Technische Universität München

Ingenieurfakultät Bau Geo Umwelt

Lehrstuhl für Statik

Stabilized Co-Simulation of Coupled Problems Including Fields and Signals

Stefan Alfred Sicklinger

Vollständiger Abdruck der von der Ingenieurfakultät Bau Geo Umwelt der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender:

Univ.-Prof. Dr.-Ing. habil. Fabian Duddeck

Prüfer der Dissertation:

1. Univ.-Prof. Dr.-Ing. Kai-Uwe Bletzinger
2. Prof. Dr. Riccardo Rossi,
   Universitat Politècnica de Catalunya - BarcelonaTech/Spanien

Die Dissertation wurde am 3. Juli 2014 bei der Technischen Universität München eingereicht und durch die Ingenieurfakultät Bau Geo Umwelt am 28. November 2014 angenommen.

Zusammenfassung

Das Berechnungsverfahren Co-Simulation wird mehr und mehr zu einer integralen und unverzichtbaren Methode zur Lösung der heutigen anspruchsvollen technischen Fragestellungen. Mit Hilfe dieser Kopplungstechnik, kann das zu lösende Gesamtproblem in einzelne Teilsysteme aufgeteilt werden, welche zur Laufzeit Informationen austauschen. Der inhärente Vorteil der Co-Simulation ist im Gegensatz zum gesamtheitlichen (monolithischen) Ansatz, dass etablierte und spezialisierte Simulationswerkzeuge wieder verwendet werden können. Darüber hinaus erlaubt die Co-Simulation verschiedene Modellierungstiefen der einzelnen Teilsystemen, dem jeweiligen Stand des Produktentwicklungszyklus entsprechend zu kombinieren und anzupassen. Leider stellt die partitionierte Behandlung der einzelnen Teilsysteme eine Herausforderungen für die numerische Stabilität und Genauigkeit dar.

Als Ziel dieser Arbeit wurde ein neuartiges Berechnungsverfahren für die Co-Simulation entwickelt, das als Interface-Jacobian-based Co-Simulation Algorithm (IJCSA) bezeichnet wird. Das Verfahren stabilisiert die Co-Simulation während die Modularität erhalten bleibt. Ferner kann dieser Algorithmus eine beliebige Anzahl von Feldern und Signalen bedienen. Selbst algebraische Schleifen können einfach aufgelöst werden, da er auf einer Residualform beruht. Des Weiteren können alle an der Co-Simulation beteiligten Subsysteme parallel gestartet werden, wodurch die Laufzeit der Simulation deutlich verkürzt wird. Ebenfalls Teil dieser Arbeit ist eine eingehende Stabilitätsbetrachtung des Algorithmus.

Abschließend wird die Anwendbarkeit des IJCSA mit mehreren industriell relevanten Beispielen veranschaulicht. Die gezeigten Beispiele reichen von einer vollständig gekoppelten und geregelten Fluid-Struktur-Signal-Wechselwirkungen bis hin zu einer vollständig gekoppelten Notbremsung einer Windturbine. Bei der Notbremsung wird die Interaktion des Generators/Getriebes, der flexiblen Verbundrotorblätter, der Steuereinheit und dem dreidimensionalen turbulenten Strömungsfeld berücksichtigt. Darüber hinaus werden die Simulationsergebnisse anhand von Messdaten aus dem National Renewable Energy Laboratory (NREL) Unsteady Aerodynamics Experiment Phase VI durchgeführt im NASA AMES Windkanal validiert.

Abstract

Co-simulation is becoming an increasingly integral and indispensable technique for solving today's challenging engineering problems. By means of this code coupling technique, the engineering problem is partitioned as an assembly of different subsystems exchanging solution information at run time. The inherent advantage of co-simulation in contrast to the monolithic approach is that it allows the (re)use of well-established and specialized simulation software to be combined, with minor alterations. Furthermore, co-simulation allows different fidelity models to be combined at different stages of the design process. Unfortunately, this partitioned treatment of the individual system poses stability and accuracy challenges.

A novel co-simulation algorithm is introduced, referred to as the Interface Jacobian-based Co-Simulation Algorithm (IJCSA), which overcomes present stability issues. The algorithm can solve co-simulation scenarios involving an arbitrary number of fields and signals. Due to the fact that the IJCSA is based on the residual form it handles algebraic loops in a natural manner. Furthermore, the individual simulators can run in parallel without flow dependency reducing the wall-clock time of the simulation, since the subsystems do not have to be executed using the classical Gauss-Seidel pattern. A thorough stability analysis of the IJCSA is presented.

In order to demonstrate the applicability, several industrially relevant examples are solved by using the IJCSA. The shown examples range from a fully coupled fluid-structure-signal interaction with closed-loop control to fully coupled emergency brake maneuver of a wind turbine. Here the interaction of the generator/gearbox, flexible composite blades, control unit and the three-dimensional flow field is taken into account. Furthermore, the simulation results are validated against measurement data from the National Renewable Energy Laboratory (NREL) Unsteady Aerodynamics Experiment Phase VI, performed in the NASA AMES wind tunnel.

Acknowledgments

This dissertation was written from 2011 to 2014 during my time as research assistant at the Chair of Structural Analysis at the Technische Universität München, Munich, Germany.

I would like to thank Prof. Dr.-Ing. Kai-Uwe Bletzinger for giving me the possibility to work in his research group. Moreover, I would like to sincerely thank him not only for his helpful and inspiring guidance as doctoral supervisor, but also for providing me the academic freedom to develop and realize new ideas and methods. I also want to thank Dr.-Ing. Roland Wüchner for fruitful and inspiring discussions.

Furthermore, I would like to thank Prof. Dr. Riccardo Rossi for being my co-examiner and for his interest in my work. Also I want to thank Univ.-Prof. Dr.-Ing. habil. Fabian Duddeck for charing the jury.

My gratitude goes also to my coworkers at the Chair of Structural Analysis for their friendly cooperation and for the pleasant time that I had working with them. In particular I would like to thank Tianyang Wang and my student Christopher Lerch for numerous inspiring and motivating discussions.

I would like to gratefully point out the cooperation with the headquarter of Dassault Systèmes SIMULIA. Especially, I want to gratefully express my appreciation towards Bruce Engelman, Vladimir Belsky, Albert Kürkchübasche and Jeff Haan who gave me ideas and valuable hints. Moreover, I would like to sincerely thank them for their hospitality during my visits.

Finally, I would like to thank my parents and my brother, without whom I would never have been able to achieve so much. Most importantly, I would like to express my deepest gratitude to Anja and my son Christian for their patience, support, encouraging inspiration and advice during all times.

Stefan Alfred Sicklinger
Technische Universität München
December 18, 2014

List of Symbols and Abbreviations

Calligraphic letters

$\mathcal{E}()$	extrapolation operator for next time step
$\mathcal{I}_i()$	interface constraint operator for interface constraint i
$\mathcal{J}()$	Jacobian extraction operator
${}^m\boldsymbol{\mathcal{R}}_i^n$	residual associated with input i at time step n at iteration m (vector)
${}^m\mathcal{R}_i^n$	residual associated with input i at time step n at iteration m (scalar)
$\mathcal{S}_i()$	operator of subsystem i

Greek letters

λ	eigenvalue
μ	dynamic viscosity
ν	kinematic viscosity
ω	eigenfrequency
ω_d	damped eigenfrequency
π	$\approx 3.141\,592\,653\,589\,793$
ρ	spectral radius
ε	Hencky strain
ϱ	density

Mathematical symbols

Σ	sum
∂	operator for partial derivative
δ	first variation
$\|\|\|_2$	euclidean norm
$\|\|\|_{\max}$	maximum norm
$\mathbb{R}$	set of real numbers

Latin letters

c_{D}	drag coefficient
c_{L}	lift coefficient
c_{P}	pressure coefficient
E	Young's modulus
$\boldsymbol{I}$	identity matrix
${}^m\boldsymbol{U}_i^n$	input of subsystem i at time step n at iteration m (vector)
${}^mU_i^n$	input of subsystem i at time step n at iteration m (scalar)
${}^m\boldsymbol{X}_i^n$	state of subsystem i at time step n at iteration m (vector)
${}^mX_i^n$	state of subsystem i at time step n at iteration m (scalar)
${}^m\boldsymbol{Y}_i^n$	output of subsystem i at time step n at iteration m (vector)
${}^mY_i^n$	output of subsystem i at time step n at iteration m (scalar)

Abbreviations:

ALE	arbitrary Lagrangian-Eulerian
BDF2	second order Backward Differentiation Formula
BE	backward Euler
CAD	computer-aided design

CFD	computational fluid dynamics
CSE	Co-Simulation Engine
CSM	computational structural mechanics
DAE	differential algebraic equation
DOF	degree of freedom
DOFs	degrees of freedom
FEM	finite element method
FSI	fluid-structure interaction
GMRES	generalized minimal residual
GS	Gauss-Seidel
GSE	global sensitivity equation
IJCSA	Interface Jacobian-based Co-Simulation Algorithm
JC	Jacobi
JFNK	Jacobian-free Newton-Krylov
NASA	National Aeronautics and Space Administration
NREL	National Renewable Energy Laboratory
ODE	ordinary differential equation
PDAE	partial differential algebraic equations
PID	proportional-integral-derivative
TR	trapezoidal rule
TUM	Technische Universität München (University of Technology, Munich)
URANS	unsteady Reynolds averaged Navier-Stokes
Re	Reynolds number
Sr	Strouhal number

CONTENTS

List of Symbols and Abbreviations v

Contents ix

1 Introduction 1

2 Mathematical and Algorithmic Framework 7
2.1 Notation for Co-Simulation of Multiple Subsystems 8
2.2 Fixed-Point Iteration 9
2.2.1 Constant Under-Relaxation 12
2.2.2 Aitken Acceleration 14
2.3 Newton Methods 19
2.3.1 Quasi Newton Method 22
2.3.2 Newton-Krylov Methods 24
2.3.3 Krylov Subspace Methods 25
2.3.4 Jacobian-free Newton-Krylov Methods . . . 26
2.4 Extrapolation 28
2.5 Elements of Numerical Analysis 30

3 Co-Simulation **33**
3.1 Monolithic 35
3.1.1 Backward Euler 35
3.1.2 Generalized-α Method 36
3.1.3 BDF2 38
3.1.4 Numerical Results 38
3.2 Partitioning Procedure – From Monolithic to Co-Simulation 39
3.2.1 Co-Simulation with Coherent Time Integration Schemes 43

3.2.2 Co-Simulation with Mixed Time Integration Schemes . . . 46
3.3 Communication Pattern 57
3.3.1 Jacobi - Parallel . . . 58
3.3.2 Gauss-Seidel - Serial . . . 59
3.4 Decomposition 60
3.5 Block Diagram 68

4 Interface Jacobian-based Co-Simulation Algorithm 69
4.1 The Algorithm for two Subsystems 70
4.2 Generalization of the Concept 72
4.3 Efficiency Enhancements 76
4.4 Usability Enhancements - Jacobian Approximation 76
4.5 Interface Jacobian Extraction 78
4.6 Stability Considerations 80
4.6.1 Gauss-Seidel Fixed-Point Iterations . . . 83
4.6.2 Jacobi Fixed-Point Iterations . . . 84
4.6.3 Interface Jacobian-based Co-Simulation Algorithm . . . 85
4.6.4 Discussion of the Stability Properties . . . 86
4.7 Examples 88
4.7.1 Truss versus Truss Problem . . . 88
4.7.2 A Multi-Code Problem . . . 103
4.7.3 BspK6 . . . 110
4.8 Conclusion 117

5 Application Examples 119
5.1 Turek Benchmark 120
5.1.1 Co-Simulation . . . 120
5.1.2 Results . . . 124
5.2 Oscillating Cylinder 126
5.2.1 CFD Validation . . . 126
5.2.2 Forced Oscillation Validation . . . 129
5.2.3 Fluid-Structure with Closed-Loop Control . 129
5.2.4 Conclusion . . . 135
5.3 NREL Phase VI Wind Turbine 135
5.3.1 Experiment . . . 136
5.3.2 CFD Model . . . 138
5.3.3 CSM Model . . . 151
5.3.4 Handling Deformations & Rotations . . . 154

5.3.5 Fluid-Structure Interaction Model 157
5.3.6 Emergency Brake Maneuver 162
5.3.7 Emergency Brake Maneuver with Flexible Blades 174

6 Conclusion and Outlook **185**

A Algebraic Loops **189**

B Aspects of Co-Simulation Software Realization **193**

Bibliography **197**

An investment in knowledge
pays the best interest.

Benjamin Franklin

CHAPTER 1

INTRODUCTION

"Storm caused wind turbine fire"[1] this headline news is one which the manufacturers and designers of wind turbines try to avoid. The failure or wrong design of a wind turbine shut down mechanism can have a catastrophic consequence as shown in Figure 1.1.

In order to design for an emergency brake maneuver load case the consideration of the interaction of all system components with the wind is critical. In order to avoid failures in the final product the virtual analysis (simulation) is an indispensable tool. The term simulation within this work is used for the combination of modeling reality and the numerical solution of the mathematical model.

In the case where simulation needs to model the interaction of a various number of technical components with their environment it usually results in so-called multiphysics simulations. Hence, multiphysics simulations involve multiple physical models or multiple concurrent physical phenomena. For instance in order to simulate the emergency brake maneuver of a wind turbine the interaction of

[1] `http://www.bbc.co.uk/news/uk-16115139` British Broadcasting Corporation [21]

Figure 1.1: Exploded wind turbine in Ardrossan, North Ayrshire, Scotland due to high winds and problems with the emergency shutdown British Broadcasting Corporation [21]

the generator/gearbox, flexible composite blades, control unit and the three-dimensional flow field is essential.

Multiphysics typically involves solving coupled systems (Definition 1.1) of partial differential algebraic equations (PDAE).

Definition 1.1: Coupled System

"Coupled systems and formulations are those applicable to multiple domains and dependent variables which usually (but not always) describe different physical phenomena and in which

(a) neither domain can be solved while separated from the other;
(b) neither set of dependent variables can be explicitly eliminated at the differential equation level.

"[a]

[a] Zienkiewicz [164]

For the solution of the coupled problem there are two main methods, namely monolithic and partitioned.

The monolithic approach discretizes the coupled problem as a whole single system and afterwards solves the resulting equation system. On the one hand the advantage of this approach is that this usually results in a robust and accurate numerical method. On the other hand a single software package needs to be able to model all the needed physics. Therefore, it is not possible to reuse any existing solvers. Furthermore, within the design process of a product the fidelity level of the simulation models are increased towards the end as more refined analysis needs to be preformed as the design of the product matures. With the monolithic approach the existing simulation cannot be reused if the fidelity level needs to be changed.

The partitioning procedure identifies the process of spatial separation of the coupled problem into multiple partitioned subsystems and a set of interface constraints (see also Felippa et al. [49]). The partitioned treatment allows that the modeling process can be done for each subsystem separately. That further implies that well-established and specialized simulation tools can be (re)used for the different subsystems. On top of that the modeling can be done by different experts at the same time for the individual subsystems. Furthermore, the partitioned approach allows different fidelity models to be combined at all stages of the design process. If the partitioning is done at the PDAEs level it is called co-simulation (see Definition 3.1). However, these advantages come for the price of numerical stability and accuracy issues.

In contrast to the partitioning where the term is linked to the spatial separation process, the term decomposition is used in this work for the process of defining the input/output relations for each individual subsystem.

The introductory example of an emergency brake maneuver of a wind turbine poses a further complication. Here, the coupling of fields (see Definition 1.2) and signals (see Definition 1.3) is needed. Since the discretizations of the composite blades and the flow field result in fields whereas the generator/gearbox and the control unit result in subsystems of type signal. Therefore, simulation techniques are needed in order to cope with field-signal coupling of an arbitrary number of subsystems.

Definition 1.2: (Physical) Field

" A field is a physical quantity that has a value for each point in space and time. "[a]

[a] Gribbin [65]

Definition 1.3: Signal

A signal is defined in general as an abstract description of one varying quantity, where the independent variable in most cases is time. [a]

[a] Frey et al. [54] and Manolakis et al. [99]

The goal of this work is the design of coupling algorithms for exactly this situation where a small number of field subsystems are coupled to a moderate number of signal subsystems. The developed algorithm which is called Interface Jacobian-based Co-Simulation Algorithm (IJCSA) combines the advantages of both monolithic and co-simulation approaches. It is based on a hybrid idea which preserves modularity and adds stability and accuracy to the classical co-simulation approaches.

As consumer products are becoming increasingly sophisticated there are a lot of technical disciplines which are benefiting of such simulation capabilities. An incomplete list includes

- Aerospace
- Automotive
- Micro-electromechanical systems (MEMS)
- Medical devices
- Structural engineering

The IJCSA is designed for a general situation where very little knowledge of the individual subsystem is available. Therefore, there exist tailored algorithms for the coupling of a specific situation that perform better than the IJCSA. For the classical co-simulation approach several solution algorithms are proposed in literature. For the coupling of two simulators there are numerous references, for instance Degroote [34], Farhat et al. [42], Felippa et al. [48], Gravouil et al. [64], and Küttler et al. [89]. The modular co-simulation of surface

coupled problems which result in field-type subsystems is presented by Schulte [130].

For the co-simulation of an arbitrary number of signals there is development towards a standardized interface (called Functional Mock-up Interface (FMI)) on the basis of functional mock-up units (FMU) ongoing. As the *FMI standard* [53] does not specify coupling algorithms the development of coupling algorithms is left to the implementors.

Tailored solutions for the co-simulation of multibody dynamics (signals) and finite element systems (fields) are presented in Busch et al. [26]. A parallel implementation for signal co-simulation is presented by Friedrich [56].

The novelty in this work is the design of a general purpose co-simulation algorithm (IJCSA) for field-signal interaction. The following bullet points summarize the properties of the proposed algorithm.

The IJCSA

- allows a general stable treatment of tightly coupled problems
- handles an arbitrary number of subsystems
- can incorporate (non)linear interface constraints at the interface level
- allows a parallel execution of all the subsystems
- maintains the full flexibility and modularity of classical co-simulation
- can handle algebraic loops

This thesis is outlined as follows:

- *Chapter 2:* Mathematical and Algorithmic Framework
  Here the operator notation of co-simulation is introduced. Furthermore, an overview of solution methods for nonlinear equation systems is given, where Jacobian-based and Jacobian-free techniques are discussed. Last but not least essential definitions of numerical analysis are given as they are used in the subsequent chapters.

- *Chapter 3:* Co-Simulation
  This chapter discusses co-simulation. Thereafter, the link between the monolithic approach and the classical co-simulation approach is shown. For the co-simulation the effect of the combination of different time integrators is discussed. Moreover,

different communication patterns are presented (Gauss-Seidel and Jacobi) and their usability and performance is discussed. Towards the end of the chapter the impact of the different decompositions on the stability is presented. Finally, the block diagram representation of co-simulation is shown.

- *Chapter 4:* Interface Jacobian-based Co-Simulation Algorithm
  The idea of the IJCSA is presented, usability and performance enhancements for the IJCSA are shown. Moreover, the extraction procedure for interface Jacobians is demonstrated. Last but not least, various benchmark cases are performed and different versions of IJCSA are compared to the classical co-simulation algorithms.

- *Chapter 5:* Application Examples
  The performance of the IJCSA for industrially relevant problems is the center point of this chapter. The first problem is a classical benchmark for FSI, the so-called FSI3 benchmark proposed by Turek et al. [148]. Afterwards, a fully coupled fluid-structure-signal interaction with closed-loop control is shown. Finally, the emergency brake maneuver of a wind turbine is presented. Here the interaction of the generator/gearbox, flexible composite blades, control unit and the three-dimensional flow field is taken into account. The simulation results are validated against measurement data from the National Renewable Energy Laboratory (NREL) Unsteady Aerodynamics Experiment Phase VI performed in the NASA AMES wind tunnel.

- *Chapter 6:* Conclusion and Outlook
  The results of the various studies performed are summarized and discussed. Moreover,the properties of the IJCSA are summarized and ideas for future research are proposed.

Any intelligent fool can make things bigger, more complex, and more violent. It takes a touch of genius – and a lot of courage – to move in the opposite direction.

Albert Einstein

CHAPTER 2

MATHEMATICAL AND ALGORITHMIC FRAMEWORK

The chapter introduces the operator notation used throughout this work.

As the solution of coupled problems usually necessitates the solution of a (non)linear interface constraint equation system, an overview of (non)linear solution methods is given in this chapter. Most methods for solving nonlinear equation systems can be sorted into two main categories. The first category is the one, where no derivative information of the residual is necessary. The second category needs at least information about the first derivative (Jacobian) of the function (residual). In the following a few members of each category are presented with the focus on the applicability to co-simulation.

Last but not least basic elements of numerical analysis are recapped as they are needed in Chapters 3 and 4.

2.1 Notation for Co-Simulation of Multiple Subsystems

The notation is introduced on the basis of the well-known two-field coupling methodology. It is generalized to a multi-code scenario afterwards.

We start with an example where we have two subsystems, each equipped with one input and one output scalar quantity. Note that there are various synonyms used for subsystem, e.g. client, code, simulator, solver, agent.

The input/output relations for the problem are given by

$$Y_1 = \mathcal{S}_1\left(U_1\right), \tag{2.1}$$

$$Y_2 = \mathcal{S}_2\left(U_2\right). \tag{2.2}$$

Each of the subsystems $\mathcal{S}_1\left(U_1\right)$ and $\mathcal{S}_2\left(U_2\right)$ have state (internal) variables in addition to the output quantities Y_1 and Y_2. The state variables are referred to as X_1 and X_2. The subsystems $\mathcal{S}_1\left(U_1\right)$ and $\mathcal{S}_2\left(U_2\right)$ are assumed to model a (non)linear relation between output $Y_\square$ and input $U_\square$ quantities, i.e. $\mathcal{S}_\square$ is in general a nonlinear operator. The coupled problem is defined by the interface constraint equations

$$\mathcal{I}_1\left(Y_1, Y_2, U_1, U_2\right) = 0, \tag{2.3}$$

$$\mathcal{I}_2\left(Y_1, Y_2, U_1, U_2\right) = 0. \tag{2.4}$$

Here $\mathcal{I}_\square$ is the interface constraint operator which can also be nonlinear in general. Note that the state variables are not needed within the interface constraint equations. The interface constraint operator is essential to the co-simulation, as it defines the coupled problem. It reflects the relations between input and output variables. The output variables can be replaced by the subsystems. Hence we arrive at

$$\mathcal{I}_1\left(\mathcal{S}_1\left(U_1\right), \mathcal{S}_2\left(U_2\right), U_1, U_2\right) = 0, \tag{2.5}$$

$$\mathcal{I}_2\left(\mathcal{S}_1\left(U_1\right), \mathcal{S}_2\left(U_2\right), U_1, U_2\right) = 0. \tag{2.6}$$

In order to be able to derive coupling algorithms for multiple subsystems a generalized notation is introduced. If an arbitrary number $\{r \in \mathbb{N} \mid r \geq 2\}$ of subsystems is present, the output-input relation for the i-th subsystem is defined as

$$\boldsymbol{Y}_i = \mathcal{S}_i\left(\boldsymbol{U}_i\right) \quad i = 1, ..., r. \tag{2.7}$$

Note here the input and output quantities are generalized to vectors. The interface constraint operator for the i-th subsystem is defined as

$$\mathcal{I}_i\left(\boldsymbol{Y}_j, \boldsymbol{U}_j, j=1,...,r\right) \quad i=1,...,r. \tag{2.8}$$

Using Equation (2.7) one has

$$\mathcal{I}_i\left(\mathcal{S}_j\left(\boldsymbol{U}_j\right), \boldsymbol{U}_j, j=1,...,r\right) \quad i=1,...,r. \tag{2.9}$$

Hence the interface constraint equation system is given by

$$\mathcal{I}_i\left(\mathcal{S}_j\left(\boldsymbol{U}_j\right), \boldsymbol{U}_j, j=1,...,r\right)=0 \quad i=1,...,r. \tag{2.10}$$

The residual components of the residual vector are given by

$$\boldsymbol{\mathcal{R}}_i=\mathcal{I}_i\left(\mathcal{S}_j\left(\boldsymbol{U}_j\right), \boldsymbol{U}_j, j=1,...,r\right) \quad i=1,...,r. \tag{2.11}$$

The global residual vector components $\boldsymbol{\mathcal{R}}_i$ define the entire co-simulation and thus the coupled problem. Furthermore, in order to solve the coupled problem the global interface residual vector

$$\boldsymbol{r}=\begin{bmatrix}\boldsymbol{\mathcal{R}}_1\\ \vdots\\ \boldsymbol{\mathcal{R}}_r\end{bmatrix} \tag{2.12}$$

needs to be minimized until a certain convergence criterion is met. In other words it is a (non)linear root finding problem that needs to be dealt with. In the next section root finding methods are reviewed.

2.2 Fixed-Point Iteration

The classical fixed-point iteration method belongs to the category of Jacobian-free methods. They rely on the residual evaluation only. Hence, a solution of all the subsystems is sufficient. This is a major advantage in contrast to Newton methods where the Jacobian is needed. However, this advantage may have the drawback of stability loss. This is discussed in detail in Chapter 3.

In order to derive the fixed-point iteration method the definition of a contraction (Definition 2.1) and the Banach fixed-point theorem (see Theorem 2.1) are a prerequisite.

Definition 2.1: Contraction

Let X be a metric space. Then a map $T : X \to X$ is called a contraction (contraction mapping) on X if there exists $q \in [0,1]$ such that

$$\left\| T(x) - T(y) \right\| \leq q \left\| x - y \right\|$$

$\forall x, y \in X$. Note q is called Lipschitz constant (see Schwarz et al. [132]).

Theorem 2.1: Banach Fixed-Point Theorem

Let $(X, \|.\|)$ be a non-empty complete metric space with a contraction $T : X \to X$. Then the equation

$$x = T(x)$$

has one solution x^*, called fixed-point and the fixed-point iteration

$$^{m+1}x = T\left(^{m}x\right) \quad m = 0, 1, 2, ...$$

converges for any $^{0}x \in X$ to x^* for $m \to \infty$.

