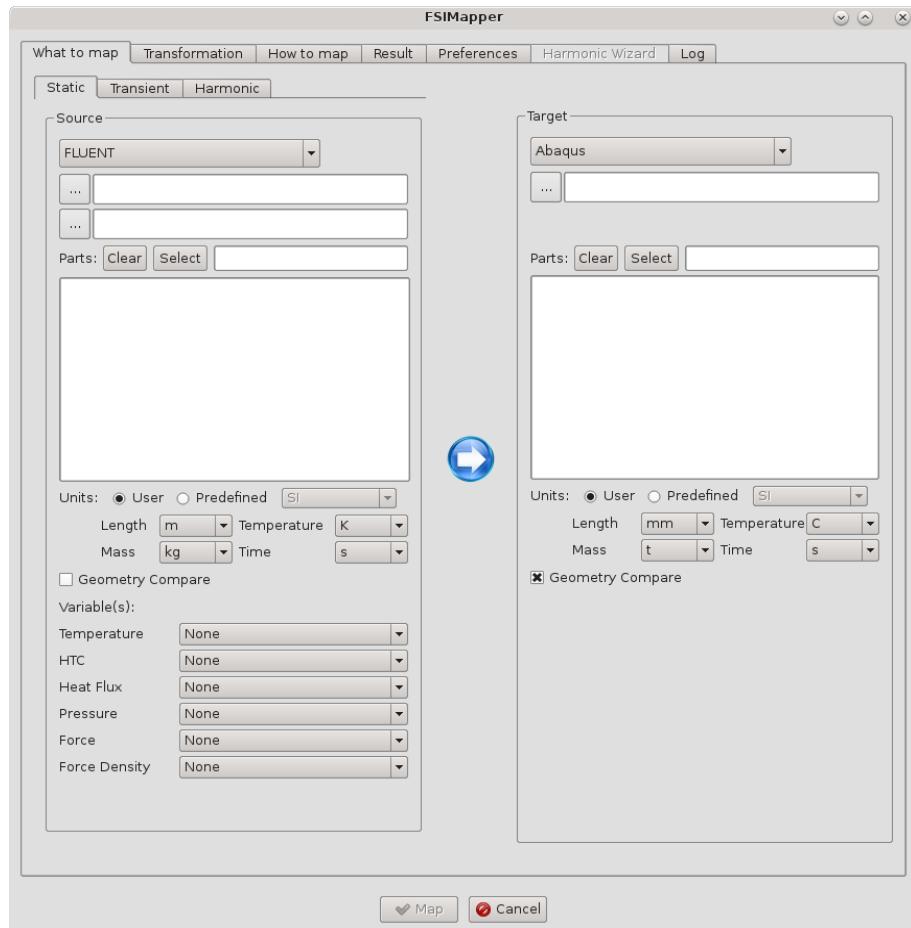


Figure 3: The “What to map” panel of the MpCCI FSIMapper

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** 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** 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.

- ① 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◀](#).

Depending on the selected source code, the following files are required:

- **FLUENT** : the native geometry (*.cas) and quantity (*.dat) file
- **FloTHERM** : resp. **FloEFD** *.flofea resp. *.efdfea 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
- **MSC NASTRAN** : ASCII file in the native bulk data format

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.

- ① 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.
- ② 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.



Figure 5: Geometry Compare check box

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](#).

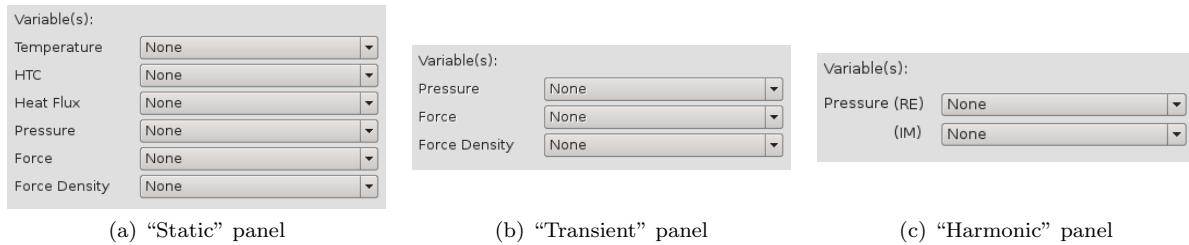


Figure 6: Quantity selection area in graphical user interface

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 and Abaqus 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

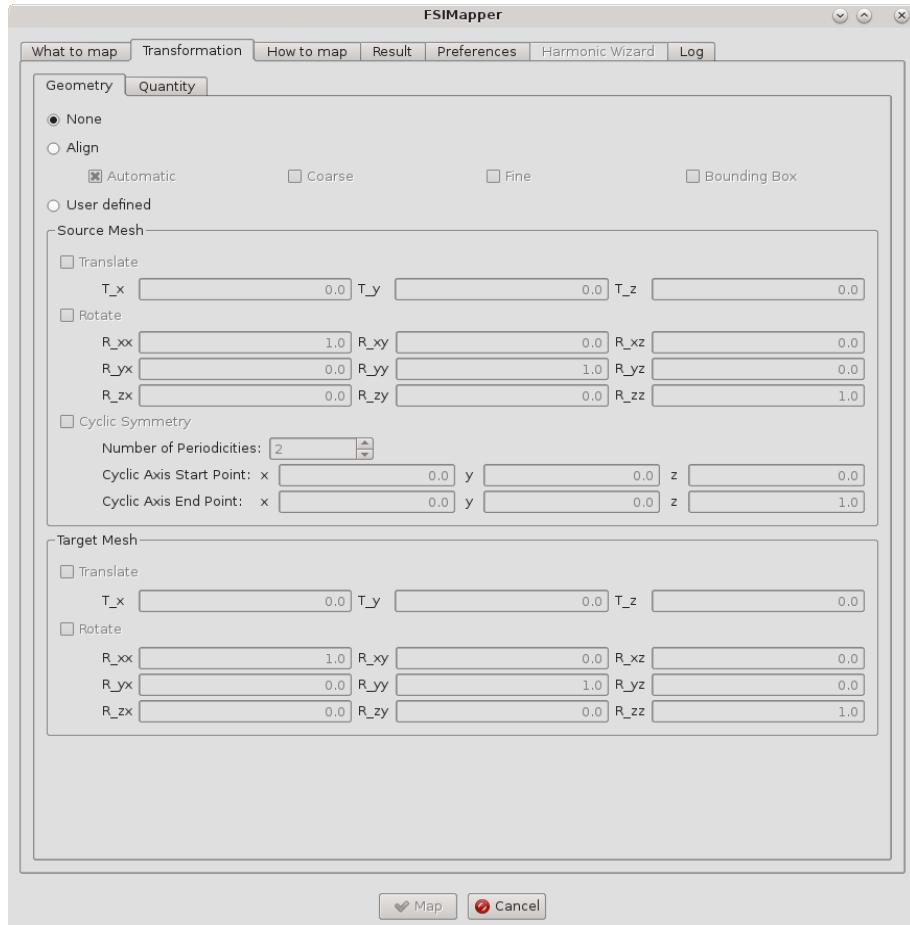


Figure 7: The “Geometry” subpanel of the MpCCI FSIMapper

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. 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 geometrie 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 3×3 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 and two points (corresponding to the source model position) which define the axis of cyclic symmetry.

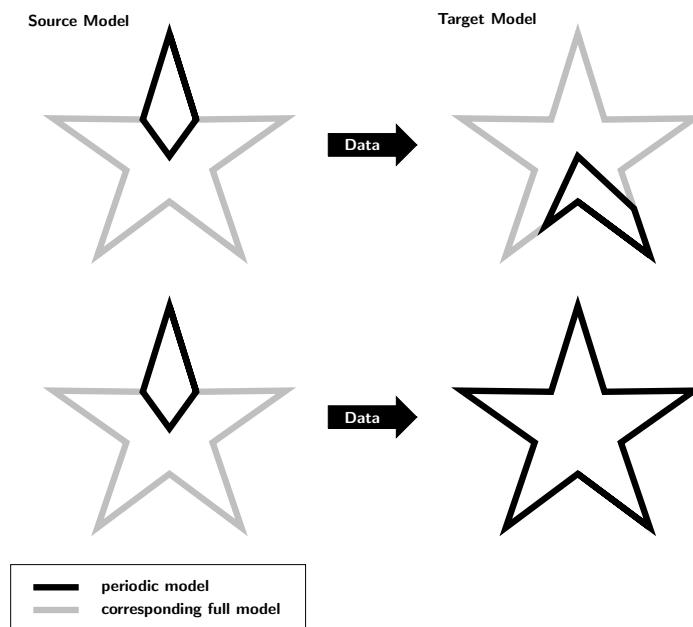


Figure 8: Mapping of data between different periodic sections (black lines) which represent the same full model (grey lines)

In the “Quantity” subpanel transformations of the quantity can be defined, currently only with respect to periodicity. Following options are available:

None

No quantity transformation is needed.

User defined**Cyclic Symmetry**

The data on cyclic symmetric models can exhibit a special periodicity, the so-called (forward/backward) nodal diameter $ND \in \mathbb{N}$ (or cyclic symmetry mode). The maximal nodal diameter is $n/2$ for an even number of periodic sections n , and $(n - 1)/2$ if n is odd.

It defines the phase lag between the data of two neighboring sections as $\frac{2\pi \cdot ND}{n}$ (forward) or $\frac{-2\pi \cdot ND}{n}$ (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 9](#) for the example of a periodic model with $n = 8$.

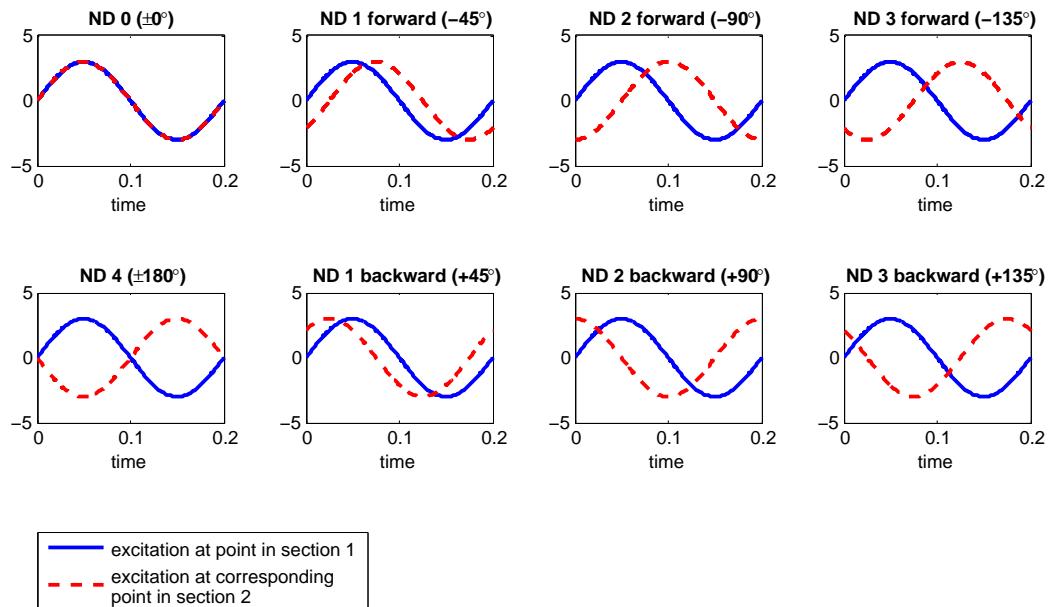


Figure 9: Phase lag between two neighboring sections for different nodal diameters on a 8-periodic model

The nodal diameters available in MpCCI FSIMapper are dependent on the number of cyclic symmetric sections n and the analysis type:

Analysis Type	n even/odd	Available Nodal Diameters
Static		0
Transient	n even	$0, n/2$
	n odd	0
Harmonic	n even	$0, 1, \dots, n/2$
	n odd	$0, 1, \dots, (n - 1)/2$

The nodal diameters $0 < ND < n/2$ are referred to as “paired”. If a paired nodal diameter is selected in a harmonic analysis, it is necessary to select both real and imaginary part of the

quantity. Both quantities are used simultaneously in order to create the data of the full model.

Only with a properly selected (forward/backward) nodal diameter, the data will be spatially continuous over the periodic boundaries in the corresponding full model which is the prerequisite of a correct mapping to a full model or a model with different section shape, as shown in [Figure 8](#). The necessary information is given by the user in the “Quantity” subpanel, see [Figure 10](#). The default is nodal diameter 0.

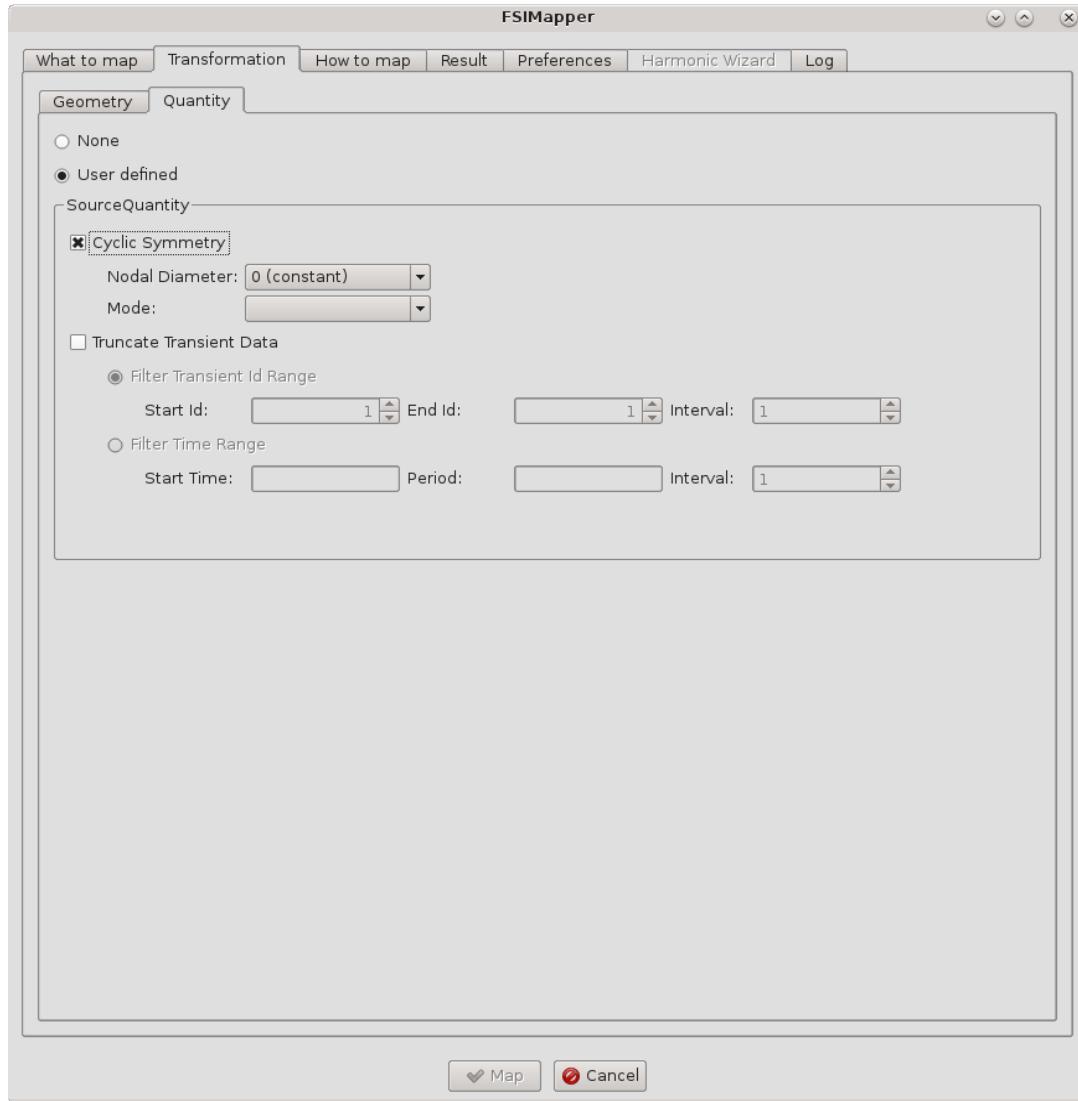


Figure 10: The “Quantity” subpanel of the MpCCI FSIMapper GUI

- (!) Which forward/backward nodal diameter needs to be chosen in harmonic analyses depends on the application. Periodic (rotating) disturbances produced by the k -th harmonic of an m -periodic part lead to a forward excitation nodal diameter $ND \in \{0, 1, \dots, [n/2]\}$, 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

- (!) 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 \vec{d} , then it is equivalent to define it as backward resp. forward nodal diameter ND with axis $-\vec{d}$.
- (!) 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.

4.3.1 Truncation of transient result data

In the “Quantity” subpanel of the “Transformation” Panel (see [Figure 10](#)) the data import of transient result data, e.g. pressure and forces, can be truncated using the following options:

Filter Transient Id Range Start Id

Read transient data beginning at this time step integer id.

End Id

Last transient time step integer id to be read from file.

Interval

Read only every n -th time step id.

Filter Time Range Start Time

Read transient data beginning at this time value.

End Time

Last transient time step value to be read from file.

Interval

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 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.

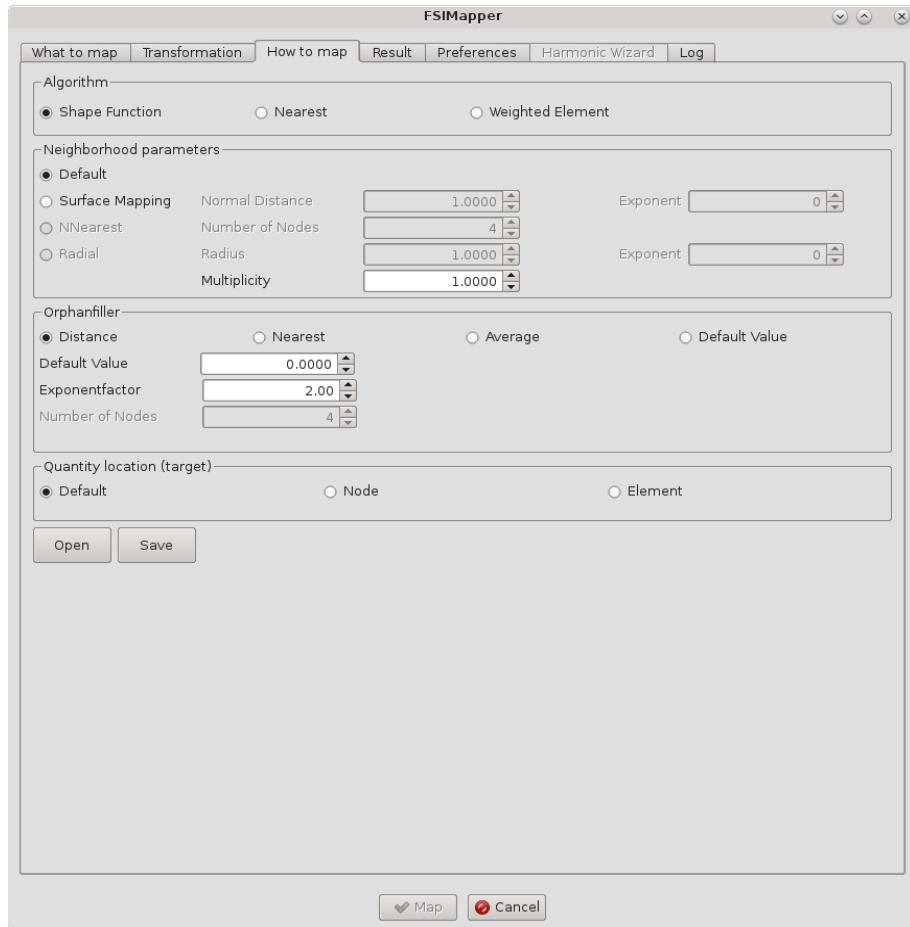


Figure 11: The “How to map” panel of the MpCCI FSIMapper

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 [▷ 7 Numerical Methods](#), 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

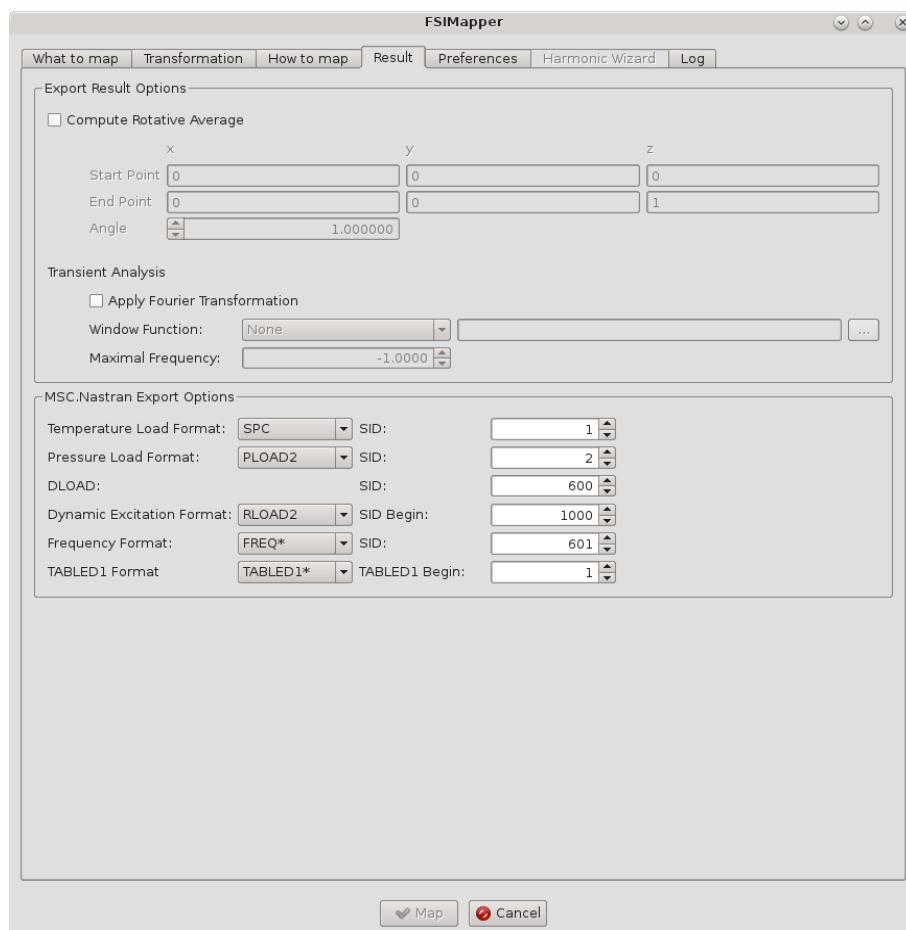


Figure 12: The “Result” panel of the MpCCI FSIMapper

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 FSIMappper 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

When using transient simulation results as mapping source, it is possible to perform a Fourier transformation on the data. The resulting complex quantities can be used in frequency response analyses.