In co-simulation the main task is to find the root(s) of the global interface residual vector equation (see Equation (2.12)). If a classical fixed-point iteration is deployed to solve the interface residual equation system the first task is to formulate a fixed-point problem. In the following it is shown how the root finding problem can be transformed into a fixed-point problem. For the derivation the root finding problem is defined by the map $F : X \to X$, where X is a metric space. Hence, the root finding problem is defined as

$$F(x) = 0. \tag{2.13}$$

Theorem 2.1 defines the fixed-point of T as

$$x^* = T\left(x^*\right). \tag{2.14}$$

If we assume that x^* is the root of $F(x)$ we have

$$F\left(x^*\right) = 0. \tag{2.15}$$

With this assumption Equation (2.14) is equivalent to

$$x^* + F\left(x^*\right) = T\left(x^*\right), \tag{2.16}$$

$$x^* - F\left(x^*\right) = T\left(x^*\right). \tag{2.17}$$

Hence, we can define the fixed-point iteration sequence for the root finding problem Equation (2.13) as

$${}^{m+1}x = {}^{m}x \pm F\left({}^{m}x\right). \tag{2.18}$$

Furthermore, the error ${}^{m+1}e_{\text{fixP}}$ can be defined by

$${}^{m+1}e_{\text{fixP}} = \left\| {}^{m+1}x - x^* \right\|. \tag{2.19}$$

For the discussion of stability properties it is favorable to rewrite Equation (2.18) into matrix form. This is only possible as long as $F(x)$ is a linear map. However, for stability considerations this is a common assumption. With this we arrive at

$${}^{m+1}\boldsymbol{x} = (\boldsymbol{I} \mp \boldsymbol{G})\,{}^{m}\boldsymbol{x} \pm \boldsymbol{c}, \tag{2.20}$$

where $F(x)$ is assumed to be

$$F(x) = \boldsymbol{G}\boldsymbol{x} - \boldsymbol{c}, \tag{2.21}$$

here $\boldsymbol{G}$ is a linear operator and $\boldsymbol{c}$ is a constant vector which represents the constant part of $T(x)$. With this, Equation (2.20) can be further simplified to

$${}^{m+1}\boldsymbol{x} = \boldsymbol{H}\,{}^{m}\boldsymbol{x} + \boldsymbol{c}. \tag{2.22}$$

Here $\boldsymbol{c}$ represents the constant part of $T(x)$. $\boldsymbol{H}$ is of special interest if convergence properties of the constructed fixed-point iteration sequence are analyzed. In particular the spectral radius of $\boldsymbol{H}$ is determining the stability. The spectral radius is defined by

$$\rho(\boldsymbol{H}) = \max_i\left(|\lambda_i|\right) \tag{2.23}$$

where λ_i are the eigenvalues of $\boldsymbol{H}$. The fixed-point iteration method converges (see Hanke-Bourgeois [70] and Strang [143]) if

$$\rho(\boldsymbol{H}) < 1. \tag{2.24}$$

With the help of the spectral radius the convergence rate for Equation (2.22) can also be determined. The convergence rate κ is given by

$$\kappa = -\log_{10}\left(\rho\left(\boldsymbol{H}\right)\right). \tag{2.25}$$

This means that the error ${}^{m+1}e_{\text{fixP}}$ (2.19) drops by κ digits per iteration.

2.2.1 Constant Under-Relaxation

In order to improve the convergence behavior of Equation (2.22) the relaxation factor $\{\alpha \in \mathbb{R}\}$ is introduced. Thus, Equation (2.18) becomes

$${}^{m+1}x = {}^{m}x + \alpha F\left({}^{m}x\right). \tag{2.26}$$

Here the possibility of a plus or a minus sign is valid. Equation (2.26) can be written in matrix form as-well. This leads to

$${}^{m+1}\boldsymbol{x} = \left(\boldsymbol{I} + \alpha \boldsymbol{G}\right){}^{m}\boldsymbol{x} - \alpha \boldsymbol{c}. \tag{2.27}$$

This can be further simplified to

$${}^{m+1}\boldsymbol{x} = \boldsymbol{H}\,{}^{m}\boldsymbol{x} - \alpha \boldsymbol{c}. \tag{2.28}$$

The latter equation is similar to Equation (2.22) as for convergence

$$\rho\left(\boldsymbol{H}\right) = \max_{i}\left(|\lambda_i|\right) \tag{2.29}$$

must hold. Note that Equation (2.27) states the core equation of the Richardson iterative method, which is discussed in detail in Hanke-Bourgeois [70] and Richardson [124]. The effect of the relaxation factor, which is a user input, is discussed in detail in Section 3.4. In general it is challenging to determine a good relaxation factor value a-priori. However, for some special cases there is helpful a-priori knowledge available as shown in the following.

Optimal Relaxation Factor for Scalar Case

The optimal relaxation factor renders

$$\rho\left(\boldsymbol{H}\right) = 0. \tag{2.30}$$

This will render the correct solution within one iteration.

Unfortunately, it is in general not possible to construct such an optimal relaxation factor. However, for the special case, where $\boldsymbol{x}$ is a scalar and $\boldsymbol{G}$ is a linear map it is possible.

In order to derive the optimal relaxation factor α_{opt} we start with the scalar form of Equation (2.27), which is

$$^{m+1}x = (1+\alpha G)\,^{m}x - \alpha c. \tag{2.31}$$

For optimal convergence we need

$$\rho(H) = 0 = 1 + \alpha_{\text{opt}} G. \tag{2.32}$$

This leads to the optimal relaxation factor

$$\alpha_{\text{opt}} = -\frac{1}{G}. \tag{2.33}$$

Optimal Relaxation Factor for Vector Case

For the case that $\boldsymbol{x}$ is a vector and $\boldsymbol{G}$ is symmetric and positive-definite the following expression guarantees an optimal choice of α such that

$$\min \rho(\boldsymbol{H}). \tag{2.34}$$

Hence, it is in general not possible to get

$$\rho(\boldsymbol{H}) = 0. \tag{2.35}$$

This means we need to minimize

$$\min \rho(\boldsymbol{I} + \alpha \boldsymbol{G}). \tag{2.36}$$

If λ_i are the eigenvalues of $\boldsymbol{G}$ this is equivalent to

$$\min |1 + \alpha \lambda_i| \quad \forall \lambda_i. \tag{2.37}$$

Furthermore, for guaranteeing convergence we need to have

$$|1 + \alpha \lambda_i| < 1 \quad \forall \lambda_i. \tag{2.38}$$

Thus we need to choose α such that

$$0 < \alpha < -\frac{2}{\lambda_{\max}}. \tag{2.39}$$

Therefore, in order to minimize Equation (2.37) we have

$$\alpha_{\text{opt}} = -\frac{2}{\lambda_{\min} + \lambda_{\max}}, \tag{2.40}$$

where

$$\lambda_{\min} = \min_i \left(\lambda_i\right) \tag{2.41}$$

and

$$\lambda_{\max} = \max_i \left(\lambda_i\right). \tag{2.42}$$

The proof for Equation (2.40) is available in Hanke-Bourgeois [70]. Note that Equation (2.40) represents the basic idea of the Chebyshev iteration method as shown in Gutknecht et al. [67].

2.2.2 Aitken Acceleration

For nonlinear problems ($\boldsymbol{G}$ is no longer linear) it is even harder to estimate an optimal relaxation factor as the optimal value for the relaxation will change during the simulation. Therefore, adaptive techniques were developed to update the relaxation factor during the iterations. One of the most prominent and for practical applications most useful methods, is discussed in the following.

The Aitken acceleration method is named after Alexander Aitken [1]. It has been reformulated for an efficient computer implementation by Irons et al. [80].

The main idea of the method is to assume a linear map $T : X \rightarrow X$. Furthermore a scalar problem is assumed $X \subseteq \mathbb{R}$. For two initial guesses $a, b \in X$ we get $\widetilde{a} = T(a), \widetilde{b} = T(b) \in X$. Based on these two points we can compute the solution c of the fixed point problem if $T : X \rightarrow X$ is a linear map (see Figure 2.1) and we arrive at

$$c = \frac{a\widetilde{b} - \widetilde{a}b}{a - b - \widetilde{a} + \widetilde{b}}. \tag{2.43}$$

In Figure 2.1 the effect of the assumption, that $T(x)$ is a linear map is demonstrated. The linearization assumption results in the blue line. If $T(x)$ is nonlinear, c will not be the fixed point x^*. Hence, more iterations need to be carried out. As a relaxation factor form of Equation (2.43) can be integrated smoothly into Equation (2.26) the relaxation factor form is derived next. In order to derive the relaxation

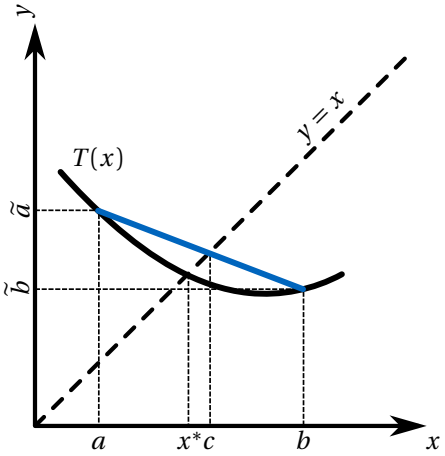


Figure 2.1: Graphical interpretation of Aitken

factor form of Equation (2.43) an iteration depended relaxation factor ${}^{b}\beta$ is defined by rewriting Equation (2.43) to

$$c = \tilde{b} + {}^{b}\beta\left(b - \tilde{b}\right), \tag{2.44}$$

$${}^{b}\beta = \frac{\tilde{b} - \tilde{a}}{a - b - \tilde{a} + \tilde{b}}. \tag{2.45}$$

For the first increment the relaxation factor ${}^{a}\beta$ needs to be given

$$b = \tilde{a} + {}^{a}\beta\left(a - \tilde{a}\right), \tag{2.46}$$

$${}^{a}\beta = {}^{\text{init}}\beta. \tag{2.47}$$

If we combine Equation (2.45) and Equation (2.46) we arrive at a recursive formula for the relaxation factor

$${}^{b}\beta = {}^{a}\beta + \left({}^{a}\beta - 1\right)\frac{b - \tilde{b}}{\left(a - \tilde{a}\right) - \left(b - \tilde{b}\right)}. \tag{2.48}$$

The notation from Irons et al. [80] is in the following converted to the notation used in this work. Equation (2.46) is equivalent to

$$b = \left(1 - {}^{a}\beta\right)T(a) + {}^{a}\beta\, a. \tag{2.49}$$

If Equation (2.16) is substituted into Equation (2.46) we arrive at

$$b = a + \left(1 - {}^{a}\beta\right) F(a). \tag{2.50}$$

In case that this equation is compared with Equation (2.26) we can identify

$$\alpha = 1 - \beta. \tag{2.51}$$

Thus Equation (2.49) can be written as

$${}^{m+1}x = {}^{m}x + \underbrace{{}^{m}\alpha}_{(1-\beta)}\left(T\left({}^{m}x\right) - {}^{m}x\right). \tag{2.52}$$

Furthermore, we can reformulate Equation (2.48) to

$${}^{m}\alpha = {}^{m-1}\alpha \frac{T\left({}^{m-1}x\right) - {}^{m-1}x}{\left(T\left({}^{m-1}x\right) - {}^{m-1}x\right) + \left({}^{m}x - T\left({}^{m}x\right)t\right)}. \tag{2.53}$$

For $T(x) = x + F(x)$ Equation (2.52) reads

$${}^{m+1}x = {}^{m}x + {}^{m}\alpha F\left({}^{m}x\right), \tag{2.54}$$

and Equation (2.53) reads

$${}^{m}\alpha = {}^{m-1}\alpha \frac{F\left({}^{m-1}x\right)}{F\left({}^{m-1}x\right) - F\left({}^{m}x\right)}. \tag{2.55}$$

Equation (2.55) represents an iteration dependent relaxation factor which is depending on previous and current evaluation of $F(x)$ only. Hence Equations (2.54) and (2.55) represent a convenient form for implementation.

However, there is one problem left in the case where $X \subseteq \mathbb{R}^n$ for $n > 1$, the division in Equation (2.55) is not possible. Therefore different versions of the Aitken Δ^2 method exist for the vector case. A comparison can be found in Macleod [97]. Within Macleod [97] the Aitken acceleration method is defined by

$$x'_{i+2} = x_{i+2} - \frac{\left(x_{i+2} - x_{i+1}\right)^2}{x_{i+2} - 2x_{i+1} + x_i}. \tag{2.56}$$

In order to compare the work of Macleod [97] with the one of Irons et al. [80] it is necessary to know how $x_i, x_{i+1}, x_{i+2}, x_{i+2}^{'}$ of Macleod [97] are connected to $a, b, \widetilde{a}, \widetilde{b}, c$ of Irons et al. [80]. With

$$\begin{aligned} x_i &= a, \\ x_{i+1} &= \widetilde{a}, \\ x_{i+1} &= b, \\ x_{i+2} &= \widetilde{b}, \\ x_{i+2}^{'} &= c, \end{aligned}$$

the notation used in Macleod [97] can be transformed to the one used in Irons et al. [80]. Thus Equation (2.56) can be converted to Equation (2.43).

One of the most prominent methods for the vector case (see Küttler et al. [89]) is the multiplication by the denominator of Equation (2.53) which results in

$$^{m}\alpha = {}^{m-1}\alpha \frac{\left(T\left({}^{m-1}\boldsymbol{x}\right) - {}^{m-1}\boldsymbol{x}\right)^{\top}\left(\left(T\left({}^{m-1}\boldsymbol{x}\right) - {}^{m-1}\boldsymbol{x}\right) + \left({}^{m}\boldsymbol{x} - T\left({}^{m}\boldsymbol{x}\right)\right)\right)}{\left\|\left(T\left({}^{m-1}\boldsymbol{x}\right) - {}^{m-1}\boldsymbol{x}\right) + \left({}^{m}\boldsymbol{x} - T\left({}^{m}\boldsymbol{x}\right)\right)\right\|^2}. \tag{2.57}$$

For a two code Gauss-Seidel communication pattern (see Section 3.3) the Aitken method is given in Algorithm 2.1. This version of the algorithm is also used in the context of fluid-structure interaction (see Küttler et al. [89]).

The Aitken method works well for cases where the global residual vector holds degrees of freedom which represent the same physical quantity (e.g. displacements for fluid-structure interaction as presented in Küttler et al. [89]). One advantage of the Aitken method is that it is simple to implement and produces little computational effort at the interface even if the global residual vector has a large number of entries. However, the convergence rate of the Aitken method is linear (proof available in Henrici [72, p. 71]) it converges significantly faster than the constant under-relaxation method as shown by Küttler et al. [89]. Figure 2.1 illustrates that the Aitken method will give the correct result after the second iteration if the problem is linear.

Algorithm 2.1: Aitken algorithm for a 2-code example with GS-pattern

```
    //  Time loop
1   for n = 0 to n = n_end do
        //  Iteration loop
2       for m = 0 to m = m_end do
            //  Solve all subsystems (sequential)
3           ^m Y_2^{n+1} = S_2( ^m U_2^n )
4           ^m Y_1^{n+1} = S_1( ^m Y_2^{n+1} )
            //  Compute and check residual
5           ^m r^n = [ ^m R_1^n ; ^m R_2^n ] = [ ^m U_1^n - ^m Y_2^{n+1} = 0 ; ^m U_2^n - ^m Y_1^{n+1} ]
6           if ||^m r^n||_ε < ε then
7               break
8           if m = 0 then
9               ^0 α^n = ^init α^n
10          else
                //  For a scalar residual
11              ^m α^n = ^{m-1} α^n ( ^{m-1} R_2^n ) / ( ^{m-1} R_2^n - ^m R_2^n )
                //  For a vector residual
12              ^m α = ^{m-1} α^n ( ^{m-1} R_2^n^T ( ^{m-1} R_2^n - ^m R_2^n ) ) / || ^{m-1} R_2^n - ^m R_2^n ||^2
            //  Apply update
13          ^{m+1} U_2^n = ^m U_2^n + ^m α^n  ^m R_2^n
        //  Initial solution for next time step
14      ^0 U_2^{n+1} = E( ^{m_end+1} U_2^k   k = 0, ..., n )
```

One of the severe disadvantages for multi-code coupling of the Aitken method is, that it has problems delivering a converged solution if the entries in the global residual vector are stemming from different physical quantities (e.g. forces and displacements) as shown by Sicklinger et al. [134].

2.3 Newton Methods

In this section a few variants of the famous Newton method are presented. Newton methods need in contrast to the classical fixed-point iteration method information of the first derivative of $F(x)$. The iteration sequence of the Newton method is given by

$$^{m+1}x = {}^{m}x - \mathcal{J}\left(F\left({}^{m}x\right)\right)^{-1} F\left({}^{m}x\right), \tag{2.58}$$

where $\mathcal{J}\left(F(x)\right)$ is the first derivative of $F(x)$. Equation (2.58) can be reformulated to

$$\mathcal{J}\left(F\left({}^{m}x\right)\right){}^{m}\Delta x = -F\left({}^{m}x\right), \tag{2.59}$$

with

$${}^{m}\Delta x = {}^{m+1}x - {}^{m}x. \tag{2.60}$$

The ordinary Newton method is summarized in algorithmic form in Algorithm 2.2. If Equation (2.58) is compared to the Banach fixed-point theorem (see Theorem 2.1) it is evident that the Newton method is also within the family of fixed point iterations method. If T is defined as

$$T\left({}^{m}x\right) = {}^{m}x - \mathcal{J}\left(F\left({}^{m}x\right)\right)^{-1} F\left({}^{m}x\right), \tag{2.61}$$

the Newton method can be written as

$$^{m+1}x = T\left({}^{m}x\right). \tag{2.62}$$

One could ask the question why to use Newton methods as the Jacobian information makes the method more bulky. There are mainly two arguments to use the Newton method. The most important one is the advantage in terms of stability (see Chapter 3). The second one is a higher convergence rate. If ${}^{0}x$ is sufficiently close to x^* it is possible to show (proof available in Deuflhard [36, p. 49]), that the error is reduced with a quadratic convergence rate. This means that

$$\left\| {}^{m+1}x - {}^{m}x \right\| \leq p \left\| {}^{m}x - {}^{m-1}x \right\|^2 \tag{2.63}$$

is true. Here p is some constant.

Algorithm 2.2: Ordinary Newton method for vector case

```
   //  Initialize
1  ⁰x = ^init x
   //  Iteration loop
2  for m = 0 to m = m_end do
   |   //  Evaluate and check residual
3  |   ᵐr = F(ᵐx)
4  |   if ||ᵐr||_ε < ε then
5  |   |   break
   |   //  Evaluate Jacobian
6  |   ᵐJ = 𝒥(F(ᵐx))
   |   //  Solve for corrector
7  |   ᵐJ · ᵐΔx = −ᵐr
   |   //  Apply update
8  |   ᵐ⁺¹x = ᵐx + ᵐΔx
```

A similar theorem holds for the convergence rate of the residuals (see Deuflhard [36, p. 77])

$$\left\| F\left({}^{m+1}x\right) \right\| \leq \widetilde{p} \left\| F\left({}^{m}x\right) \right\|^2 . \tag{2.64}$$

Intuitively it means that the number of correct digits at least doubles for every iteration.

A few variants of Newton methods with respect to applications to co-simulation should be discussed in following. The following definitions are an excerpt of Deuflhard [36] which is an excellent reference for Newton methods.

- Ordinary Newton method
  The basic Newton method according to Equation (2.59) which shows a quadratic convergence rate.
- Simplified or Modified Newton method
  The initial derivative is kept throughout the iterations. This results in a general linear convergence rate.
- Newton-like method
  The finite dimensional Jacobian is replaced by some approxi-

mation of the Jacobian. This generally results in a linear convergence rate and might render a smaller convergence radius.

- Exact Newton method
  The resulting linear equation system is solved in a numerically exact manner i.e. by using a direct solver and proper scaling.

- Inexact Newton method
  The resulting linear equation system is not solved in a numerically exact manner. This maybe due to the use of an iterative solver.

- Secant method
  For scalar problems the tangent (Jacobian) is replaced by the secant. It works very well for scalar problems where the Jacobian is difficult to compute and it shows a superlinear convergence rate.

- Quasi-Newton method
  Extends the secant method to finite dimensions (Jacobian rank update)

With these definitions the classical fixed-point iteration methods (see Section 2.2) can be seen as a subclass of the Newton method. They belong in the category of Newton-like methods.

If the Jacobian is replaced by some approximation it harms in general the convergence rate. For most cases this results in a linear convergence rate.

Example 2.1 should be used to illustrate the difference between quadratic, superlinear and linear convergence rate. Within Example 2.1 the ordinary, the quasi Newton and the modified Newton method are compared.

Example 2.1: Newton Example

The following example should demonstrate the convergence behaviors of different variants of the Newton method. The residual vector of the example is given by

$$F(x_1, x_2) = \begin{bmatrix} \sin(x_1) - \cos(x_2) \\ \cos(x_1) - \sin(x_2) \end{bmatrix}. \tag{2.65}$$

The Jacobian of the residual vector is a two by two matrix and is given by

$$\mathcal{J}\left(F\left(x_1,x_2\right)\right)=\begin{bmatrix}\cos\left(x_1\right) & \sin\left(x_2\right)\\ -\sin\left(x_1\right) & -\cos\left(x_2\right)\end{bmatrix}. \tag{2.66}$$

The convergence criteria is set to $\left\|{}^m\boldsymbol{r}\right\| < 1\cdot 10^{-6}$ for all cases.
The results for the ordinary Newton method can be found in Table 2.1. The results are aligned with theory and show a quadratic convergence rate.
If the initial Jacobian is kept constant throughout the iterations the convergence rate drops from quadratic to linear (see Table 2.2). The linear convergence rate can be improved by using a quasi Newton method to superlinear (see Table 2.3 and Table 2.4).

Table 2.1: Behavior of the ordinary Newton method for Example 2.1

iteration	$\left\|F\left({}^m x\right)\right\|$	$\left\|{}^m\Delta x\right\|$	${}^m e_{\text{fixP}}$
0	1.414213562373095 1	1.414213562373095 1	0.303492827833503 6
1	0.425916830318592 3	0.308239298872401 4	0.004746471038897 9
2	0.006712511144309	0.004746488861175 9	0.000000017822278 0
3	0.000000025204507 2	0.000000017822278 0	0.000000000000000 2

2.3.1 Quasi Newton Method

The convergence rate of the modified Newton method can be improved to superlinear by using the famous Broyden's 'good' rank-1 update as shown by Broyden et al. [23]. This algorithm can be found in an implementation-friendly form in Deuflhard [36, p. 62]. The algorithm is based on the idea of the secant method, where the Jacobian is approximated by

$${}^m\widetilde{J}\,{}^m\Delta x = F\left({}^{m+1}x\right) - F\left({}^m x\right). \tag{2.67}$$

Here ${}^m\widetilde{J}$ is the secant approximation of the Jacobian in iteration m. Broyden et al. [23] transferred this idea from the scalar case the multidimensional vector case by using the following update rule:

$${}^m\widetilde{J} = {}^{m-1}\widetilde{J} + \frac{F\left({}^m x\right) - F\left({}^{m-1}x\right) - {}^{m-1}\widetilde{J}\,{}^{m-1}\Delta x}{\left\|{}^{m-1}\Delta x\right\|^2}\,{}^{m-1}\Delta x^{\top} \tag{2.68}$$

Similar to Equation (2.58) we need to solve

$${}^{m+1}x = {}^m x - {}^m\widetilde{J}^{-1}F\left({}^m x\right), \tag{2.69}$$

Table 2.2: Behavior of the modified Newton method for Example 2.1

iteration	$\left\|F({}^m x)\right\|$	$\left\|{}^m x \Delta\right\|$	${}^m e_{\text{fixP}}$
0	1.414213562373095 1	1.414213562373095 1	0.303492827833503 6
1	0.425916830318592 3	0.425916830318592 3	0.122424002485088 8
2	0.172917526956656 4	0.172917526956656 4	0.050493524471567 7
3	0.071393456157483 4	0.071393456157483 4	0.020899931685915 7
4	0.029555890963451 6	0.029555890963451 6	0.008655959277535 9
5	0.012241298573054 8	0.012241298573054 8	0.003585339295518 9
6	0.005070430025874 4	0.005070430025874 4	0.001485090730355 5
7	0.002100235066218 5	0.002100235066218 5	0.000615144335863 0
8	0.000869945435162 0	0.000869945435162 0	0.000254801099290
9	0.000360343168386 6	0.000360343168386 6	0.000105542069087 6
10	0.000149259025366 0	0.000149259025366 0	0.000043716956278 5
11	0.000061825112464 8	0.000061825112464 8	0.000018108156186 3
12	0.000025608800067 6	0.000025608800067 6	0.000007500643881 3
13	0.000010607512303 4	0.000010607512303 4	0.000003106868422 1
14	0.000004393775459 0	0.000004393775459 0	0.000001286907036 9
15	0.000001819961385 0	0.000001819961385 0	0.000000533054348 1
16	0.000000753852688 6	0.000000753852688 6	0.000000220798340 5

Table 2.3: Behavior of the quasi Newton method (Broyden's rank 1 update) for Example 2.1 where ${}^0\widetilde{\boldsymbol{J}} = {}^0\boldsymbol{J}$

iteration	$\left\|F({}^m x)\right\|$	$\left\|{}^m \Delta x\right\|$	${}^m e_{\text{fixP}}$
0	1.414213562373095 1	1.414213562373095 1	0.303492827833503 6
1	0.425916830318592 3	0.327334062994542 8	0.023841235161039 2
2	0.033715001075671 5	0.024010670132413 9	0.000169434971374 6
3	0.000239617233885 1	0.000169442940232 9	0.000000007968858 3
4	0.000000011269667 6	0.000000007968858 4	0.000000000000000 2

in order to proceed to the next Newton iteration. Hence, Broyden et al. [23] further suggested to use the Sherman–Morrison formula

Table 2.4: Behavior of the quasi Newton method (Broyden's rank 1 update) for Example 2.1 where ${}^0\widetilde{\boldsymbol{J}} = \boldsymbol{I}$

iteration	$\left\|F({}^m x)\right\|$	$\left\|{}^m \Delta x\right\|$	${}^m e_{\text{fixP}}$
0	1.414213562373095 1	1.414213562373095 1	3.367919300532252 5
1	1.414213562373095 1	3.076399077971968 8	1.388527951665040 8
2	1.003648926252681 1	0.986173563597888 7	0.936376679636708 3
3	1.204966549794268 1	1.270893539140174 8	0.344112368779571 5
4	0.481020255548720 0	0.346211156198226 5	0.002100010463912 8
5	0.002956781995747 3	0.002105412488969 7	0.000005402025347 7
6	0.000007606205141 3	0.000005402407704 5	0.000000000382357 0
7	0.000000000538369 5	0.000000000382356 9	0.000000000000001 0

(see Strang [143]) to directly update the inverse of the Jacobian approximation ${}^m\widetilde{J}^{-1}$, which results in

$$ {}^m\widetilde{J}^{-1} = {}^{m-1}\widetilde{J}^{-1} + \frac{{}^{m-1}\Delta x - {}^{m-1}\widetilde{J}^{-1}\left(F({}^m x) - F\left({}^{m-1}x\right)\right)}{{}^{m-1}\Delta x^{\top}\,{}^m\widetilde{J}^{-1}\left(F({}^m x) - F({}^{m-1}x)\right)} \left({}^{m-1}\Delta x^{\top}\,{}^{m-1}\widetilde{J}^{-1}\right). \quad (2.70) $$

A comprehensive presentation of Broyden's 'good' rank-1 update is given in Algorithm 2.3.