To export loads for this type of analyses select “Apply Fourier Transformation” when having selected the “Transient” panel in “What to map” panel.

- (!) 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 devided by the number of time steps minus 1. A warning is displayed in the Log panel.

The transient data is mapped to the target model, which is shown in MpCCI Visualizer. The mapped quantities are then Fourier transformed and written to the MpCCI FSIMapper output file.

The transient signal $(s(t_k))_{k=0}^{n-1}$ can be prepared for the Fourier transformation by multiplying it with a so-called window function w . Several predefined window functions¹ are offered:

- **Rectangular**

$$w(k) = 1, \quad k = 0, \dots, n - 1$$

- **Von-Hann**

$$w(k) = 0.5 - 0.5 \cdot \cos\left(\frac{2\pi k}{n-1}\right), \quad k = 0, \dots, n - 1$$

- **Hamming**

$$w(k) = 0.54 - 0.46 \cdot \cos\left(\frac{2\pi k}{n-1}\right), \quad k = 0, \dots, n - 1$$

- **Blackman**

$$w(k) = 0.42 - 0.5 \cdot \cos\left(\frac{2\pi k}{n-1}\right) + 0.08 \cdot \cos\left(\frac{4\pi k}{n-1}\right), \quad k = 0, \dots, n - 1$$

- **Blackman-Harris**

$$w(k) = a_0 - a_1 \cos\left(\frac{2\pi k}{n-1}\right) + a_2 \cos\left(\frac{4\pi k}{n-1}\right) - a_3 \cos\left(\frac{6\pi k}{n-1}\right), \quad k = 0, \dots, n - 1$$

with $a_0 = 0.35875$, $a_1 = 0.48829$, $a_2 = 0.14128$, $a_3 = 0.01168$.

- **Blackman-Nuttall**

$$w(k) = a_0 - a_1 \cos\left(\frac{2\pi k}{n-1}\right) + a_2 \cos\left(\frac{4\pi k}{n-1}\right) - a_3 \cos\left(\frac{6\pi k}{n-1}\right), \quad k = 0, \dots, n - 1$$

with $a_0 = 0.3635819$, $a_1 = 0.4891775$, $a_2 = 0.1365995$, $a_3 = 0.0106411$.

¹taken from <https://de.wikipedia.org/wiki/Fensterfunktion>, July 5, 2016

- **Barlett**

$$w(k) = \frac{2}{n-1} \cdot \left(\frac{n-1}{2} - \left| k - \frac{n-1}{2} \right| \right), \quad k = 0, \dots, n-1$$

- **Barlett-Hann**

$$w(k) = 0.62 - 0.48 \cdot \left| \frac{k}{n-1} - 0.5 \right| - 0.38 \cdot \cos \left(\frac{2\pi k}{n-1} \right), \quad k = 0, \dots, n-1$$

- **Cosinus**

$$w(k) = \cos \left(\frac{\pi k}{n-1} - \frac{\pi}{2} \right), \quad k = 0, \dots, n-1$$

- **Tukey**

$$w(k) = \begin{cases} \frac{1}{2} \left[1 + \cos \left(\frac{2\pi k}{\alpha(n-1)} - \pi \right) \right], & \text{if } 0 \leq k \leq \frac{\alpha(n-1)}{2} \\ 1, & \text{if } \frac{\alpha(n-1)}{2} \leq k \leq (n-1)(1 - \frac{\alpha}{2}) \\ \frac{1}{2} \left[1 + \cos \left(\frac{2\pi k}{\alpha(n-1)} - \pi(\frac{2}{\alpha} - 1) \right) \right], & \text{if } (n-1)(1 - \frac{\alpha}{2}) \leq k \leq (n-1) \end{cases}$$

with $\alpha = 0.5$.

- **Lanczos**

$$w(k) = \operatorname{sinc} \left(\frac{2k}{n-1} - 1 \right), \quad k = 0, \dots, n-1$$

- **File**

The user can define a one-columned file, where the n window function values $w(k)$, $k = 0, \dots, n-1$ are defined.

The Fourier transformation is applied to the signal $(s(t_k) \cdot w(k))_{k=0}^{n-1}$.

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

$$q_j = A \cdot \cos(2\pi \cdot j/n + \phi), \quad j = 1, \dots, n$$

with the amplitude $A = \sqrt{x^2 + y^2}$ and the phase shift $\phi = \operatorname{atan2}(y, x)$. For vector quantities this is done component-wise.

The ccvx-file can be read into MpCCI Visualizer by **File → Open...**. By the “Play” button the n 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.



Figure 13: The Preferences panel of the MpCCI FSIMapper

4.6 The “Preferences” Panel

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

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

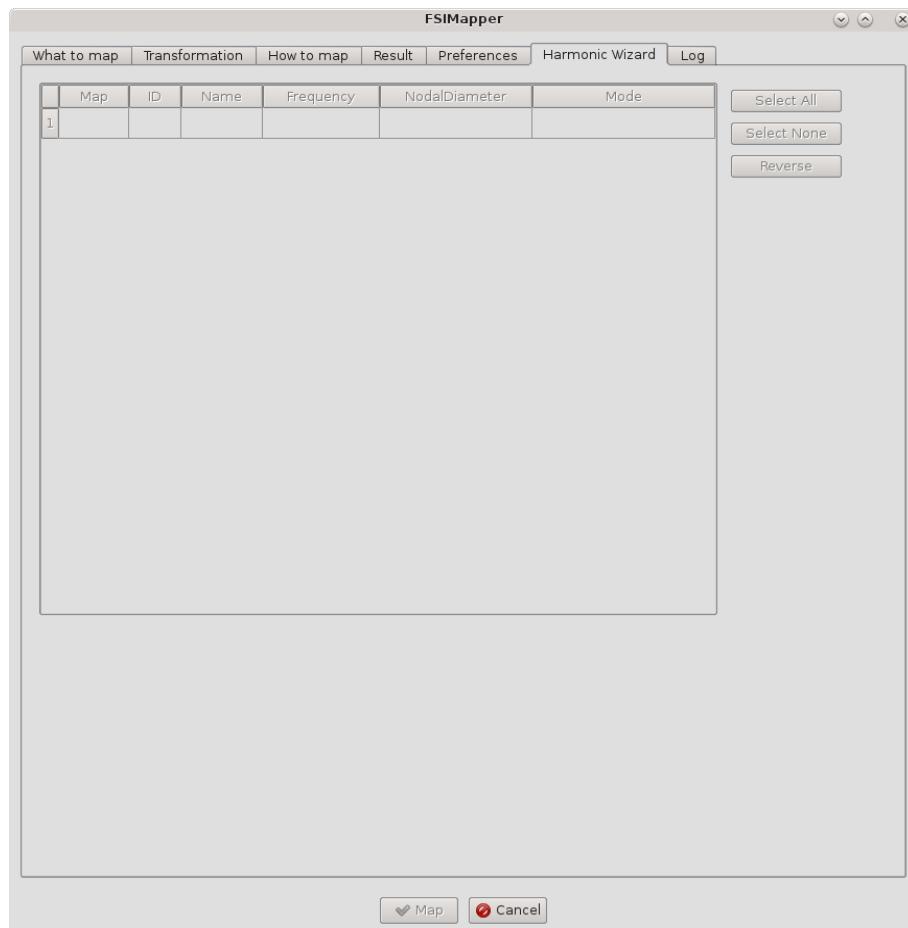


Figure 14: The “Harmonic Wizard” panel of the MpCCI FSIMapper

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

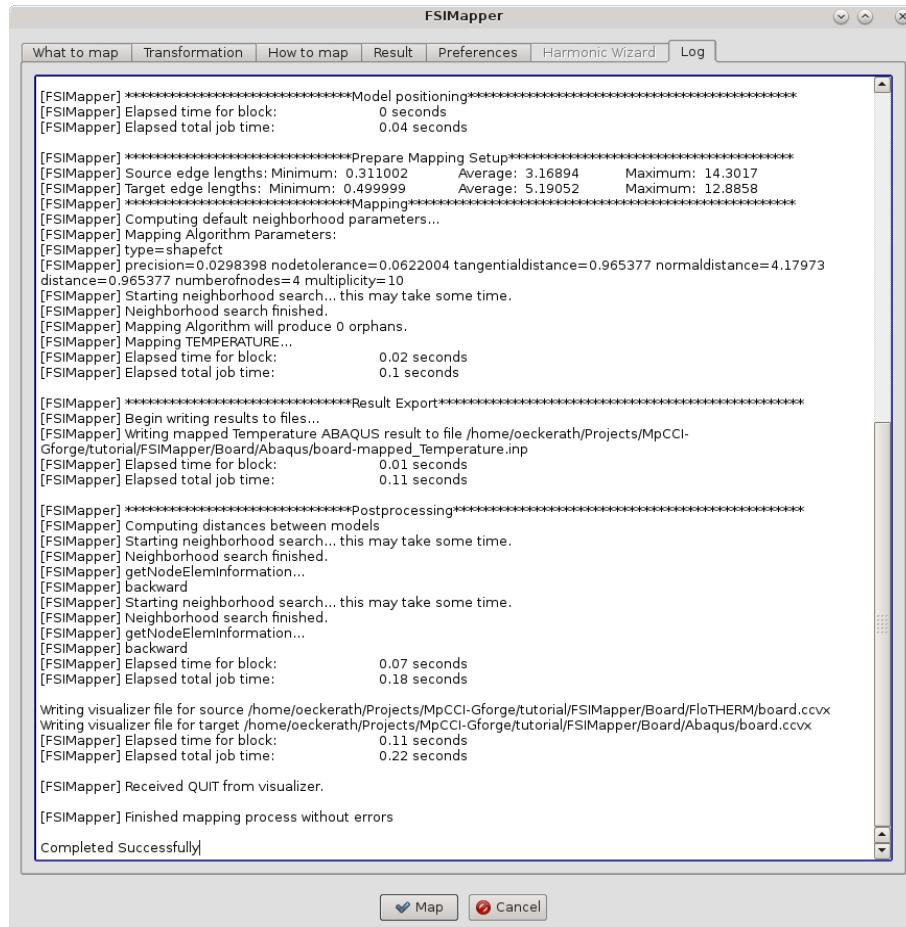


Figure 15: The “Log” panel of the MpCCI FSIMapper

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 all saved scalar 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 16.0.x are supported and more recent versions should be compatible if native file format does not change.

 Only zone types “wall” and “fluid” can be coupled with MpCCI FSIMapper.

5.2.1 Supported Quantities

Quantity	Dim.	Default Unit	Location	Comments
TEMPERATURE	Scalar	K	fluid	
FILMTEMPERATURE	Scalar	K	wall	
HTC (computed)	Scalar	W/(m ² K)	wall	$\alpha = q/(T_w - T_c)$
wallfuncHTC	Scalar	W/(m ² K)	wall	stored in UDM0
WALLHEATFLUX	Scalar	W/m ²	wall	
STATICPRESSURE	Scalar	N/m ²	wall	
ABSPRESSURE	Scalar	N/m ²	wall	STATICPRESSURE + operating pressure
OVERPRESSURE	Scalar	N/m ²	wall	STATICPRESSURE + operating pressure - reference pressure

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
- α = heat transfer coefficient, W/(m²K)
- A = heat transfer surface area, m²

- ΔT = 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

Quantity	Dim.	Default Unit	Location	comments
TEMPERATURE	Scalar	K	wall	
HEATTRANSFERCOEF	Scalar	W/(m ² K)	wall	
HEATFLUX	Scalar	W/m ²	wall	
PRESSURE	Scalar	N/m ²	wall	
REP_x	Scalar	N/m ²	volume	complex harmonic pressures (NLH method)
IMP_x				

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

Quantity	Dim.	Unit	Location
FILMTEMPERATURE	Scalar	K	Face
HTC	Scalar	W/(m ² K)	Face
PRESSURE	Scalar	N/m ²	Face

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

Quantity	Dim.	Default Unit	Location
TEMPERATURE	Scalar	C	solid

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

Quantity	Dim.	Default Unit	Location
TEMPERATURE	Scalar	K	solid/wall
HTC	Scalar	W/(m ² K)	wall
SURFACE_HEAT_FLUX	Scalar	W/m ²	wall
PRESSURE	Scalar	N/m ²	wall
RELATIVE_PRESSURE	Scalar	N/m ²	wall

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

5.7.3 Supported Quantities

Quantity	Dim.	Default Unit	Location	Comments
SFD	Vector	N/m ²	wall	Surface Force Density

MpCCI FSIMapper	VTK Element Type
3D_SURF_TRIA3	5
3D_SURF_TRIA6	22
3D_SURF_QUAD4	9
3D_SURF_QUAD8	23
2D_SURF_LINE2	3
2D_SURF_LINE3	21

Table 1: Supported MagNet (.vtk) structural elements.

5.8 Abaqus

5.8.1 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).

5.8.2 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 SI-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:

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.

5.8.3 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_TransientPressure.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"`

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.

5.8.4 Elements

5.8.5 Supported Quantities

Quantity	Dim.	Unit (see Table 4)	Location	Abaqus Keyword
TEMPERATURE	Scalar	θ	*Elset	<code>*TEMPERATURE</code>
FILMTEMPERATURE	Scalar	θ	*Surface	<code>*SFILM</code>
HTC	Scalar	$JT^{-1}L^{-2}\theta^{-1}$	*Surface	<code>*SFILM</code>
WALLHEATFLUX	Scalar	$JL^{-2}T^{-1}$	*Surface	<code>*DFLUX</code>
PRESSURE	Scalar	$ML^{-1}\theta^{-2}$	*Surface	<code>*DLOAD</code>
FORCE	Vector	MLT^{-2}	*Surface	<code>*CLOAD</code>

MpCCI FSIMapper	stress/displ.	heat transf./mass diff.	heat transf. conv./diff.	acoustic
3D_VOL_HEX8	C3D8	DC3D8	DCC3D8	AC3D8
3D_VOL_HEX20	C3D20	DC3D20	DCC3D20	AC3D20
3D_VOL_HEX27	C3D27	DC3D27	DCC3D27	AC3D27
3D_VOL_TET4	C3D4	DC3D4	DCC3D4	AC3D4
3D_VOL_TET10	C3D10	DC3D10	DCC3D10	AC3D10
3D_VOL_WEDGE6	C3D6	DC3D6	DCC3D6	AC3D6
3D_VOL_WEDGE15	C3D15	DC3D15	DCC3D15	AC3D15
2D_VOL_TRIA3	CPS3			
2D_VOL_QUAD4	CPS4			
2D_VOL_TRIA6	CPS6			
2D_VOL_QUAD8	CPS8			

Table 2: Supported Abaqus solid (continuum) elements. Type variants like reduced integration or coupled temperature-displacement are handled as base type.

MpCCI FSIMapper	stress/displ.	heat transfer
3D_SHELL_TRIA3	S3	DS3
3D_SHELL_TRIA6	S6	DS6
3D_SHELL_QUAD4	S4	DS4
3D_SHELL_QUAD8	S8	DS8
3D_SHELL_QUAD9	S9	

Table 3: Supported Abaqus structural elements. Type variants like reduced integration or coupled temperature-displacement are handled as base type.

Unit Symbol	Length L	Mass M	Temperature θ	Time T	Energy J

Table 4: Abaqus unit symbols.

5.9 ANSYS

5.9.1 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.

5.9.2 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 5](#).

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

ANSYS Element Types	MpCCI FSIMapper
PLANE13, PLANE25, PLANE42, PLANE55, PLANE67, PLANE181, PLANE182, SHELL28, SHELL41, SHELL43, SHELL57, SHELL63, SHELL131 SHELL143, SHELL157, HYPER56, VISCO106, CPT212, SURF252,	3D_SHELL_TRIA3, 3D_SHELL_QUAD4
PLANE2, PLANE35	3D_SHELL_TRIA3, 3D_SHELL_TRIA6
SHELL91, SHELL93, SHELL99, SHELL132, SHELL150, SHELL281	3D_SHELL_TRIA6, 3D_SHELL_QUAD4, 3D_SHELL_QUAD8
PLANE53, PLANE145, PLANE223, PLANE77, PLANE78, PLANE82, PLANE83, PLANE121, PLANE183, PLANE146, PLANE230 SURF152, SURF154 CPT213, HYPER74, VISCO88, VISCO108,	
SOLID5, SOLID45, SOLID46, SOLID62, SOLID64, SOLID65, SOLID69, SOLID70, SOLID96, SOLID97, SOLID164, SOLID285, SOLID185, HYPER58, HYPER86, VISCO107, SOLSH190, CPT215	3D_VOL_TET4, 3D_VOL_WEDGE6, 3D_VOL_HEX8, 3D_VOL_PYRAM5
SOLID87, SOLID92, SOLID98, SOLID123, SOLID127, SOLID148, SOLID168, SOLID186, SOLID187, SOLID227, SOLID232, SOLID237, CPT217	3D_VOL_TET4, 3D_VOL_TET10
SOLID90, SOLID95, SOLID122, SOLID117, SOLID122, SOLID128, SOLID147, SOLID186, SOLID191, SOLID226, SOLID231, SOLID236, VISCO89, CPT216	3D_VOL_TET4, 3D_VOL_TET10, 3D_VOL_WEDGE6, 3D_VOL_WEDGE15, 3D_VOL_HEX8, 3D_VOL_HEX20, 3D_VOL_PYRAM5, 3D_VOL_PYRAM13
MESH200	MPCCI_ETYP_LINE2, MPCCI_ETYP_TRIA3, MPCCI_ETYP_QUAD4, MPCCI_ETYP_QUAD8

Table 5: Supported ANSYS element types.

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 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. 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 6](#) 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.

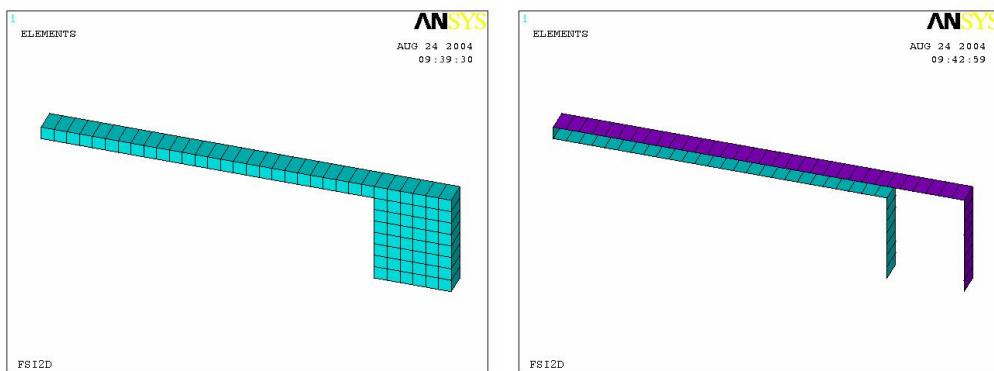


Figure 16: ANSYS volume model and additional dummy SHELL63 elements for surface coupling

<i>Quantity Element</i>	TEMPERA-TURE	HTC	WALL-HEATFLUX	PRESSURE
SOLID5				+
SOLID45				+
SHELL57	+	+	+	
SHELL63				+
SHELL93				+
SURF151	+	+	+	

Table 6: Quantities supported by MpCCI FSIMapper for different ANSYS elements.

5.9.3 Supported Quantities

Quantity	Dim.	Unit	Location	ANSYS Keyword
TEMPERATURE	Scalar	θ	Solid/Shell	BF,,,TEMP
HTC	Scalar	$JT^{-1}L^{-2}\theta^{-1}$	Shell	SFE,,,CONV
WALLHEATFLUX	Scalar	$JL^{-2}T^{-1}$	Shell	SFE,,,HFLUX
PRESSURE	Scalar	$ML^{-1}\theta^{-2}$	Shell	SFE,,,PRES

5.9.4 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 ▷3 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.

5.10 MSC NASTRAN

5.10.1 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).

5.10.2 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.

5.10.3 Elements

MSC NASTRAN	MpCCI FSIMapper
CTRIA3	3D_SHELL_TRIA3
CTRIA6	3D_SHELL_TRIA6
CQUAD4	3D_SHELL_QUAD4
CQUAD8	3D_SHELL_QUAD8
CQUAD	3D_SHELL_QUAD9
CTETRA	3D_VOL_TET4/10
CHEXA	3D_VOL_HEX8/20
CPENTA	3D_VOL_WEDGE6

Table 7: Supported MSC NASTRAN structural elements.