The performance of Algorithm 2.3 is demonstrated with the help of Example 2.1 in Table 2.3 and Table 2.4. Furthermore it is demonstrated that the influence of the initial choice of the global interface Jacobian can have a significant impact on the convergence. For a correct initial Jacobian ${}^0\widetilde{\boldsymbol{J}} = {}^0\boldsymbol{J}$ Example 2.1 needs 5 iterations for convergence (see Table 2.3). Whereas if the initial Jacobian is set to be the identity matrix ${}^0\widetilde{\boldsymbol{J}} = \boldsymbol{I}$ Example 2.1 needs 8 iterations for convergence (see Table 2.4).

2.3.2 Newton-Krylov Methods

If the Newton method is combined with a Krylov subspace method for the solution of the linear equation system this is called Newton-Krylov method. Because of the usage of an iterative Krylov linear equation solver Newton-Krylov methods are also in the category of inexact Newton methods. If Quasi-Newton methods or Newton-like methods are combined with Krylov subspace methods it is possible to design

Algorithm 2.3: Broyden's 'good' rank-1 update

// Initialize
${}^0\boldsymbol{x} = {}^{\text{init}}\boldsymbol{x}$
${}^0\widetilde{\boldsymbol{J}} = {}^{\text{init}}\widetilde{\boldsymbol{J}}$
// Iteration loop
for $m = 0$ *to* $m = m_{\text{end}}$ **do**
  // Evaluate and check residual
  ${}^m\boldsymbol{r} = F({}^m\boldsymbol{x})$
  if $\|{}^m\boldsymbol{r}\|_\epsilon < \epsilon$ **then**
    break
  // Rank-1 update of approximative Jacobian
  if $m \neq 0$ **then**
    $${}^m\widetilde{\boldsymbol{J}} = {}^{m-1}\widetilde{\boldsymbol{J}} + \frac{{}^m\boldsymbol{r} - {}^{m-1}\boldsymbol{r} - {}^{m-1}\widetilde{\boldsymbol{J}}\,{}^{m-1}\Delta\boldsymbol{x}}{\|{}^{m-1}\Delta\boldsymbol{x}\|^2}\,{}^{m-1}\Delta\boldsymbol{x}^\top$$
  // Solve for corrector
  ${}^m\widetilde{\boldsymbol{J}} \cdot {}^m\Delta\boldsymbol{x} = -{}^m\boldsymbol{r}$
  // Apply update
  ${}^{m+1}\boldsymbol{x} = {}^m\boldsymbol{x} + {}^m\Delta\boldsymbol{x}$

Jacobian-free Newton-Krylov (JFNK) methods (see Brown et al. [22]). For further information and a good overview of different techniques the reader is referred to Knoll et al. [88]. Despite of this excellent reference a short introduction to JFNK is given in the following.

2.3.3 Krylov Subspace Methods

Krylov subspace methods are iterative solution methods for large linear systems. They were introduced as iterative methods by Reid et al. [121].

These methods are generalized projection methods according to Saad [127] for solving $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{b} \in \mathbb{R}^n$. They are using the jth Krylov subspace

$$\mathcal{K}_j(\boldsymbol{A}, {}_0\boldsymbol{a}) = \text{span}\{{}_0\boldsymbol{a}, \boldsymbol{A}\,{}_0\boldsymbol{a}, \boldsymbol{A}^2\,{}_0\boldsymbol{a}, \ldots, \boldsymbol{A}^{j-1}\,{}_0\boldsymbol{a}\}, \tag{2.71}$$

where ${}_0\boldsymbol{a} = \boldsymbol{b} - \boldsymbol{A}_0\boldsymbol{x}$ is the initial linear residual for some given ${}_0\boldsymbol{x}$. This idea forms the whole family of Krylov subspace methods, which includes for instance conjugate gradients, Arnoldi method, Lanczos

method and the generalized minimal residual method (GMRES). The GMRES method is especially interesting as it can be used to solve large sparse nonsymmetric linear equation systems. For illustration purposes the GMRES algorithm is given in Algorithm 2.4. Please note that only the GMRES iteration index is denoted for an improved readability.

2.3.4 Jacobian-free Newton-Krylov Methods

The central idea of JFNK methods is to use a Krylov subspace method to solve the linear Equation System (2.58) which needs to be solved in every Newton iteration m of the ordinary Newton method. For instance if the GMRES method is used as linear solution methods, the kth GMRES iteration will minimize $\left\| {}_k\boldsymbol{a} \right\|_2$, where

$$ {}^m_k\boldsymbol{a} = -{}^m\boldsymbol{r} - {}^m\boldsymbol{J} \cdot {}^m_k\Delta\boldsymbol{x}, \tag{2.72} $$

in a least-squares sense (see also Knoll et al. [88]). The key feature to construct a matrix-free methods is that GMRES requires the action of the Jacobian only in the form of matrix–vector products, which may be approximated by

$$ \mathcal{J}\left(F(\boldsymbol{u})\right) \cdot \boldsymbol{v} \approx \frac{F(\boldsymbol{u} + \epsilon\boldsymbol{v}) - F(\boldsymbol{u})}{\epsilon}, \tag{2.73} $$

according to Brown et al. [22], where $\epsilon \in \mathbb{R}$ is a small number. Hence we can now formulate a basic version of the JFNK method in Algorithm 2.5. The GMRES algorithm is here indicated via a function call. A good reference for the implementation of a GMRES algorithm is pretested by Ayachour [7]. The behavior of the JFNK methods presented in Algorithm 2.5 is demonstrated with the help of Example 2.1 in Table 2.5. Please note that for each Newton iteration two GMRES iterations were performed.

The JFNK method shows a very good performance for the simple Example 2.1. For bigger problems preconditioning is required in order to achieve convergence (see Knoll et al. [88]). For very ill-conditioned problems it can be a very difficult task to find a good preconditioner (see for instance Benzi et al. [17]). The applicability of JFNK methods for the solution of simple elliptic coupled problems is discussed in Kerkhoven et al. [87]. Here it is found that if the clustering of the eigenvalues of the global interface Jacobian is at 1 preconditioning of

Algorithm 2.4: GMRES method according to Hanke-Bourgeois [70]

// Initialize
$\boldsymbol{a}_0 = -\boldsymbol{r} - \boldsymbol{J} \cdot \Delta \boldsymbol{x}_0$
$d_0 = \|\boldsymbol{a}_0\|_2$
$\boldsymbol{v}_1 = {}^{\boldsymbol{a}_0}/_{d_0}$
// GMRES Iteration loop
for $k = 1$ *to* $k = k_{\text{end}}$ **do**
  // Use Equation (2.73)
  $\boldsymbol{z}_{k+1} = \boldsymbol{J}\boldsymbol{v}_k \approx {}^{\left(F\left({}^{m}\boldsymbol{x} + \epsilon \boldsymbol{v}_k\right) - F\left({}^{m}\boldsymbol{x}\right)\right)}/_{\epsilon}$
  // Arnoldi process
  for $i = 1$ *to* $i = k$ **do**
    $h_{ik} = \boldsymbol{v}_i^{\top} \boldsymbol{z}_{k+1}$
    $\boldsymbol{z}_{k+1} = \boldsymbol{z}_{k+1} - h_{ik}\boldsymbol{v}_i$
  $w = \|\boldsymbol{z}_{k+1}\|_2$
  // Apply old Givens rotations on h_k
  for $i = 1$ *to* $i = k-1$ **do**
    $\widetilde{h} = c_i h_{ik} + s_i h_{i+1,k}$
    $h_{i+1,k} = -s_i h_{ik} + c_i h_{i+1,k}$
    $h_{ik} = \widetilde{h}$
  // Determine new Givens rotation
  if $w \leq |h_{kk}|$ **then**
    $t_k = {}^{w}/_{|h_{kk}|}$
    $c_k = {}^{h_{kk}}/_{|h_{kk}|\sqrt{1+t_k^2}}$
    $s_k = {}^{t_k}/_{\sqrt{1+t_k^2}}$
  else
    $t_k = {}^{h_{kk}}/_{w}$
    $c_k = {}^{t_k}/_{\sqrt{1+t_k^2}}$
    $s_k = {}^{1}/_{\sqrt{1+t_k^2}}$
  // Apply Givens rotation
  $h_{kk} = c_k h_{kk} + s_k w$
  // Complement d to a vector $d \in \mathbb{R}^{k+1}$
  $d_k = -s_k d_{k-1}$ // $|d_k|$ corresponds to $\|\boldsymbol{a}_k\|_2$
  $d_{k-1} = c_k d_{k-1}$
  if $|d_k| < \epsilon$ **then**
    // Trivial solve as $\boldsymbol{H}$ is an upper Hessenberg
    $\boldsymbol{H}\boldsymbol{y}_k = \boldsymbol{d}$
    $\Delta \boldsymbol{x}_k = \Delta \boldsymbol{x}_0 + \boldsymbol{V}_k \boldsymbol{y}_k$ // $V_k = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k]$
    break

Algorithm 2.5: Basic Jacobian-free Newton-Krylov method

```
    //  Initialize
1   ^0x = ^init x
    //  Iteration loop
2   for m = 0 to m = m_end do
        //  Evaluate and check residual
3       ^m r = F(^m x)
4       if ||^m r||_ε < ε then
5           break
        //  Use Algorithm 2.4 and Equation (2.73) for
        corrector
6       ^m Δx = GMRES(^m x, −^m r)
        //  Apply update
7       ^{m+1} x = ^m x + ^m Δx
```

Table 2.5: Behavior of the JFNK method for Example 2.1

iteration	$\left\|F(^m x)\right\|$	$\left\|^m \Delta x\right\|$	$^m e_{\text{fixP}}$
0	1.4142135623730951	1.4142135567740073	0.3034928222344159
1	0.4259168225819227	0.3082392919753713	0.0047464697409562
2	0.0067125092788733	0.0047464875528316	0.0000000178118777
3	0.0000000251897958	0.0000000180575680	0.0000000029776933

the GMRES method is not required for convergence. However this is rarely the case for general co-simulation scenarios.

This is the main reason why JFNK methods are difficult to use in general co-simulation scenarios, where the details of each subsystem might not be known nor accessible. Nevertheless as also illustrated by Example 2.1 and Table 2.5, the JFNK methods can be a very good choice for special kinds of co-simulation for instance fluid-structure interaction or structure-structure interaction.

2.4 Extrapolation

So far methods have been discussed which allow the solution of the interface constraint equation within the iteration loop. An open ques-

tion is still how the solution of the converged time step can be extrapolated to the next one. This is accomplished with the help of the extrapolation operator $\mathcal{E}$ (e.g. last line of Algorithm 2.1). This operator takes as input an arbitrary number of previous converged time steps of the inputs and extrapolates in time the next input ${}^{0}\boldsymbol{U}_i^{n+1}$.

For practical applications the use of previous converged time steps is rather small ($r < 3$). The simplest case of extrapolation is zero-order hold (ZOH), where the previous converged time step is taken as start solution for the next time step.

$$ {}^{0}\boldsymbol{U}_i^{n+1} = \mathcal{E}_{\mathrm{ZOH}}\left({}^{m_{\mathrm{end}}+1}\boldsymbol{U}_i^{n}\right) = {}^{m_{\mathrm{end}}+1}\boldsymbol{U}_i^{n} \tag{2.74} $$

A good compromise between storage requirements and approximation error is first-order hold (FOH). Within this method a linear extrapolator is build-up by using two previous converged time steps. This reads

$$ \begin{aligned} {}^{0}\boldsymbol{U}_i^{n+1} = \mathcal{E}_{\mathrm{FOH}}\left({}^{m_{\mathrm{end}}+1}\boldsymbol{U}_i^{n-1}, {}^{m_{\mathrm{end}}+1}\boldsymbol{U}_i^{n}\right) = \\ {}^{m_{\mathrm{end}}+1}\boldsymbol{U}_i^{n-1} + \frac{t^{n+1}-t^{n-1}}{t^{n}-t^{n-1}}\left({}^{m_{\mathrm{end}}+1}\boldsymbol{U}_i^{n} - {}^{m_{\mathrm{end}}+1}\boldsymbol{U}_i^{n-1}\right). \end{aligned} \tag{2.75} $$

If a constant time step is used throughout the entire simulation the equation above is reduced to

$$ \begin{aligned} {}^{0}\boldsymbol{U}_i^{n+1} = \mathcal{E}_{\mathrm{FOH}}\left({}^{m_{\mathrm{end}}+1}\boldsymbol{U}_i^{n-1}, {}^{m_{\mathrm{end}}+1}\boldsymbol{U}_i^{n}\right) = \\ 2\ {}^{m_{\mathrm{end}}+1}\boldsymbol{U}_i^{n} - {}^{m_{\mathrm{end}}+1}\boldsymbol{U}_i^{n-1}. \end{aligned} \tag{2.76} $$

Besides the two presented extrapolators there are a lot of different ones available in the literature. They are mostly all tuned towards the use between two subsystems, where the subsystems are limited to a special choice of time integrators. Even though they are not applicable for general co-simulation scenarios they can perform very well for the special case they are designed for.

Some prominent loosely coupled fluid-structure interaction algorithms which exploit the knowledge of the time integrators can be found in Farhat et al. [44], Felippa et al. [49], Felippa et al. [51], and Piperno et al. [114, 115]. Loosely coupled means that one does not iterate within the time step ($m_{\mathrm{end}} = 0$). Note that the operator $\mathcal{E}$ is sometimes also called predictor, this is especially true within the design of loosely coupled partitioned algorithms. In this work the

term extrapolator is used for the predicted input variables of a new time step. The term predictor is used for predicted new input values when a new interface iteration is started. Moreover, as the scope of this work is the design of general co-simulation algorithms, the design of extrapolators which are geared towards a specific choice of subsystem integrators or towards a specific number of subsystems is not the focus of this work.

2.5 Elements of Numerical Analysis

At the end of this chapter a few basic definitions of numerical analysis should be recapped as they are essential to the following chapters.

When algorithms are implemented in software and executed on a central processing unit (CPU) it is important to realize that these CPUs usually work with floating-point arithmetic. This means that the rational numbers $\mathbb{R}$ are approximated by numbers with a finite amount of digits. This has the consequence that during the calculation a round-off error needs to be taken into account. A further complication with finite precision is that multiplication and addition are no longer necessarily associative. There is an excellent overview paper by Goldberg [63] which shows the implications of floating-point arithmetic. Further references on the analysis of round-off error propagation are presented by Henrici [72, 73] and Wilkinson [155]. The reader is referred to these references as the round-off error analysis is beyond the scope of this work.

If algorithms are analyzed consistency is one basic property. Consistency can be associated with the local error of the numerical method (see Definition 2.2).

Definition 2.2: Consistency

Let $\widetilde{\phi}\left(x^{n+1}\right)$ be the numerical solution at time step $n+1$ for the case, where the numerical method was started from the exact value $\phi\left(x^{n}\right)$ at time step n. The local error of the method can be defined by

$$e_{\mathrm{locT}}^{t^{n+1}} = \left\| \widetilde{\boldsymbol{\phi}}^{t^{n+1}} - \boldsymbol{\phi}^{t^{n+1}} \right\|. \tag{2.77}$$

The numerical method is called consistent if

$$\lim_{h\to 0} e_{\mathrm{locT}}^{t^{n+1}} = 0, \tag{2.78}$$

where h is the time step size.

Consistency is a necessary but not sufficient condition for convergence. For convergence a method needs to be consistent and stable.

Hence, stability and the closely linked condition are defined next.

Definition 2.3: Stability and Condition

Let $F : \mathbb{R}^n \to \mathbb{R}^m$ be the problem which depends on x and $\widetilde{F}$ should be the numerical algorithm which approximates the problem. $\widetilde{x}$ is the disturbed input data.
In order to estimate the error we can use the triangle inequality:

$$\left\| F(x) - \widetilde{F}\left(\widetilde{x}\right) \right\| \leq \underbrace{\left\| F(x) - F\left(\widetilde{x}\right) \right\|}_{\text{condition}} + \underbrace{\left\| F\left(\widetilde{x}\right) - \widetilde{F}\left(\widetilde{x}\right) \right\|}_{\text{stability}}. \tag{2.79}$$

Note that condition is a property of the problem itself and stability is a property of the algorithm.

A consequence of consistency and stability is convergence. It is linked to the global error of the numerical method and is therefore very important in accuracy analysis as noted by Felippa et al. [49].

Definition 2.4: Convergence

A numerical method is said to be convergent if the numerical solution $\hat{\boldsymbol{\phi}}$ approaches the exact solution $\boldsymbol{\phi}$ as the step size h goes to zero.
Convergence can also be defined by using the global error. For the global error

$$e_{\mathrm{gloT}}^{t^{n+1}} = \left\| \hat{\boldsymbol{\phi}}^{t^{n+1}} - \boldsymbol{\phi}^{t^{n+1}} \right\|, \tag{2.80}$$

if the following relation holds

$$\max_n \left\| e_{\mathrm{gloT}}^{t^n} \right\| \to 0 \quad \text{for} \quad h \to 0, \tag{2.81}$$

the method is converging.
The method is said to have an accuracy of order m if

$$e_{\mathrm{gloT}}^{t^n} = \mathcal{O}(h^m), \tag{2.82}$$

at a specific time step n.

The last definition within this chapter introduces an error norm for the global interface residuals vector. In this case it is favorable to normalize the residual by the number of DOFs in order to get residual numbers which have the same physical meaning as demonstrated in Küttler et al. [89]. Let $\boldsymbol{\phi} \in \mathbb{R}^n$ then we define

$$\left\|\boldsymbol{\phi}\right\|_\epsilon = \frac{1}{\sqrt{n}} \left\|\boldsymbol{\phi}\right\|_2. \tag{2.83}$$

I think there's a world market
for about five computers.

Thomas Watson

CHAPTER 3

CO-SIMULATION

This chapter focuses on co-simulation. As a first step it is shown how the co-simulation model can be derived from the monolithic problem. Afterwards aspects of co-simulation like decomposition, communication patterns and block diagrams are discussed.

These discussion points provide the necessary background to derive the Interface Jacobian-based Co-Simulation Algorithm (IJCSA) introduced by Sicklinger et al. [134] in Chapter 4. The IJCSA is designed for robust co-simulation with an arbitrary number of subsystems.

Definition 3.1: Co-Simulation

In co-simulation the different subsystems which form a coupled problem are modeled and simulated in a segregated manner. Hence, the modeling is done on the subsystem level without having the coupled problem in mind. Furthermore, the coupled simulation is carried out by running the subsystems in a black-box manner. During the simulation the subsystems will exchange data.

Before the transition from monolithic to co-simulation is discussed, co-simulation should be defined. The Definition 3.1 used within this work is close to the one of Geimer et al. [59].

In order to discuss the properties of co-simulation a linear model is used. This procedure is done in a lot of disciplines, where some prominent references are presented by Busch et al. [27], Causin et al. [28], Dettmer et al. [35], Felippa et al. [51], and Strang [143]. Even

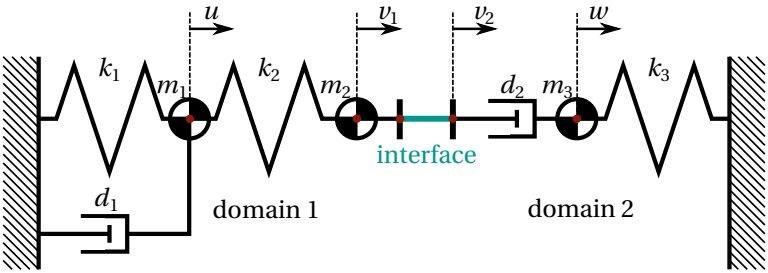


Figure 3.1: Monolithic/co-simulation test problem

though the used linear model (see Figure 3.1) is motivated from structural mechanics the properties can be carried over to fluid-structure interaction (see also Dettmer et al. [35]) as well as domain 2 can be seen as an idealization of a fluid model. The model problem is an initial value problem (see also Table 3.1).

The model problem (Figure 3.1) has 3 DOFs for the monolithic case and 4 DOFs for the co-simulation case. Hence, subsystem 1 and subsystem 2 have each one input, one output and one state variable. Therefore, the model problem is used to show the accuracy behavior of the state and the interface variables for different combinations of time integrators. Before the co-simulation is analyzed the accuracy properties for the monolithic system are shown for different time integrators. In order to keep the discussion more general first and second order single and multi-step time integrators are used. All the presented time integrators are in the class of linear multistep methods (LMS) according to Felippa et al. [49].

3.1 Monolithic

The following linear system of ODEs describes the model problem shown in (Figure 3.1).

$$\begin{aligned} \ddot{u} + \frac{d_1}{m_1}\dot{u} + \frac{k_1}{m_1}u + \frac{k_2}{m_1}(u-v) &= 0 \\ \ddot{v} + \frac{d_2}{m_2}(\dot{v}-\dot{w}) + \frac{k_2}{m_2}(v-u) &= 0 \\ \ddot{w} + \frac{d_2}{m_3}(\dot{w}-\dot{v}) + \frac{k_3}{m_3}w &= 0 \end{aligned} \tag{3.1}$$

Note that there is one second order ODE per DOF. In order to apply a numerical time integration procedure, Equation Set (3.1) is rearranged in matrix form.

$$\underbrace{\begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix}}_{\boldsymbol{M}} \underbrace{\begin{bmatrix} \ddot{u} \\ \ddot{v} \\ \ddot{w} \end{bmatrix}}_{\ddot{\boldsymbol{\phi}}} + \underbrace{\begin{bmatrix} d_1 & 0 & 0 \\ 0 & d_2 & -d_2 \\ 0 & -d_2 & d_2 \end{bmatrix}}_{\boldsymbol{D}} \underbrace{\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix}}_{\dot{\boldsymbol{\phi}}} + \underbrace{\begin{bmatrix} k_1+k_2 & -k_2 & 0 \\ -k_2 & k_2 & 0 \\ 0 & 0 & k_3 \end{bmatrix}}_{\boldsymbol{K}} \underbrace{\begin{bmatrix} u \\ v \\ w \end{bmatrix}}_{\boldsymbol{\phi}} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}}_{\boldsymbol{f}} \tag{3.2}$$

Equation (3.2) represents the monolithic continuous form of the model problem. In order to solve the continuous problem different numerical time integrators are used in the following.

3.1.1 Backward Euler

The most basic implicit time integrator is backward Euler (BE). The backward Euler time integrator is also called implicit Euler or first

order backward differentiation formula (BDF1). The BE operator is given for $\boldsymbol{\phi}$ and $\dot{\boldsymbol{\phi}}$

$$\boldsymbol{\phi}^{n+1} = \boldsymbol{\phi}^{n} + h\dot{\boldsymbol{\phi}}^{n+1}, \tag{3.3}$$

$$\dot{\boldsymbol{\phi}}^{n+1} = \dot{\boldsymbol{\phi}}^{n} + h\ddot{\boldsymbol{\phi}}^{n+1}, \tag{3.4}$$

where h is the time step. Rewriting these equations renders an approximation for the velocity, namely

$$\dot{\boldsymbol{\phi}}^{n+1} = \frac{1}{h}\left(\boldsymbol{\phi}^{n+1} - \boldsymbol{\phi}^{n}\right), \tag{3.5}$$

and one for the acceleration

$$\ddot{\boldsymbol{\phi}}^{n+1} = \frac{1}{h^2}\left(\boldsymbol{\phi}^{n+1} - 2\boldsymbol{\phi}^{n} + \boldsymbol{\phi}^{n-1}\right). \tag{3.6}$$

With these two approximations the BE time discretized ODE System (3.2) can be written as

$$\left(\frac{1}{h^2}\boldsymbol{M} + \frac{1}{h}\boldsymbol{D} + \boldsymbol{K}\right)\boldsymbol{\phi}^{n+1} = \left(\frac{2}{h^2}\boldsymbol{M} + \frac{1}{h}\boldsymbol{D}\right)\boldsymbol{\phi}^{n} - \left(\frac{1}{h^2}\boldsymbol{M}\right)\boldsymbol{\phi}^{n-1} + h\boldsymbol{f}^{n+1}. \tag{3.7}$$

The initial conditions $\boldsymbol{\phi}^{\text{init}}$ and $\dot{\boldsymbol{\phi}}^{\text{init}}$ can be set for the BE method with the following equations:

$$\boldsymbol{\phi}^{0} = \boldsymbol{\phi}^{\text{init}} \tag{3.8}$$

$$\boldsymbol{\phi}^{-1} = \boldsymbol{\phi}^{\text{init}} - h\dot{\boldsymbol{\phi}}^{\text{init}} \tag{3.9}$$

3.1.2 Generalized-α Method

The generalized-α method is a very prominent method in structural dynamics, which was introduced by Chung et al. [31]. It is a second order accurate single step method with four user parameters. By setting different values for the four user parameters $\alpha_{\text{m}}, \alpha_{\text{f}}, \beta, \gamma$ the Generalized-α method can be transformed into various time integrators (e.g. trapezoidal rule, Newmark method, Hilber-Hughes-Taylor method presented by Hilber et al. [75] and more).