5.10.4 Supported Quantities

Quantity	Dim.	Default Unit	Location	MSC NASTRAN Keyword
(FILM)TEMPERATURE	Scalar	θ	Shell/Volume	TEMP SPC
PRESSURE	Scalar	$ML^{-1}\theta^{-2}$	Shell	PLOAD PLOAD2 PLOAD4 RLOAD1 RLOAD2
FORCE	Vector	MLT^{-2}	Node	FORCE TLOAD1 RLOAD1 RLOAD2

5.11 LS-Dyna

MpCCI FSIMapper supports writing of LS-Dyna files in ASCII keyword data format.

5.11.1 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.

5.11.2 Elements

LS-Dyna	MpCCI FSIMapper
3 node shell elements	3D_SHELL_TRIA3
4 node shell elements	3D_SHELL_QUAD4

Table 8: Supported LS-Dyna structural elements.

5.11.3 Supported Quantities

Quantity	Dim.	Default Unit	Location	LS-Dyna Keyword
FORCE	Vector	MLT^{-2}	Node	*DATABASE_FREQUENCY_BINARY_D3SSD *FREQUENCY_DOMAIN_SSDFORCE *DEFINE_CURVE

5.11.4 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.efdf
> 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” ([▷ 5.9.4 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:

Zonename	Zonetype	ZoneID
----------	----------	--------

6.1.1.1 Example

As an example the output of the FLUENT scanner for a simple case file “exampleFluent.cas” is shown below.

```
> fsimapper scan FLUENT ./exampleFluent.cas
<CPLDATA>
# This file was automatically created for the MpCCI GUI.
#@ Code: "FLUENT"
#@ Scan: "....../MpCCI/codes/FLUENT/Scanner.pm"
#@ File: "exampleFluent.cas"
#@ Path: "...."
#@ Date: "...."
#@ User: "...."
#@ Host: "...."
#
# MpCCI relevant information:
#   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:

```
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 FLOEFDLIB in the generic MpCCI FSIMapper scan command, e.g.

```
> fsimapper scan FLOEFDLIB 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 FLOEFDLIB ./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
FLOEFDLIB
sourceMeshPartIds
2
```

A mapping of the surface **Surface-1** would require to add

```
sourceFileType
FLOEFDLIB
```

```
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.

```
> 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

```
> 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 [▷ 5.9.4 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 [▷ 5.9 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 (**SOLIDIS__**) 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 **SOLIDIS__** 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 [6.1.6.1 Example](#) 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.

```
> 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:

```
targetFileType
ABAQUS
targetMeshPartIds
2
```

A mapping of temperature on elset SET-1 would require to add

```
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.

```
> fsimapper scan ENSIGHT exampleEnSight.case
<CPDATA>
# This file was automatically created for the MpCCI GUI.
#@ Code: "EnSight"
#@ Scan: "1"
#@ File: "exampleEnSight.case"
#@ Path: "..."
```

```

#@ Date: "..."
#@ User: "..."
#@ Host: "..."
#
# MpCCI relevant information:
#   Model dimensions : unknown
#   Coordinate system : unknown
#   Solution type     : unknown
#   Load cases        : unknown
#   Unit system       : unknown
#   Precision         : unknown
#
      Total_Temperature      SCALAR      1
</CPLDATA>
<CPLDATA>
#@ Code: "EnSight"
#@ File: "exampleEnSight.case"
#@ Path: "..."
wall_1          S      2
inlet           S      3
wall_2          S      4
flexible_pin   S      5
outlet          S      6
Region_1        V      1
</CPLDATA>

```

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

```

sourceFileType
ENSIGHTGOLD

```

to the MpCCI FSIMapper configuration file if the case EnSight is used as source format

```
targetFileType  
ENSIGHTGOLD
```

otherwise.

6.2 Comparing Geometries

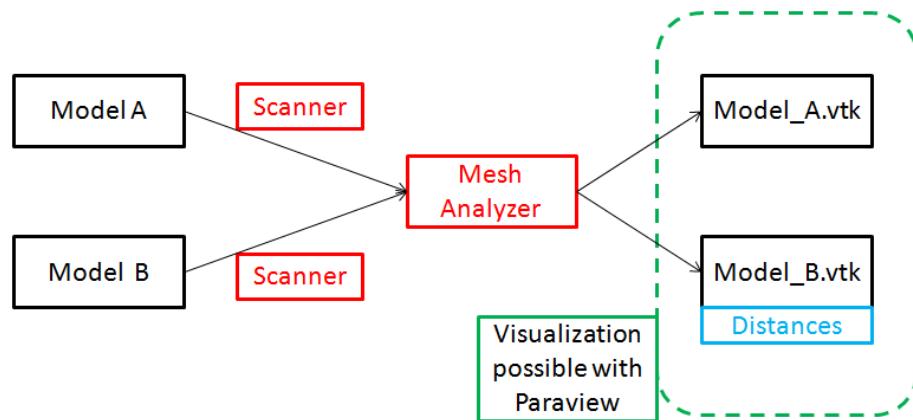


Figure 17: The workflow of the MpCCI FSIMapper in batch usage when comparing two geometries

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:

```
> 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 ([▷5.8 Abaqus](#)), ANSYS or MSC NASTRAN ([▷5.10 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: $\text{Pa} = \frac{\text{kg}}{\text{m s}^2}$, 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:

```
#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.

Keyword	Value
sourceFileType	FLUENT → Code FLUENT FLOTHERMMAPLIB → Code FloTHERM FLOEFDMAPLIB → Code FloEFD ENSIGHTGOLD → EnSight Case Gold
targetFileType	ABAQUS → Code Abaqus ANSYS → Code ANSYS NASTRAN → Code MSC NASTRAN ENSIGHTGOLD → EnSight Case Gold
sourceQuantityMode	ABSPRESSURE ⇒ Read and map pressure STATICPRESSURE ⇒ Read and map static pressure OVERPRESSURE ⇒ Read and map overpressure FILMTEMPHTC ⇒ Read and map surface film temperature and htc 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 WALLHEATFLUX ⇒ Read and map surface heat flux GEOMETRYCOMPARE ⇒ Compare both geometries default: NONE
targetQuantityMode	GEOMETRYCOMPARE default: NONE
sourceMeshPartIds targetMeshPartIds	the zone ids of source and target mesh from the file scanner separated by whitespace
sourceMeshLengthUnit targetMeshLengthUnit	m, ft, in, cm or mm default: m
sourceTemperatureUnit targetTemperatureUnit	K or C default: K
sourceMeshMassUnit targetMeshMassUnit	kg, t or g default: kg
sourceMeshTimeUnit targetMeshTimeUnit	s or ms default: s
targetQuantityLocation	NODE, ELEMENT or DEFAULT default: DEFAULT = same location as source
optionMapAlgoType	type=shapefct type=weightedelement type=nearest default: type=shapefct
optionUseNeighborhoodDefaultParameters	true or false default: false
optionNeighborhoodSearch	list of parameters [targetMeshLengthUnit unit] necessary for the neighborhood search: normaldistance, numberofnodes and radius
optionMultiplicity	value that increases neighborhood searching range default: 1
orphanFillType	type=distance type=average type=nearest type=none default: type=distance
fillOrphansValue	value that is assigned to orphaned nodes or elements. default: 0
optionComputeRotativeAverage	true or false default: false
rotationAngle	angle in angular degree
rotationAxisBeginPoint	x-, y- and z-coordinate of starting point separated by whitespace
rotationAxisEndPoint	x-, y- and z-coordinate of end point separated by whitespace

temperatureScaleValue	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 $600/500 = 1.2$
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

A detailed description of the mapping algorithms and the concept of orphan filling can be found in [▷ 7 Numerical Methods](#). The meaning of the parameters for the different mapping algorithms is also explained in this chapter in [▷ 7.2 Parameters for the Mapping](#). The easiest solution for many applications might be to use the default parameters.

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.

```

sourceFileType
FLUENT

targetFileType
ABAQUS

sourceQuantityMode
FILMTEMPHTC

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 Numerical Methods

7.1 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.

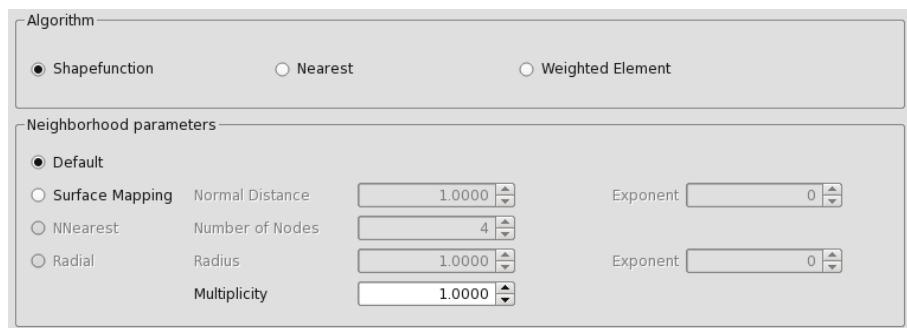


Figure 18: Mapping algorithms and parameters in MpCCI FSIMapper

7.2 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”.

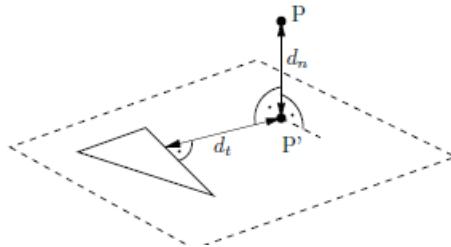


Figure 19: Normal distance d_n computation for element to node neighborhood used in shape function and weighted element mapping

A reasonable choice for the distance parameter might be

$$\text{Normal Distance} = \max(\text{SourceMaxEdgeLength}, \text{TargetMaxEdgeLength})$$

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.

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.

7.3 Orphan Filling

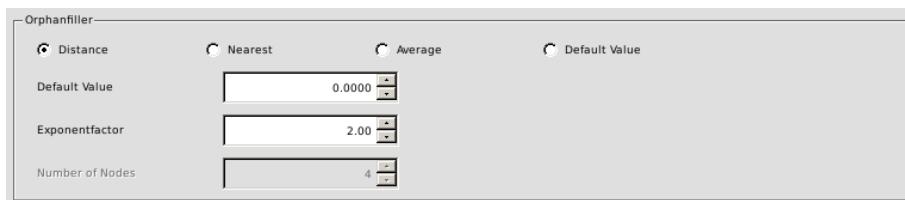


Figure 20: Methods and parameters for orphan filling in MpCCI FSIMapper

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 **Exponentfactor**. This parameter determines to which power the distance - in the denominator - is taken. The default value for the **Exponentfactor** 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.

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 21](#)), 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.

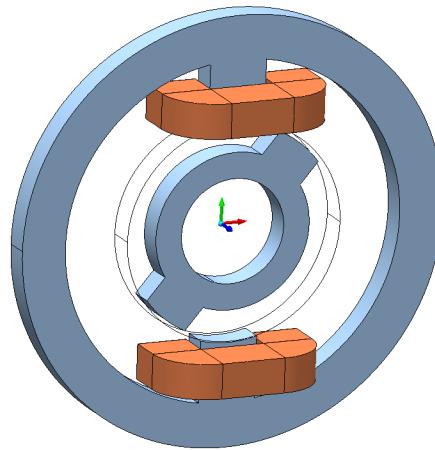


Figure 21: TEAM-24 model in MagNet

The files concerning this tutorial are located in

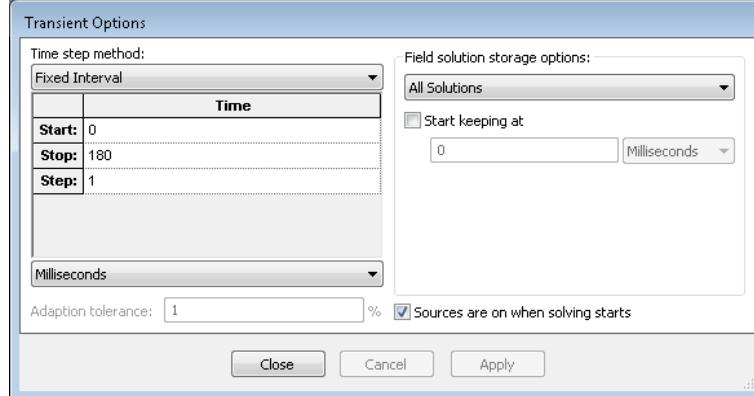
- MpCCI FSIMapper as part of the MpCCI installation:
"*<MpCCI_home>/tutorial/FSIMapper/Team24-Stator_ElectromagneticForces*"
- MpCCI FSIMapper standalone installation:
"*<MpCCIFSIMapper_home>/tutorial/Team24-Stator_ElectromagneticForces*"

Before mapping, copy the files to your working directory.

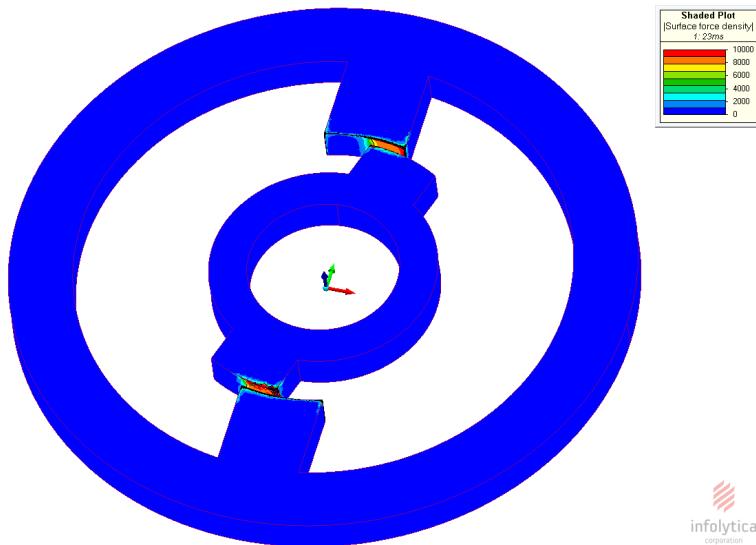
8.1.2 Source Result File

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 22](#)) 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 23](#).



[Figure 22: Options for transient MagNet simulation](#)



[Figure 23: Surface force density at 23 ms computed by MagNet](#)

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 24](#)):

- 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

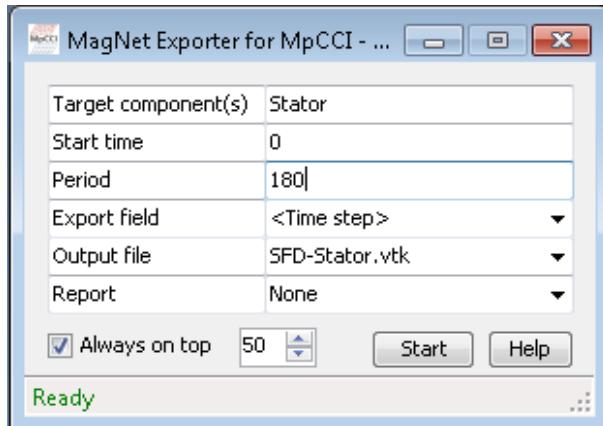


Figure 24: MagNet's "Exporter for MpCCI"

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 25](#).

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 25](#)), the prerequisites for a correct Fourier transformation are fulfilled.

- (!) In the .vtk file the SI unit system is used.

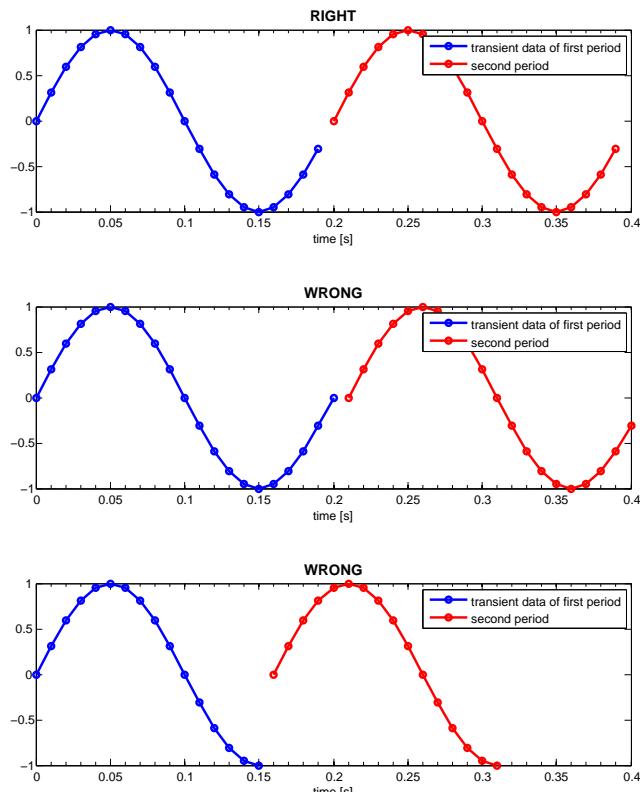


Figure 25: How to define transient data used by the Fourier transformation (without windowing). The blue dots are the data points provided for the mapping (here, a discrete sine-signal of frequency 5 Hz and sampling time step 0.01 s). The red dots are generated by stacking the data behind the end of 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 26](#). The mapping surface is defined by shell elements (green) which use the same nodes as the volume elements (blue).

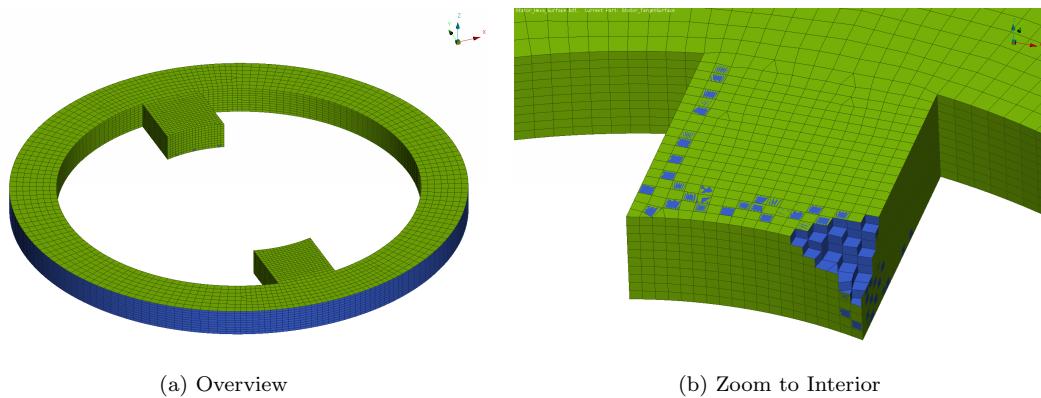


Figure 26: Target structural mesh (blue) with shell elements (green) defining the mapping surface

The unit system used is *mm-t-s* which leads to force unit *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 27](#)):

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 *mm-t-s*.

This configuration maps for each time step the electromechanical 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 28](#).

Pressing the **Map** button starts the mapping process.

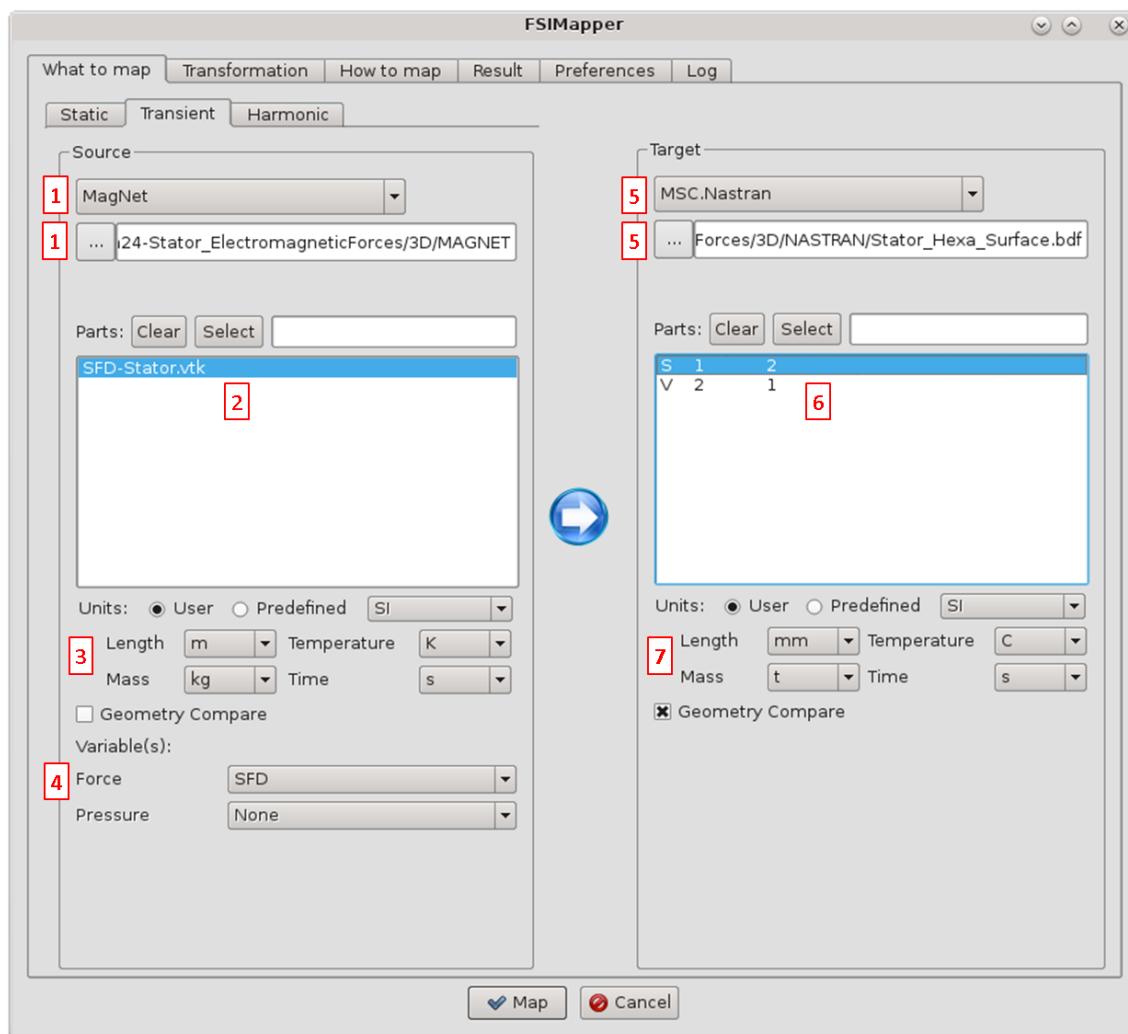


Figure 27: The “What to map” panel

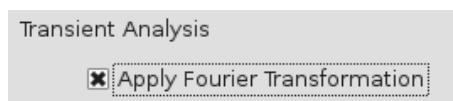


Figure 28: The “Apply Fourier Transformation” option in the “Result” panel

When the mapping finishes, the MpCCI Visualizer shows the mapping results for each time step (cf. Figure 29 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.