The following approximations are used for the velocity and the acceleration:

$$\dot{\boldsymbol{\phi}}^{n+1} = \dot{\boldsymbol{\phi}}^{n} + h\left(\gamma \ddot{\boldsymbol{\phi}}^{n+1} + \left(1-\gamma\right)\ddot{\boldsymbol{\phi}}^{n}\right) \tag{3.10}$$

$$\ddot{\boldsymbol{\phi}}^{n+1} = \frac{1}{h^2\beta}\boldsymbol{\phi}^{n+1} - \frac{1}{h^2\beta}\boldsymbol{\phi}^{n} - \frac{1}{h\beta}\dot{\boldsymbol{\phi}}^{n} + \left(1-\frac{1}{2\beta}\right)\ddot{\boldsymbol{\phi}}^{n} \tag{3.11}$$

With these approximations the generalized-α time discretized ODE System (3.2) can be written as

$$\begin{aligned}
&\left(\left(\frac{1}{h^2\beta} - \frac{\alpha_\mathrm{m}}{h^2\beta}\right)\boldsymbol{M} + \left(\frac{\gamma}{h\beta} - \frac{\alpha_\mathrm{f}\gamma}{h\beta}\right)\boldsymbol{D} + \left(1-\alpha_\mathrm{f}\right)\boldsymbol{K}\right)\boldsymbol{\phi}^{n+1} = \\
&\boldsymbol{M}\left(\left(\frac{1}{h^2\beta} - \frac{\alpha_\mathrm{m}}{h^2\beta}\right)\boldsymbol{\phi}^{n} + \left(\frac{1}{h\beta} - \frac{\alpha_\mathrm{m}}{h\beta}\right)\dot{\boldsymbol{\phi}}^{n} + \left(\frac{1}{2\beta} - \frac{\alpha_\mathrm{m}}{2\beta} - 1\right)\ddot{\boldsymbol{\phi}}^{n}\right) + \\
&\boldsymbol{D}\left(\left(\frac{\gamma}{h\beta} - \frac{\alpha_\mathrm{f}\gamma}{h\beta}\right)\boldsymbol{\phi}^{n} + \left(\frac{\gamma}{\beta} - 1 - \frac{\alpha_\mathrm{f}\gamma}{\beta}\right)\dot{\boldsymbol{\phi}}^{n} + \right. \\
&\qquad \left.\left(\frac{h\gamma}{2\beta} + \alpha_\mathrm{f}h - h - \frac{\alpha_\mathrm{f}h\gamma}{2\beta}\right)\ddot{\boldsymbol{\phi}}^{n}\right) - \boldsymbol{K}\left(\alpha_\mathrm{f}\boldsymbol{\phi}^{n}\right) + \boldsymbol{f}^{n+1-\alpha_\mathrm{f}}.
\end{aligned} \tag{3.12}$$

The following parameter values are used for the subsequent discussions:

$$\begin{aligned}
\alpha_\mathrm{m} &= 0.5 \\
\alpha_\mathrm{f} &= 0.5 \\
\beta &= 0.25 \\
\gamma &= 0.5
\end{aligned}$$

These values result in a second order accurate, unconditionally stable scheme with minimal numerical dissipation. The initial conditions $\boldsymbol{\phi}^{\mathrm{init}}$ and $\dot{\boldsymbol{\phi}}^{\mathrm{init}}$ can be set for the generalized-α method as shown by Chung et al. [31] with the following equations:

$$\boldsymbol{\phi}^{0} = \boldsymbol{\phi}^{\mathrm{init}} \tag{3.13}$$

$$\dot{\boldsymbol{\phi}}^{0} = \dot{\boldsymbol{\phi}}^{\mathrm{init}} \tag{3.14}$$

$$\ddot{\boldsymbol{\phi}}^{0} = \boldsymbol{M}^{-1}\left(\boldsymbol{f}^{0} - \boldsymbol{D}\dot{\boldsymbol{\phi}}^{0} - \boldsymbol{K}\boldsymbol{\phi}^{0}\right) \tag{3.15}$$

3.1.3 BDF2

The second order Backward Differentiation Formula (BDF2) method is a two step second order accurate method. More details of the Backward Differentiation Formula can for instance be found in Ascher et al. [5]. The BDF2 operator for approximating the velocity is given by

$$\dot{\boldsymbol{\phi}}^{n+1} = \frac{1}{h}\left(\frac{3}{2}\boldsymbol{\phi}^{n+1} - 2\boldsymbol{\phi}^{n} + \frac{1}{2}\boldsymbol{\phi}^{n-1}\right), \tag{3.16}$$

and for the acceleration by

$$\ddot{\boldsymbol{\phi}}^{n+1} = \frac{1}{h^2}\left(\frac{9}{4}\boldsymbol{\phi}^{n+1} - 6\boldsymbol{\phi}^{n} + \frac{11}{2}\boldsymbol{\phi}^{n-1} - 2\boldsymbol{\phi}^{n-2} + \frac{1}{4}\boldsymbol{\phi}^{n-3}\right). \tag{3.17}$$

If the method is applied to the second order ODE System (3.2) the following linear equation system is obtained:

$$\begin{aligned}\left(\frac{9}{4h^2}\boldsymbol{M} + \frac{3}{2h}\boldsymbol{D} + \boldsymbol{K}\right)\boldsymbol{\phi}^{n+1} = \\ \boldsymbol{M}\left(\frac{6}{h^2}\boldsymbol{\phi}^{n} - \frac{11}{2h^2}\boldsymbol{\phi}^{n-1} + \frac{2}{h^2}\boldsymbol{\phi}^{n-2} - \frac{1}{4h^2}\boldsymbol{\phi}^{n-3}\right) + \\ \boldsymbol{D}\left(\frac{2}{h}\boldsymbol{\phi}^{n} - \frac{1}{2h}\boldsymbol{\phi}^{n-1}\right) + \boldsymbol{f}^{n+1}\end{aligned} \tag{3.18}$$

Setting the initial conditions $\boldsymbol{\phi}^{\text{init}}$ and $\dot{\boldsymbol{\phi}}^{\text{init}}$ for the second order Backward Differentiation Formula is more complicated. In order to get the method started values for $\boldsymbol{\phi}^{-3}$, $\boldsymbol{\phi}^{-2}$, $\boldsymbol{\phi}^{-1}$ and $\boldsymbol{\phi}^{0}$ are needed. This can be accomplished by running the BE method for the first two time steps and afterwards switching to BDF2. However, for initial value problems this will lower the order of accuracy. Another possibility is to use the generalized-α method to get the BDF2 method started. This conserves the second order of accuracy of the BDF2 method.

3.1.4 Numerical Results

For the presented three time integrators the accuracy order is plotted in Figure 3.2 for the model problem presented in Figure 3.1. Clearly, the numerical examples match the theoretical expectations. The BE method shows a first order of accuracy for displacements. Whereas the other two methods show a second order of accuracy.

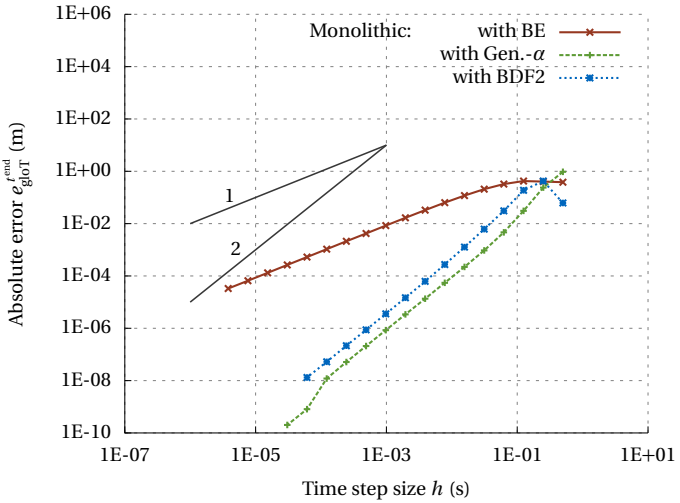


Figure 3.2: Absolute global error (2.80) of the displacements over time step size h

For the system parameter in Table 3.1 the results for displacements u, v, w can be found in Figure 3.3.

Note that the reference solution for the test problem was obtained by using Richardson extrapolation.

3.2 Partitioning Procedure – From Monolithic to Co-Simulation

Within this section the transition process from the monolithic problem to the co-simulation is illustrated and investigated. Co-Simulation implies so-called differential partitioning according to Felippa et al. [49] of the monolithic system. Hence the differential equation system is partitioned. In contrast to the differential partitioning there is also the so-called algebraic partitioning according to Felippa et al. [49], where the monolithic differential equation system is first discretized and afterwards partitioned.

The maximum flexibility is obtained by using the differential partitioning approach or in other words co-simulation (Definition 3.1).

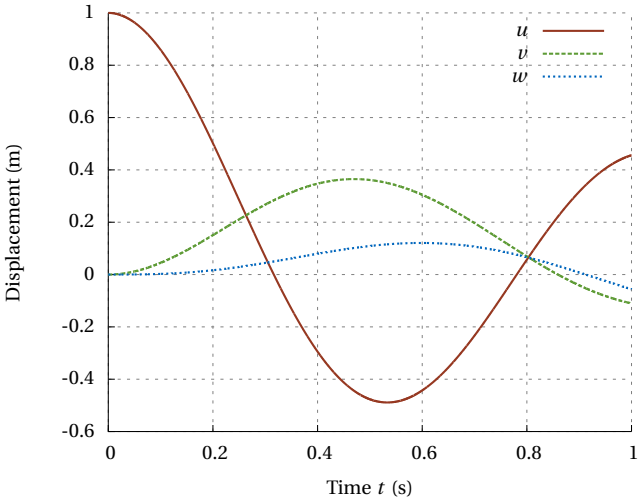


Figure 3.3: Solution over time

Table 3.1: System parameters for the model problem

Property	Symbol	Value	Unit
Mass	m_1	0.1	kg
Mass	m_2	0.2	kg
Mass	m_3	0.3	kg
Damping coefficient	d_1	0.1	N/s
Damping coefficient	d_2	0.5	N/s
Spring stiffness	k_1	1.0	N/m
Spring stiffness	k_2	2.0	N/m
Spring stiffness	k_3	3.0	N/m
Initial displacement	u^{init}	1.0	m
Initial velocity	$\dot{u}^{\text{init}}$	0.0	m/s

The notation introduced in Section 2.1 will be used to describe the differential partitioned ODE system. Hence, the first step is to partition the monolithic second order ODE System (3.1) into two domains.

For the first domain we get

$$m_1 \ddot{u}_1 + d_1 \dot{u}_1 + k_1 u_1 + k_2 \left(u_1 - v_1\right) = 0, \tag{3.19}$$

$$m_2 \ddot{v}_1 + k_2 \left(v_1 - u_1\right) = f_1. \tag{3.20}$$

The equations for the first domain can be rearranged in matrix notation. This renders

$$\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \begin{bmatrix} \ddot{u}_1 \\ \ddot{v}_1 \end{bmatrix} + \begin{bmatrix} d_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{u}_1 \\ \dot{v}_1 \end{bmatrix} + \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \end{bmatrix} = \begin{bmatrix} 0 \\ f_1 \end{bmatrix}. \tag{3.21}$$

For the second domain we get

$$d_2 \left(\dot{v}_2 - \dot{w}_2\right) = f_2, \tag{3.22}$$

$$m_3 \ddot{w}_2 + d_2 \left(\dot{w}_2 - \dot{v}_2\right) + k_3 w_2 = 0, \tag{3.23}$$

and in matrix notation we have

$$\begin{bmatrix} 0 & 0 \\ 0 & m_3 \end{bmatrix} \begin{bmatrix} \ddot{v}_2 \\ \ddot{w}_2 \end{bmatrix} + \begin{bmatrix} d_2 & -d_2 \\ -d_2 & d_2 \end{bmatrix} \begin{bmatrix} \dot{v}_2 \\ \dot{w}_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & k_3 \end{bmatrix} \begin{bmatrix} v_2 \\ w_2 \end{bmatrix} = \begin{bmatrix} f_2 \\ 0 \end{bmatrix}. \tag{3.24}$$

If the partitioned Equation Sets (3.21) and (3.24) are compared to the monolithic Equation Set (3.2) it is evident that some terms vanish due to the partitioning. This information loss needs to be compensated by an appropriate set of interface constraint equations.

Hence as a next step a complete set of equation is established which describes the monolithic problem in a co-simulation sense. Note that in this case the monolithic ODE system will be reformulated

in a set of differential algebraic equations (DAEs). This is also the case in fluid-structure interaction. Here the reader is referred to Dörfel [38] and Dörfel et al. [39].

The general notation for this problem with two subsystems is given by

$$Y_1 = \mathcal{S}_1\left(U_1\right), \tag{3.25}$$

$$Y_2 = \mathcal{S}_2\left(U_2\right). \tag{3.26}$$

By using the information of the partitioned Equation Sets (3.21) and (3.24) we can rewrite the general notation to

$$v_1 = \mathcal{S}_1\left(f_1\right), \tag{3.27}$$

$$v_2 = \mathcal{S}_2\left(f_2\right), \tag{3.28}$$

where v_1 and v_2 are the interface displacements for domain 1 and domain 2. In order to retain the coupled problem the interface constraint equations need to be added. In general form they read

$$\mathcal{I}_1\left(Y_1, Y_2, U_1, U_2\right) = 0, \tag{3.29}$$

$$\mathcal{I}_2\left(Y_1, Y_2, U_1, U_2\right) = 0. \tag{3.30}$$

Again the general notation applied to the partitioned Equation Sets (3.21) and (3.24) reads

$$\mathcal{I}_1\left(f_1, f_2\right) = f_1 + f_2 \qquad = 0, \tag{3.31}$$

$$\mathcal{I}_2\left(v_1, v_2\right) = v_1 - v_2 = \mathcal{S}_1\left(f_1\right) - \mathcal{S}_2\left(f_2\right) = 0. \tag{3.32}$$

The first interface constraint Equation (3.31) ensures the force balance at the interface. The latter Equation (3.32) the kinematic compatibility of the displacements.

The Partitioning and Decomposition Procedure

For a practical co-simulation scenario the input and output quantities are predefined through the subsystems and can not be changed. However, if the monolithic problem is decomposed this is a decision which needs to be made (see Section 3.2). The implications on the stability of the co-simulation of a certain decomposition are discussed in detail in Section 3.4.

A hint for choosing interface constraint equations and input and output quantities is that the constructed system should be as close as possible to the monolithic system.

This means for the example depicted in Figure 3.1 the inputs for each of the two subsystems are forces and that the interface constraint equations need to comply with kinematic compatibility and force balance.

Nevertheless a "good" choice of interface constraint equations and input/output quantities is in general problem dependent.

3.2.1 Co-Simulation with Coherent Time Integration Schemes

In the following it is shown that the co-simulation will not harm the order of accuracy, if the same time integrators are used throughout all subsystems of the co-simulation. Furthermore it is shown that for this case the derived quantities (i.e. velocity and acceleration) at the interface are the same as for the monolithic case.

Example 3.1: Backward Euler for Domain 1 and Domain 2

If the backward Euler time integrator Equation (3.7) is applied to Equation (3.21) we arrive at

$$\left(\frac{1}{h}\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} + \begin{bmatrix} d_1 & 0 \\ 0 & 0 \end{bmatrix} + h\begin{bmatrix} k_1+k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix}\right)\begin{bmatrix} u_1^{n+1} \\ v_1^{n+1} \end{bmatrix} =$$
$$\left(\frac{2}{h}\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} + \begin{bmatrix} d_1 & 0 \\ 0 & 0 \end{bmatrix}\right)\begin{bmatrix} u_1^{n} \\ v_1^{n} \end{bmatrix} - \left(\frac{1}{h}\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}\right)\begin{bmatrix} u_1^{n-1} \\ v_1^{n-1} \end{bmatrix} + h\begin{bmatrix} 0 \\ f_1^{n+1} \end{bmatrix}. \quad (3.33)$$

Here the input U_1 is the force f_1. Subsystem 1 $v_1 = \mathcal{S}_1\left(f_1\right)$ solves for the interface displacement v_1 and for the state displacement u_1 which is not shown to the interface.
The same procedure is applied to Equation (3.24) and thus we arrive at

$$\left(\frac{1}{h}\begin{bmatrix} 0 & 0 \\ 0 & m_3 \end{bmatrix} + \begin{bmatrix} d_2 & -d_2 \\ -d_2 & d_2 \end{bmatrix} + h\begin{bmatrix} 0 & 0 \\ 0 & k_3 \end{bmatrix}\right)\begin{bmatrix} v_2^{n+1} \\ w_2^{n+1} \end{bmatrix} =$$
$$\left(\frac{2}{h}\begin{bmatrix} 0 & 0 \\ 0 & m_3 \end{bmatrix} + \begin{bmatrix} d_2 & -d_2 \\ -d_2 & d_2 \end{bmatrix}\right)\begin{bmatrix} v_2^{n} \\ w_2^{n} \end{bmatrix} - \left(\frac{1}{h}\begin{bmatrix} 0 & 0 \\ 0 & m_3 \end{bmatrix}\right)\begin{bmatrix} v_2^{n-1} \\ w_2^{n-1} \end{bmatrix} + h\begin{bmatrix} f_2^{n+1} \\ 0 \end{bmatrix}. \quad (3.34)$$

For the second subsystem the input U_2 is the force f_2. The subsystem 2 $v_2 = \mathcal{S}_2\left(f_2\right)$ solves for the interface displacement v_2 and for the state displacement w_2.

The first case to investigate is Example 3.1. Here subsystem 1 and subsystem 2 are discretized via the backward Euler time integrator. This should result in a first order accurate co-simulation in time for displacements, velocities and accelerations. The results should be equal for the monolithic and co-simulation case if the interface constraint Equations (3.31) and (3.32) are satisfied precisely.

The graph in Figure 3.4(a) shows the accuracy order for the monolithic and the co-simulation solution of the problem. It is evident that the results meet the expectations. Displacements, velocities and accelerations converge with first order in time. The results are the same for the monolithic case and for the co-simulation case.

Note that the interface constraint equations are satisfied to machine precision by using the Interface Jacobian-based Co-Simulation Algorithm, which is described in detail in Chapter 4.

As a next step the second order accurate generalized-α method is used for the co-simulation of the model problem. Similar to Example 3.1 the co-simulation and the monolithic solution show the same order of accuracy namely, second order for displacements, velocities and accelerations (see Figure 3.4(b)).

Example 3.2: Generalized-α Method for Domain 1 and Domain 2

The following example is similar to Example 3.1. However, the generalized-α method (3.12) is used to integrate Equation (3.21) and thus we arrive at

$$\left(\left(\frac{1}{h^2\beta}-\frac{\alpha_\mathrm{m}}{h^2\beta}\right)\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}+\left(\frac{\gamma}{h\beta}-\frac{\alpha_\mathrm{f}\gamma}{h\beta}\right)\begin{bmatrix} d_1 & 0 \\ 0 & 0 \end{bmatrix}+\left(1-\alpha_\mathrm{f}\right)\begin{bmatrix} k_1+k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix}\right)$$

$$\begin{bmatrix} u_1^{n+1} \\ v_1^{n+1} \end{bmatrix}=\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}\left(\left(\frac{1}{h^2\beta}-\frac{\alpha_\mathrm{m}}{h^2\beta}\right)\begin{bmatrix} u_1^n \\ v_1^n \end{bmatrix}+\left(\frac{1}{h\beta}-\frac{\alpha_\mathrm{m}}{h\beta}\right)\begin{bmatrix} \dot{u}_1^n \\ \dot{v}_1^n \end{bmatrix}+\right.$$

$$\left.\left(\frac{1}{2\beta}-\frac{\alpha_\mathrm{m}}{2\beta}-1\right)\begin{bmatrix} \ddot{u}_1^n \\ \ddot{v}_1^n \end{bmatrix}\right)+\begin{bmatrix} d_1 & 0 \\ 0 & 0 \end{bmatrix}\left(\left(\frac{\gamma}{h\beta}-\frac{\alpha_\mathrm{f}\gamma}{h\beta}\right)\begin{bmatrix} u_1^n \\ v_1^n \end{bmatrix}+\left(\frac{\gamma}{\beta}-1-\frac{\alpha_\mathrm{f}\gamma}{\beta}\right)\begin{bmatrix} \dot{u}_1^n \\ \dot{v}_1^n \end{bmatrix}+\right.$$

$$\left.\left(\frac{h\gamma}{2\beta}+\alpha_\mathrm{f}h-h-\frac{\alpha_\mathrm{f}h\gamma}{2\beta}\right)\begin{bmatrix} \ddot{u}_1^n \\ \ddot{v}_1^n \end{bmatrix}\right)-\begin{bmatrix} k_1+k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix}\left(\alpha_\mathrm{f}\begin{bmatrix} u_1^n \\ v_1^n \end{bmatrix}\right)+\begin{bmatrix} 0 \\ f_1^{n+1} \end{bmatrix}. \quad (3.35)$$

For the first subsystem input, output and state variables are chosen to Example 3.1 accordingly. The same integration procedure is applied to Equation (3.24) and thus we arrive at

$$\left(\left(\frac{1}{h^2\beta}-\frac{\alpha_\mathrm{m}}{h^2\beta}\right)\begin{bmatrix} 0 & 0 \\ 0 & m_3 \end{bmatrix}+\left(\frac{\gamma}{h\beta}-\frac{\alpha_\mathrm{f}\gamma}{h\beta}\right)\begin{bmatrix} d_2 & -d_2 \\ -d_2 & d_2 \end{bmatrix}+\left(1-\alpha_\mathrm{f}\right)\begin{bmatrix} 0 & 0 \\ 0 & k_3 \end{bmatrix}\right)$$

$$\begin{bmatrix} v_2^{n+1} \\ w_2^{n+1} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & m_3 \end{bmatrix} \left(\left(\frac{1}{h^2\beta} - \frac{\alpha_\mathrm{m}}{h^2\beta} \right) \begin{bmatrix} v_2^n \\ w_2^n \end{bmatrix} + \left(\frac{1}{h\beta} - \frac{\alpha_\mathrm{m}}{h\beta} \right) \begin{bmatrix} \dot{v}_2^n \\ \dot{w}_2^n \end{bmatrix} + \left(\frac{1}{2\beta} - \frac{\alpha_\mathrm{m}}{2\beta} - 1 \right) \begin{bmatrix} \ddot{v}_2^n \\ \ddot{w}_2^n \end{bmatrix} \right) + \begin{bmatrix} d_2 & -d_2 \\ -d_2 & d_2 \end{bmatrix} \left(\left(\frac{\gamma}{h\beta} - \frac{\alpha_\mathrm{f}\gamma}{h\beta} \right) \begin{bmatrix} v_2^n \\ w_2^n \end{bmatrix} + \left(\frac{\gamma}{\beta} - 1 - \frac{\alpha_\mathrm{f}\gamma}{\beta} \right) \begin{bmatrix} \dot{v}_2^n \\ \dot{w}_2^n \end{bmatrix} + \left(\frac{h\gamma}{2\beta} + \alpha_\mathrm{f} h - h - \frac{\alpha_\mathrm{f} h\gamma}{2\beta} \right) \begin{bmatrix} \ddot{v}_2^n \\ \ddot{w}_2^n \end{bmatrix} \right) - \begin{bmatrix} 0 & 0 \\ 0 & k_3 \end{bmatrix} \left(\alpha_\mathrm{f} \begin{bmatrix} v_2^n \\ w_2^n \end{bmatrix} \right) + \begin{bmatrix} f_2^{n+1} \\ 0 \end{bmatrix}. \tag{3.36}$$

For the second subsystem input, output and state variables are chosen to Example 3.1 accordingly.

The last case is Example 3.3 where the two step, second order accurate BDF2 method is discussed. Similar to Example 3.2 the co-simulation and the monolithic solution for Example 3.3 show a second order of accuracy (see Figure 3.4(c)).

Conclusion

The model problem (see Figure 3.1) demonstrates that the monolithic and the co-simulation solution are identical as long as the interface constraint equations are satisfied exactly and the same numerical discretization methods are used within all subsystem. In the terminology of Felippa et al. [49] this means that it has been demonstrated with the help of Example 3.1 to Example 3.3 that a coherent (time) discretization throughout all subsystems renders an algebraic partitioning of the monolithic problem.

In order to summarize this section we can note the following facts: Co-simulation can be derived from the monolithic system in a consistent manner. If the same discretization is used in all subsystems as for the monolithic problem and if the interface constraint equations are solved sufficiently accurate the co-simulation solution is equivalent to the monolithic one.

Example 3.3: Second Order Backward Differentiation Formula for Domain 1 and Domain 2

The following example is similar to Example 3.1. However, the two step method BDF2 (3.18) is used to integrate Equation (3.21) and thus we arrive at

$$\left(\frac{9}{4h^2} \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} + \frac{3}{2h} \begin{bmatrix} d_1 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix} \right) \begin{bmatrix} u_1^{n+1} \\ v_1^{n+1} \end{bmatrix} =$$

$$\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \left(\frac{6}{h^2} \begin{bmatrix} u_1^n \\ v_1^n \end{bmatrix} - \frac{11}{2h^2} \begin{bmatrix} u_1^{n-1} \\ v_1^{n-1} \end{bmatrix} + \frac{2}{h^2} \begin{bmatrix} u_1^{n-2} \\ v_1^{n-2} \end{bmatrix} - \frac{1}{4h^2} \begin{bmatrix} u_1^{n-3} \\ v_1^{n-3} \end{bmatrix} \right) + \begin{bmatrix} d_1 & 0 \\ 0 & 0 \end{bmatrix} \left(\frac{2}{h} \begin{bmatrix} u_1^n \\ v_1^n \end{bmatrix} - \frac{1}{2h} \begin{bmatrix} u_1^{n-1} \\ v_1^{n-1} \end{bmatrix} \right) + \begin{bmatrix} 0 \\ f_1^{n+1} \end{bmatrix}. \quad (3.37)$$

For the first subsystem input, output and state variables are chosen to Example 3.1 accordingly. The same integration procedure is applied to Equation (3.24) and thus we arrive at

$$\left(\frac{9}{4h^2} \begin{bmatrix} 0 & 0 \\ 0 & m_3 \end{bmatrix} + \frac{3}{2h} \begin{bmatrix} d_2 & -d_2 \\ -d_2 & d_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & k_3 \end{bmatrix} \right) \begin{bmatrix} v_2^{n+1} \\ w_2^{n+1} \end{bmatrix} =$$
$$\begin{bmatrix} 0 & 0 \\ 0 & m_3 \end{bmatrix} \left(\frac{6}{h^2} \begin{bmatrix} v_2^n \\ w_2^n \end{bmatrix} - \frac{11}{2h^2} \begin{bmatrix} v_2^{n-1} \\ w_2^{n-1} \end{bmatrix} + \frac{2}{h^2} \begin{bmatrix} v_2^{n-2} \\ w_2^{n-2} \end{bmatrix} - \frac{1}{4h^2} \begin{bmatrix} v_2^{n-3} \\ w_2^{n-3} \end{bmatrix} \right) + \begin{bmatrix} d_2 & -d_2 \\ -d_2 & d_2 \end{bmatrix} \left(\frac{2}{h} \begin{bmatrix} v_2^n \\ w_2^n \end{bmatrix} - \frac{1}{2h} \begin{bmatrix} v_2^{n-1} \\ w_2^{n-1} \end{bmatrix} \right) + \begin{bmatrix} f_2^{n+1} \\ 0 \end{bmatrix}. \quad (3.38)$$

For the second subsystem input, output and state variables are chosen to Example 3.1 accordingly.

3.2.2 Co-Simulation with Mixed Time Integration Schemes

After discussing the case of equal time discretization for all subsystems, the discussion is focused on the more general and more practical relevant case where different time integrators in the different subsystems are present. As this leads to a high number of possible combinations, the investigation is limited to representative cases. However, the methods used in following can be applied to different combinations of time integrators as well, besides the ones presented.

Example 3.4: Generalized-α Method for Domain 1 and BDF2 for Domain 2

The first subsystem is integrated with the generalized-α method.

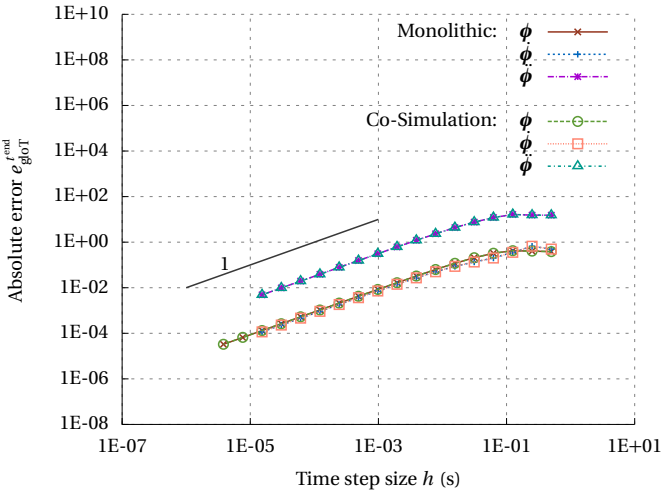
$$\left(\left(\frac{1}{h^2\beta} - \frac{\alpha_m}{h^2\beta} \right) \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} + \left(\frac{\gamma}{h\beta} - \frac{\alpha_f\gamma}{h\beta} \right) \begin{bmatrix} d_1 & 0 \\ 0 & 0 \end{bmatrix} + (1-\alpha_f) \begin{bmatrix} k_1+k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix} \right)$$
$$\begin{bmatrix} u_1^{n+1} \\ v_1^{n+1} \end{bmatrix} = \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \left(\left(\frac{1}{h^2\beta} - \frac{\alpha_m}{h^2\beta} \right) \begin{bmatrix} u_1^n \\ v_1^n \end{bmatrix} + \left(\frac{1}{h\beta} - \frac{\alpha_m}{h\beta} \right) \begin{bmatrix} \dot{u}_1^n \\ \dot{v}_1^n \end{bmatrix} + \right.$$

$$\left(\frac{1}{2\beta}-\frac{\alpha_\mathrm{m}}{2\beta}-1\right)\begin{bmatrix}\ddot{u}_1^n\\ \ddot{v}_1^n\end{bmatrix}+\begin{bmatrix}d_1 & 0\\ 0 & 0\end{bmatrix}\left(\left(\frac{\gamma}{h\beta}-\frac{\alpha_\mathrm{f}\gamma}{h\beta}\right)\begin{bmatrix}u_1^n\\ v_1^n\end{bmatrix}+\left(\frac{\gamma}{\beta}-1-\frac{\alpha_\mathrm{f}\gamma}{\beta}\right)\begin{bmatrix}\dot{u}_1^n\\ \dot{v}_1^n\end{bmatrix}+\left(\frac{h\gamma}{2\beta}+\alpha_\mathrm{f}h-h-\frac{\alpha_\mathrm{f}h\gamma}{2\beta}\right)\begin{bmatrix}\ddot{u}_1^n\\ \ddot{v}_1^n\end{bmatrix}\right)-\begin{bmatrix}k_1+k_2 & -k_2\\ -k_2 & k_2\end{bmatrix}\left(\alpha_\mathrm{f}\begin{bmatrix}u_1^n\\ v_1^n\end{bmatrix}+\begin{bmatrix}0\\ f_1^{n+1}\end{bmatrix}\right). \tag{3.39}$$

The second subsystem (3.24) is integrated with the BDF2 method and thus we arrive at

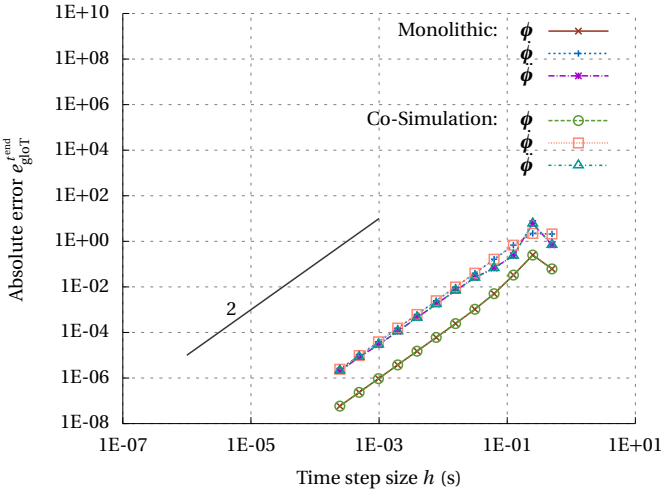
$$\left(\frac{9}{4h^2}\begin{bmatrix}0 & 0\\ 0 & m_3\end{bmatrix}+\frac{3}{2h}\begin{bmatrix}d_2 & -d_2\\ -d_2 & d_2\end{bmatrix}+\begin{bmatrix}0 & 0\\ 0 & k_3\end{bmatrix}\right)\begin{bmatrix}v_2^{n+1}\\ w_2^{n+1}\end{bmatrix}=\begin{bmatrix}0 & 0\\ 0 & m_3\end{bmatrix}\left(\frac{6}{h^2}\begin{bmatrix}v_2^n\\ w_2^n\end{bmatrix}-\frac{11}{2h^2}\begin{bmatrix}v_2^{n-1}\\ w_2^{n-1}\end{bmatrix}+\frac{2}{h^2}\begin{bmatrix}v_2^{n-2}\\ w_2^{n-2}\end{bmatrix}-\frac{1}{4h^2}\begin{bmatrix}v_2^{n-3}\\ w_2^{n-3}\end{bmatrix}\right)+\begin{bmatrix}d_2 & -d_2\\ -d_2 & d_2\end{bmatrix}\left(\frac{2}{h}\begin{bmatrix}v_2^n\\ w_2^n\end{bmatrix}-\frac{1}{2h}\begin{bmatrix}v_2^{n-1}\\ w_2^{n-1}\end{bmatrix}\right)+\begin{bmatrix}f_2^{n+1}\\ 0\end{bmatrix}. \tag{3.40}$$

Within this example two different second order accurate time integrators are combined. The overall order of accuracy can be seen in Figure 3.5(a).
Note that the input and output relations for this example are the same as for Example 3.1.

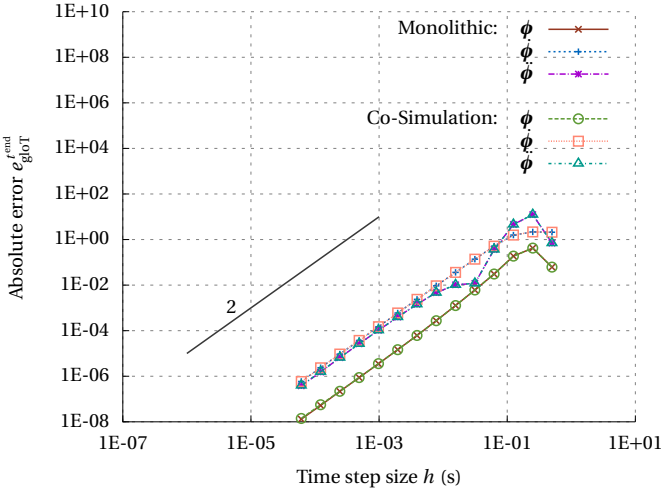


(a) Backward Euler (Example 3.1)

Figure 3.4: Absolute global error (2.80) over time step size for displacement, velocity and acceleration



(b) Generalized-α method (Example 3.2)



(c) BDF2 (Example 3.3)

Figure 3.4: Absolute global error (2.80) over time step size for displacement, velocity and acceleration

Due to the interface constraint equations the displacements are compatible at the interface for Example 3.4 and Example 3.5. If the same time integrators are used between the subsystems the compatibility of velocities and accelerations is also achieved. For the case of different time integrators this is not the case. Only the quantities which are enforced by the interface constraint equations are compatible at the interface. This is demonstrated for Example 3.4 in Figure 3.5(b). Here the generalized-α method for domain 1 is coupled to the BDF2 for domain 2. The displacements are enforced to be kinematic compatible at the interface. Figure 3.5(b) shows the matching displacements for the left and right interface node. Moreover Figure 3.5(b) also demonstrates that the velocities and accelerations are not matching at the interface for Example 3.4 due to the different time integration schemes in domain 1 and domain 2.

Even though Example 3.4 uses two second order accurate time integrators, the coupled solution does not show second order of accuracy (see Figure 3.5(a)), this has also been shown by Farhat et al. [44] and Joosten et al. [84]. However, for this combination of time integrators and coupling conditions the overall numerical method is consistent for the interface DOFs.

An accuracy order loss does not necessarily occur for other combinations of time integrators. If the backward Euler method is chosen for the first domain and the BDF2 method for the second domain (see Example 3.5) the resulting method is still first order accurate for DOFs at the interface as depicted in Figure 3.6. This is the maximum order which can be expected as the BE integrator is only first order accurate.

Example 3.5: BE Method for Domain 1 and BDF2 for Domain 2

The first subsystem is integrated with the backward Euler time integrator Equation (3.7).

$$\left(\frac{1}{h}\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} + \begin{bmatrix} d_1 & 0 \\ 0 & 0 \end{bmatrix} + h\begin{bmatrix} k_1+k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix}\right)\begin{bmatrix} u_1^{n+1} \\ v_1^{n+1} \end{bmatrix} =$$
$$\left(\frac{2}{h}\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} + \begin{bmatrix} d_1 & 0 \\ 0 & 0 \end{bmatrix}\right)\begin{bmatrix} u_1^{n} \\ v_1^{n} \end{bmatrix} - \left(\frac{1}{h}\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}\right)\begin{bmatrix} u_1^{n-1} \\ v_1^{n-1} \end{bmatrix} + h\begin{bmatrix} 0 \\ f_1^{n+1} \end{bmatrix}. \quad (3.41)$$

The second subsystem (3.24) is integrated with the BDF2 method and thus we arrive at

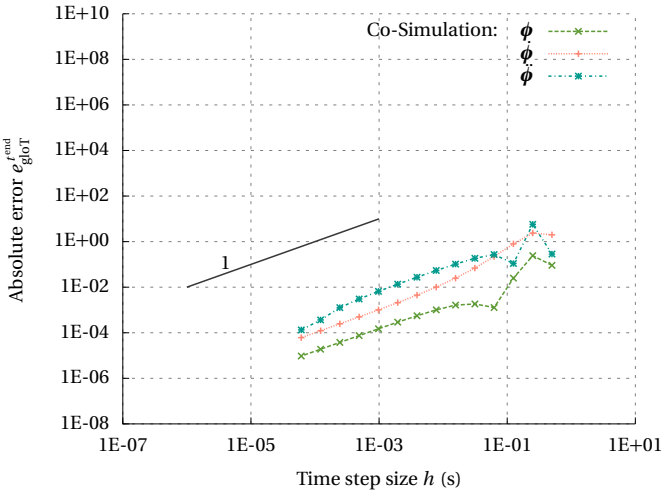
$$\left(\frac{9}{4h^2}\begin{bmatrix} 0 & 0 \\ 0 & m_3 \end{bmatrix} + \frac{3}{2h}\begin{bmatrix} d_2 & -d_2 \\ -d_2 & d_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & k_3 \end{bmatrix}\right)\begin{bmatrix} v_2^{n+1} \\ w_2^{n+1} \end{bmatrix} =$$

$$\begin{bmatrix} 0 & 0 \\ 0 & m_3 \end{bmatrix} \left(\frac{6}{h^2} \begin{bmatrix} v_2^n \\ w_2^n \end{bmatrix} - \frac{11}{2h^2} \begin{bmatrix} v_2^{n-1} \\ w_2^{n-1} \end{bmatrix} + \frac{2}{h^2} \begin{bmatrix} v_2^{n-2} \\ w_2^{n-2} \end{bmatrix} - \frac{1}{4h^2} \begin{bmatrix} v_2^{n-3} \\ w_2^{n-3} \end{bmatrix} \right) + \begin{bmatrix} d_2 & -d_2 \\ -d_2 & d_2 \end{bmatrix} \left(\frac{2}{h} \begin{bmatrix} v_2^n \\ w_2^n \end{bmatrix} - \frac{1}{2h} \begin{bmatrix} v_2^{n-1} \\ w_2^{n-1} \end{bmatrix} \right) + \begin{bmatrix} f_2^{n+1} \\ 0 \end{bmatrix}. \quad (3.42)$$

Within this example two different first order and second order accurate time integrators are combined.
The overall order of accuracy can be seen in Figure 3.6.
Note that the input and output relations for this example are the same as for Example 3.1.

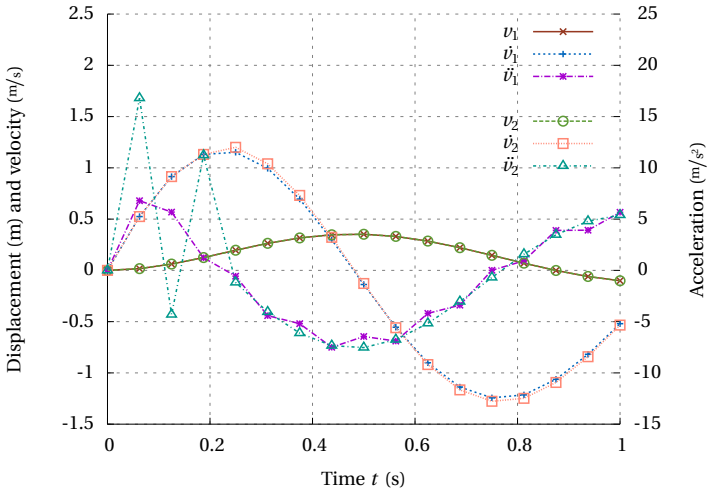
Another important parameter to investigate is the type of coupling conditions. Throughout Example 3.1 to Example 3.5 the displacements and forces were enforced at the interface. In the following it is shown what happens if the displacement compatibility constraint is replaced by a velocity compatibility constraint.

The velocity coupling (enforcing equal interface velocity) and acceleration coupling for Example 3.5 show the same order of accuracy as for displacement coupling as depicted in Figure 3.7. Hence the par-



(a) Absolute global error (2.80) over time step size for displacement, velocity and acceleration

Figure 3.5: Generalized-α versus BDF2 method (Example 3.4)



(b) Displacements, velocities and accelerations for the co-simulation case at the interface

Figure 3.5: Generalized-α versus BDF2 method (Example 3.4)

ticular combination of time integrators of Example 3.5 (BE and BDF2) is first order for displacement, velocity and acceleration coupling for the model problem shown in Figure 3.1. Note that this is also true for the combination of BE and generalized-α which is not shown in detail in order to focus the discussion.

Moreover the behavior of the order of accuracy for Example 3.4 is similar for displacement, velocity and acceleration coupling as well (compare Figure 3.5(a) with Figure 3.8(a)). However, by coupling the accelerations the high frequency oscillation in the solution can be reduced to a minimum (compare Figure 3.5(b) with Figure 3.8(b)).

Conclusion & Remedy

As we have seen the combination of mixed time integration schemes for the individual subsystems is pretentious. In the best case the overall order of accuracy is the minimum of the orders of accuracy of all subsystems.

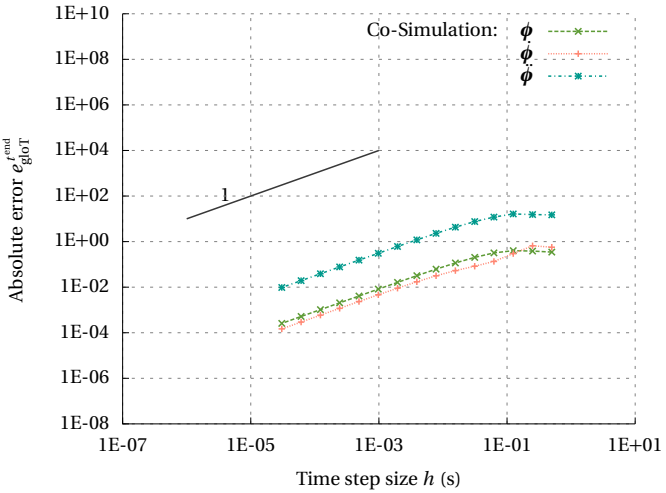


Figure 3.6: Absolute global error (2.80) over time step size for displacement, velocity and acceleration for Example 3.5

Moreover with the help of Example 3.4 it has been demonstrated that the overall oder of accuracy can be less than the minimum of all subsystems.

This phenomena is well known in literature. In the fluid-structure interaction community it is discussed by Farhat et al. [45] and Joosten et al. [84]. A recent publication from the structure-structure interaction community is provided by Li et al. [94] and Mahjoubi et al. [98].

In order to solve the problem of accuracy order loss due to the coupling of incompatible time integration schemes several ideas are discussed.

Joosten et al. [84] suggests to interpolate the transfered forces in time, so that they can be evaluated at the same point in time by the different time integration schemes. However, this idea relies on the use of a generalized-α type method for all subsystems.

The idea of Li et al. [94] is to couple the velocities in an integral sense by applying a weighted residual form of the velocity coupling constraint which preserves the energy over the interface. This idea is tested by coupling an explicit second order accurate Lax-Wendroff

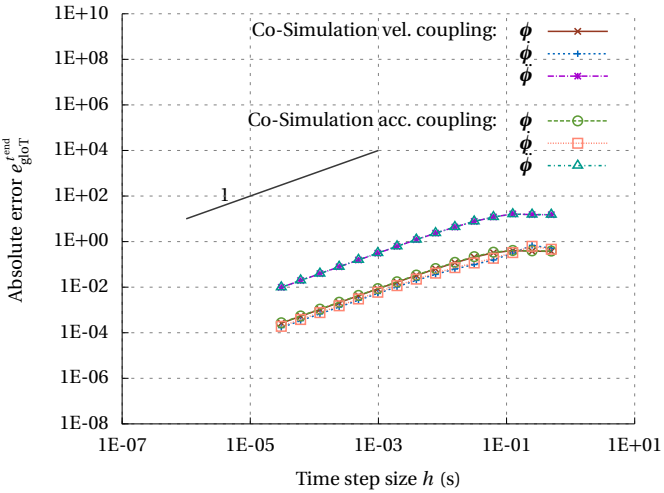


Figure 3.7: Absolute global error (2.80) over time step size for displacement, velocity and acceleration for velocity and acceleration coupling of Example 3.5

scheme with a fourth order Runge-Kutta scheme in Li et al. [94]. However in the case where the interface residual is zero it is shown in the following that the weighted residual form of Li et al. [94] reduces to the strong form of the interface constraint, namely

$$\mathcal{I}_2\left(\dot{\upsilon}_1, \dot{\upsilon}_2\right) = \dot{\upsilon}_1 - \dot{\upsilon}_2 = \mathcal{S}_1\left(f_1\right) - \mathcal{S}_2\left(f_2\right) = 0. \tag{3.43}$$

Li et al. [94] introduces the interface constraint by

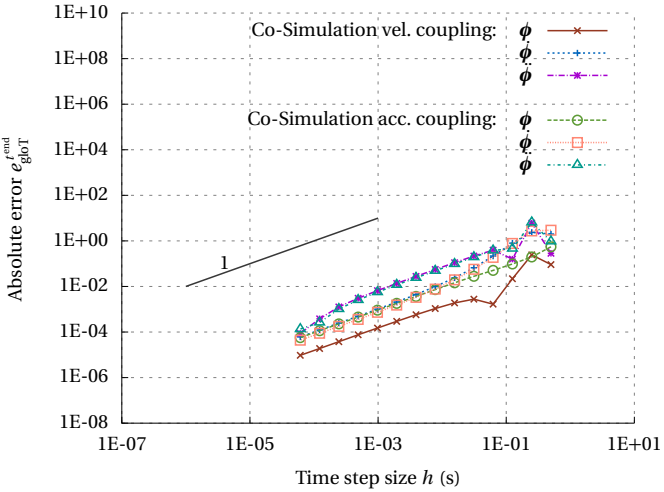
$$\overline{f} \int_{t^n}^{t^{n+1}} \dot{\upsilon}_1(t) - \dot{\upsilon}_2(t)\,\mathrm{d}t = 0. \tag{3.44}$$

Li et al. [94] shows that Equation (3.44) is equivalent to

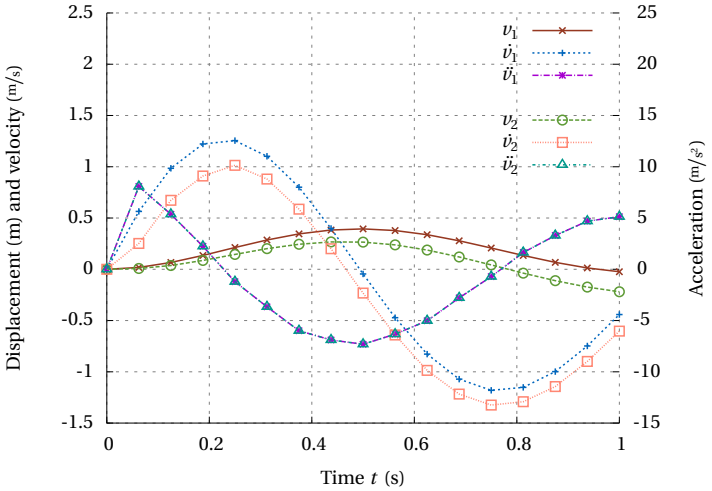
$$\overline{\dot{\upsilon}}_1 - \overline{\dot{\upsilon}}_2 = 0, \tag{3.45}$$

where $\overline{\dot{\upsilon}}$ is an average interface velocity, hence is defined by

$$\overline{\dot{\upsilon}} = \frac{\dot{\upsilon}^{n+1} + \dot{\upsilon}^n}{2}. \tag{3.46}$$



(a) Absolute global error (2.80) over time step size for displacement, velocity and acceleration for velocity and acceleration coupling



(b) Displacements, velocities and accelerations for the co-simulation case at the interface for acceleration coupling

Figure 3.8: Generalized-α versus BDF2 method for velocity and acceleration coupling (Example 3.4)

With that, Equation (3.44) is equal to

$$\dot{v}_1^{n+1} - \dot{v}_2^{n+1} = \dot{v}_2^{n} - \dot{v}_1^{n}. \tag{3.47}$$

If the interface residual is reduces to zero within every time step the right hand side of Equation (3.47) is zero and the integral form of the interface constraint is equal to Equation (3.43).

After discussing all those ideas the question remains how to restore the second oder accuracy for Example 3.4. Let us first examine what exactly causes the loss of accuracy. Therefore it needs to be repeated that following interface constraints were applied

$$\mathcal{I}_1\left(f_1, f_2\right) = f_1 + f_2 \qquad = 0, \tag{3.48}$$

$$\mathcal{I}_2\left(v_1, v_2\right) = v_1 - v_2 = \mathcal{S}_1\left(f_1\right) - \mathcal{S}_2\left(f_2\right) = 0. \tag{3.49}$$

Let us have a closer look into the first constraint. Here the forces at every point in time $n+1$ need to be in equilibrium

$$f_1^{n+1} + f_2^{n+1} = 0. \tag{3.50}$$

If the BDF2 method is applied to $\mathcal{S}_2$ we have

$$\mathcal{S}_2\left(f_2^{n+1}\right) = v_2^{n+1}. \tag{3.51}$$

Example 3.4 makes the same assumption for the generalized-α method, namely

$$\mathcal{S}_1\left(f_1^{n+1}\right) = v_1^{n+1}. \tag{3.52}$$

However, the integration parameter is set to $\alpha_\mathrm{f} = 0.5$. As Chung et al. [31] has shown the load needs also to be interpolated for the generalized-α method by

$$f = (1-\alpha_\mathrm{f}) f^{n+1} + \alpha_\mathrm{f} f^{n}. \tag{3.53}$$

This shows that the assumption of Equation (3.52) is not correct. The correct statement for $\alpha_\mathrm{f} = 0.5$ is

$$\mathcal{S}_1\left(f_1^{n+\frac{1}{2}}\right) = v_1^{n+1}. \tag{3.54}$$

So we can conclude if the load is interpolated inside $\mathcal{S}_1$ to the time step $n+1/2$ one should be able to restore second order accuracy for Example 3.4. Example 3.6 clearly shows that this is the case as demonstrated by Figure 3.9.

Example 3.6: Generalized-α Method for Domain 1 and BDF2 for Domain 2 (with Interpolation in Time for Domain 1)

The first subsystem is integrated with the generalized-α method. The load which is applied to the system is interpolated from the current input force f_1^{n+1} and the old input force f_1^n by

$$f^{n+(1-\alpha_f)h} = (1-\alpha_f) f^{n+1} + \alpha_f f^n. \tag{3.55}$$

Afterwards the generalized-α method is applied.

$$\begin{aligned}
&\left(\left(\frac{1}{h^2\beta} - \frac{\alpha_m}{h^2\beta}\right)\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} + \left(\frac{\gamma}{h\beta} - \frac{\alpha_f\gamma}{h\beta}\right)\begin{bmatrix} d_1 & 0 \\ 0 & 0 \end{bmatrix} + (1-\alpha_f)\begin{bmatrix} k_1+k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix}\right) \\
&\begin{bmatrix} u_1^{n+1} \\ v_1^{n+1} \end{bmatrix} = \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}\left(\left(\frac{1}{h^2\beta} - \frac{\alpha_m}{h^2\beta}\right)\begin{bmatrix} u_1^n \\ v_1^n \end{bmatrix} + \left(\frac{1}{h\beta} - \frac{\alpha_m}{h\beta}\right)\begin{bmatrix} \dot{u}_1^n \\ \dot{v}_1^n \end{bmatrix} + \right. \\
&\left.\left(\frac{1}{2\beta} - \frac{\alpha_m}{2\beta} - 1\right)\begin{bmatrix} \ddot{u}_1^n \\ \ddot{v}_1^n \end{bmatrix}\right) + \begin{bmatrix} d_1 & 0 \\ 0 & 0 \end{bmatrix}\left(\left(\frac{\gamma}{h\beta} - \frac{\alpha_f\gamma}{h\beta}\right)\begin{bmatrix} u_1^n \\ v_1^n \end{bmatrix} + \left(\frac{\gamma}{\beta} - 1 - \frac{\alpha_f\gamma}{\beta}\right)\begin{bmatrix} \dot{u}_1^n \\ \dot{v}_1^n \end{bmatrix} + \right. \\
&\left.\left(\frac{h\gamma}{2\beta} + \alpha_f h - h - \frac{\alpha_f h\gamma}{2\beta}\right)\begin{bmatrix} \ddot{u}_1^n \\ \ddot{v}_1^n \end{bmatrix}\right) - \begin{bmatrix} k_1+k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix}\left(\alpha_f\begin{bmatrix} u_1^n \\ v_1^n \end{bmatrix}\right) + \begin{bmatrix} 0 \\ f_1^{n+(1-\alpha_f)h} \end{bmatrix}.
\end{aligned} \tag{3.56}$$

The second subsystem (3.24) is integrated with the BDF2 method and thus we arrive at

$$\begin{aligned}
&\left(\frac{9}{4h^2}\begin{bmatrix} 0 & 0 \\ 0 & m_3 \end{bmatrix} + \frac{3}{2h}\begin{bmatrix} d_2 & -d_2 \\ -d_2 & d_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & k_3 \end{bmatrix}\right)\begin{bmatrix} v_2^{n+1} \\ w_2^{n+1} \end{bmatrix} = \\
&\begin{bmatrix} 0 & 0 \\ 0 & m_3 \end{bmatrix}\left(\frac{6}{h^2}\begin{bmatrix} v_2^n \\ w_2^n \end{bmatrix} - \frac{11}{2h^2}\begin{bmatrix} v_2^{n-1} \\ w_2^{n-1} \end{bmatrix} + \frac{2}{h^2}\begin{bmatrix} v_2^{n-2} \\ w_2^{n-2} \end{bmatrix} - \frac{1}{4h^2}\begin{bmatrix} v_2^{n-3} \\ w_2^{n-3} \end{bmatrix}\right) + \\
&\qquad\qquad \begin{bmatrix} d_2 & -d_2 \\ -d_2 & d_2 \end{bmatrix}\left(\frac{2}{h}\begin{bmatrix} v_2^n \\ w_2^n \end{bmatrix} - \frac{1}{2h}\begin{bmatrix} v_2^{n-1} \\ w_2^{n-1} \end{bmatrix}\right) + \begin{bmatrix} f_2^{n+1} \\ 0 \end{bmatrix}.
\end{aligned} \tag{3.57}$$

Within this example two different second order accurate time integrators are combined. The overall order of accuracy can be seen in Figure 3.9.
Note that the input and output relations for this example are the same as for Example 3.1.