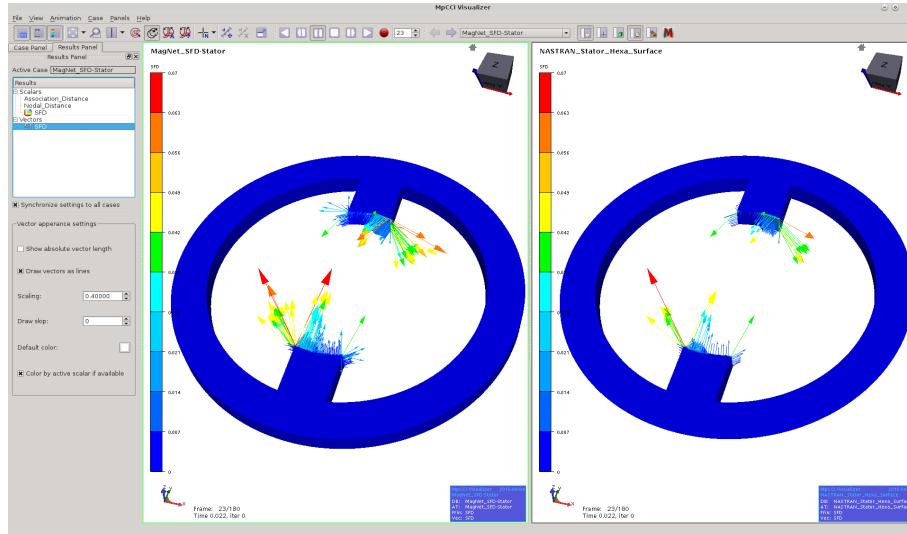


Figure 29: Transient mapping result shown in MpCCI Visualizer at 23 ms

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

$$0 \text{ Hz} \quad \text{to} \quad \frac{\lfloor L/2 \rfloor}{\tau \cdot L} = 500 \text{ Hz} \quad \text{in} \quad \frac{1}{\tau \cdot L} = 5.5 \text{ Hz} \quad \text{steps}$$

MpCCI FSIMapper 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:

```
$
$ DLOAD = 600
$ FREQ  = 601
$
$-----2-----3-----4-----5-----6-----7-----8-----9-----0-----
FREQ*          601 0.00000000E+00 5.555555556E+00 1.111111111E+01
*      1.666666667E+01 2.22222222E+01 ...
...
$
$-----2-----3-----4-----5-----6-----7-----8-----9-----0-----
$ Node 1, DOF 1
RLOAD1      600      1                  1      2
DAREA       1      1      11.00E+00
$ Real part for each frequency
TABLED1*           1
*
*      0.00000000E+00 5.050313394E-03 5.555555556E+00-1.127201827E-03
*      1.111111111E+01-2.292065593E-03 1.666666667E+01 1.449636200E-03
*      2.22222222E+01 2.616820479E-04 ...
```

```

...
$ Imaginary part for each frequency
TABLED1*          2
*
*      0.00000000E+00 0.00000000E+00 5.555555556E+00-3.482561041E-03
*      1.11111111E+01 1.660795053E-03 1.666666667E+01 1.071458938E-03
*      2.22222222E+01-8.023966910E-04 ...
...

```

Moreover, MpCCI FSIMapper creates the file "Stator_Hexa_Surface-mapped_FreqRespForce.ccvx" in order to post-process the Fourier transformed forces. It is opened in MpCCI Visualizer and shows the corresponding transient force fluctuations (at 10 pseudo time steps) for each frequency. Open **Panels**→**Results Panel** and select in the "Results" list e.g. "Frequency_005.555555555" as "Scalar" and "Vector" quantity, see Figure 30a and 30b. Switch off **View**→**Perspective Projection** and select in the **Panels**→**Settings Panel** the outline representation as shown in Figure 30c.

Pressing the  button animates the 10 pseudo time steps. With the value given in **File**→**Preferences**→**Maximum animation FPS** the animation speed is set. In this way one can visualize the frequency content of the transient signal, see Figure 31. 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.

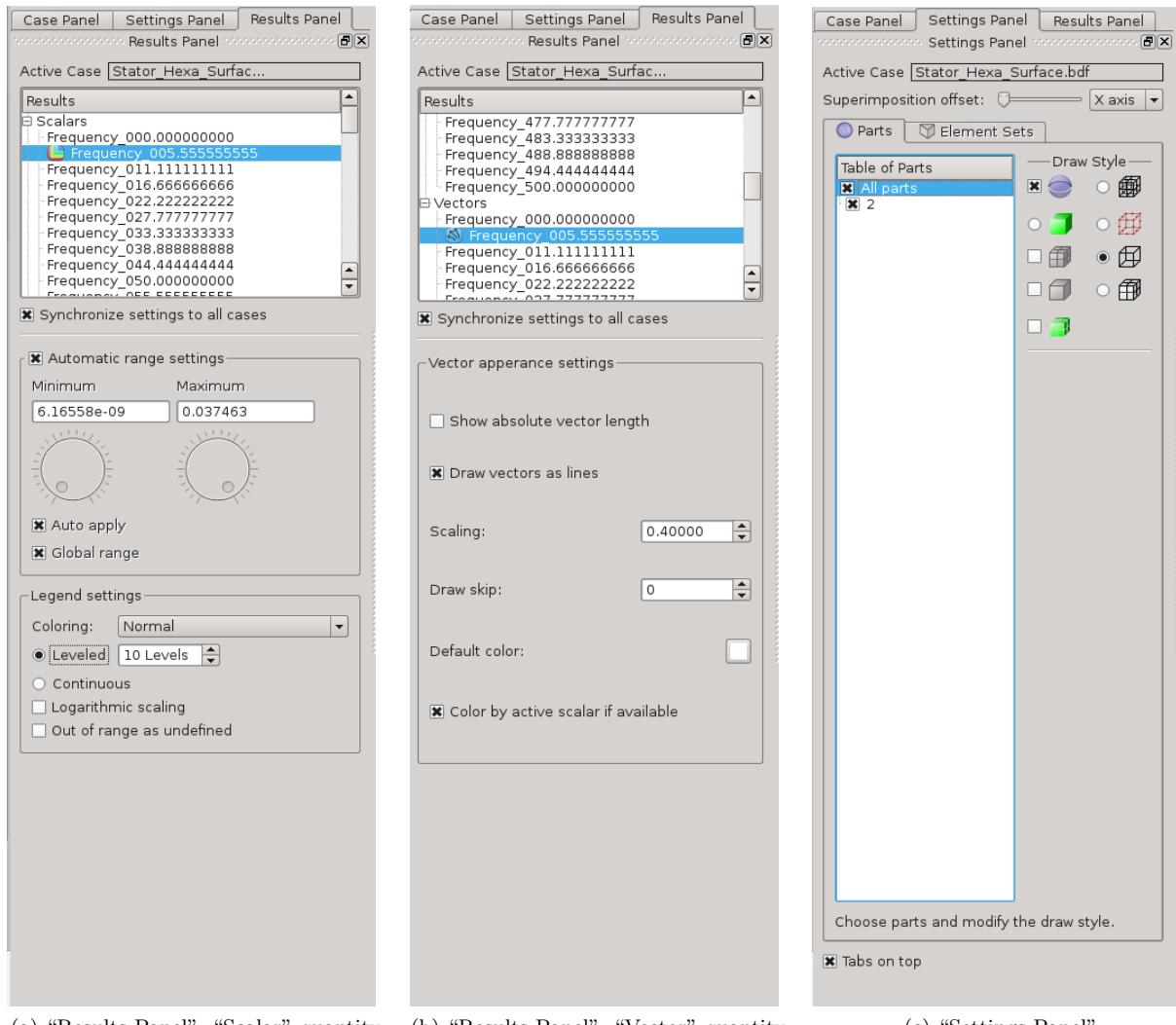


Figure 30: MpCCI Visualizer settings

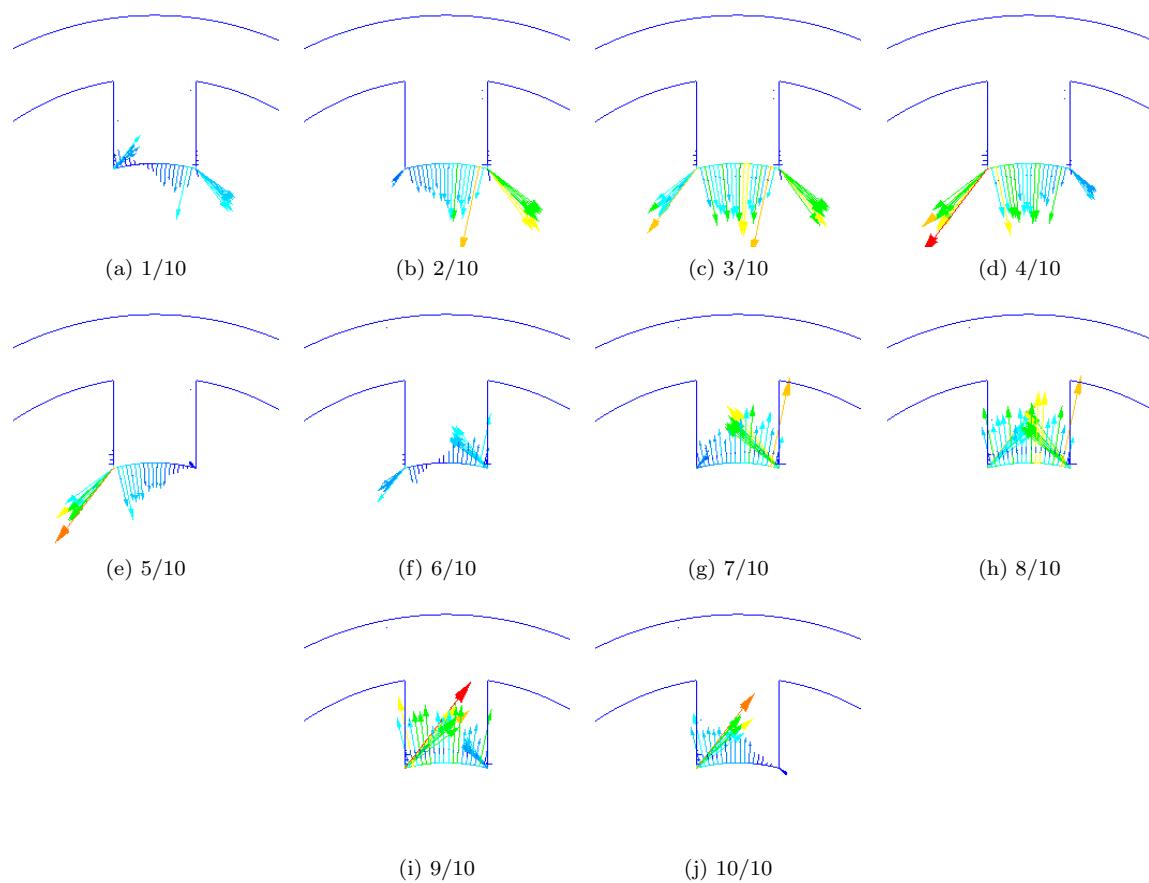


Figure 31: Corresponding transient force excitation at 5.5 Hz generated for 10 pseudo time steps