As a final remark it can be stated that one needs to take care when mixed time integrators are coupled in a co-simulation scenario. But if everything is done in a consistent manner a simulation result which is as accurate as the monolithic one can be obtained.

A useful feature for the end user is an automatic estimation of the overall order of accuracy during the co-simulation as this gives feedback to the user about the quality of the obtained results. The

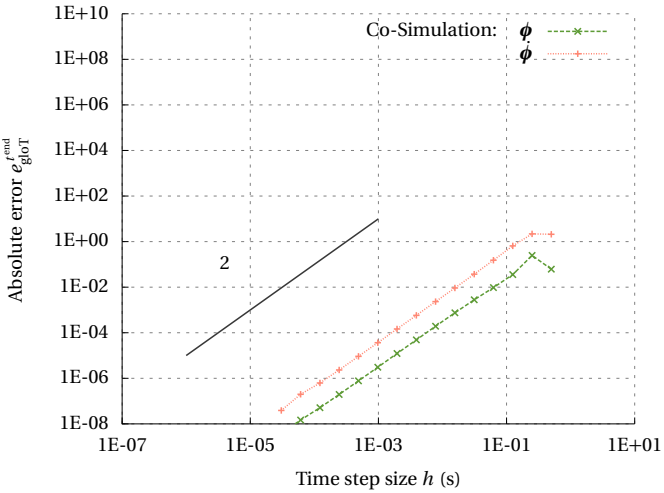


Figure 3.9: Absolute global error (2.80) over time step size for displacement and velocity for generalized-α with time interpolation versus BDF2 method (Example 3.6)

ideas of Busch et al. [25] for automatic time step control can be used in order allow for an automatic estimation.

3.3 Communication Pattern

The definition of co-simulation (Definition 3.1) states the exchange of data between the individual subsystems as an essential property of co-simulation. The choice of the order for the information exchange is part of the discussion in this section. There are mainly two possibilities how to exchange data, namely in a parallel and a serial manner. The parallel information exchange is called Jacobi (JC) and the serial Gauss-Seidel (GS) as their properties are similar to linear iterative solvers.

The implications of the two communication patterns on stability and on the convergence rate of the interface constraint equation set are discussed in detail in Sections 3.4 and 4.6.

3.3.1 Jacobi - Parallel

The Jacobi communication pattern is shown for the iterative case in Figure 3.10. Here, it is assumed that the iterations are converged after the second iteration. However, it is straight forward to extend the figure for any number of iterations. Furthermore, the figure assumes that only two subsystems are present, otherwise the variety of different combinations will distract from the focus of the discussion.

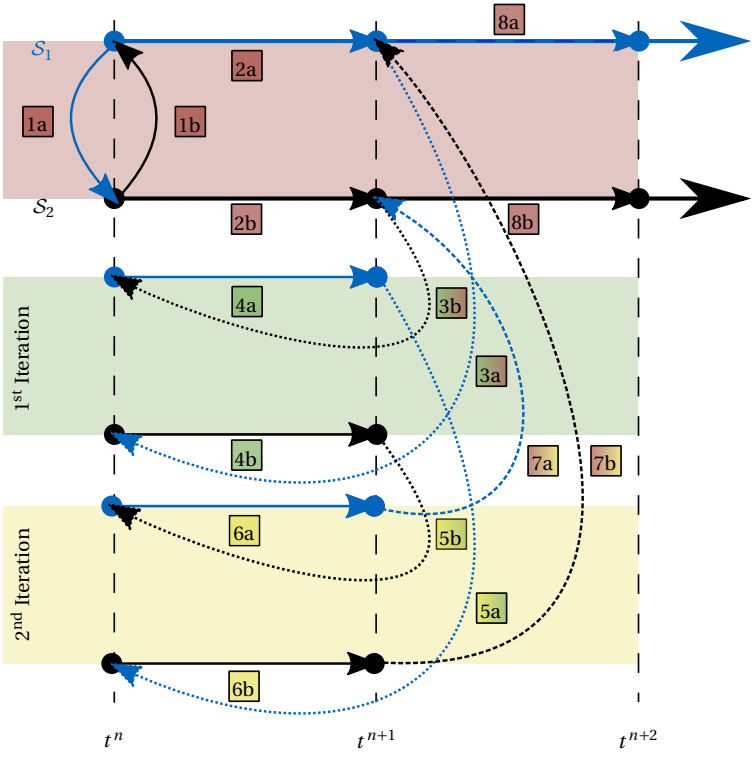


Figure 3.10: Iteration pattern for Jacobi

For an arbitrary multi-code scenario the advantage of the Jacobi pattern is that it allows for parallel execution of the subsystems. In Figure 3.10 this is indicated by a and b. For instance the execution of $\mathcal{S}_1$ and $\mathcal{S}_2$ can be done at the same time (steps 2a and 2b). This is a huge benefit as all subsystems can be executed in parallel and there

is no waiting of an individual subsystem on the outcome of another subsystem necessary within one time step. Hence, this pattern will lead to a co-simulation with no data flow dependency within the time step.

3.3.2 Gauss-Seidel - Serial

In fluid-structure interaction the Gauss-Seidel scheme is commonly used. There, Farhat et al. [45] also calls it conventional serial staggered (CSS) approach within the context of loose coupling. The GS communication pattern for two iterations is plotted in Figure 3.11. However, in a practical use case as many iterations as needed are performed in order to meet a certain convergence criteria.

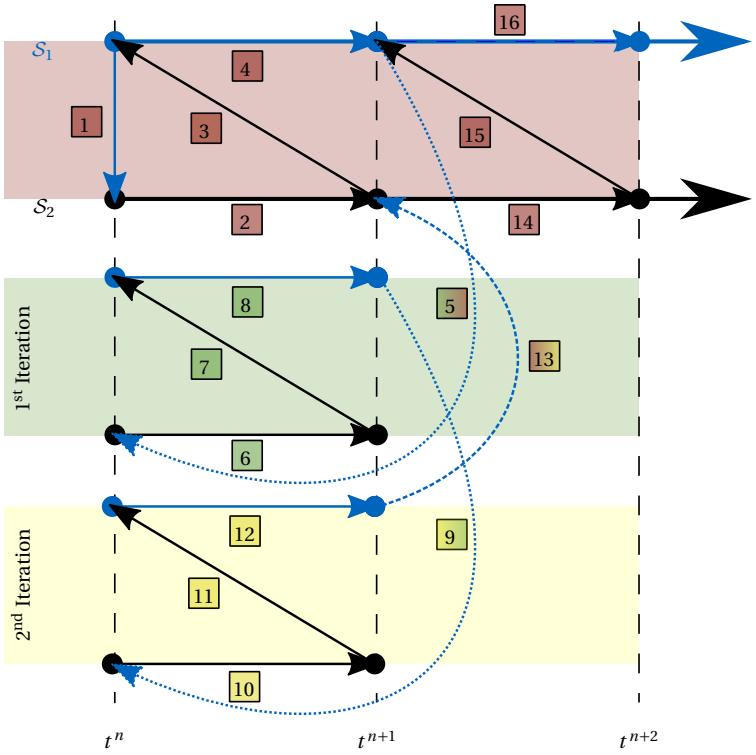


Figure 3.11: Iteration pattern for Gauss-Seidel

The advantage of the GS pattern is a higher convergence rate of the fixed-point iteration method with respect to the JC pattern (see Section 4.6).

However, within a GS pattern based co-simulation the subsystem need to wait on each other. This can already lead to an increase of the overall runtime by a factor of two for a two subsystem co-simulation, where the individual subsystem runtime is equal (e.g. fluid-structure interaction when fluid and structure need the same wall-clock time for one time step). Moreover, the more subsystems there are, the worse this drawback gets.

Furthermore, the design of the GS pattern, where the output of one subsystem is directly feed as an input to another subsystem, has implications on the possible decompositions of the problem as the next section will show.

3.4 Decomposition

In the following the fixed point iteration method should be illustrated in the context of co-simulation. A linear steady state spring example is used. With this example the discussion can be focused on the decomposition. Examples with dynamic effects bring more effects as shown in Section 3.2. Hence, the example in Figure 3.12 helps to demonstrate effects which arise from the decomposition of the monolithic problem.

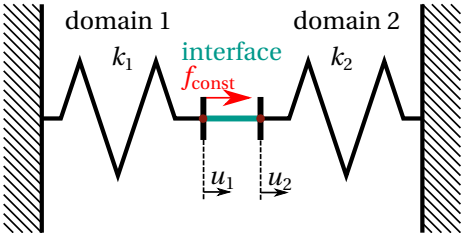


Figure 3.12: Model problem for stability considerations

Four different decompositions of the steady state model problem are presented in the subsequent section in combination with the different communication patterns JC and GS. This results in six different possible iterative methods. These will be called Dirichlet/Dirichlet

(JC), Dirichlet/Neumann (JC and GS), Neumann/Dirichlet (JC and GS) and Neumann/Neumann (JC) borrowing the terminology from corresponding domain decomposition algorithms introduced by Quarteroni et al. [120]. Some of these cases are also well investigated with respect to fluid-structure interaction by Causin et al. [28]. The different decompositions map to the examples in the following manner:

- Dirichlet/Dirichlet → Example 3.7
- Dirichlet/Neumann → Example 3.8
- Neumann/Dirichlet → Example 3.9
- Neumann/Neumann → Example 3.10

Mixed methods which will result in Robin type decompositions (see Badia et al. [8]) are not discussed as they are less general in terms of applicability to general co-simulation problems.

Example 3.7: Fixed-point formulation - Dirichlet-Dirichlet decomposition

In order to give an example for a Dirichlet-Dirichlet decomposition the linear steady state spring example (see Figure 3.12) is used. The general operator notation for the problem is defined as

$$\begin{aligned} \mathcal{S}_1(U_1) &= Y_1, \\ \mathcal{S}_2(U_2) &= Y_2. \end{aligned}$$

Note this decomposition will lead to a Jacobi communication pattern. For two linear springs, which are decomposed in a Dirichlet-Dirichlet manner (each subsystem has a displacement as input) this can be written as

$$\begin{aligned} \mathcal{S}_1(u_1) &= k_1 \cdot u_1 - f_{\text{const}} = f_1, \\ \mathcal{S}_2(u_2) &= k_2 \cdot u_2 \qquad\quad = f_2. \end{aligned}$$

Furthermore, the interface constraint operators (in this case functions) for this problem read

$$\begin{aligned} \mathcal{R}_1 &= \mathcal{I}_1(U_1, U_2) \qquad\quad = u_1 - u_2 = 0, \\ \mathcal{R}_2 &= \mathcal{I}_2\big(\mathcal{S}_1(U_1), \mathcal{S}_2(U_2)\big) = f_1 + f_2 \;\; = 0. \end{aligned}$$

The task is to find the roots to these two real-valued functions. We use Equation (2.26) to formulate the fixed-point iteration

$$\begin{bmatrix} {}^{k+1}U_1 \\ {}^{k+1}U_2 \end{bmatrix} = \begin{bmatrix} {}^{k}U_1 \\ {}^{k}U_2 \end{bmatrix} + \alpha \begin{bmatrix} {}^{k}U_1 - {}^{k}U_2 \\ \mathcal{S}_1\left({}^{k}U_1\right) + \mathcal{S}_2\left({}^{k}U_2\right) \end{bmatrix}.$$

This is equivalent to

$$\begin{bmatrix} {}^{k+1}u_1 \\ {}^{k+1}u_2 \end{bmatrix} = \begin{bmatrix} {}^{k}u_1 \\ {}^{k}u_2 \end{bmatrix} + \alpha \begin{bmatrix} {}^{k}u_1 - {}^{k}u_2 \\ k_1 \cdot {}^{k}u_1 - f_{\text{const}} + k_2 \cdot {}^{k}u_2 \end{bmatrix}.$$

In order to investigate the convergence properties of this fixed-point iteration the system is reformulated to match the structure of Equation (2.22). This renders

$$\begin{bmatrix} {}^{k+1}u_1 \\ {}^{k+1}u_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 1+\alpha & -\alpha \\ \alpha k_1 & 1+\alpha k_2 \end{bmatrix}}_{\boldsymbol{H}} \begin{bmatrix} {}^{k}u_1 \\ {}^{k}u_2 \end{bmatrix} - \begin{bmatrix} 0 \\ \alpha f_{\text{const}} \end{bmatrix}.$$

For a converging fixed-point iteration

$$\rho(\boldsymbol{H}) < 1$$

has to be true. The spectral radius of $\boldsymbol{H}$ is given by

$$\rho(\boldsymbol{H}) = \max\left(\left|\frac{1}{2}\alpha k_2 + \frac{1}{2}\alpha + 1 \pm \frac{1}{2}\alpha\sqrt{k_2^2 - 4k_1 - 2k_2 + 1}\right|\right).$$

The graph of the spectral radius is depicted in Figure 3.14(b).

Example 3.8: Fixed-point formulation - Dirichlet-Neumann decomposition

For an example of a Dirichlet-Neumann decomposition the linear steady state spring example (see Figure 3.12) is used again. Note this decomposition can lead to a Jacobi and a Gauss-Seidel communication pattern. For two linear springs, which are decomposed in a Dirichlet-Neumann manner (one subsystem has a displacement as input the other a force) this can be written as

$$\begin{aligned} \mathcal{S}_1(u_1) &= k_1 \cdot u_1 - f_{\text{const}} = f_1, \\ \mathcal{S}_2(f_2) &= \frac{f_2}{k_2} \qquad\qquad\quad = u_2. \end{aligned}$$

Furthermore, the interface constraint operators (in this case functions) for this problem read

$$\begin{aligned} \mathcal{R}_1 &= \mathcal{I}_1\left(U_1, \mathcal{S}_2(U_2)\right) = u_1 - u_2 = 0, \\ \mathcal{R}_2 &= \mathcal{I}_2\left(\mathcal{S}_1(U_1), U_2\right) = f_1 + f_2 \;\; = 0. \end{aligned}$$

The task is to find the roots to these two real-valued functions. We use Equation (2.26) to formulate the fixed-point iteration

$$\begin{bmatrix} {}^{k+1}U_1 \\ {}^{k+1}U_2 \end{bmatrix} = \begin{bmatrix} {}^{k}U_1 \\ {}^{k}U_2 \end{bmatrix} + \alpha \begin{bmatrix} {}^{k}U_1 - \mathcal{S}_2\left({}^{k}U_2\right) \\ \mathcal{S}_1\left({}^{k}U_1\right) + {}^{k}U_2 \end{bmatrix}.$$

This is equivalent to

$$\begin{bmatrix} {}^{k+1}u_1 \\ {}^{k+1}f_2 \end{bmatrix} = \begin{bmatrix} {}^{k}u_1 \\ {}^{k}f_2 \end{bmatrix} + \alpha \begin{bmatrix} {}^{k}u_1 - \frac{{}^{k}f_2}{k_2} \\ k_1 \cdot {}^{k}u_1 - f_{\text{const}} + {}^{k}f_2 \end{bmatrix}.$$

In order to investigate the convergence properties of this fixed-point iteration the system is reformulated to match the structure of Equation (2.22). This renders

$$\begin{bmatrix} {}^{k+1}u_1 \\ {}^{k+1}f_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 1+\alpha & -\frac{\alpha}{k_2} \\ \alpha k_1 & 1+\alpha \end{bmatrix}}_{\boldsymbol{H}} \begin{bmatrix} {}^{k}u_1 \\ {}^{k}f_2 \end{bmatrix} - \begin{bmatrix} 0 \\ \alpha f_{\text{const}} \end{bmatrix}.$$

The spectral radius of $\boldsymbol{H}$ is given by

$$\rho(\boldsymbol{H}) = \max\left(\left|\pm\frac{\alpha\sqrt{-k_1 k_2}+\alpha k_2 + k_2}{k_2}\right|\right).$$

The graph of the spectral radius is shown in Figure 3.14(c). Note for a Dirichlet-Neumann decomposition of the problem a Gauss-Seidel communication pattern can be used. It can either eliminate the displacement or the force from the interface residual equation.

$$\mathcal{S}_1\left(\mathcal{S}_2\left(f_2\right)\right) = k_1\frac{f_2}{k_2} - f_{\text{const}} = f_1 \qquad \mathcal{S}_2\left(-\mathcal{S}_1\left(u_1\right)\right) = \frac{f_{\text{const}} - k_1 u_1}{k_2} = u_2$$

$$^{k+1}f = {}^{k}f + \alpha\left({}^{k}f + k_1\frac{{}^{k}f}{k_2} - f_{\text{const}}\right) \qquad {}^{k+1}u = {}^{k}u + \alpha\left({}^{k}u - \frac{f_{\text{const}} - k_1\,{}^{k}u}{k_2}\right)$$

$$\rho(H) = \max\left(\left|1+\alpha\left(1+\frac{k_1}{k_2}\right)\right|\right) \qquad \rho(H) = \max\left(\left|1+\alpha\left(1+\frac{k_1}{k_2}\right)\right|\right)$$

The graph of the spectral radius for eliminated displacements is shown in Figure 3.14(d) and for eliminated forces in Figure 3.14(e).

Example 3.9: Fixed-point formulation - Neumann-Dirichlet decomposition

For an example of a Dirichlet-Neumann decomposition the linear steady state spring example (see Figure 3.12) is used again. Note this decomposition can lead to a Jacobi and a Gauss-Seidel communication. For two linear springs, which are decomposed in a Dirichlet-Neumann manner (one subsystem has a displacement as input the other a force) this can be written as

$$\begin{aligned}\mathcal{S}_1\left(f_1\right) &= \frac{f_1 + f_{\text{const}}}{k_1} = u_1,\\ \mathcal{S}_2\left(u_2\right) &= u_2 \cdot k_2 \qquad = f_2.\end{aligned}$$

Furthermore, the interface constraint operators (in this case functions) for this problem read

$$\begin{aligned}\mathcal{R}_1 &= \mathcal{I}_1\left(U_1, \mathcal{S}_2\left(U_2\right)\right) = u_1 - u_2 = 0,\\ \mathcal{R}_2 &= \mathcal{I}_2\left(\mathcal{S}_1\left(U_1\right), U_2\right) = f_1 + f_2 \;\; = 0.\end{aligned}$$

The task is to find the roots to these two real-valued functions. We use Equation (2.26) to formulate the fixed-point iteration

$$\begin{bmatrix}{}^{k+1}U_1\\ {}^{k+1}U_2\end{bmatrix} = \begin{bmatrix}{}^{k}U_1\\ {}^{k}U_2\end{bmatrix} + \alpha\begin{bmatrix}\mathcal{S}_1\left({}^{k}U_1\right) - {}^{k}U_2\\ {}^{k}U_1 + \mathcal{S}_2\left({}^{k}U_2\right)\end{bmatrix}.$$

This is equivalent to

$$\begin{bmatrix}{}^{k+1}f_1\\ {}^{k+1}u_2\end{bmatrix} = \begin{bmatrix}{}^{k}f_1\\ {}^{k}u_2\end{bmatrix} + \alpha\begin{bmatrix}\frac{{}^{k}f_1 + f_{\text{const}}}{k_1} - {}^{k}u_2\\ {}^{k}f_1 + {}^{k}u_2 \cdot k_2\end{bmatrix}.$$

In order to investigate the convergence properties of this fixed-point iteration the system is reformulated to match the structure of Equation (2.22). This renders

$$\begin{bmatrix}{}^{k+1}f_1\\ {}^{k+1}u_2\end{bmatrix} = \underbrace{\begin{bmatrix}1+\frac{\alpha}{k_1} & -\alpha\\ \alpha & 1+\alpha k_2\end{bmatrix}}_{\boldsymbol{H}}\begin{bmatrix}{}^{k}f_1\\ {}^{k}u_2\end{bmatrix} + \begin{bmatrix}\frac{\alpha f_{\text{const}}}{k_1}\\ 0\end{bmatrix}.$$

The spectral radius of $\boldsymbol{H}$ is given by

$$\rho(\boldsymbol{H}) = \max\left(\left|\frac{\alpha k_2 k_1 + \alpha + 2k_1 \pm \alpha\sqrt{k_1^2 k_2^2 - 4k_1^2 - 2k_1 k_2 + 1}}{2k_1}\right|\right).$$

The graph of the spectral radius is shown in Figure 3.15(b). Note for a Dirichlet-Neumann decomposition of the problem a Gauss-Seidel communication pattern can be used. It can either eliminate the displacement or the force from the interface residual equation.

$$\mathcal{S}_2\left(\mathcal{S}_1\left(f_1\right)\right) = \frac{k_2}{k_1}\left(f_1 + f_{\text{const}}\right) = f_2 \qquad \mathcal{S}_1\left(-\mathcal{S}_2\left(u_2\right)\right) = \frac{f_{\text{const}} - k_2 u_2}{k_1} = u_1$$

$${}^{k+1}f = {}^{k}f + \alpha\left({}^{k}f + \frac{k_2}{k_1}\left({}^{k}f + f_{\text{const}}\right)\right) \qquad {}^{k+1}u = {}^{k}u + \alpha\left(\frac{f_{\text{const}} - k_2\,{}^{k}u}{k_1} - {}^{k}u\right)$$

$$\rho(H) = \max\left(\left|1 + \alpha\left(1 + \frac{k_2}{k_1}\right)\right|\right) \qquad \rho(H) = \max\left(\left|1 - \alpha\left(1 + \frac{k_2}{k_1}\right)\right|\right)$$

The graph of the spectral radius for eliminated displacements is shown in Figure 3.15(c) and for eliminated forces in Figure 3.15(d).

Example 3.10: Fixed-point formulation - Neumann-Neumann decomposition

For an example of a Neumann-Neumann decomposition the linear steady state spring example (see Figure 3.12) is used again. The general operator notation for the problem is defined as

$$\mathcal{S}_1\left(U_1\right) = Y_1,$$
$$\mathcal{S}_2\left(U_2\right) = Y_2.$$

Note this decomposition will lead to a Jacobi communication pattern. For two linear springs which are decomposed in a Neumann-Neumann manner (each subsystem has a force as input) this can be written as

$$\mathcal{S}_1\left(f_1\right) = \frac{f_1 + f_{\text{const}}}{k_1} = u_1,$$
$$\mathcal{S}_2\left(f_2\right) = \frac{f_2}{k_2} = u_2.$$

Furthermore, the interface constraint operators (in this case functions) for this problem read

$$\mathcal{R}_1 = \mathcal{I}_1\left(\mathcal{S}_1\left(U_1\right), \mathcal{S}_2\left(U_2\right)\right) = u_1 - u_2 = 0,$$
$$\mathcal{R}_2 = \mathcal{I}_2\left(U_1, U_2\right) = f_1 + f_2 = 0.$$

The task is to find the roots to these two real-valued functions. We use Equation (2.26) to formulate the fixed-point iteration

$$\begin{bmatrix} {}^{k+1}U_1 \\ {}^{k+1}U_2 \end{bmatrix} = \begin{bmatrix} {}^{k}U_1 \\ {}^{k}U_2 \end{bmatrix} + \alpha \begin{bmatrix} \mathcal{S}_1\left({}^{k}U_1\right) - \mathcal{S}_2\left({}^{k}U_2\right) \\ {}^{k}U_1 + {}^{k}U_2 \end{bmatrix}.$$

This is equivalent to

$$\begin{bmatrix} {}^{k+1}f_1 \\ {}^{k+1}f_2 \end{bmatrix} = \begin{bmatrix} {}^{k}f_1 \\ {}^{k}f_2 \end{bmatrix} + \alpha \begin{bmatrix} \frac{{}^{k}f_1 + f_{\text{const}}}{k_1} - \frac{{}^{k}f_2}{k_2} \\ {}^{k}f_1 + {}^{k}f_2 \end{bmatrix}.$$

In order to investigate the convergence properties of this fixed-point iteration the system is reformulated to match the structure of Equation (2.22). This renders

$$\begin{bmatrix} {}^{k+1}f_1 \\ {}^{k+1}f_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 1+\frac{\alpha}{k_1} & -\frac{\alpha}{k_2} \\ \alpha & 1+\alpha \end{bmatrix}}_{\boldsymbol{H}} \begin{bmatrix} {}^{k}f_1 \\ {}^{k}f_2 \end{bmatrix} + \begin{bmatrix} \frac{\alpha}{k_1} f_{\text{const}} \\ 0 \end{bmatrix}.$$

The spectral radius of $\boldsymbol{H}$ is given by

$$\rho(\boldsymbol{H}) = \max\left(\left| \frac{\alpha k_1 k_2 + \alpha k_2 + 2k_1 k_2 \pm \alpha\sqrt{k_1^2 k_2^2 - 4k_1^2 k_2 - 2k_1 k_2^2 + k_2^2}}{2k_1 k_2} \right| \right).$$

The graph of the spectral radius is shown in Figure 3.15(e).

The stability maps for 5 different relaxation factor values for Example 3.7 to Example 3.10 are shown in Figure 3.14 and Figure 3.15. By analyzing these graphs of the steady state model problem the following conclusions can be drawn:

All GS based patterns show a good coverage of the stability. This is even true in the nonphysical stiffness space ($k_\square < 0$). However, the best stability in the physical stiffness space ($k_\square > 0$) is achieved by using the Neumann/Neumann decomposition (see Example 3.10).

The best decomposition in terms of stability and efficiency is usually highly problem dependent. However, in general the decomposition should not lead to an ill-posed subsystem. In order to demonstrate the consequences of a numerically unfavorable decomposition a small example is presented in Figure 3.13. If the input of subsystem 2 is set to be a force f_2, subsystem 2 is not solvable anymore as it is missing a Dirichlet boundary condition and will result in a pure Neumann problem.