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:

```
$
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](#).

- ① 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 splitted 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 1st harmonic pressure excitation
2. map 2nd harmonic pressure excitation
3. map 3rd 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 $\omega = 17200 \text{ rpm} = 286.6 \text{ Hz}$.

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:
" $<MpCCIFSIMapper_home>/tutorial/Rotor37$ "

Before mapping, copy the files to your working directory.

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 32](#). $m \cdot \omega$ refers to as blade passing frequency.

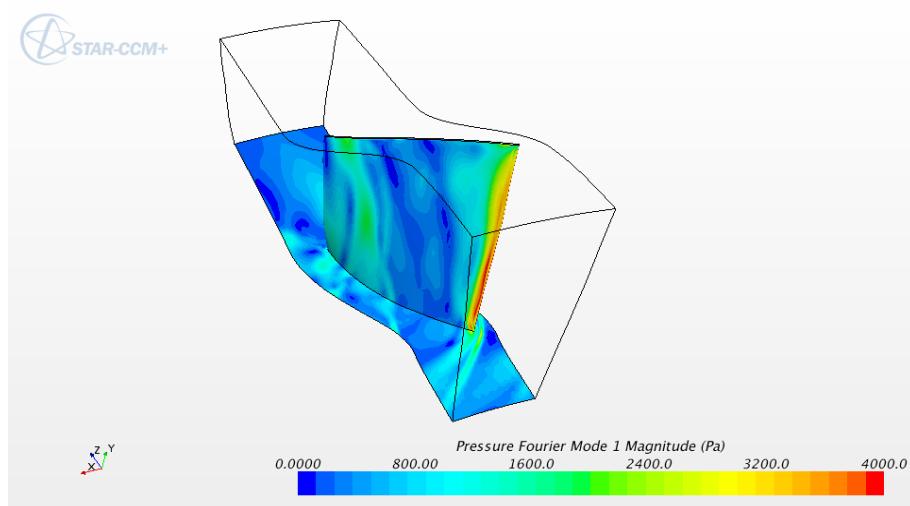


Figure 32: Magnitude of the first pressure harmonic

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 33](#).

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").

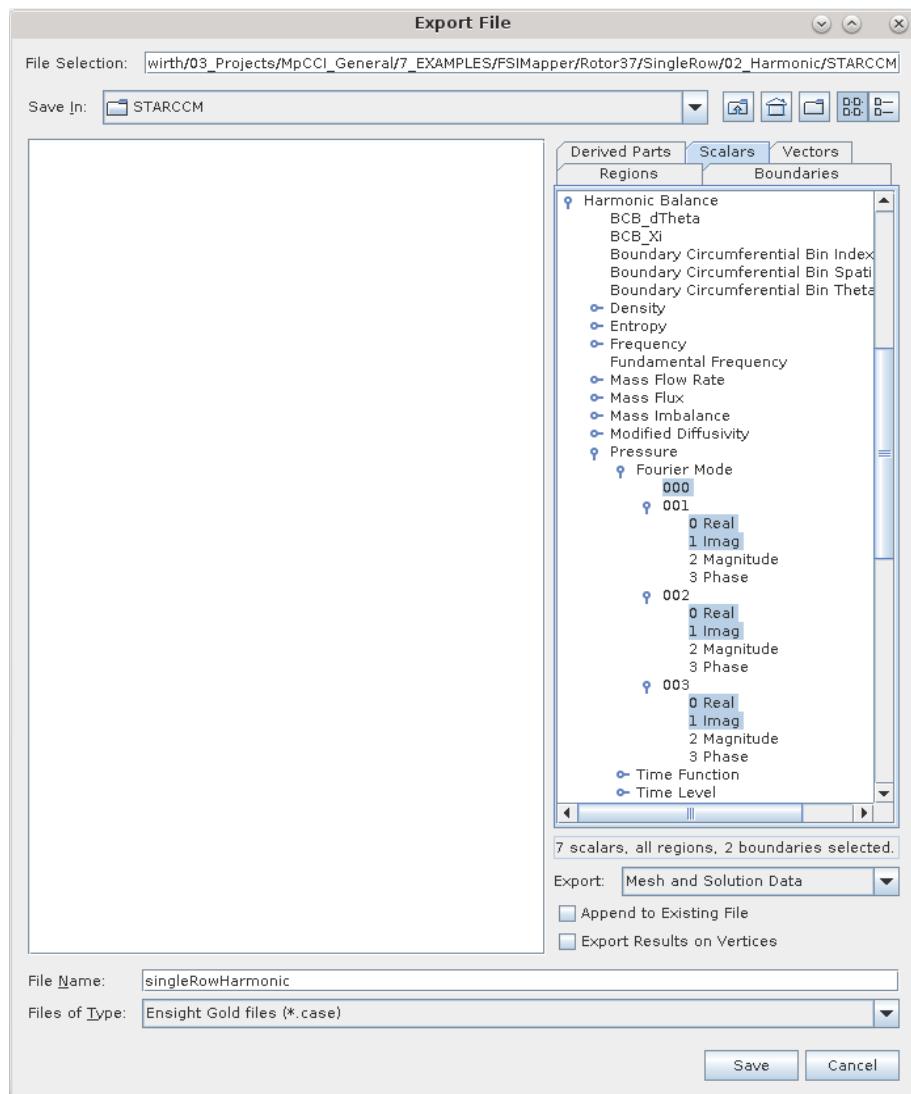


Figure 33: Export the harmonic results as EnSight Gold case file

8.2.1.3 Target Mesh File

The structural target mesh file "rotor37_fine.inp" comprises the whole rotor, as shown in [Figure 34](#). Second order tetra elements were used.

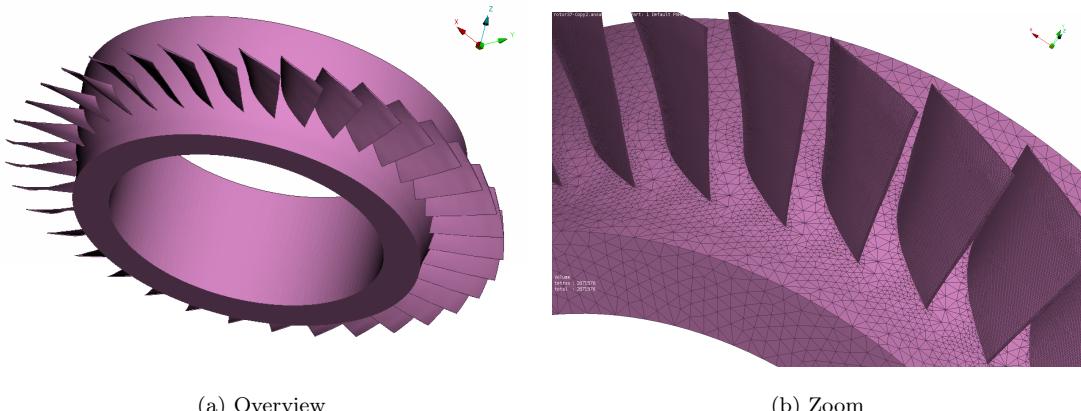


Figure 34: Target structural mesh of the full Rotor37

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 35](#)):

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 *mm-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 36](#)):

- 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.

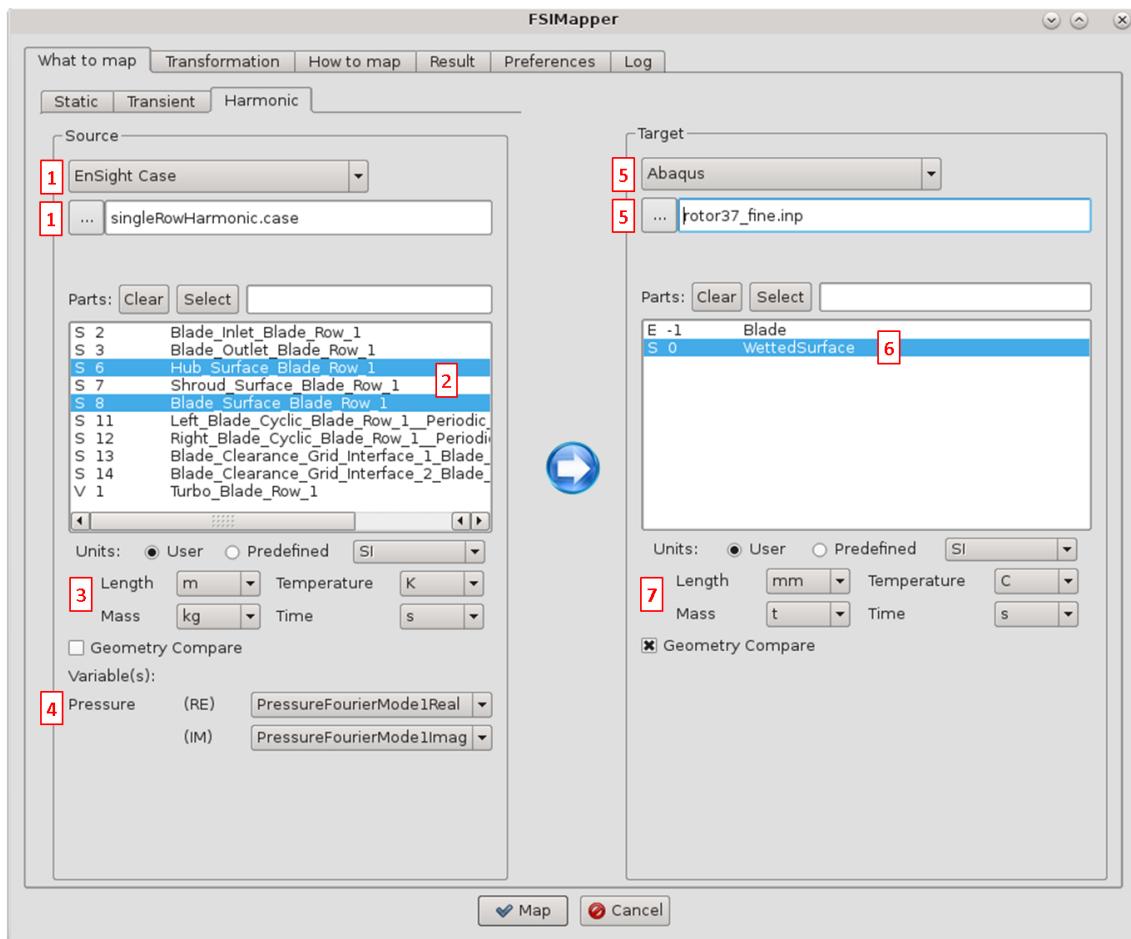


Figure 35: The “What to map” panel

This can be seen by the formula given in a note in [▷ 4.3 The “Transformation” Panel ◁](#):

$$ND = k \cdot m - b \cdot n = 1 \cdot 48 - b \cdot 36 = 12 \quad \text{with } b = 1$$

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.

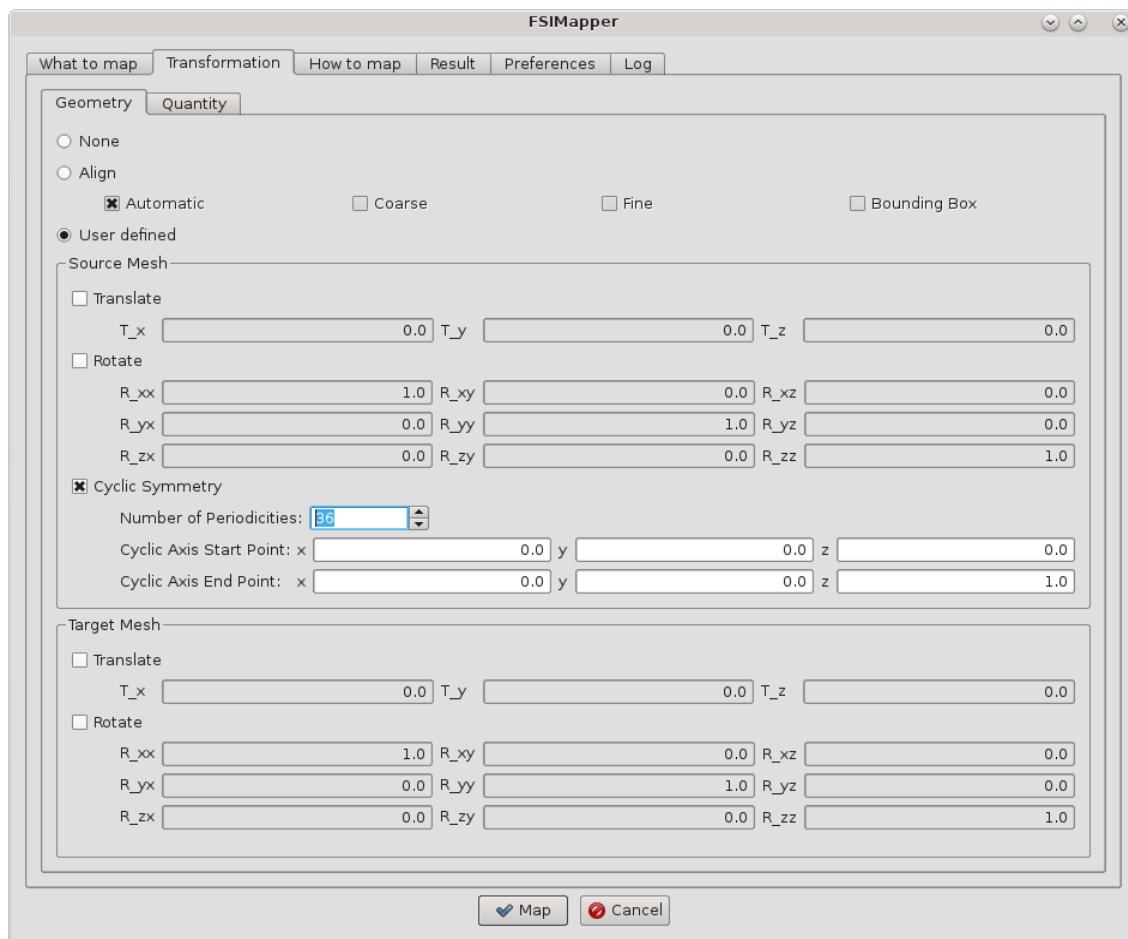


Figure 36: The “Transformation/Geometry” subpanel

When the mapping finishes, the MpCCI Visualizer shows the mapping results, cf. Figure 37.

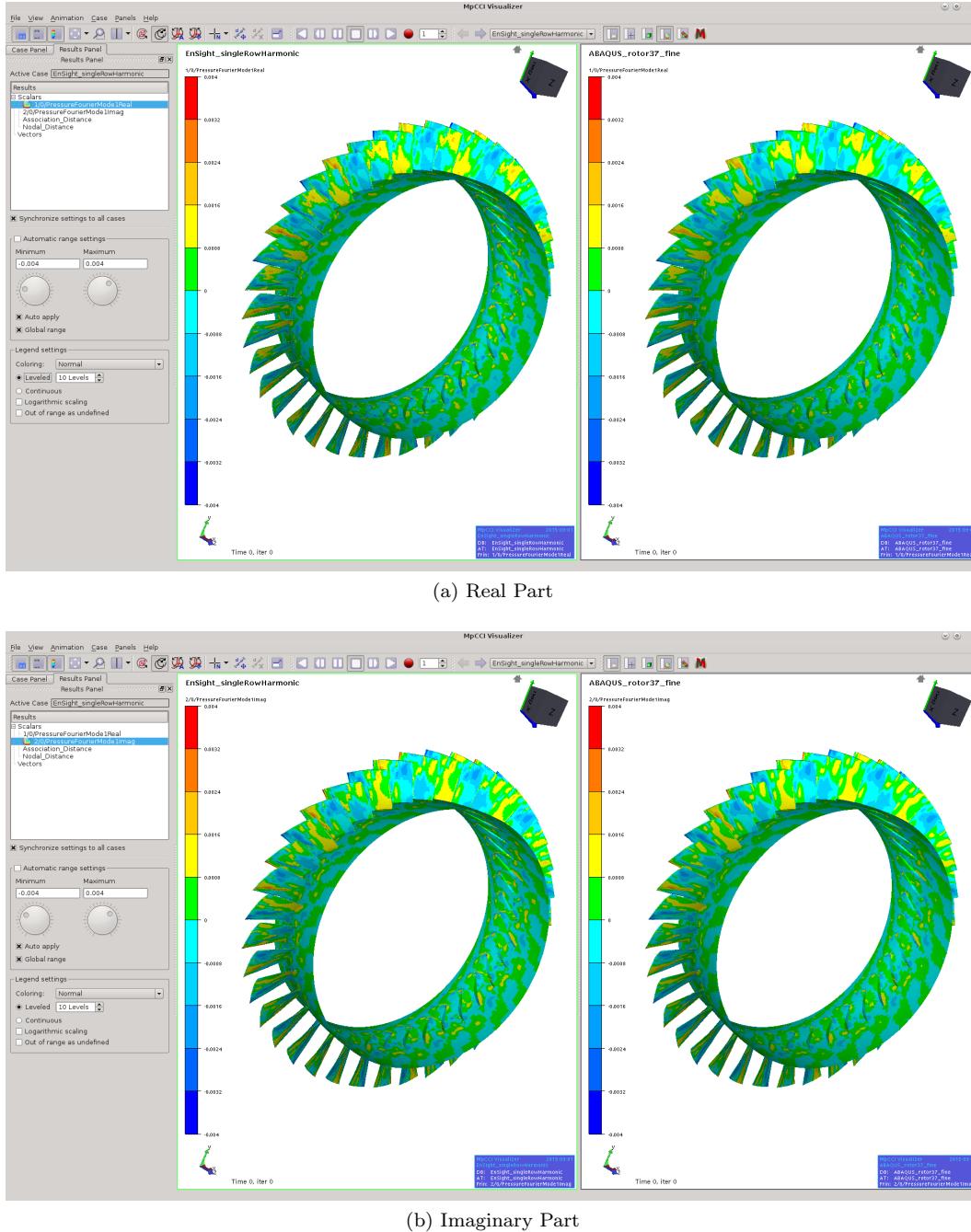


Figure 37: Harmonic mapping result shown in MpCCI Visualizer

The periodic mesh and data were transformed by the given information in order to create the full source model and the corresponding data. The first harmonic complex pressure excitation has been revolved such that the data are spatially continuous over the periodic boundaries, see Figure 38. This is only the case for the correctly selected nodal diameter and excitation mode.

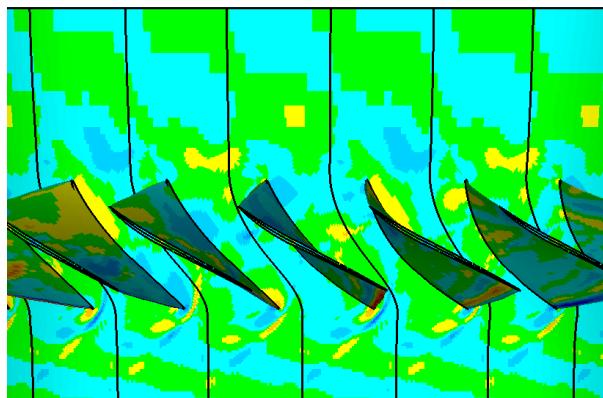


Figure 38: Cyclic Symmetry transformation creates continuous data over the periodic boundaries

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, -0.00009977
473, P1, -0.00007182
10650, P1, -0.00031675
10651, P1, -0.00037232
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.<extension>".

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 39](#). 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.

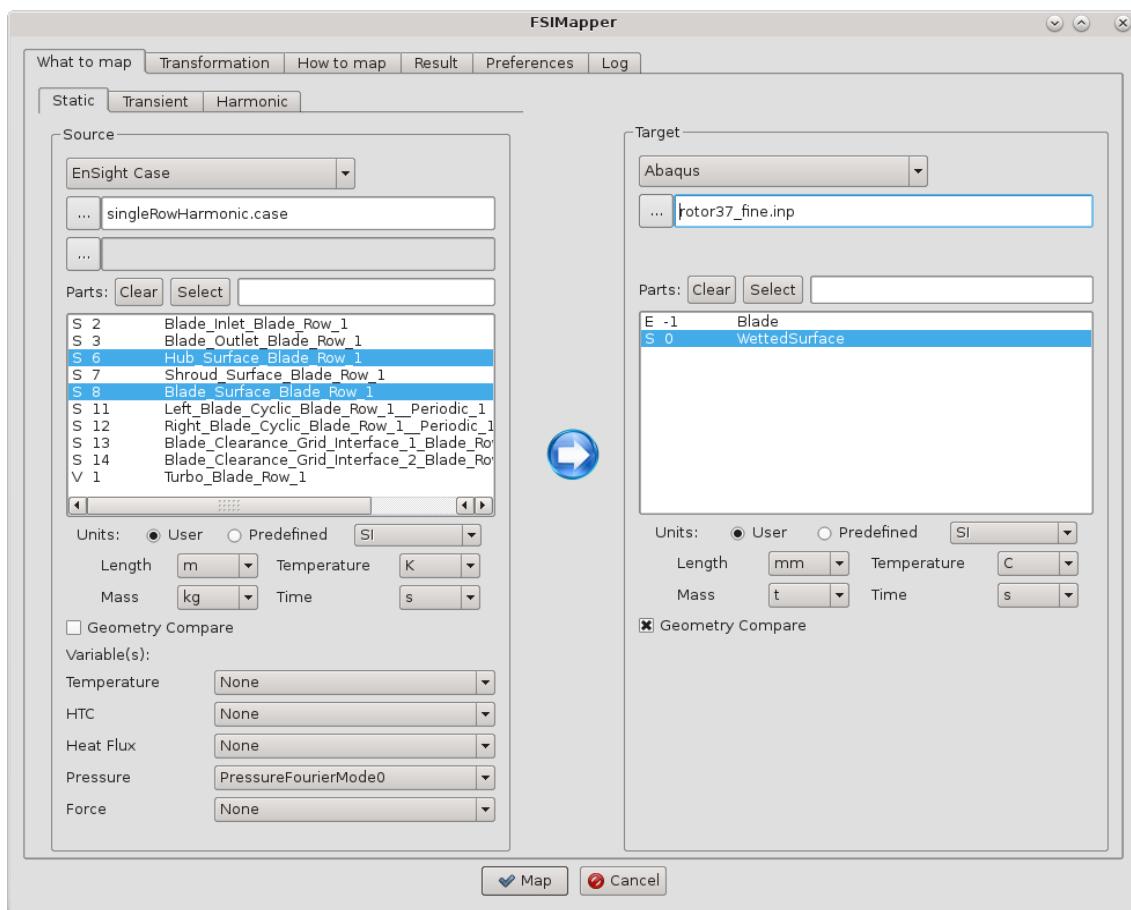


Figure 39: The “What to map/Static” panel

The mapped static pressure is shown in [Figure 40](#).

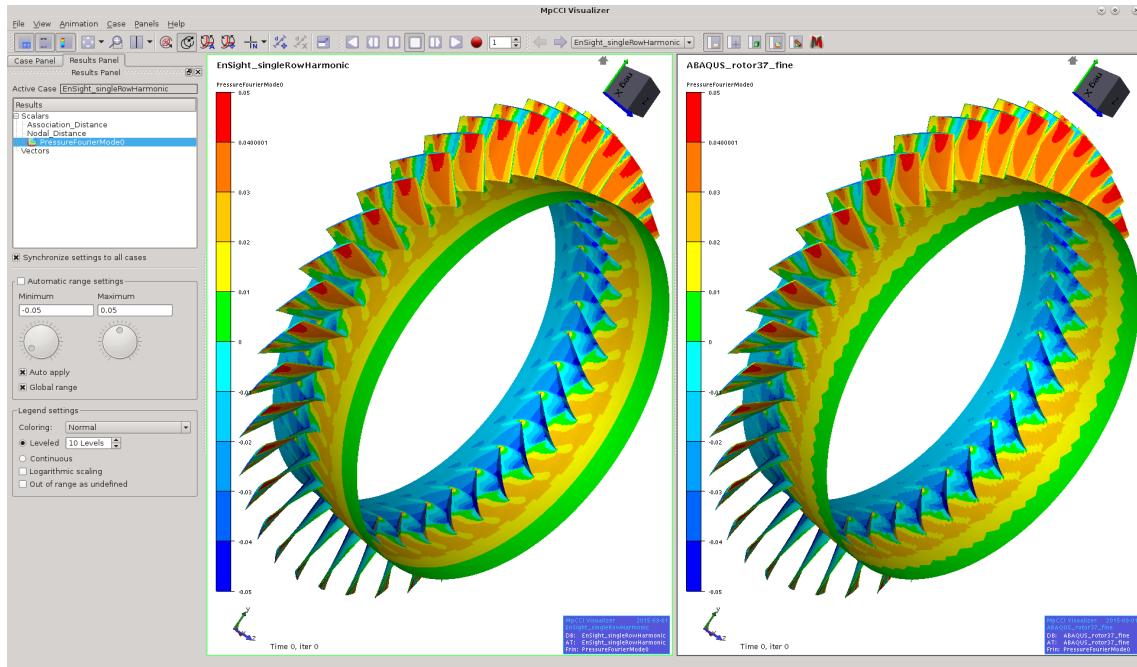


Figure 40: Mapping result of the time-averaged pressure

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:

```
**
*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
**
*STEADY 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
**
*STEADY STATE 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.

- ① For the sake of completeness use in the first step also the centrifugal and the coriolis forces (by the keyword ***DLOAD**).
- ① 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**.

- ① 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. 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:
" <MpCCIFSIMapper_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 41. $m \cdot \omega$ 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 42. 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`.

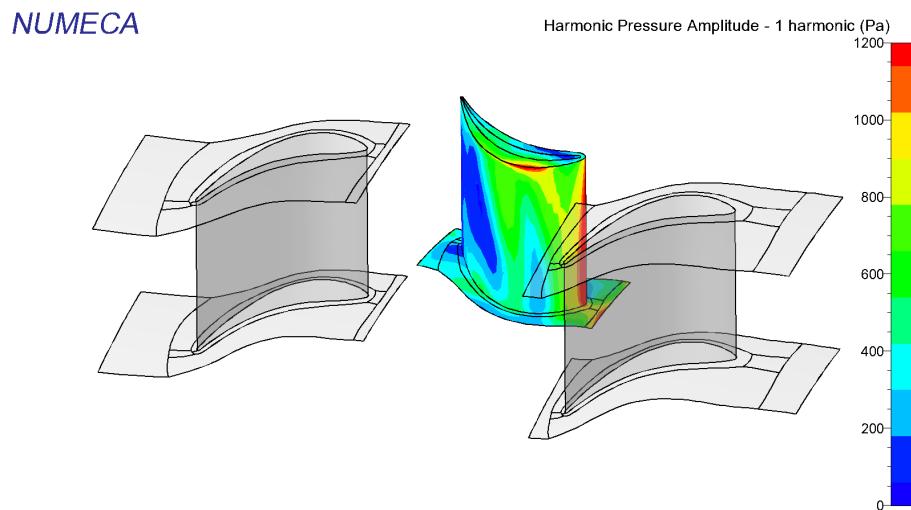


Figure 41: Magnitude of the first pressure harmonic at 2100 Hz

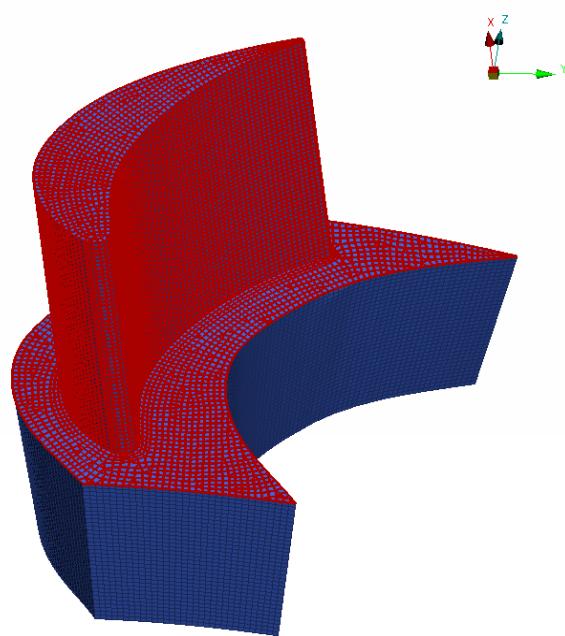


Figure 42: Target structural mesh (blue) with wetted surface definition (red) of one periodic section of the axial turbine “Aachen”

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 ($k = 2$: forward nodal diameter 10, $k = 3$: 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 43](#)):

1. Select “FINE/Turbo” in the source file type drop-down menu and choose the file “Tutorial2_NLH_Rank1_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 *mm-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 44](#)):

- 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 45](#).

Pressing the  button starts the mapping process.

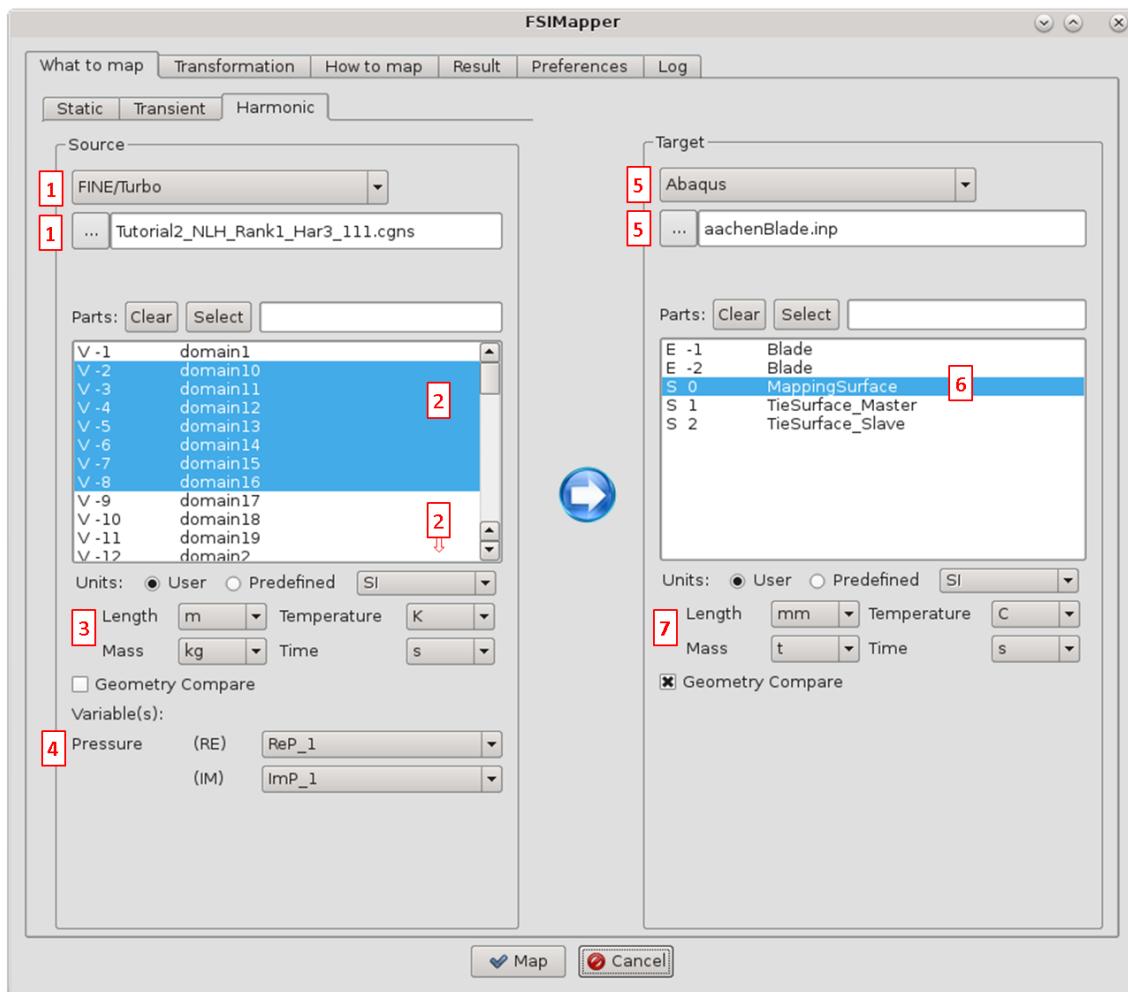


Figure 43: The “What to map” panel

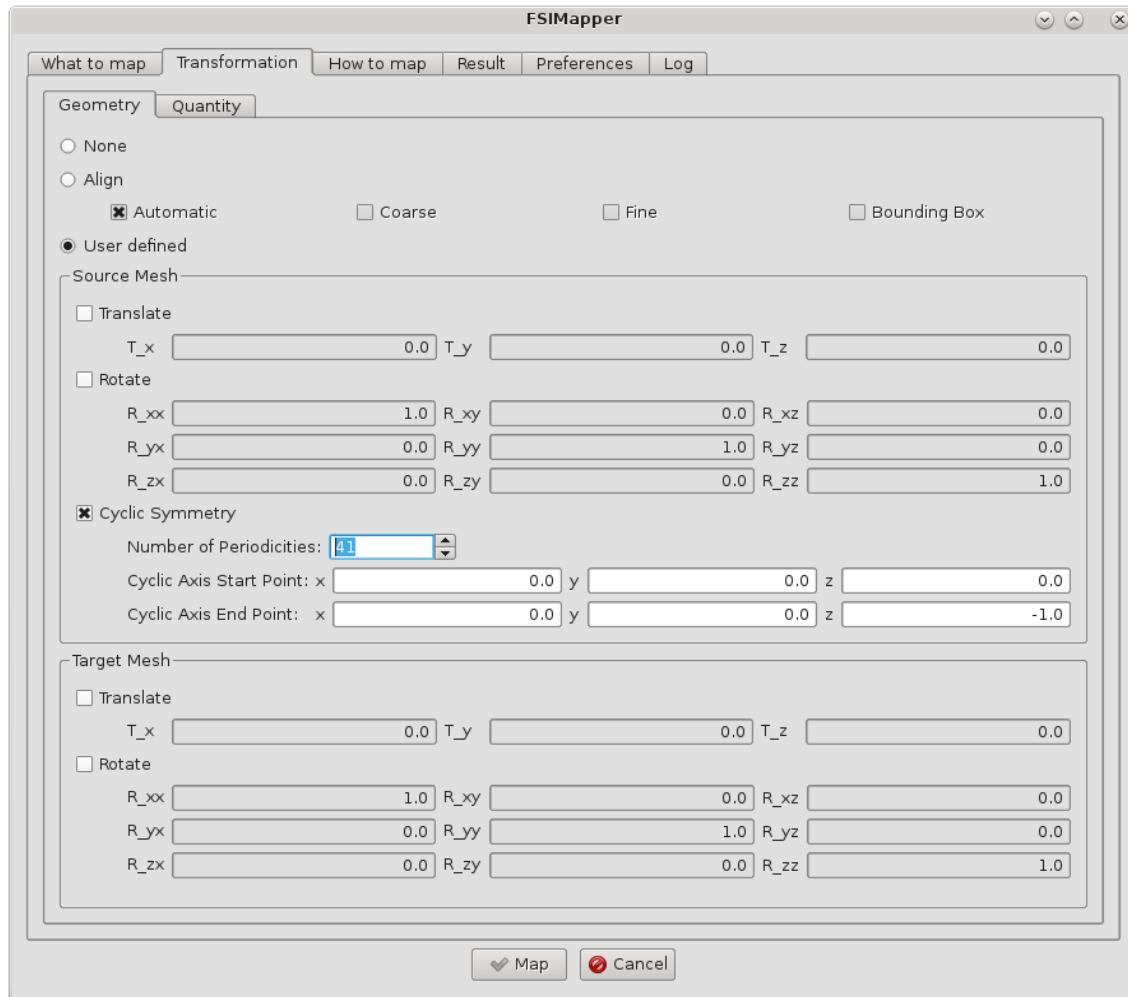


Figure 44: The “Transformation/Geometry” panel

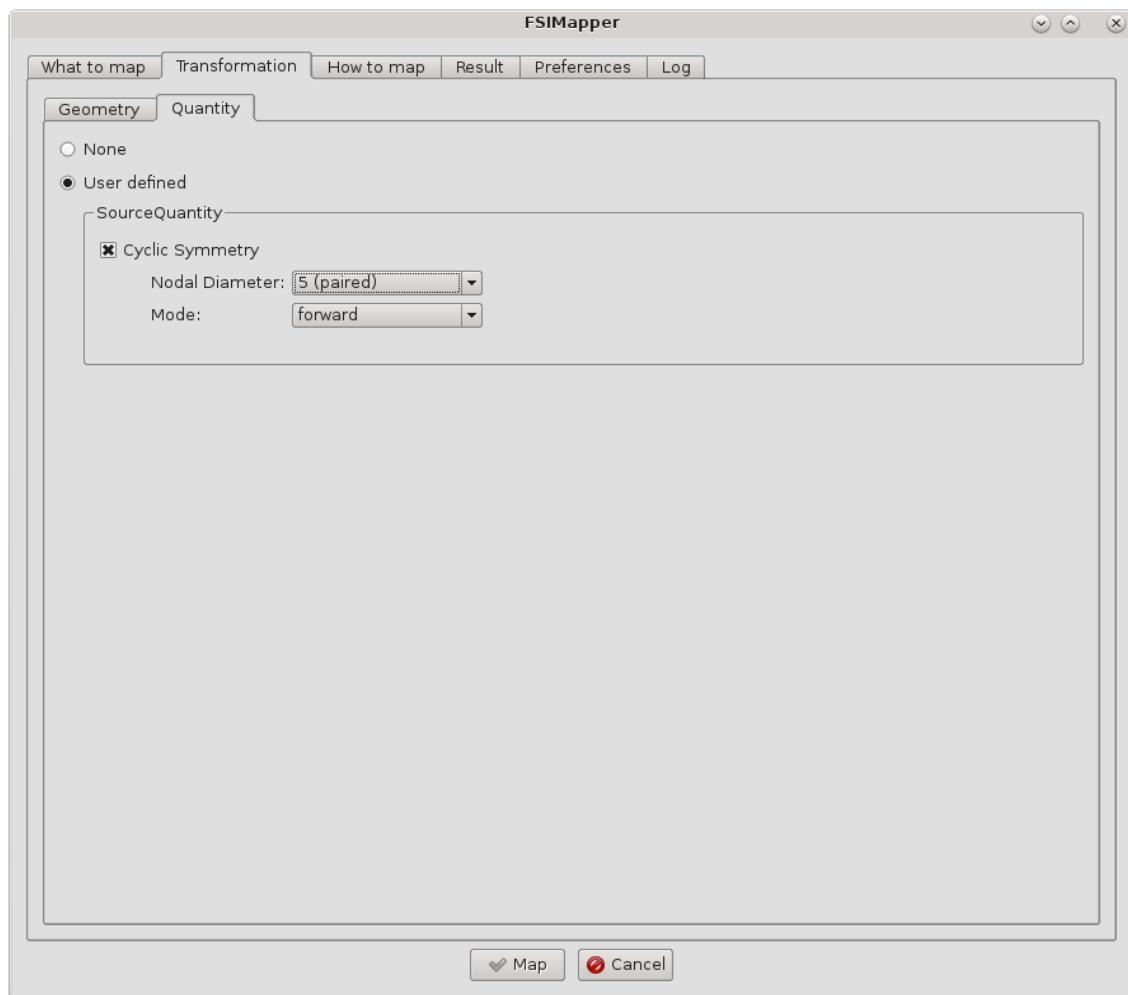


Figure 45: The “Transformation/Quantity” panel

When the mapping finishes, the MpCCI Visualizer shows the mapping results, cf. [Figure 46](#).

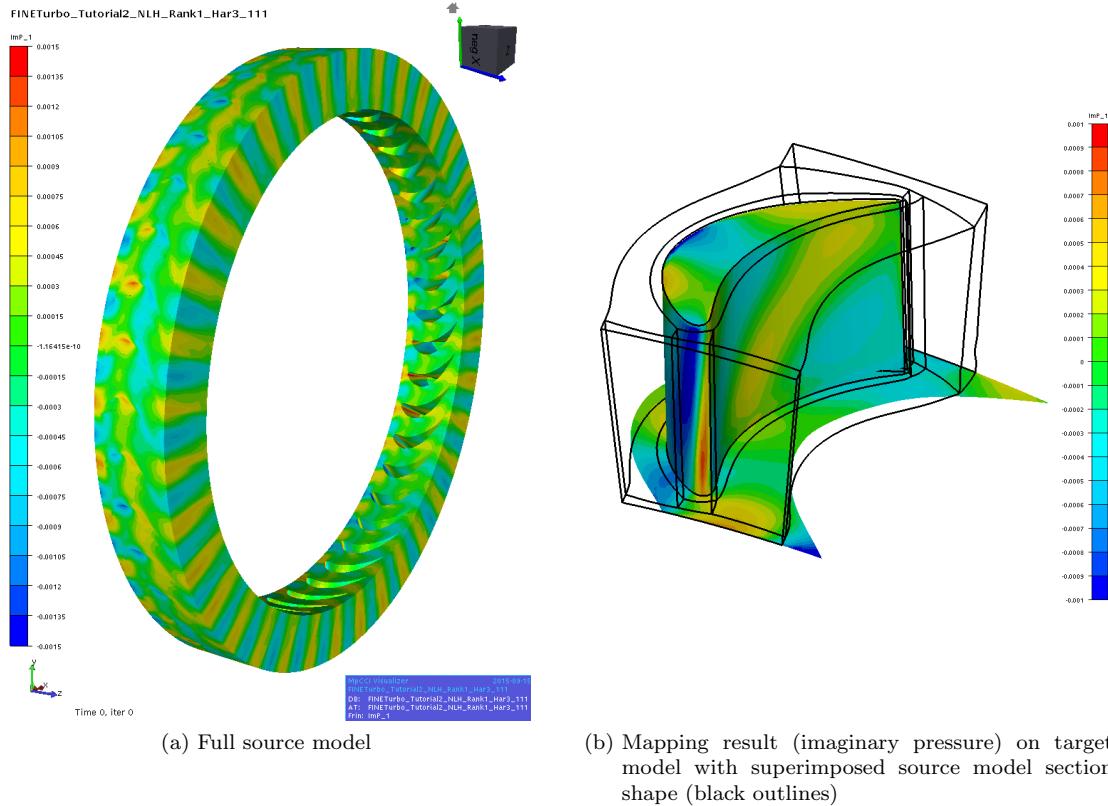


Figure 46: Harmonic mapping result shown in MpCCI Visualizer

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 46a). The first harmonic complex pressure excitation has been revolved such that the data are spatially 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 46b. 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:

```
**
** Exported real part of surface pressure
** *CLOAD, 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
...
```

```
**
** 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
**
```

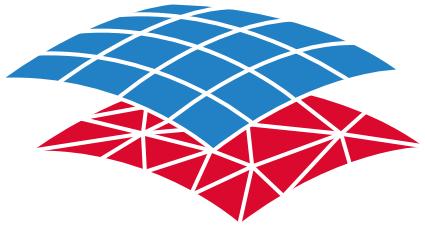
```
*END STEP  
**
```

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.

- ➊ For the sake of completeness use also the mean pressure (by a “Static” mapping), the centrifugal and the coriolis forces (by the keyword ***DLOAD**).
- ➋ The damping in the system has fundamental influence to the magnitude of blade vibration, so it is recommended to define it.
- ➌ 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**.



MpCCI
CouplingEnvironment

Part XI

Appendix

Version 4.5.0

MpCCI 4.5.0-1 Documentation
Part XI Appendix
PDF version
March 30, 2017

MpCCI is a registered trademark of Fraunhofer SCAI
www.mpcci.de



Fraunhofer Institute for Algorithms and Scientific Computing SCAI
Schloss Birlinghoven, 53754 Sankt Augustin, Germany

Abaqus and SIMULIA are trademarks or registered trademarks of Dassault Systèmes
ANSYS, FLUENT and ANSYS Icepak are trademarks or registered trademarks of Ansys, Inc.
Elmer is an open source software developed by CSC
FINE/Open and FINE/Turbo are trademarks of NUMECA International
Flowmaster is a registered trademark of Flowmaster Group NV
JMAG is a registered trademark of The JSOL Corporation
MATLAB is a registered trademark of The MathWorks, Inc.
MSC Adams, MSC Marc, MD NASTRAN and MSC NASTRAN are trademarks or registered trademarks of
MSC.Software Corporation
OpenFOAM is a registered trademark of OpenCFD Ltd.
PERMAS is a registered trademark of Intes GmbH
RadTherm, TAITherm is a registered trademark of ThermoAnalytics Inc.
SIMPACK is a registered trademark of SIMPACK AG.
STAR-CCM+ and STAR-CD are registered trademarks of CD adapco Group

ActivePerl has a Community License Copyright of Active State Corp.
FlexNet Publisher is a registered trademark of Flexera Software.
Java is a registered trademark of Oracle and/or its affiliates.
Linux is a registered trademark of Linus Torvalds
Mac OS X is a registered trademark of Apple Inc.
OpenSSH has a copyright by Tatu Ylonen, Espoo, Finland
Perl has a copyright by Larry Wall and others
UNIX is a registered trademark of The Open Group
Windows, Windows XP, Windows Vista and Windows 7 are registered trademarks of Microsoft Corp.

XI Appendix – Contents

Quantity Reference	4
Literature	39
Glossary	40
Keyword Index	43

Quantity Reference

Quantity	ANSYS	Abaqus	MSC Adams	FINE/Open	FINE/Turbo	FLUENT	Flowmaster	ANSYS Icepak	JMAG	MATLAB	MSC Marc	MSC NASTRAN	OpenFOAM	RadTherm	SIMPACK	STAR-CCM+	STAR-CD
AbsPressure	s/r	r		s	s	s/r			s/r	r		s			s	s/r	
Acceleration		s/r	s/r						s/r								
AcstPressure						s/r										s/r	
AngularAcceleration		s/r	s/r						s/r								
AngularCoordinate	s/r	s/r	s/r						s/r					s			
AngularVelocity	s/r	s/r	s/r			r			s/r					s	r		
BodyForce	s/r	s/r				s/r			s/r							s/r	
CGAngle	s/r					s/r										s/r	
CGOmega	s/r					s/r										s/r	
CGPosition	s/r					s/r										s/r	
CGVelocity	s/r					s/r										s/r	
ChargeDensity	s/r					s/r										s/r	
Current1	s/r					s/r										s/r	
Current2	s/r					s/r										s/r	
Current3	s/r					s/r										s/r	
Current4	s/r					s/r										s/r	
CurrentDensity	s/r					s/r		s	s/r							s/r	
DeltaTime	s/r	s/r	s/r	s	s	s/r	s/r	s/r	s	s/r	s/r	s/r	s/r	s	r	s/r	s/r
Density				s	s	s/r											s/r
DynPressure						s											s
ElectrCond1	s/r					s/r			r	s/r							s/r
ElectrCond3	s/r					s/r			r	s/r							s/r
ElectrCondX	s/r					s/r				s/r							s/r
ElectrCondY	s/r					s/r				s/r							s/r
ElectrCondZ	s/r					s/r				s/r							s/r
ElectricField	s/r					s/r											
ElectricFlux	s/r					s/r											s/r
ElectricPot	s/r									s/r							
ElectrRes1	s/r					s/r				s/r							s/r
ElectrRes3	s/r					s/r				s/r							s/r
ElectrResX	s/r					s/r				s/r							s/r
ElectrResY	s/r					s/r				s/r							s/r
ElectrResZ	s/r					s/r				s/r							s/r
Enthalpy	s/r					s/r		s/r									s/r
FilmTemp	r	r				s/r		s/r		s/r	r		s	r		s	s/r
Force	s/r	s/r	s/r			s			s/r						s/r	s	
gs00																	
gs01																	
gs02																	
gs03																	
gs04																	
gs05																	
gs06																	

Quantity	ANSYS	Abaqus	MSC Adams	FINE/Open	FINE/Turbo	FLUENT	Flowmaster	ANSYS Icepak	JMAG	MATLAB	MSC Marc	MSC NASTRAN	OpenFOAM	RadTherm	SIMPACK	STAR-CCM+	STAR-CD
gs07																	
gv00																	
gv01																	
gv02																	
gv03																	
gv04																	
gv05																	
gv06																	
gv07																	
HeatFlux	s/r					s/r		s/r									s/r
HeatRate		r				s						s				s	
HeatSource	s/r					s/r		s/r									s/r
IntFlag	s/r					s/r		s/r									s/r
IterationNo	s/r			s		s/r		s/r						s		s/r	s/r
JouleHeat	s/r					s/r		s/r	s	s/r							s/r
JouleHeatLin						s/r		s/r									
LorentzForce	s/r					s/r			s	s/r							s/r
MagneticField	s/r					s/r											
MagneticFlux	s/r					s/r				s/r							s/r
MassFlowRate						s/r	s/r			s/r			s/r			s/r	s/r
MassFlowVect																	
MassFluxRate						s/r	s/r			s/r			s/r			s/r	s/r
MassFluxVect																	
NPosition	s	s		r	r	s/r			s/r	s/r	s	s	r			r	r
OverPressure	s/r	r		s	s	s/r					r		s			s	s/r
PhysicalTime	s/r			s	s	s/r	s/r	s/r					s		s/r	s/r	
PointPosition	s/r	s/r	s/r							s/r					s		
PorePressure		s/r				s/r											s/r
PorousFlow		s/r															
RealFlag	s/r		s/r			s/r		s/r		s/r					r		s/r
RefPressure	s/r			s	s	s/r									s/r	s/r	
RelWallForce	r	r		s	s	s/r					r	r	s		s/r	s/r	
Residual	s/r					s/r		s/r					s			s/r	
SpecificHeat	s/r					s/r		s/r		s/r							s/r
StaticPressure																	s/r
Temperature	s/r	s/r				s/r	s/r	s/r	r	s/r		s/r			s/r		s/r
ThermCond1	s/r					s/r		s/r									s/r
ThermCond3	s/r					s/r		s/r									s/r
ThermCondX	s/r					s/r		s/r									s/r
ThermCondY	s/r					s/r		s/r									s/r
ThermCondZ	s/r					s/r		s/r									s/r
TimeStepNo	s/r			s		s/r		s/r						s		s/r	s/r
Torque	s/r	s/r	s/r							s/r				s/r		s	
TotalPressure						s/r	s/r			s/r			s/r		s/r	s/r	

Quantity	ANSYS	Abaqus	MSC Adams	FINE/Open	FINE/Turbo	FLUENT	Flowmaster	ANSYS Icepak	JMAG	MATLAB	MSC Marc	MSC NASTRAN	OpenFOAM	RadTherm	SIMPACK	STAR-CCM+	STAR-CD
TotalTemp																	
Velocity	r	s/r	s/r	s		s/r				s/r		s	s/r		s	s/r	s/r
VelocityMagnitude						s/r	s/r			s/r							
Voltage1	s/r					s/r											s/r
Voltage2	s/r					s/r											s/r
Voltage3	s/r					s/r											s/r
Voltage4	s/r					s/r											s/r
VolumeFlow																	
VolumeFlowRate																	
WallForce	r	s/r			s	s/r				s/r	r	r	s			s/r	s/r
WallHeatFlux	s/r	r		s	s	s/r		s/r			r		s	r		s/r	s/r
WallHTCoeff	r	r				s/r		s/r		s/r	r		s	r		s	s/r
WallTemp	s/r	s/r		s/r	s/r	s/r		s/r		s/r	s		s/r	s		s/r	s/r
YI00																	
YI01																	
YI02																	
YI03																	
YI04																	
YI05																	
YI06																	
YI07																	
YI08																	
YI09																	

Table 10: 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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Line, Face, Volume	Element	Direct	ETAB Direct
Abaqus	Face	Code		Buffer
FINE/Open	Face	Node, Element	Direct	
FINE/Turbo	Face	Code	Direct	
FLUENT	Face, Volume	Code	Dir	UDM
MATLAB	Face	Code, Element	Direct	Direct
MSC Marc	Line, Face, Volume	Element		Direct
OpenFOAM	Line, Face, Volume	Code	Dir	
STAR-CCM+	Face	Code	Direct	
STAR-CD	Face, Volume	Code	Direct	SCALAR

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

Code	Coupling Dimensions	Location	Send option	Receive option
Abaqus	Point	Code	Direct	Buffer
MSC Adams	Point	Node	Direct	Usermemory
MATLAB	Point	Code	Direct	Direct

AcstPressure 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

Code	Coupling Dimensions	Location	Send option	Receive option
FLUENT	Volume	Code	Dir	UDM
STAR-CD	Volume	Code	Direct	SCALAR

AngularAcceleration 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

Code	Coupling Dimensions	Location	Send option	Receive option
Abaqus	Point	Code	Direct	Buffer
MSC Adams	Point	Node	Direct	Usermemory
MATLAB	Point	Code	Direct	Direct

AngularCoordinate quaternion (i,j,k,w) [-]