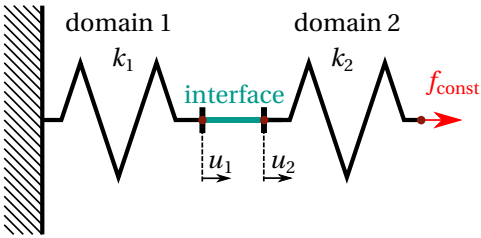
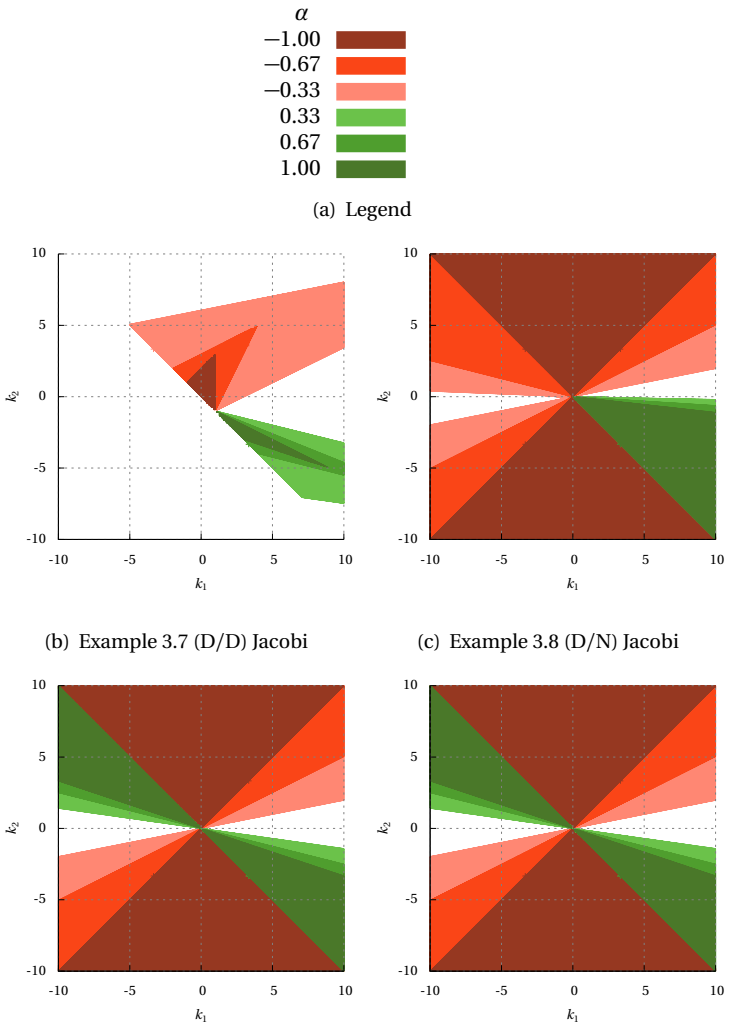


Figure 3.13: Model problem for decomposition considerations



(a) Legend

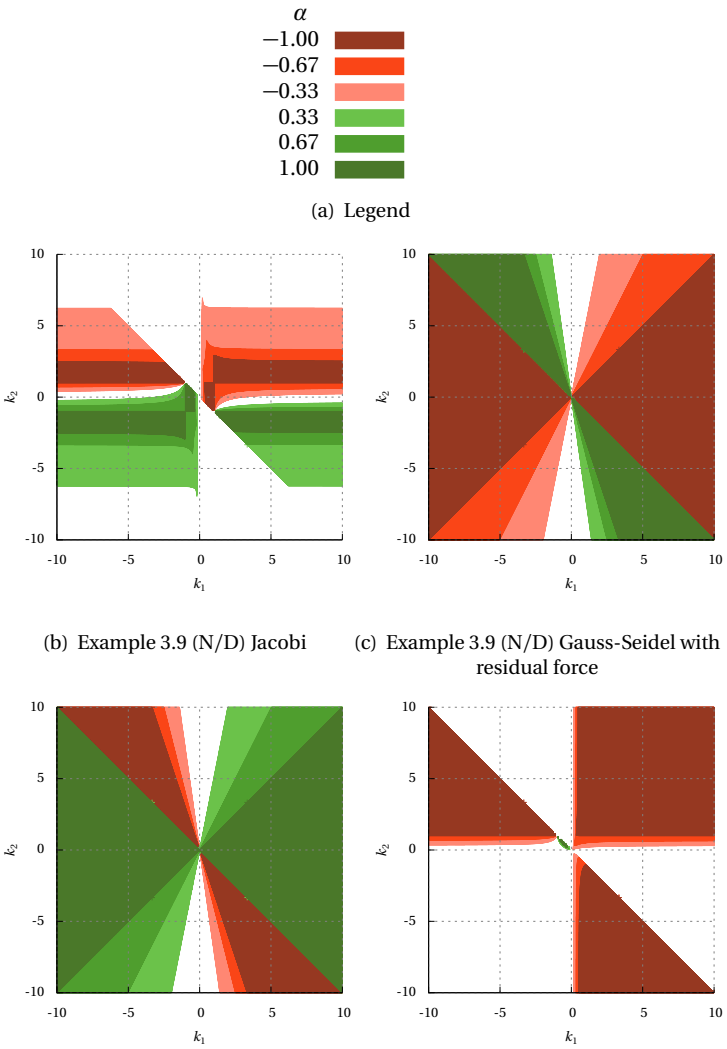
(b) Example 3.7 (D/D) Jacobi

(c) Example 3.8 (D/N) Jacobi

(d) Example 3.8 (D/N) Gauss-Seidel with residual force

(e) Example 3.8 (D/N) Gauss-Seidel with residual displacement

Figure 3.14: Stability maps for the linear steady state spring example (see Figure 3.12) - stable within colored regions



(a) Legend

(b) Example 3.9 (N/D) Jacobi

(c) Example 3.9 (N/D) Gauss-Seidel with residual force

(d) Example 3.9 (N/D) Gauss-Seidel with residual displacement

(e) Example 3.10 (N/N) Jacobi

Figure 3.15: Stability maps for the linear steady state spring example (see Figure 3.12) - stable within colored regions

3.5 Block Diagram

A block diagram is a diagram of a system in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks.[1]

The block diagram is typically the representation of a co-simulation with that the user is confronted with. As it can represent the interface constraint equations. However, using the block diagram for representing the interface constraint equations has the drawback that this can not represent all different possible decompositions of a co-simulation.

For instance if we consider fluid-structure interaction which is from the decompositions point of view similar to the model problem of Section 3.4. A typical block diagram for fluid-structure interaction can be seen in Figure 3.16. As long as the block diagram does only

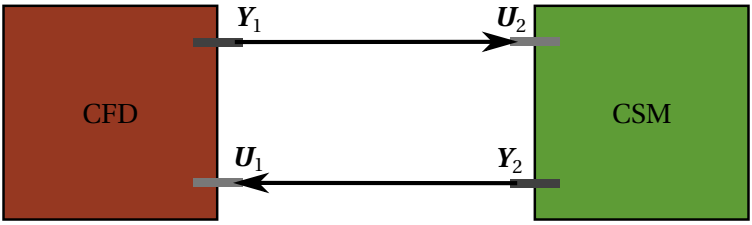


Figure 3.16: Block diagram for fluid-structure interaction

contain subsystem blocks and no algorithm block it is not possible to represent the Dirichlet/Dirichlet or Neumann/Neumann decomposition cases. Hence, for co-simulation scenarios where the full flexibility should be preserved it is best to add an algorithm block which then allows to represent all possible decomposition cases.

The block diagram can also be represented as a graph, where the signal flow for the co-simulation can be described by a directed graph with the subsystems as the nodes and the exchanged data as the edges. This is of advantage when a graphical user interface is implemented as the implementation can leverage graph theory.

[1] SEVOCAB: Software and Systems Engineering Vocabulary. Term: block diagram. retrieved 31 July 2008.

Research is what I'm doing when I don't know what I'm doing.

Wernher von Braun

CHAPTER 4

INTERFACE JACOBIAN-BASED CO-SIMULATION ALGORITHM

The previous chapter discussed the different aspects of co-simulation and the associated drawbacks, such as stability and accuracy issues. Within this work a new kind of co-simulation algorithm is proposed which is based on a Jacobi communication pattern, therefore it is suited for a large number of subsystems. In order to overcome stability issues the algorithm uses interface Jacobians for stabilization. Hence, the algorithm will need more information from the subsystems than fixed point iteration based methods. The proposed algorithm is called Interface Jacobian-based Co-Simulation Algorithm (IJCSA), it is a hybrid algorithm which combines the advantages of the monolithic approach and co-simulation. As long as the subsystems can provide their own interface Jacobian the full modularity of co-simulation is preserved. The interface Jacobian information of each subsystem is assembled at interface level to a global Jacobian which stabilizes the entire co-simulation. The algorithm is formulated in residual form, so it handles cycles within the graph (block-diagram) without special treatment, as the cycles do not occur if the problem is formulated in a residual form (see Bastian et al. [11] and Appendix A).

The following derivations and discussions of the IJCSA are based on Sicklinger et al. [134].

4.1 The Algorithm for two Subsystems

Similar to Section 2.1 the IJCSA is illustrated on the basis of the well-known two-subsystem coupling methodology, it is generalized to a multi-code scenario with vectorial input/output quantities afterwards. The input and output relations for the problem are given by

$$Y_1 = \mathcal{S}_1\left(U_1\right), \tag{4.1}$$

$$Y_2 = \mathcal{S}_2\left(U_2\right). \tag{4.2}$$

Each of the subsystems $\mathcal{S}_1\left(U_1\right)$ and $\mathcal{S}_2\left(U_2\right)$ have state (internal) variables in addition to the output quantities Y_1 and Y_2. The state variables are referred to as X_1 and X_2. The point of departure for the derivation of the IJCSA are the interface constraint equations

$$\mathcal{I}_1\left(\mathcal{S}_1\left(U_1\right), \mathcal{S}_2\left(U_2\right), U_1, U_2\right) = 0, \tag{4.3}$$

$$\mathcal{I}_2\left(\mathcal{S}_1\left(U_1\right), \mathcal{S}_2\left(U_2\right), U_1, U_2\right) = 0. \tag{4.4}$$

The interface constraint operators are essential to the IJCSA, as they reflect the relations between input and output variables. The basic idea of the IJCSA is to formulate the Newton method at interface level where in contrast to a monolithic approach a much smaller system needs to be solved. However the effort for the reduction process needs to be taken into account as well. In order to solve Equation (4.3) and Equation (4.4) with the Newton method the interface residuals need to be defined as

$$\mathcal{R}_1 = \mathcal{I}_1\left(\mathcal{S}_1\left(U_1\right), \mathcal{S}_2\left(U_2\right), U_1, U_2\right), \tag{4.5}$$

$$\mathcal{R}_2 = \mathcal{I}_2\left(\mathcal{S}_1\left(U_1\right), \mathcal{S}_2\left(U_2\right), U_1, U_2\right). \tag{4.6}$$

The solution of Equation (4.3) and Equation (4.4) renders a set of input values $U_\square$ which satisfy the interface constraint equations.

The iteration sequence for the Newton method can be written for a vectorial quantity $\boldsymbol{\phi}$ as

$${}^{m+1}\boldsymbol{\phi} = {}^{m}\boldsymbol{\phi} - \mathcal{J}\left(\boldsymbol{r}\left({}^{m}\boldsymbol{\phi}\right)\right)^{-1}\boldsymbol{r}\left({}^{m}\boldsymbol{\phi}\right), \tag{4.7}$$

which is equivalent to

$$\mathcal{J}\left(\boldsymbol{r}\left({}^{m}\boldsymbol{\phi}\right)\right)\underbrace{\left({}^{m+1}\boldsymbol{\phi}-{}^{m}\boldsymbol{\phi}\right)}_{\Delta^{m+1}\boldsymbol{\phi}}=-\boldsymbol{r}\left({}^{m}\boldsymbol{\phi}\right), \tag{4.8}$$

$$\mathcal{J}\left(\boldsymbol{r}\left({}^{m}\boldsymbol{\phi}\right)\right)\Delta^{m+1}\boldsymbol{\phi}=-\boldsymbol{r}\left({}^{m}\boldsymbol{\phi}\right), \tag{4.9}$$

where $\boldsymbol{\phi}$ is the vector of unknowns, $\boldsymbol{r}$ is the residual vector and $\mathcal{J}$ is the Jacobian operator. The dimension of the Jacobian matrix is only dependent on the number of input variables at the interface level. The iteration index is denoted with m. For the two-subsystem example $\boldsymbol{\phi}$ and $\boldsymbol{r}$ are defined by

$$\boldsymbol{\phi}=\begin{bmatrix}U_1\\U_2\end{bmatrix} \tag{4.10}$$

and

$$\boldsymbol{r}=\begin{bmatrix}\mathcal{R}_1\\\mathcal{R}_2\end{bmatrix}. \tag{4.11}$$

The linear interface system of the Newton method can be written by

$$\begin{bmatrix}\frac{\partial\mathcal{R}_1}{\partial U_1} & \frac{\partial\mathcal{R}_1}{\partial U_2}\\ \frac{\partial\mathcal{R}_2}{\partial U_1} & \frac{\partial\mathcal{R}_2}{\partial U_2}\end{bmatrix}\begin{bmatrix}\Delta U_1\\\Delta U_2\end{bmatrix}=-\begin{bmatrix}\mathcal{R}_1\\\mathcal{R}_2\end{bmatrix}. \tag{4.12}$$

If we assume the following interface conditions

$$\mathcal{I}_1\left(\mathcal{S}_2\left(U_2\right),U_1\right)=U_1-Y_2=0, \tag{4.13}$$

$$\mathcal{I}_2\left(\mathcal{S}_1\left(U_1\right),U_2\right)=U_2-Y_1=0, \tag{4.14}$$

Equation System (4.12) reads

$$\begin{bmatrix}\mathrm{I} & -\frac{\partial\mathcal{S}_2}{\partial U_2}\\ -\frac{\partial\mathcal{S}_1}{\partial U_1} & \mathrm{I}\end{bmatrix}\begin{bmatrix}\Delta U_1\\\Delta U_2\end{bmatrix}=-\begin{bmatrix}\mathcal{R}_1\\\mathcal{R}_2\end{bmatrix}. \tag{4.15}$$

It is evident that the final form of the interface equation system is

$$\begin{bmatrix} \mathrm{I} & -\frac{\partial Y_2}{\partial U_2} \\ -\frac{\partial Y_1}{\partial U_1} & \mathrm{I} \end{bmatrix} \begin{bmatrix} \Delta U_1 \\ \Delta U_2 \end{bmatrix} = - \begin{bmatrix} \mathcal{R}_1 \\ \mathcal{R}_2 \end{bmatrix}. \tag{4.16}$$

Where $\partial Y_\square / \partial U_\square$ represents the derivative of the output with respect to the input.

The algorithm for the case of two subsystems is described in Algorithm 4.1 on the basis of the previous considerations. Note that if the global Jacobian matrix $\boldsymbol{J}_{\text{global}}$ is set to the identity matrix Algorithm 4.1 reduces to the classical fixed-point iteration scheme. This is shown in Algorithm 4.2.

4.2 Generalization of the Concept

In Section 2.1 the generalized interface residual components were defined by

$$\boldsymbol{\mathcal{R}}_i = \mathcal{I}_i \left(\mathcal{S}_j \left(\boldsymbol{U}_j \right), \boldsymbol{U}_j, j = 1, ..., r \right) \quad i = 1, ..., r. \tag{4.17}$$

With this definition the generalized global interface Jacobian system is given by

$$\begin{bmatrix} \frac{\partial \mathcal{I}_1}{\partial \mathcal{S}_1} \frac{\partial \mathcal{S}_1}{\partial \boldsymbol{U}_1} + \frac{\partial \mathcal{I}_1}{\partial \boldsymbol{U}_1} & \cdots & \frac{\partial \mathcal{I}_1}{\partial \mathcal{S}_r} \frac{\partial \mathcal{S}_r}{\partial \boldsymbol{U}_r} + \frac{\partial \mathcal{I}_1}{\partial \boldsymbol{U}_r} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathcal{I}_r}{\partial \mathcal{S}_1} \frac{\partial \mathcal{S}_1}{\partial \boldsymbol{U}_1} + \frac{\partial \mathcal{I}_r}{\partial \boldsymbol{U}_1} & \cdots & \frac{\partial \mathcal{I}_r}{\partial \mathcal{S}_r} \frac{\partial \mathcal{S}_r}{\partial \boldsymbol{U}_r} + \frac{\partial \mathcal{I}_r}{\partial \boldsymbol{U}_r} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{U}_1 \\ \vdots \\ \Delta \boldsymbol{U}_r \end{bmatrix} = - \begin{bmatrix} \boldsymbol{\mathcal{R}}_1 \\ \vdots \\ \boldsymbol{\mathcal{R}}_r \end{bmatrix}. \tag{4.18}$$

This is equivalent to

$$\begin{bmatrix} \frac{\partial \mathcal{I}_1}{\partial \boldsymbol{Y}_1} \frac{\partial \boldsymbol{Y}_1}{\partial \boldsymbol{U}_1} + \frac{\partial \mathcal{I}_1}{\partial \boldsymbol{U}_1} & \cdots & \frac{\partial \mathcal{I}_1}{\partial \boldsymbol{Y}_r} \frac{\partial \boldsymbol{Y}_r}{\partial \boldsymbol{U}_r} + \frac{\partial \mathcal{I}_1}{\partial \boldsymbol{U}_r} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathcal{I}_r}{\partial \boldsymbol{Y}_1} \frac{\partial \boldsymbol{Y}_1}{\partial \boldsymbol{U}_1} + \frac{\partial \mathcal{I}_r}{\partial \boldsymbol{U}_1} & \cdots & \frac{\partial \mathcal{I}_r}{\partial \boldsymbol{Y}_r} \frac{\partial \boldsymbol{Y}_r}{\partial \boldsymbol{U}_r} + \frac{\partial \mathcal{I}_r}{\partial \boldsymbol{U}_r} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{U}_1 \\ \vdots \\ \Delta \boldsymbol{U}_r \end{bmatrix} = - \begin{bmatrix} \boldsymbol{\mathcal{R}}_1 \\ \vdots \\ \boldsymbol{\mathcal{R}}_r \end{bmatrix}. \tag{4.19}$$

Algorithm 4.1: Interface Jacobian-based Co-Simulation Algorithm for a 2-code example

// Time loop

for $n = 0$ *to* $n = n_{\text{end}}$ **do**

  // Iteration loop

  for $m = 0$ *to* $m = m_{\text{end}}$ **do**

    // Solve for all subsystems in parallel

    ${}^{m}Y_1^{n+1} = \mathcal{S}_1\left({}^{m}U_1^{n}\right)$

    ${}^{m}Y_2^{n+1} = \mathcal{S}_2\left({}^{m}U_2^{n}\right)$

    // Compute and check residual

    $${}^{m}\boldsymbol{r}^{n} = \begin{bmatrix} {}^{m}\mathcal{R}_1^{n} \\ {}^{m}\mathcal{R}_2^{n} \end{bmatrix} = \begin{bmatrix} {}^{m}U_1^{n} - {}^{m}Y_2^{n+1} \\ {}^{m}U_2^{n} - {}^{m}Y_1^{n+1} \end{bmatrix}$$

    if $\|{}^{m}\boldsymbol{r}^{n}\|_{\epsilon} < \epsilon$ **then**

      break

    // Get parts of interface Jacobian

    ${}^{m}J_1^{n} = \mathcal{J}\left(\mathcal{S}_1\left({}^{m}U_1^{n}\right)\right) = {}^{m}\frac{\partial Y_1}{\partial U_1}{}^{n} := \frac{\partial Y_1}{\partial U_1}$

    ${}^{m}J_2^{n} = \mathcal{J}\left(\mathcal{S}_2\left({}^{m}U_2^{n}\right)\right) = {}^{m}\frac{\partial Y_2}{\partial U_2}{}^{n} := \frac{\partial Y_2}{\partial U_2}$

    // Assemble global interface Jacobian

    $${}^{m}\boldsymbol{J}_{\text{global}}^{n} = \mathcal{A}_{\mathcal{J}}\left({}^{m}J_1^{n}, {}^{m}J_2^{n}\right) = \begin{bmatrix} 1 & -\frac{\partial Y_2}{\partial U_2} \\ -\frac{\partial Y_1}{\partial U_1} & 1 \end{bmatrix}$$

    // Solve for corrector

    ${}^{m}\boldsymbol{J}_{\text{global}}^{n} \cdot {}^{m}\Delta\boldsymbol{c}^{n} = -{}^{m}\boldsymbol{r}^{n}$

    // Apply corrector

    ${}^{m+1}U_1^{n} = {}^{m}U_1^{n} + {}^{m}\Delta c_1^{n}$

    ${}^{m+1}U_2^{n} = {}^{m}U_2^{n} + {}^{m}\Delta c_2^{n}$

  // Initial solution for next time step

  ${}^{0}U_1^{n+1} = \mathcal{E}\left({}^{m_{\text{end}}+1}U_1^{k} \quad k = 0, \ldots, n\right)$

  ${}^{0}U_2^{n+1} = \mathcal{E}\left({}^{m_{\text{end}}+1}U_2^{k} \quad k = 0, \ldots, n\right)$

Algorithm 4.2: Constant under-relaxation algorithm for a 2-code example

// Time loop

1 **for** $n = 0$ *to* $n = n_{\text{end}}$ **do**

// Iteration loop

2 **for** $m = 0$ *to* $m = m_{\text{end}}$ **do**

// Solve all subsystems (parallel)

3 $^{m}Y_1^{n+1} = \mathcal{S}_1\left(^{m}U_1^{n}\right)$

4 $^{m}Y_2^{n+1} = \mathcal{S}_2\left(^{m}U_2^{n}\right)$

// Compute and check residual

5 $$^{m}\boldsymbol{r}^{n} = \begin{bmatrix} ^{m}\mathcal{R}_1^{n} \\ ^{m}\mathcal{R}_2^{n} \end{bmatrix} = \begin{bmatrix} ^{m}U_1^{n} - ^{m}Y_2^{n+1} \\ ^{m}U_2^{n} - ^{m}Y_1^{n+1} \end{bmatrix}$$

6 **if** $\|^{m}\boldsymbol{r}^{n}\|_{\epsilon} < \epsilon$ **then**

7 break

// Apply update

8 $^{m+1}U_1^{n} = {}^{m}U_1^{n} + \alpha\, {}^{m}\mathcal{R}_1^{n}$

9 $^{m+1}U_2^{n} = {}^{m}U_2^{n} + \alpha\, {}^{m}\mathcal{R}_2^{n}$

// Initial solution for next time step

10 $^{0}U_1^{n+1} = \mathcal{E}\left(^{m_{\text{end}}+1}U_1^{k} \quad k = 0, \ldots, n\right)$

11 $^{0}U_2^{n+1} = \mathcal{E}\left(^{m_{\text{end}}+1}U_2^{k} \quad k = 0, \ldots, n\right)$

With the global interface Jacobian matrix given in Equation (4.19) the general version of the IJCSA is presented in Algorithm 4.3.

Note that the assembly operator is denoted by $\mathcal{A}_{\mathcal{J}}$. The previous derivations show that the entries of the interface Jacobian matrix are combined by two basic kinds of derivatives. The first part ($\partial\mathcal{I}_j/\partial\boldsymbol{Y}_i$ and $\partial\mathcal{I}_j/\partial\boldsymbol{U}_i$) of the interface Jacobian needs to be provided by the interface for each subsystem. This part reflects the interface constraint equations. The second part needs to be provided by each individual subsystem $\partial\boldsymbol{Y}_i/\partial\boldsymbol{U}_i$, it represents the subsystem's sensitivity. Hence, it is a measure for the change of the output of the subsystem if the subsystem's input is perturbed.

Algorithm 4.3: Interface Jacobian-based Co-Simulation Algorithm

// Time loop

for $n = 0$ *to* $n = n_{\text{end}}$ **do**

  // Iteration loop

  for $m = 0$ *to* $m = m_{\text{end}}$ **do**

    // Solve for all subsystems in parallel

    ${}^{m}\boldsymbol{Y}_i^{n+1} = \mathcal{S}_i\left({}^{m}\boldsymbol{U}_i^{n}\right)$

    // Compute and check residual

    $${}^{m}\boldsymbol{r}^{n} = \begin{bmatrix} {}^{m}\mathcal{R}_1^{n} \\ \vdots \\ {}^{m}\mathcal{R}_r^{n} \end{bmatrix} = \begin{bmatrix} \mathcal{I}_1\left({}^{m}\boldsymbol{Y}_j^{n+1}, {}^{m}\boldsymbol{U}_j^{n} \quad j = 1,\ldots,r\right) \\ \vdots \\ \mathcal{I}_r\left({}^{m}\boldsymbol{Y}_j^{n+1}, {}^{m}\boldsymbol{U}_j^{n} \quad j = 1,\ldots,r\right) \end{bmatrix}$$

    if $\|{}^{m}\boldsymbol{r}^{n}\|_{\epsilon} < \epsilon$ **then**

      break

    // Get parts of interface Jacobian

    $${}^{m}\boldsymbol{J}_i^{n} = \mathcal{J}\left(\mathcal{S}_i\left({}^{m}\boldsymbol{U}_i^{n}\right)\right) = {}^{m}\frac{\partial \boldsymbol{Y}_i}{\partial \boldsymbol{U}_i}{}^{n} := \frac{\partial \boldsymbol{Y}_i}{\partial \boldsymbol{U}_i}$$

    // Assemble global interface Jacobian

    $${}^{m}\boldsymbol{J}_{\text{global}}^{n} = \mathcal{A}_{\mathcal{J}}\left({}^{m}\boldsymbol{J}_i^{n}\right) = \begin{bmatrix} \frac{\partial \mathcal{I}_1}{\partial \boldsymbol{Y}_1}\frac{\partial \boldsymbol{Y}_1}{\partial \boldsymbol{U}_1} + \frac{\partial \mathcal{I}_1}{\partial \boldsymbol{U}_1} & \cdots & \frac{\partial \mathcal{I}_1}{\partial \boldsymbol{Y}_r}\frac{\partial \boldsymbol{Y}_r}{\partial \boldsymbol{U}_r} + \frac{\partial \mathcal{I}_1}{\partial \boldsymbol{U}_r} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathcal{I}_r}{\partial \boldsymbol{Y}_1}\frac{\partial \boldsymbol{Y}_1}{\partial \boldsymbol{U}_1} + \frac{\partial \mathcal{I}_r}{\partial \boldsymbol{U}_1} & \cdots & \frac{\partial \mathcal{I}_r}{\partial \boldsymbol{Y}_r}\frac{\partial \boldsymbol{Y}_r}{\partial \boldsymbol{U}_r} + \frac{\partial \mathcal{I}_r}{\partial \boldsymbol{U}_r} \end{bmatrix} \quad i = 1,\ldots,r$$

    // Solve for corrector

    ${}^{m}\boldsymbol{J}_{\text{global}}^{n} \cdot {}^{m}\Delta\boldsymbol{c}^{n} = -{}^{m}\boldsymbol{r}^{n}$

    // Apply corrector

    ${}^{m+1}\boldsymbol{U}_i^{n} = {}^{m}\boldsymbol{U}_i^{n} + {}^{m}\Delta\boldsymbol{c}_i^{n} \quad i = 1,\ldots,r$

  // Initial solution for next time step

  ${}^{0}\boldsymbol{U}_i^{n+1} = \mathcal{E}\left({}^{m_{\text{end}}+1}\boldsymbol{U}_i^{k} \quad k = 0,\ldots,n\right) \quad i = 1,\ldots,r$

4.3 Efficiency Enhancements

The most time is generally spend for the solution of all subsystems in Algorithm 4.3 in line 3. This is in general a computationally expensive operation. The cost for the solution operation can be dramatically reduced by using an approximation.