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Point	Node	Direct	Direct
Abaqus	Point	Code	Direct	Buffer
MSC Adams	Point	Node	Direct	Usermemory
MATLAB	Point	Code	Direct	Direct
SIMPACK	Point	Node	Direct	Direct

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Point	Node	Direct	Direct
Abaqus	Point	Code	Direct	Buffer
MSC Adams	Point	Node	Direct	Usermemory
FLUENT	Point	Code		Buf
MATLAB	Point	Code	Direct	Direct
SIMPACK	Point	Node	Direct	
STAR-CCM+	Point	Code		Direct

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Volume	Node, Element	Direct ETAB	Direct
Abaqus	Volume	Code	Direct	Buffer
FLUENT	Volume	Code	UDM	UDM
MATLAB	Volume	Node, Element	Direct	Direct
STAR-CD	Volume	Code	SCALAR	SCALAR

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Global	global	APDL	APDL
FLUENT	Global	global	Dir	Dir
STAR-CD	Global	global	Direct	Direct

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Global	global	APDL	APDL
FLUENT	Global	global	Dir	Dir
STAR-CD	Global	global	Direct	Direct

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Global	global	APDL	APDL
FLUENT	Global	global	Dir	Dir
STAR-CD	Global	global	Direct	Direct

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Global	global	APDL	APDL
FLUENT	Global	global	Dir	Dir
STAR-CD	Global	global	Direct	Direct

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Line, Volume	Element	Direct	Direct
FLUENT	Volume	Code	UDM	UDM
STAR-CD	Volume	Code	SCALAR	SCALAR

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Global	global	APDL	APDL
FLUENT	Global	global	Dir	Dir
STAR-CD	Global	global	Direct	Direct

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Global	global	APDL	APDL
FLUENT	Global	global	Dir	Dir
STAR-CD	Global	global	Direct	Direct

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Global	global	APDL	APDL
FLUENT	Global	global	Dir	Dir
STAR-CD	Global	global	Direct	Direct

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Global	global	APDL	APDL
FLUENT	Global	global	Dir	Dir
STAR-CD	Global	global	Direct	Direct

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Line, Face, Volume	Node, Element	Direct	Direct
FLUENT	Face, Volume	Code	UDM UDS	UDM
JMAG	Volume	Element	Direct	
MATLAB	Volume	Node, Element	Direct	Direct
STAR-CD	Face, Volume	Code	SCALAR	SCALAR

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Global	global	APDL	APDL
Abaqus	Global	global	Direct	Direct
MSC Adams	Global	global	Direct	Usermemory
FINE/Open	Global	global	Direct	
FINE/Turbo	Global	global	Direct	
FLUENT	Global	global	Dir	Dir
Flowmaster	Global	global	Direct	Direct
ANSYS Icepak	Global	global	Dir	Dir
JMAG	Global	global	Direct	
MATLAB	Global	global	Direct	Direct
MSC Marc	Global	global	Direct	Direct
MSC NASTRAN	Global	global	Direct	Direct
OpenFOAM	Global	global	Dir	Dir
RadTherm	Global	global	Direct	
SIMPACK	Global	global		Direct
STAR-CCM+	Global	global	Direct	Direct
STAR-CD	Global	global	Direct	Direct

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

Code	Coupling Dimensions	Location	Send option	Receive option
FINE/Open	Volume	Node, Element	Direct	
FINE/Turbo	Volume	Code	Direct	
FLUENT	Volume	Code	Dir	UDM
STAR-CD	Volume	Code	Direct	SCALAR

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

Code	Coupling Dimensions	Location	Send option	Receive option
FLUENT	Face	Code	Dir	
STAR-CD	Face	Code	Direct	

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Line, Face, Volume	Element	Direct	Direct
FLUENT	Face, Volume	Code	UDM	UDM
JMAG	Volume	Node		Direct
MATLAB	Volume	Node, Element	Direct	Direct
STAR-CD	Face, Volume	Code	SCALAR	SCALAR

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Line, Face, Volume	Element	Direct	Direct
FLUENT	Face, Volume	Code	UDM	UDM
JMAG	Volume	Node		Direct
MATLAB	Volume	Node, Element	Direct	Direct
STAR-CD	Face, Volume	Code	SCALAR	SCALAR

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Line, Face, Volume	Element	Direct	Direct
FLUENT	Volume	Code	UDM	UDM
MATLAB	Volume	Node, Element	Direct	Direct
STAR-CD	Volume	Code	SCALAR	SCALAR

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Line, Face, Volume	Element	Direct	Direct
FLUENT	Volume	Code	UDM	UDM
MATLAB	Volume	Node, Element	Direct	Direct
STAR-CD	Volume	Code	SCALAR	SCALAR

ElectrCondZ 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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Line, Face, Volume	Element	Direct	Direct
FLUENT	Volume	Code	UDM	UDM
MATLAB	Volume	Node, Element	Direct	Direct
STAR-CD	Volume	Code	SCALAR	SCALAR

ElectricField 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
 Coupling Dimensions: Line, Face, Volume

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Line, Volume	Element	Direct	Direct
FLUENT	Volume	Code	UDM UDS	UDM

ElectricFlux Electric flux vector [C/m²]

Code API symbol: MPCCI_QID_ELECTRICFLUX
 Default value: 0.0
 Dimension: Vector
 Physical meaning: Boundary condition: face normal gradient
 Interpolation type: flux density
 Coupling Dimensions: Line, Face, Volume

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Line, Volume	Element	Direct	Direct
FLUENT	Face, Volume	Code	UDM UDS	UDM
STAR-CD	Face, Volume	Code	SCALAR	SCALAR

ElectricPot Electric Potential [V]

Code API symbol: MPCCI_QID_ELECTRICPOT
 Default value: 0.0
 Dimension: Scalar
 Physical meaning: Boundary condition: value
 Interpolation type: field
 Coupling Dimensions: Line, Face, Volume

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Line, Face, Volume	Node	Direct	Direct
MATLAB	Volume	Node, Element	Direct	Direct

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Line, Face, Volume	Element	Direct	Direct
FLUENT	Face, Volume	Code	UDM	UDM
MATLAB	Volume	Node, Element	Direct	Direct
STAR-CD	Face, Volume	Code	SCALAR	SCALAR

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Line, Face, Volume	Element	Direct	Direct
FLUENT	Face, Volume	Code	UDM	UDM
MATLAB	Volume	Node, Element	Direct	Direct
STAR-CD	Face, Volume	Code	SCALAR	SCALAR

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Line, Face, Volume	Element	Direct	Direct
FLUENT	Volume	Code	UDM	UDM
MATLAB	Volume	Node, Element	Direct	Direct
STAR-CD	Volume	Code	SCALAR	SCALAR

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Line, Face, Volume	Element	Direct	Direct
FLUENT	Volume	Code	UDM	UDM
MATLAB	Volume	Node, Element	Direct	Direct
STAR-CD	Volume	Code	SCALAR	SCALAR

ElectrResZ Electric resistivity - z [ohm 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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Line, Face, Volume	Element	Direct	Direct
FLUENT	Volume	Code	UDM	UDM
MATLAB	Volume	Node, Element	Direct	Direct
STAR-CD	Volume	Code	SCALAR	SCALAR

Enthalpy Enthalpy density [W/m³]

Code API symbol: MPCCI_QID_ENTHALPY
 Default value: 0.0
 Dimension: Scalar
 Physical meaning: General
 Interpolation type: flux density
 Coupling Dimensions: Volume

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Volume	Node, Element	Direct	Direct
FLUENT	Volume	Code	Dir	UDM
ANSYS Icepak	Volume	Code	Dir	UDM
STAR-CD	Volume	Code	Direct	SCALAR

FilmTemp Film temperature [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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Face	Element		Direct
Abaqus	Face	Code		Buffer
FLUENT	Face	Code	Dir	UDM
ANSYS Icepak	Face	Code	Dir	UDM
MATLAB	Face	Code, Element	Direct	Direct
MSC Marc	Line, Face	Element		Direct
OpenFOAM	Line, Face	Code	Dir	
RadTherm	Face	Code		Direct
STAR-CCM+	Face	Code	Direct	
STAR-CD	Face	Code	Direct	SCALAR

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Point	Node	Direct	Direct
Abaqus	Point, Face	Code	Direct	Buffer
MSC Adams	Point	Node	Direct	Usermemory
FLUENT	Face	Code	Dir	
MATLAB	Point	Code	Direct	Direct
SIMPACK	Point	Node	Direct	Direct
STAR-CCM+	Face	Code	Direct	

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.

gv05 Global vector 05 [-]

Code API symbol: MPCCI_QID_UGV_05
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.

gv06 Global vector 06 [-]

Code API symbol: MPCCI_QID_UGV_06
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.

gv07 Global vector 07 [-]

Code API symbol: MPCCI_QID_UGV_07
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.

HeatFlux Heat flux density vector [W/m²]

Code API symbol: MPCCI_QID_HEATFLUX
 Default value: 0.0
 Dimension: Vector
 Physical meaning: Boundary condition: face normal gradient
 Interpolation type: flux density
 Coupling Dimensions: Volume

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Volume	Node, Element	Direct	ETAB
FLUENT	Volume	Code	UDM	UDM
ANSYS Icepak	Volume	Code	UDM	UDM
STAR-CD	Volume	Code	Direct	SCALAR

HeatRate Heat rate [W]

Code API symbol: MPCCI_QID_HEATRATE
 Default value: 0.0
 Dimension: Scalar
 Physical meaning: Boundary condition: value
 Interpolation type: field
 Coupling Dimensions: Point, Line, Face, Volume

Code	Coupling Dimensions	Location	Send option	Receive option
Abaqus	Face	Code		Buffer
FLUENT	Face	Code	Dir	
OpenFOAM	Face	Code	Dir	
STAR-CCM+	Face	Code	Direct	

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Volume	Node, Element	Direct	ETAB
FLUENT	Volume	Code	UDM	UDM
ANSYS Icepak	Volume	Code	UDM	UDM
STAR-CD	Volume	Code	SCALAR	SCALAR

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Global	global	APDL	APDL
FLUENT	Global	global	Dir	Dir
ANSYS Icepak	Global	global	Dir	Dir
STAR-CD	Global	global	Direct	Direct

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Global	global	APDL	APDL
FINE/Open	Global	global	Direct	
FLUENT	Global	global	Dir	Dir
ANSYS Icepak	Global	global	Dir	Dir
RadTherm	Global	global	Direct	
STAR-CCM+	Global	global	Direct	Direct
STAR-CD	Global	global	Direct	Direct

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Volume	Node, Element	Direct	ETAB
FLUENT	Volume	Code	UDM	UDM
ANSYS Icepak	Volume	Code	UDM	UDM
JMAG	Volume	Element	Direct	
MATLAB	Volume	Node, Element	Direct	Direct
STAR-CD	Volume	Code	SCALAR	SCALAR

JouleHeatLin Joule heat linearization [W/m³ K]

Code API symbol: MPCCI_QID_JOULEHEATLIN
 Default value: 0.0
 Dimension: Scalar
 Physical meaning: Energy source
 Interpolation type: flux density
 Coupling Dimensions: Volume

Code	Coupling Dimensions	Location	Send option	Receive option
FLUENT	Volume	Code	UDM	UDM
ANSYS Icepak	Volume	Code	UDM	UDM

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Volume	Node, Element	Direct ETAB	Direct
FLUENT	Volume	Code	UDM	UDM
JMAG	Volume	Element	Direct	
MATLAB	Volume	Node, Element	Direct	Direct
STAR-CD	Volume	Code	SCALAR	SCALAR

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Line, Volume	Element	Direct	Direct
FLUENT	Volume	Code	UDM UDS	UDM

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Line, Face, Volume	Node, Element	Direct	Direct
FLUENT	Face, Volume	Code	UDM UDS	UDM
MATLAB	Volume	Node, Element	Direct	Direct
STAR-CD	Face, Volume	Code	SCALAR	SCALAR

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

Code	Coupling Dimensions	Location	Send option	Receive option
FLUENT	Face	Code	Dir	UDM
Flowmaster	Point	Code	Direct	Direct
MATLAB	Point	Code	Direct	Direct
OpenFOAM	Face	Code	Dir	Dir
STAR-CCM+	Face	Code	Direct	Direct
STAR-CD	Face	Code	Direct	SCALAR

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

Code	Coupling Dimensions	Location	Send option	Receive option
FLUENT	Face	Code	Dir	UDM
Flowmaster	Point	Code	Direct	Direct
MATLAB	Point	Code	Direct	Direct
OpenFOAM	Face	Code	Dir	Dir
STAR-CCM+	Face	Code	Direct	Direct
STAR-CD	Face	Code	Direct	SCALAR

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Line, Face, Volume	Node	Direct	
Abaqus	Face, Volume	Code	Direct	
FINE/Open	Face	Node		Direct
FINE/Turbo	Face	Code		Direct
FLUENT	Face, Volume	Code	Dir	Buf
JMAG	Face, Volume	Node	Direct	Direct
MATLAB	Face, Volume	Code	Direct	Direct
MSC Marc	Line, Face, Volume	Node	Direct	
MSC NASTRAN	Line, Face	Code	Direct	
OpenFOAM	Line, Face, Volume	Code		Dir
STAR-CCM+	Face	Code		Direct
STAR-CD	Face, Volume	Code		Buffered

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Line, Face, Volume	Element	Direct ETAB	Direct
Abaqus	Face	Code		Buffer
FINE/Open	Face	Node, Element	Direct	
FINE/Turbo	Face	Code	Direct	
FLUENT	Face, Volume	Code	Dir	UDM
MSC Marc	Line, Face, Volume	Element		Direct
OpenFOAM	Line, Face, Volume	Code	Dir	
STAR-CCM+	Face	Code	Direct	
STAR-CD	Face, Volume	Code	Direct	SCALAR

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Global	global	APDL	APDL
FINE/Open	Global	global	Direct	
FINE/Turbo	Global	global	Direct	
FLUENT	Global	global	Dir	Dir
Flowmaster	Global	global	Direct	Direct
ANSYS Icepak	Global	global	Dir	Dir
RadTherm	Global	global	Direct	
STAR-CCM+	Global	global	Direct	Direct
STAR-CD	Global	global	Direct	Direct

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Point	Node	Direct	Direct
Abaqus	Point	Code	Direct	Buffer
MSC Adams	Point	Node	Direct	Usermemory
MATLAB	Point	Code	Direct	Direct
SIMPACK	Point	Node	Direct	

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

Code	Coupling Dimensions	Location	Send option	Receive option
Abaqus	Face, Volume	Code	Direct	Buffer
FLUENT	Volume	Code	Dir	UDM
STAR-CD	Volume	Code	Direct	SCALAR

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

Code	Coupling Dimensions	Location	Send option	Receive option
Abaqus	Face, Volume	Code	Direct	Buffer

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Global	global	APDL	APDL
MSC Adams	Global	global	Direct	Usermemory
FLUENT	Global	global	Dir	Dir
ANSYS Icepak	Global	global	Dir	Dir
MATLAB	Global	global	Direct	Direct
SIMPACK	Global	global		Direct
STAR-CD	Global	global	Direct	Direct

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Global	global	APDL	APDL
FINE/Open	Global	global	Direct	
FINE/Turbo	Global	global	Direct	
FLUENT	Global	global	Dir	Dir
STAR-CCM+	Global	global	Direct	Direct
STAR-CD	Global	global	Direct	Direct

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Face	Node		Direct
Abaqus	Face	Code		Buffer
FINE/Open	Face	Element	Direct	
FINE/Turbo	Face	Code	Direct	
FLUENT	Face	Code	Dir	UDM
MSC Marc	Line, Face	Node		Direct
MSC NASTRAN	Line, Face	Code		Direct
OpenFOAM	Line, Face	Code	Dir	
STAR-CCM+	Face	Code	Direct	Direct
STAR-CD	Face	Code	Direct	SCALAR

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Global	global	APDL	APDL
FLUENT	Global	global	Dir	Dir
ANSYS Icepak	Global	global	Dir	Dir
RadTherm	Global	global	Direct	
STAR-CD	Global	global	Direct	Direct

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Volume	Element	Direct	Direct
FLUENT	Volume	Code	Dir	UDM
ANSYS Icepak	Volume	Code	Dir	UDM
MATLAB	Volume	Node, Element	Direct	Direct
STAR-CD	Volume	Code	Direct	SCALAR

StaticPressure Static pressure [N/m²]

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

Code	Coupling Dimensions	Location	Send option	Receive option
STAR-CCM+	Face	Code	Direct	Direct

Temperature Temperature [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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Line, Volume	Node, Element	Direct	ETAB
Abaqus	Volume	Code	Direct	Buffer
FLUENT	Face, Volume	Code	Dir	UDM
Flowmaster	Point	Code	Direct	Direct
ANSYS Icepak	Volume	Code	Dir	UDM
JMAG	Face, Volume	Node		Direct
MATLAB	Point, Volume	Code, Element	Direct	Direct
OpenFOAM	Line, Face, Volume	Code	Dir	Dir
STAR-CCM+	Face	Code	Direct	Direct
STAR-CD	Volume	Code	Direct	SCALAR

ThermCond1 Thermal conductivity - xyz [W/m 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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Line, Face, Volume	Element	Direct	Direct
FLUENT	Face, Volume	Code	Dir	UDM
ANSYS Icepak	Face, Volume	Code	Dir	UDM
MATLAB	Volume	Node, Element	Direct	Direct
STAR-CD	Face, Volume	Code	Direct	SCALAR

ThermCond3 Thermal conductivity -(x,y,z) [W/m K]

Code API symbol: MPCCI_QID_THERMCOND3
 Default value: 0.0
 Dimension: Vector
 Physical meaning: Material property/general property
 Interpolation type: field
 Coupling Dimensions: Line, Face, Volume

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Line, Face, Volume	Element	Direct	Direct
FLUENT	Face, Volume	Code	Dir	UDM
ANSYS Icepak	Face, Volume	Code	Dir	UDM
MATLAB	Volume	Node, Element	Direct	Direct
STAR-CD	Face, Volume	Code	Direct	SCALAR

ThermCondX Thermal conductivity - x [W/m K]

Code API symbol: MPCCI_QID_THERMCONDX
 Default value: 0.0
 Dimension: Scalar
 Physical meaning: Material property/general property
 Interpolation type: field
 Coupling Dimensions: Line, Face, Volume

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Line, Face, Volume	Element	Direct	Direct
FLUENT	Volume	Code	Dir	UDM
ANSYS Icepak	Volume	Code	Dir	UDM
MATLAB	Volume	Node, Element	Direct	Direct
STAR-CD	Volume	Code	Direct	SCALAR

ThermCondY Thermal conductivity - y [W/m K]

Code API symbol: MPCCI_QID_THERMCONDY
 Default value: 0.0
 Dimension: Scalar
 Physical meaning: Material property/general property
 Interpolation type: field
 Coupling Dimensions: Line, Face, Volume

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Line, Face, Volume	Element	Direct	Direct
FLUENT	Volume	Code	Dir	UDM
ANSYS Icepak	Volume	Code	Dir	UDM
MATLAB	Volume	Node, Element	Direct	Direct
STAR-CD	Volume	Code	Direct	SCALAR

ThermCondZ Thermal conductivity - z [W/m K]

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Line, Face, Volume	Element	Direct	Direct
FLUENT	Volume	Code	Dir	UDM
ANSYS Icepak	Volume	Code	Dir	UDM
MATLAB	Volume	Node, Element	Direct	Direct
STAR-CD	Volume	Code	Direct	SCALAR

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Global	global	APDL	APDL
FINE/Open	Global	global	Direct	
FLUENT	Global	global	Dir	Dir
ANSYS Icepak	Global	global	Dir	Dir
RadTherm	Global	global	Direct	
STAR-CCM+	Global	global	Direct	Direct
STAR-CD	Global	global	Direct	Direct

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Point	Node	Direct	Direct
Abaqus	Point	Code	Direct	Buffer
MSC Adams	Point	Node	Direct	Usermemory
MATLAB	Point	Code	Direct	Direct
SIMPACK	Point	Node	Direct	Direct
STAR-CCM+	Face	Code	Direct	

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

Code	Coupling Dimensions	Location	Send option	Receive option
FLUENT	Face	Code	Dir	UDM
Flowmaster	Point	Code	Direct	Direct
MATLAB	Point	Code	Direct	Direct
OpenFOAM	Line, Face, Volume	Code	Dir	Dir
STAR-CCM+	Face	Code	Direct	Direct
STAR-CD	Face	Code	Direct	SCALAR

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Volume	Element		Direct
Abaqus	Point, Face, Volume	Code	Direct	Buffer
MSC Adams	Point	Node	Direct	Usermemory
FINE/Open	Face	Node, Element	Direct	
FLUENT	Point, Face, Volume	Code	Dir	UDM Buf
MATLAB	Point	Code	Direct	Direct
MSC NASTRAN	Line, Face	Code	Direct	
OpenFOAM	Line, Face, Volume	Code	Dir	Dir
SIMPACK	Point	Node	Direct	
STAR-CCM+	Point, Face	Code	Direct	Direct
STAR-CD	Face, Volume	Code	Direct	SCALAR

VelocityMagnitude 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

Code	Coupling Dimensions	Location	Send option	Receive option
FLUENT	Face	Code	Dir	UDM
Flowmaster	Point	Code	Direct	Direct
MATLAB	Point	Code	Direct	Direct

Voltage1 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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Global	global	APDL	APDL
FLUENT	Global	global	Dir	Dir
STAR-CD	Global	global	Direct	Direct

Voltage2 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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Global	global	APDL	APDL
FLUENT	Global	global	Dir	Dir
STAR-CD	Global	global	Direct	Direct

Voltage3 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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Global	global	APDL	APDL
FLUENT	Global	global	Dir	Dir
STAR-CD	Global	global	Direct	Direct

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Global	global	APDL	APDL
FLUENT	Global	global	Dir	Dir
STAR-CD	Global	global	Direct	Direct

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Face	Node		Direct
Abaqus	Face	Code	Direct	Buffer
FINE/Turbo	Face	Code	Direct	
FLUENT	Face	Code	Dir	UDM
MATLAB	Face	Code, Element	Direct	Direct
MSC Marc	Line, Face	Node		Direct
MSC NASTRAN	Line, Face	Code		Direct
OpenFOAM	Line, Face	Code	Dir	
STAR-CCM+	Face	Code	Direct	Direct
STAR-CD	Face	Code	Direct	SCALAR

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Face	Element	Direct	Direct
Abaqus	Face	Code		Buffer
FINE/Open	Face	Element	Direct	
FINE/Turbo	Face	Code	Direct	
FLUENT	Face	Code	Dir	UDM
ANSYS Icepak	Face	Code	Dir	UDM
MSC Marc	Line, Face	Element		Direct
OpenFOAM	Line, Face	Code	Dir	
RadTherm	Face	Code		Direct
STAR-CCM+	Face	Code	Direct	Direct
STAR-CD	Face	Code	Direct	SCALAR

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Face	Element		Direct
Abaqus	Face	Code		Buffer
FLUENT	Face	Code	Dir	UDM
ANSYS Icepak	Face	Code	Dir	UDM
MATLAB	Face	Code, Element	Direct	Direct
MSC Marc	Line, Face	Element		Direct
OpenFOAM	Line, Face	Code	Dir	
RadTherm	Face	Code		Direct
STAR-CCM+	Face	Code	Direct	
STAR-CD	Face	Code	Direct	SCALAR

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

Code	Coupling Dimensions	Location	Send option	Receive option
ANSYS	Face	Node, Element	Direct	Direct
Abaqus	Face	Code	Direct	Buffer
FINE/Open	Face	Element	Direct	Direct
FINE/Turbo	Face	Code	Direct	Direct
FLUENT	Face	Code	Dir	UDM
ANSYS Icepak	Face	Code	Dir	UDM
MATLAB	Face	Code, Element	Direct	Direct
MSC Marc	Line, Face	Node	Direct	
OpenFOAM	Line, Face	Code	Dir	Dir
RadTherm	Face	Code	Direct	
STAR-CCM+	Face	Code	Direct	Direct
STAR-CD	Face	Code	Direct	SCALAR

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.

YI08 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.

YI09 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

- Karetta, F. and M. Lindmayer, 1998: Simulation of gasdynamic and electromagnetic processes in low voltage switching arcs. *IEEE Transactions on components, packaging, and manufacturing technology-Part A*, 21(1): 96-103, IEEE.
- Koch, M. D., 2016: *Quasi-Newton Methods for Unstable Partitioned Fluid-Structure Interactions*. Master's thesis, Institut für Numerische Simulation, Rheinische Friedrich-Wilhelms-Universität Bonn.
- Lyttle, I., B. Pulido, A. Zolfaghari, and K. Parker, 2006: CUSTOMIZATION OF MPCCI FOR USE BY DEVELOPMENT TEAMS: MAGNETO-THERMAL SIMULATIONS. K. Wolf, ed., *Proceedings of the 7th MpCCI User Forum*, Fraunhofer Institute SCAI.
- Mok, D. P., 2001: *Partitionierte Lösungsansätze in der Strukturdynamik und der Fluid-Struktur-Interaktion*. Ph.D. thesis, Institut für Baustatik und Baudynamik, Universität Stuttgart.
- Schwartz, R. L., T. Phoenix, and B. D. Foy, 2005: *Learning Perl*. 4th edn., O'Reilly, covers Perl 5.8.
- Walhorn, E., B. Hübner, and D. Dinkler, 2002: Starke Kopplung von Fluid und Struktur mit Raum-Zeit-Elementen. *Fluid-Struktur-Wechselwirkung*, VDI-Bericht 1682, 221–240, VDI Verlag GmbH.
- Wall, W. A., 1999: *Fluid-Struktur-Wechselwirkungen mit stabilisierten Finiten Elementen*. Ph.D. thesis, Institut für Baustatik, Universität Stuttgart.
- Wirth, N. and A. Oeckerath, 2015: Analysis of flow-induced vibrations in turbomachinery by mapping of complex fluid pressures. *International Journal of Multiphysics*, 9(2): 195-208.
- Zienkiewicz, O. C. and R. Taylor, 2000: *The Finite Element Method*. Butterworth-Heinemann.

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.

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 the [▷ II-5 Supported Platforms in MpCCI 4.5 ◁](#).

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 Algorithms ◁](#).

coupling component

A coupling component is a part of a coupling region, typically a set of elements, see description of *coupling region*.

Coupling Step

The Coupling Step is a panel in the wizard-like MpCCI GUI where the *coupling regions* and their quantities are defined, see [▷ V-4.5 Coupling Step ◁](#).

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-2.5 Coupling Step – Defining Coupling Regions and Quantities ◁](#) and [▷ V-4.5 Coupling Step ◁](#)).

coupling type

A set of typical *coupling types* can be selected in the *Coupling Step* of the MpCCI GUI. Each type corresponds to a predefined selection of quantities for a coupling dimension, see [▷ V-4.5 Coupling Step ◁](#) for details.

Edit Step

The Edit 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-4.7 Edit Step ◁](#).

exchange

Other term for *transfer*.

field

One of the two interpolation principles (see also *flux*). Field interpolation is applied for quantities which do not depend on the element area, e.g. density, electric resistivity or temperature. See ▷ [V-3.3.2.2 Flux and Field Interpolation](#) ◤.

flux

One of the two interpolation principles (see also *field*). Flux interpolation is applied for quantities for which the integral over an area/volume must be preserved, e.g. forces or mass flux. See ▷ [V-3.3.2.2 Flux and 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 ▷ [V-4.8 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 ▷ [V-3.4.1 Course of the 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 ▷ [V-3.3 Data Exchange](#) ◤.

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 ▷ [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 ▷ [V-4.6 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 ▷ [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 ▷ [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 *Coupling Step* of MpCCI. A quantity has a typical SI-unit.

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-4.9 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-6 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 FSIMapper
- XI Appendix

Important entries are **bold**, *italic* entries refer to the glossary.

- Abaqus, **VI-15**
- acoustics, **V-20**
- ActivePerl, **III-9**
- Adams, *see* MSC Adams
- adaptive relaxation, **V-36**
- Additional Scalars, **VI-207**
- Aitken relaxation, **V-36**
- align, **VI-13**
- angular interpolation, **V-104**
- ANSYS, **VI-30**
- API Kit, **VIII-9**
- applied-force coupling, **VII-122**
- architecture, **V-12, V-12, XI-40, V-134, V-136**
 - supported by MpCCI, **II-20**
- association, **IV-5, VIII-19, V-27, XI-40, V-43**
- Automotive Thermal Management, **IX-4**
- axisymmetry, **VII-85**
- backup, **V-150**
- baffle shift, **V-104**
- baffle thickness definition, **VIII-57, VIII-57**
- batch execution, **V-62, V-76, V-151**
- beam element types, **VIII-71**
- bounding box check, **V-23**
- CFD, **IV-7, IV-8, I-9, V-20**
- cgs units, **VII-11**
- check
 - mesh, **V-22**
- checker, **VIII-28, VIII-36**
- clean
 - temporary files, **V-158**
- client, **XI-40, V-60**
- cluster, **V-63**
- code adapter, **IV-6, VI-11, XI-40**
- code configuration directory, **VIII-22**
- code integration, **VIII-6, VIII-7**
 - step-by-step, **VIII-12**
- command line interface, **V-115**
- complex start, **V-59**
- Computational Fluid Dynamics, *see* CFD
- configuration, **VIII-49**
- configuration directory, **VIII-22**
- conformal mesh, **V-104**
- connection, **VIII-40**
- control parameters, **V-102**
- control process, *see* tracefile
- coordinate transformation, **VI-13**
- coupled system, **IV-4**
- coupling, **V-20**
- coupling algorithm, **XI-40, V-43**
- coupling algorithms, **V-44**
- coupling component, **XI-40**
- coupling manager functions, **VIII-44**
- coupling point, **VII-124**
- coupling region, **XI-40**
- Coupling Step, **IV-10, XI-40**, *see* GUI
- coupling type, **V-21, XI-40, V-97**
- data flow
 - visualizer, **V-168**
- data structure, **VIII-71**
- debugging, **V-13, V-138**
- directory
 - MpCCI resource, **V-16**
 - code configuration, **VIII-22**
- domain
 - physical, **V-20**
- domain check, **V-23, V-104**
- domain check visualizer, **V-24**

- download, [III-7](#)
 driven cavity, [VII-22](#)
 driver function, [VIII-62](#)
 duc heater, [VII-93](#)
- Edit Step, [XI-40](#), *see* GUI
 elastic flap, [VII-30](#)
 elastic flap water, [VII-44](#)
 elastic foundation, [VIII-9](#)
 electromagnetic, [VII-65](#), [VII-75](#)
 electromagnetism, [V-21](#)
 electrothermal analysis, [V-22](#)
 element definition, [VIII-54](#), [VIII-54](#)
 element-element relationship, *see* intersection
 endianness, [V-61](#)
 environment variable, [V-11](#), [III-14](#), [VIII-28](#), [V-133](#), [V-138](#)
 example
 code integration, [VIII-9](#)
 exchange, [XI-40](#), [VIII-58](#), [VIII-70](#)
 before solution, [IV-24](#)
 initial, [V-13](#), [VIII-47](#)
 exhaust, [VII-54](#)
- FE, [IV-7](#), [IV-8](#), [I-9](#)
 FEM, [V-20](#)
 field, [XI-41](#)
 file system, [V-113](#)
 files, [V-15](#)
 temporary, [V-17](#)
 finalization, [V-43](#)
 FINE/Open, [VI-53](#)
 FINE/Turbo, [VI-60](#)
 Finite Element, *see* FE
 firewall, [V-61](#)
 flap
 elastic, [VII-30](#)
 flapwater
 elastic, [VII-44](#)
 FLEXlm, *see* license
 FlexNet Publisher, [III-17](#)
 Flowmaster, [VI-68](#)
 FLUENT, [VI-76](#)
 fluid domain, [IV-8](#)
 Fluid Dynamics, *see* CFD
 fluid-structure interaction, *see* FSI
 Fluid-structure interaction pump simulation, [IX-13](#)
 flux, [XI-41](#)
 force/displacement coupling, [VII-122](#)
 foundation example, [VIII-9](#)
 FSI, [VII-11](#), [V-21](#), [VII-22](#), [VII-30](#), [VII-44](#), [VII-85](#)
- MpCCI FSIMapper, [X-15](#)
 Batch, [X-44](#)
 command, [V-119](#)
 How to map, [X-25](#)
 Transformation, [X-21](#)
 What to map, [X-16](#)
- GLOBUS, [V-74](#), [V-157](#)
 glossary, [XI-40](#)
 Go Step, [XI-41](#), *see* GUI
 Graphical User Interface, *see* GUI
 grid, [XI-41](#)
 GUI, [V-76](#), [V-120](#)
 check, [V-111](#)
 configuration file, [VIII-23](#)
 Coupling Step, [IV-10](#)
 automatic region generation, [V-89](#)
 region generation, [V-89](#)
 rules, [V-89](#)
 Edit Step, [IV-17](#)
 options, [V-102](#)
 Go Step, [IV-18](#), [VIII-26](#), [V-109](#)
 menus, [V-77](#)
 Models Step, [IV-9](#), [VIII-24](#)
 Monitors Step, [IV-17](#)
 options, [V-101](#)
 properties, [V-77](#)
 Start, [IV-9](#)
 title, [V-77](#)
- heat transfer, [V-21](#)
 hexahedral element types, [VIII-77](#)
 HOME, [III-14](#)
 home
 MpCCI, [V-10](#)
 home directory, [V-12](#)
 hostlist file, [V-61](#)
- ANSYS Icepak, [VI-50](#)
 info, [VI-12](#), [VIII-38](#)
 information, [V-133](#)
 initial exchange, [V-13](#), [IV-18](#), [VIII-47](#)
 initialization, [VIII-40](#), [XI-41](#), [V-43](#), [VIII-48](#)
 installation, [III-5](#), [V-141](#)
 Perl, [III-32](#)
 Windows, [III-12](#)
 multi-platform, [III-11](#)
 prerequisites, [III-7](#)
 interpolation, [IV-5](#), [V-29](#), [XI-41](#), [V-57](#), [V-104](#)
 iteration, [V-43](#)
 iterative coupling, [V-50](#)
- Java, [III-9](#)

- JMAG, [VI-95](#)
 job, [V-104](#)
 job control, [V-149](#)
 job scheduler, [V-63](#)
 killer, [VIII-28](#), [VIII-38](#)
 killing jobs, [V-159](#)
 license, [III-17](#), [V-141](#), [V-142](#), [V-144](#)
 FSIMapper, [III-18](#)
 Morpher, [III-18](#)
 LoadLeveler, [V-73](#), [V-156](#)
 loop, [VIII-91](#)
 LSF, [V-70](#), [V-153](#)
 macro, [VIII-71](#)
 Magneto-Thermal, [VII-65](#), [VII-75](#)
 manifold, [VII-54](#)
 Marc, *see* MSC Marc
 MATLAB, [VI-102](#)
 mesh, [XI-41](#)
 mesh check, [V-22](#)
 mesh definition, [VIII-50](#), [VIII-68](#)
 mesh deletion, [VIII-52](#)
 mesh motion, [V-23](#), [V-39](#), [V-104](#)
 Microsoft Windows, *see* Windows
 mixed solution type coupling, [V-48](#)
 model file, [VI-13](#)
 model preparation, [IV-7](#)
 Models Step, [XI-41](#), *see* GUI
 monitor
 options, [V-102](#)
 Monitors Step, [XI-41](#)
 morpher
 MpCCI, [VII-30](#), [VII-44](#), [V-128](#), [V-192](#)
 OpenFOAM, [VI-153](#)
 STAR-CCM+, [VI-188](#)
 pro-STAR, [VI-213](#)
 mount, [V-113](#)
 moving reference frame, [VIII-56](#), [VIII-56](#)
 moving reference frame correction (MRF correct),
 [V-104](#)
 MpCCI
 Run Implicit FSI, [VI-84](#)
 API, [VIII-6](#)
 Control Panel (FLUENT), [VI-84](#)
 FSIMapper, [X-6](#)
 Grid Morpher, [VII-30](#), [VII-44](#), [V-128](#), [VI-153](#), [V-192](#), [VI-213](#)
 GUI, *see* GUI, [V-76](#)
 Logviewer, [V-121](#)
 Monitor, [V-122](#)
 Observer, [V-131](#)
 Tools, [V-125](#)
 Visualizer, *see* Visualizer, [V-168](#)
 MPCCI_ARCH, *see* architecture
 MPCCI_DEBUG, [V-13](#)
 MPCCI_HOME, [V-10](#), [V-139](#)
 MPCCI_HOSTLIST_FILE, [V-62](#)
 MPCCI_TINFO, [V-13](#)
 MPCCI_LICENSE_FILE, [III-22](#)
 MPCCI_NOREMESH, [III-24](#)
 MPCCI_RSHTYPE, [III-24](#), [V-62](#)
 MPCCI_TMPDIR, [V-17](#)
 MS HPC, [V-68](#)
 MSC Adams, [VI-115](#)
 MSC Marc, [VI-128](#)
 MSC NASTRAN, [VI-136](#)
 MSI, *see* windows installer
 multi-platform installation, [III-11](#)
 multiphysics, [IV-4](#), [V-20](#), [XI-41](#)
 N1GE, [V-72](#), [V-155](#)
 Nastran, *see* MSC NASTRAN
 neighborhood search, [XI-41](#), *see* association
 network, [V-21](#), [III-24](#), [V-60](#), [V-111](#)
 node definition, [VIII-53](#), [VIII-53](#)
 non-matching time steps, [V-57](#)
 nozzle, [VII-85](#)
 online monitor, [V-102](#)
 OpenFOAM, [VI-145](#)
 Morpher, [VI-153](#)
 OpenPBS, [V-71](#), [V-154](#)
 OpenSSH, [III-9](#)
 orphaned nodes, [V-30](#), [XI-41](#)
 output, [VIII-45](#), [V-107](#)
 overlap check, [V-104](#)
 parallel run, [V-60](#)
 part definition, [VIII-50](#)
 part deletion, [VIII-52](#)
 PATH, [V-10](#), [III-11](#)
 PBS, [V-71](#), [V-154](#)
 PBSPro, [V-71](#), [V-154](#)
 periodic components, [V-85](#)
 periodic model, [V-42](#)
 periodic models, [VII-141](#)
 Perl, [III-9](#), [III-14](#), [VIII-35](#)
 installation, [III-32](#)
 physical domain, [V-20](#)
 pipe nozzle, [VII-85](#)
 platform, *see* architecture
 Plug-In, [VI-204](#)
 point-element relationship, *see* minimal distance
 polygonal element types, [VIII-78](#)

- polyhedral element types, [VIII-79](#)
 port, [V-61](#)
 post-processing, [IV-27](#)
 prefix, [VIII-46](#)
 prerequisites
 installation, [III-7](#)
 pressure
 reference, [V-21](#)
 pro-STAR
 Grid Morpher, [VI-213](#)
 ptoj, [V-162](#)
 pyramid element types, [VIII-75](#)
 quadrilateral element types, [VIII-73](#)
 quantity, [IV-4](#), [VIII-25](#), [XI-41](#)
 descriptor, [VIII-90](#)
 reference, [XI-4](#)
 Quasi-Newton methods, [VII-22](#), [V-37](#)
 queuing system, [V-63](#)
 radiation, [V-21](#)
 RadTherm, [VI-159](#)
 ramping, [V-34](#)
 reference pressure, [V-21](#)
 FLUENT, [VI-77](#)
 OpenFOAM, [VI-146](#)
 region, *see* coupling region
 relation search, [V-106](#)
 relaxation, [V-33](#)
 adaptive, [V-36](#)
 Aitken, [V-36](#)
 Quasi-Newton, [V-37](#)
 ramping, [V-34](#)
 releases, [VI-12](#)
 remeshing, [VIII-60](#)
 remote computing, [V-60](#)
 remote file browser, [XI-42](#), [V-111](#)
 remote shell, [V-19](#), [III-24](#), [V-62](#), [V-113](#), [V-137](#),
 [V-145](#)
 testing, [III-25](#)
 requirements
 code integration, [VIII-7](#)
 resource directory, [V-16](#)
 restart, [V-59](#)
 results, [IV-26](#)
 MpCCI-RSH, [III-10](#)
 Preparing the .rhosts file, [III-29](#)
 rsh, *see* remote shell
 scanner, [VI-13](#), [VIII-28](#), [VIII-35](#), [XI-42](#)
 Start, [IV-9](#)
 server, [XI-42](#), [V-60](#), [V-163](#)
 SGE, [V-72](#), [V-155](#)
 SGEEE, [V-72](#), [V-155](#)
 shape function, [V-29](#)
 SIMPACK, [VI-172](#)
 simulation code, [VI-11](#), [III-15](#), [XI-42](#)
 Slave node mark, [V-25](#)
 software
 MpCCI, [V-9](#)
 third party, [V-19](#)
 solid mechanics, [V-20](#)
 spring mass, [VII-122](#)
 ssh, *see* remote shell
 stability, [VII-122](#)
 staggered method, [IV-5](#), [XI-42](#)
 STAR-CCM+, [VI-179](#)
 Morpher, [VI-188](#)
 STAR-CD, [VI-203](#)
 Additional Scalars, [VI-207](#)
 Plug-In, [VI-204](#)
 Restart, [VI-204](#), [VI-209](#)
 starter, [VIII-28](#), [VIII-37](#)
 starting MpCCI, [V-118](#)
 stationary problems, [V-44](#)
 step-by-step code integration, [VIII-12](#)
 stopper, [VIII-28](#), [VIII-37](#)
 storage index, [VI-207](#)
 subcmd, [VIII-38](#)
 subcommands
 list, [V-116](#)
 subcycling, [V-55](#)
 surface coupling, [IV-4](#)
 symmetry
 axial, [VII-85](#)
 temporary files, [V-17](#)
 terminal window, [V-132](#)
 testing
 installation, [III-14](#)
 license server, [III-23](#)
 MpCCI installation, [V-146](#)
 remote shell, [III-25](#)
 tetrahedral element types, [VIII-74](#)
 thermal coupling, [V-22](#), [VII-65](#), [VII-75](#)
 third party software, [V-19](#)
 time step size
 exchange, [V-54](#)
 time tolerance, [V-104](#)
 Torque, [V-71](#), [V-154](#)
 tracefile, [IV-26](#), [XI-42](#), [V-107](#), [V-168](#)
 transfer, [VIII-58](#), [VIII-58](#)
 transfer information, [VIII-47](#)
 transformation matrix, [VIII-56](#)
 transient problems, [V-44](#)

triangle element types, [VIII-72](#)

unit, [VII-85](#)

units, [VII-11](#), [VI-14](#)

user-defined function, [VII-85](#)

version

mpCCI, [V-136](#)

vibration, [VII-11](#)

Visualizer, [IV-26](#), [V-123](#), [V-168](#)

 Menus and Toolbars, [V-170](#)

 Mouse Interaction, [V-183](#)

 Panels, [V-176](#)

 Preferences Server Dialog, [V-187](#)

 Preferences Viewer Dialog, [V-185](#)

 Viewport Area, [V-182](#)

volume coupling, [IV-4](#)

vortex street, [VII-11](#)

wedge element types, [VIII-76](#)

Windows

 rcp, [III-28](#)

 rsh, [III-28](#)

 rlogin, [III-28](#)

Windows, [III-14](#), [III-28](#), [III-28](#)

windows installer, [III-12](#)

writer, [V-107](#)

xterm, [V-132](#)

Y-Junction, [VII-105](#)