During the extraction process of ${}^{\partial \boldsymbol{Y}_i}/_{\partial \boldsymbol{U}_i}$ a by-product is ${}^{\partial \boldsymbol{X}_i}/_{\partial \boldsymbol{U}_i}$ which is a derivative of the state variables with respect to the input variable. The derivative is used in the following Taylor series approximation:

$$ {}^{m+1}\boldsymbol{X}_i = {}^{m}\boldsymbol{X}_i + {}^{m}\frac{\partial \boldsymbol{X}_i}{\partial \boldsymbol{U}_i}\, {}^{m}\Delta \boldsymbol{U}_i + \mathcal{O}\left({}^{m}\Delta \boldsymbol{U}_i^2\right) \tag{4.20} $$

For the output variables this approximation is not used. The approximated state variables are used in combination with a nonlinear map g between the state and the output variables. A more general g may also depend on the input variables. That is

$$ {}^{m+1}\boldsymbol{Y}_i = g\left({}^{m+1}\boldsymbol{X}_i, {}^{m+1}\boldsymbol{U}_i\right). \tag{4.21} $$

This relation is the one which is used inside the simulators $\mathcal{S}_{\approx i}$.

Assuming that g is nonlinear, Algorithm 4.4 represents a more efficient version of Algorithm 4.3. Note that only line 3 of Algorithm 4.3 is modified to

$$ {}^{m}\boldsymbol{Y}_i^{n+1} \approx \mathcal{S}_{\approx i}\left({}^{m-1}\Delta \boldsymbol{c}_i^{n}, {}^{m}\boldsymbol{U}_i^{n}\right). \tag{4.22} $$

Equation (4.20) can be modified using the notation and indices in Algorithm 4.4 to

$$ {}^{m}\boldsymbol{X}_{\approx i} \approx {}^{m-1}\boldsymbol{X}_i + {}^{m-1}\frac{\partial \boldsymbol{X}_i}{\partial \boldsymbol{U}_i}\, {}^{m-1}\Delta \boldsymbol{c}_i^{n}, \tag{4.23} $$

and for the output update Equation (4.21) we arrive at

$$ {}^{m}\boldsymbol{Y}_i \approx g\left({}^{m}\boldsymbol{X}_{\approx i}, {}^{m}\boldsymbol{U}_i\right). \tag{4.24} $$

4.4 Usability Enhancements - Jacobian Approximation

In some situations it is cumbersome or even impossible to provide interface Jacobian within the subsystems. In order to apply the IJCSA

Algorithm 4.4: Enhanced version of the IJCSA

// Time loop

1 **for** $n = 0$ *to* $n = n_{\text{end}}$ **do**

// Iteration loop

2 **for** $m = 0$ *to* $m = m_{\text{end}}$ **do**

3 **if** $m = 0$ **then**

// Solve for all subsystems in parallel

4 $${}^{m}\boldsymbol{Y}_i^{n+1} = \mathcal{S}_i\left({}^{m}\boldsymbol{U}_i^{n}\right)$$

5 **else**

// Approx. solve for all subsystems in parallel

6 $${}^{m}\boldsymbol{Y}_i^{n+1} = \mathcal{S}_{\approx i}\left({}^{m-1}\Delta\boldsymbol{c}_i^{n}, {}^{m}\boldsymbol{U}_i^{n}\right)$$

// Compute and check residual

7 $${}^{m}\boldsymbol{r}^{n} = \begin{bmatrix} {}^{m}\mathcal{R}_1^{n} \\ \vdots \\ {}^{m}\mathcal{R}_r^{n} \end{bmatrix} = \begin{bmatrix} \mathcal{I}_1\left({}^{m}\boldsymbol{Y}_j^{n+1}, {}^{m}\boldsymbol{U}_j^{n} \quad j = 1,...,r\right) \\ \vdots \\ \mathcal{I}_r\left({}^{m}\boldsymbol{Y}_j^{n+1}, {}^{m}\boldsymbol{U}_j^{n} \quad j = 1,...,r\right) \end{bmatrix}$$

8 **if** $\|{}^{m}\boldsymbol{r}^{n}\|_{\epsilon} < \epsilon$ **then**

9 break

// Get parts of interface Jacobian

10 $${}^{m}\boldsymbol{J}_i^{n} = \mathcal{J}\left(\mathcal{S}_i\left({}^{m}\boldsymbol{U}_i^{n}\right)\right) = {}^{m}\frac{\partial \boldsymbol{Y}_i}{\partial \boldsymbol{U}_i}{}^{n} := \frac{\partial \boldsymbol{Y}_i}{\partial \boldsymbol{U}_i}$$

// Assemble global interface Jacobian

11 $${}^{m}\boldsymbol{J}_{\text{global}}^{n} = \mathcal{A}_{\boldsymbol{J}}\left({}^{m}\boldsymbol{J}_i^{n}\right) = \begin{bmatrix} \frac{\partial \mathcal{I}_1}{\partial \boldsymbol{Y}_1}\frac{\partial \boldsymbol{Y}_1}{\partial \boldsymbol{U}_1} + \frac{\partial \mathcal{I}_1}{\partial \boldsymbol{U}_1} & \cdots & \frac{\partial \mathcal{I}_1}{\partial \boldsymbol{Y}_r}\frac{\partial \boldsymbol{Y}_r}{\partial \boldsymbol{U}_r} + \frac{\partial \mathcal{I}_1}{\partial \boldsymbol{U}_r} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathcal{I}_r}{\partial \boldsymbol{Y}_1}\frac{\partial \boldsymbol{Y}_1}{\partial \boldsymbol{U}_1} + \frac{\partial \mathcal{I}_r}{\partial \boldsymbol{U}_1} & \cdots & \frac{\partial \mathcal{I}_r}{\partial \boldsymbol{Y}_r}\frac{\partial \boldsymbol{Y}_r}{\partial \boldsymbol{U}_r} + \frac{\partial \mathcal{I}_r}{\partial \boldsymbol{U}_r} \end{bmatrix} \quad i = 1,...,r$$

// Solve for corrector

12 $${}^{m}\boldsymbol{J}_{\text{global}}^{n} \cdot {}^{m}\Delta\boldsymbol{c}^{n} = -{}^{m}\boldsymbol{r}^{n}$$

// Apply corrector

13 $${}^{m+1}\boldsymbol{U}_i^{n} = {}^{m}\boldsymbol{U}_i^{n} + {}^{m}\Delta\boldsymbol{c}_i^{n}$$

// Initial solution for next time step

14 $${}^{0}\boldsymbol{U}_i^{n+1} = \mathcal{E}\left({}^{m_{\text{end}}+1}\boldsymbol{U}_i^{k} \quad k = 0,...,n\right) \quad i = 1,...,r$$

to general co-simulation scenarios it is needed to handle such situations where some subsystems are not able to provide their interface Jacobian.

In the case where the interface quantities are scalar quantities the secant method can give a very good compromise between stability and efficiency on the one hand and applicability of the IJCSA to complicated subsystems (e.g. CFD solvers) on the other hand. The secant extension of the IJCSA is shown in Algorithm 4.5. It was successfully applied and tested for wind turbines (see Section 5.3).

4.5 Interface Jacobian Extraction

Static condensation methods introduced by Wilson [157] can be exploited for the extraction of interface Jacobians. These methods can be implemented for sparse systems in an efficient manner without the need of the explicit computation of any matrix inverse as noted by Felippa [50].

If we assume the following linear relation which is modeled by a subsystem

$$\underbrace{\begin{bmatrix} \boldsymbol{A_{YY}} & \boldsymbol{A_{YX}} \\ \boldsymbol{A_{XY}} & \boldsymbol{A_{XX}} \end{bmatrix}}_{\boldsymbol{A}} \begin{bmatrix} \boldsymbol{Y} \\ \boldsymbol{X} \end{bmatrix} = \begin{bmatrix} \boldsymbol{U} \\ \boldsymbol{b} \end{bmatrix}, \tag{4.25}$$

where $\boldsymbol{A_{YY}}$ and $\boldsymbol{A_{XX}}$ are square invertible matrices. The state vector of the subsystem is abbreviated with $\boldsymbol{X}$, the input vector with $\boldsymbol{U}$ and the output vector of the subsystem is denoted by $\boldsymbol{Y}$.

The second equation of Equation System (4.25) can be written as

$$\boldsymbol{X} = \boldsymbol{A_{XX}}^{-1}\left(\boldsymbol{b} - \boldsymbol{A_{XY}}\,\boldsymbol{Y}\right). \tag{4.26}$$

If this equation is plugged into the first equation of Equation System (4.25) equation

$$\underbrace{\left(\boldsymbol{A_{YY}} - \boldsymbol{A_{YX}}\boldsymbol{A_{XX}}^{-1}\boldsymbol{A_{XY}}\right)}_{\boldsymbol{A}_{\text{Schur}}}\boldsymbol{Y} = \boldsymbol{U} - \boldsymbol{A_{YX}}\boldsymbol{A_{XX}}^{-1}\boldsymbol{b} \tag{4.27}$$

is obtained.

Algorithm 4.5: Secant version of the IJCSA

// Time loop

for $n = 0$ *to* $n = n_{\text{end}}$ **do**

  // Iteration loop

  for $m = 0$ *to* $m = m_{\text{end}}$ **do**

    // Solve for all subsystems in parallel

    ${}^{m}Y_i^{n+1} = \mathcal{S}_i\left({}^{m}U_i^{n}\right)$

    // Compute and check residual

    $${}^{m}\boldsymbol{r}^{n} = \begin{bmatrix} {}^{m}\mathcal{R}_1^{n} \\ \vdots \\ {}^{m}\mathcal{R}_r^{n} \end{bmatrix} = \begin{bmatrix} \mathcal{I}_1\left({}^{m}Y_j^{n+1}, {}^{m}U_j^{n} \quad j = 1,\ldots,r\right) \\ \vdots \\ \mathcal{I}_r\left({}^{m}Y_j^{n+1}, {}^{m}U_j^{n} \quad j = 1,\ldots,r\right) \end{bmatrix}$$

    if $\|{}^{m}\boldsymbol{r}^{n}\|_{\epsilon} < \epsilon$ **then**

      break

    // Get parts of interface Jacobian if possible

    ${}^{m}J_i^{n} = \mathcal{J}\left(\mathcal{S}_i\left({}^{m}U_i^{n}\right)\right) = {}^{m}\frac{\partial Y_i}{\partial U_i}{}^{n} := \frac{\partial Y_i}{\partial U_i}$

    // Use secant approximation if not possible to approximate parts of interface Jacobian

    if $m = 0$ **then**

      ${}^{m}J_i^{n} \approx {}^{\text{init}}J_i^{n}$

    else

      $${}^{m}J_i^{n} \approx \frac{{}^{m}Y_i^{n+1} - {}^{m-1}Y_i^{n+1}}{{}^{m}U_i^{n} - {}^{m-1}U_i^{n}}$$

    // Assemble global interface Jacobian

    $${}^{m}\boldsymbol{J}_{\text{global}}^{n} = \mathcal{A}_{\mathcal{J}}\left({}^{m}J_i^{n}\right) = \begin{bmatrix} \frac{\partial \mathcal{I}_1}{\partial Y_1}\frac{\partial Y_1}{\partial U_1} + \frac{\partial \mathcal{I}_1}{\partial U_1} & \cdots & \frac{\partial \mathcal{I}_1}{\partial Y_r}\frac{\partial Y_r}{\partial U_r} + \frac{\partial \mathcal{I}_1}{\partial U_r} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathcal{I}_r}{\partial Y_1}\frac{\partial Y_1}{\partial U_1} + \frac{\partial \mathcal{I}_r}{\partial U_1} & \cdots & \frac{\partial \mathcal{I}_r}{\partial Y_r}\frac{\partial Y_r}{\partial U_r} + \frac{\partial \mathcal{I}_r}{\partial U_r} \end{bmatrix} \quad i = 1,\ldots,r$$

    // Solve for corrector

    ${}^{m}\boldsymbol{J}_{\text{global}}^{n} \cdot {}^{m}\Delta\boldsymbol{c}^{n} = -{}^{m}\boldsymbol{r}^{n}$

    // Apply corrector

    ${}^{m+1}U_i^{n} = {}^{m}U_i^{n} + {}^{m}\Delta c_i^{n}$

  // Initial solution for next time step

  ${}^{0}U_i^{n+1} = \mathcal{E}\left({}^{m_{\text{end}}+1}U_i^{k} \quad k = 0,\ldots,n\right) \quad i = 1,\ldots,r$

Here the Schur complement of $\boldsymbol{A}$ with respect to $\boldsymbol{A}_{XX}$ is denoted by $\boldsymbol{A}_{\text{Schur}}$. The name Schur complement goes back to the definition of the mathematician Issai Schur whose original work was published in German, an English translation of his work can be found in Schur [131]. Furthermore, it can be shown (e.g. Boyd et al. [20] and Strang [144]) that if $\boldsymbol{A}$ is positive definite $\boldsymbol{A}_{\text{Schur}}$ is also positive definite. For the determinants the following holds (see Zhang [162])

$$\det(\boldsymbol{A}) = \det\left(\boldsymbol{A}_{XX}\right) \cdot \det\left(\boldsymbol{A}_{\text{Schur}}\right). \tag{4.28}$$

The formulation of the system involving the Schur complement has better numerical properties than the original Equation System (4.25), which is discussed by Badia et al. [8]. This also holds for the extraction of the interface Jacobian as the interface Jacobian is given by

$$\frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{U}} = \boldsymbol{A}_{\text{Schur}}{}^{-1}. \tag{4.29}$$

Note that these ideas can be carried over to the nonlinear regime by using the so called global sensitivity equation (GSE). Please see Section 4.7.1 for more details.

4.6 Stability Considerations

An undamped two-mass oscillator system is used to represent a coupled second-order initial value problem. The model problem is used to analyze the stability properties of the IJCSA and compare them with classical fixed-point iteration algorithms (Gauss-Seidel and Jacobi). The setting of the model problem is shown in Figure 4.1. The coupled system is decomposed in two domains (domain 1 and domain 2).

The interface of the two domains is formed by a massless rigid link between the masses m_1 and m_2. The properties of the partitioned systems are derived from the monolithic quantities. The monolithic stiffness is denoted by k and the monolithic mass is denoted by m. Hence the eigenfrequency of the monolithic system is given by $\omega = \sqrt{k/m}$.

With the initial conditions $u(0) = 0$ and $\dot{u}(0) = 1$ the analytical solution of the monolithic problem is given by $u(t) = 1/\omega \sin(\omega t)$, where $u(t)$ is the time dependent interface displacement. Let us introduce f being the force exerted by the mass m_2 onto m_1.

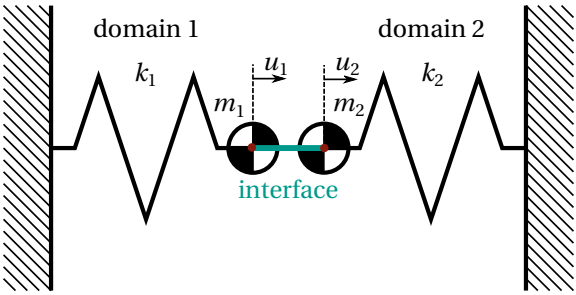


Figure 4.1: Model problem for stability considerations

Furthermore, it is needed to define the quantities m_1, m_2, k_1 and k_2 of the two subsystems. Therefore, the dimensionless parameters β_1 and β_2 are introduced where $\{\beta_1 \in \mathbb{R} \,|\, 0 \leq \beta_1 \leq 1\}$ and $\{\beta_2 \in \mathbb{R} \,|\, 0 \leq \beta_2 \leq 1\}$. Thus for domain 1 the system parameters are given by

$$m_1 = \beta_1 m, \tag{4.30}$$

$$k_1 = \beta_2 k. \tag{4.31}$$

For domain 2 one has

$$m_2 = \left(1-\beta_1\right) m, \tag{4.32}$$

$$k_2 = \left(1-\beta_2\right) k. \tag{4.33}$$

With these parameters the governing ordinary differential equations for the two domains can be defined. Hence, for domain 1 we have

$$m_1 \ddot{u}_1 + k_1 u_1 = f_1, \tag{4.34}$$

$$\beta_1 m \ddot{u}_1 + \beta_2 k u_1 = f_1, \tag{4.35}$$

and for domain 2 we arrive at

$$m_2 \ddot{u}_2 + k_2 u_2 = -f_2, \tag{4.36}$$

$$\left(1-\beta_1\right) m \ddot{u}_2 + \left(1-\beta_2\right) k u_2 = -f_2. \tag{4.37}$$

Equations (4.35) and (4.37) can be reformulated in operator notation to

$$Y_1 = \mathcal{S}_1\left(U_1\right), \tag{4.38}$$

$$Y_2 = \mathcal{S}_2\left(U_2\right). \tag{4.39}$$

For a Dirichlet-Neumann decomposition of the model problem (see Section 3.4) the interface constraint equations read

$$u_1 = u_2 = u, \tag{4.40}$$
$$f_1 = f_2. \tag{4.41}$$

Having that, the output and the input quantities are defined for the model problem, as follows

$$U_1 = f_1, \tag{4.42}$$
$$U_2 = u_2, \tag{4.43}$$
$$Y_1 = u_1, \tag{4.44}$$
$$Y_2 = f_2. \tag{4.45}$$

Hence, the interface constraint operators are given by

$$\mathcal{I}_1\left(\mathcal{S}_2\left(U_2\right), U_1\right) = U_1 - Y_2 = 0, \tag{4.46}$$
$$\mathcal{I}_2\left(\mathcal{S}_1\left(U_1\right), U_2\right) = U_2 - Y_1 = 0. \tag{4.47}$$

In order to be able to discuss numerical stability properties of the different coupling algorithms a numerical time integrator needs to be employed. As this is a model problem the BE (see Section 3.1.1) time integrator is chosen, however any time integrator could be taken for the discussion, in general. The BE method is used for both domains. Generally, different time integrators for the different domains may be present (see Section 3.2). The stability discussion is not limited to the particular choice of BE for both domains. However, it keeps the formulas to a reasonable level of complexity without harming the generality.

With the help of the BE time integrator, Equations (4.35) and (4.37) can be discretized in time. This leads to

$$u_1^{n+1} = \frac{h^2}{\beta_1 m + \beta_2 k h^2} f_1^{n+1} + \frac{\beta_1 m}{\beta_1 m + \beta_2 k h^2}\left(2u_1^n - u_1^{n-1}\right), \tag{4.48}$$

$$f_2^{n+1} = \frac{\beta_1 m - m - k h^2 + \beta_2 k h^2}{h^2} u_2^{n+1} + \frac{\left(1-\beta_1\right) m}{h^2}\left(2u_2^n - u_2^{n-1}\right). \tag{4.49}$$

4.6.1 Gauss-Seidel Fixed-Point Iterations

As already discussed in Section 3.3, in fluid-structure interaction the GS scheme is commonly used. Therefore we analyze this pattern first. The communication pattern for two iterations is plotted in Figure 3.11. As we focus on iterative schemes in this work iteration counters need to be added to Equations (4.48) and (4.49). Furthermore, the partitioned treatment requires that the time index of the input quantities has to be changed which results in

$$ {}^{m}u_1^{n+1} = \frac{h^2}{\beta_1 m + \beta_2 k h^2} {}^{m}f_1^{n} + \frac{\beta_1 m}{\beta_1 m + \beta_2 k h^2}\left(2u_1^n - u_1^{n-1}\right), \quad (4.50) $$

$$ {}^{m}f_2^{n+1} = \frac{\beta_1 m - m - k h^2 + \beta_2 k h^2}{h^2} {}^{m}u_2^{n} + \frac{\left(1-\beta_1\right) m}{h^2}\left(2u_2^n - u_2^{n-1}\right). \quad (4.51) $$

By analyzing the Gauss-Seidel communication pattern (see steps 3, 7 and 11 in Figure 3.11) it is obvious that the following holds:

$$ {}^{m}f_1^{n} = {}^{m}f_2^{n+1} \quad (4.52) $$

Therefore Equations (4.50) and (4.51) can be coupled into one equation. After sorting the variables one has

$$ {}^{m}u^{n+1} = \underbrace{\frac{\beta_1 m - m - k h^2 + \beta_2 k h^2}{\beta_1 m + \beta_2 k h^2}}_{H_{\mathrm{GS}}} {}^{m}u^{n} + \frac{m}{\beta_1 m + \beta_2 k h^2}\left(2u^n - u^{n-1}\right), \quad (4.53) $$

where H_{GS} is the so called convergence factor. Note that the operator notation for this problem can be simplified if the Gauss-Seidel communication pattern is used. Hence Equations (4.38) and (4.39) can be combined to

$$ Y_1 = \mathcal{S}_1\left(\mathcal{S}_2\left(U_2\right)\right). \quad (4.54) $$

For convergence of the Gauss-Seidel iterations the following must hold (see Section 2.2):

$$ |H_{\mathrm{GS}}| < 1 \quad (4.55) $$

Note that for this particular model problem, which has one scalar interface variable, an optimal relaxation factor can be determined to provide the converged solution of the coupled problem after the first iteration. If relaxation is included the update rule is modified and instead of using

$$^{m+1}U_2 = {}^{m}Y_1, \tag{4.56}$$

the relaxation factor $\{\alpha \in \mathbb{R}\}$ is incorporated in the update rule as follows

$$^{m+1}U_2 = {}^{m}U_2 + \alpha\left({}^{m}U_2 - {}^{m}Y_1\right). \tag{4.57}$$

With the help of the convergence factor this can be written as

$$^{m+1}U_2 = {}^{m}U_2 + \alpha\left({}^{m}U_2 - H_{\mathrm{GS}}\,{}^{m}U_2 - c\right), \tag{4.58}$$

$$^{m+1}U_2 = \left(1 + \alpha - \alpha H_{\mathrm{GS}}\right){}^{m}U_2 - c\,. \tag{4.59}$$

Similar to Section 2.2.1 for optimal convergence

$$1 + \alpha - \alpha H_{\mathrm{GS}} = 0, \tag{4.60}$$

must be true.

Hence the optimal relaxation factor for the model problem is given by

$$\alpha_{\mathrm{opt}} = \frac{1}{H_{\mathrm{GS}} - 1}. \tag{4.61}$$

This is also discussed in Joosten et al. [83, p. 767].

4.6.2 Jacobi Fixed-Point Iterations

In order to determine the convergence factor for the Jacobi scheme, Equations (4.50) and (4.51) are written in the following matrix from

$$\begin{bmatrix} {}^{m}u_1^{n+1} \\ {}^{m}f_2^{n+1} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & \frac{h^2}{\beta_1 m + \beta_2 k h^2} \\ \frac{\beta_1 m - m - k h^2 + \beta_2 k h^2}{h^2} & 0 \end{bmatrix}}_{\mathbf{H}_{\mathrm{JC}}} \begin{bmatrix} {}^{m}u_1^{n} \\ {}^{m}f_2^{n} \end{bmatrix} + \begin{bmatrix} \frac{\beta_1 m}{\beta_1 m + \beta_2 k h^2} \\ \frac{(1-\beta_1)m}{h^2} \end{bmatrix} \left(2u_2^n - u_2^{n-1}\right). \tag{4.62}$$

As the convergence factor $\mathbf{H}_{\mathrm{JC}}$ is no longer a scalar the convergence criterion (4.55) needs to be extended to a multi-dimensional space. In a multi-dimensional space the spectral radius is introduced. Hence, Equation (2.24) must be satisfied for convergence. Thus we have

$$\rho\left(\mathbf{H}_{\mathrm{JC}}\right) < 1, \tag{4.63}$$

for the Jacobi scheme. Therefore Jacobi iterations will converge for the model problem if and only if

$$\sqrt{\left| \frac{h^2}{\beta_1 m + \beta_2 k h^2} \frac{\beta_1 m - m - k h^2 + \beta_2 k h^2}{h^2} \right|} < 1, \tag{4.64}$$

is satisfied. This may be further simplified to

$$\sqrt{\left| \frac{\beta_1 - 1 - \omega^2 h^2 + \beta_2 \omega^2 h^2}{\beta_1 + \beta_2 \omega^2 h^2} \right|} < 1, \tag{4.65}$$

which is the square root of the convergence factor for the Gauss-Seidel H_{GS} pattern.

4.6.3 Interface Jacobian-based Co-Simulation Algorithm

As the IJCSA is applied to a linear problem it needs to converge to the correct solution within one iteration. In order to proof that the Newton method converges for the model problem Theorem 2.12 of Deuflhard [36] is used. As the model problem is linear the only assumption left to check is for which model properties $\boldsymbol{J}_{\mathrm{global}}$ is invertible.

$$\boldsymbol{J}_{\mathrm{global}} = \begin{bmatrix} 1 & -\frac{\partial Y_2}{\partial U_2} \\ -\frac{\partial Y_1}{\partial U_1} & 1 \end{bmatrix} \tag{4.66}$$

For the model problem the interface Jacobian matrix is defined by

$$\boldsymbol{J}_{\mathrm{global}} = \begin{bmatrix} 1 & -\frac{\beta_1 m - m - k h^2 + \beta_2 k h^2}{h^2} \\ -\frac{h^2}{\beta_1 m + \beta_2 k h^2} & 1 \end{bmatrix} = \text{const.} \tag{4.67}$$

For a 2×2 matrix the inverse is given by

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{a\,d - b\,c} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}. \tag{4.68}$$

That means that the global interface Jacobian is not invertible if

$$\frac{\beta_1 m - m - k h^2 + \beta_2 k h^2}{h^2} \frac{h^2}{\beta_1 m + \beta_2 k h^2} = 1. \tag{4.69}$$

This is only the case if

$$m + k h^2 = 0, \tag{4.70}$$

which is not the case for a physical meaningful parameter set. Therefore we conclude that the IJCSA is unconditionally stable for the model problem.

4.6.4 Discussion of the Stability Properties

In the following the results of the stability considerations for GS without relaxation, JC without relaxation and the IJCSA should be discussed. One interesting limit case is $h \to 0$, this case represents the consistency of the algorithm. Thus we have

$$\lim_{h \to 0} \left(\rho \left(\boldsymbol{H}_{\mathrm{JC}} \right) \right) = \sqrt{\left| \frac{\beta_1 - 1}{\beta_1} \right|}, \tag{4.71}$$

$$\lim_{h \to 0} \left(\left| H_{\mathrm{GS}} \right| \right) = \left| \frac{\beta_1 - 1}{\beta_1} \right|. \tag{4.72}$$

For $h \to 0$ IJCSA is converging as long as the mass of the model problem is larger than zero $m > 0$.

As the convergence factor of the GS pattern is the square of the one of the JC pattern it is enough to analyze the convergence factor of the GS method. The limit of stability is the same for the JC and the GS method, the difference is the rate of convergence. The smaller the convergence factor is the faster the convergence rate is i.e. GS converges faster than the JC pattern. The convergence factor for GS may also be written as

$$\left| H_{\mathrm{GS}} \right| = \left| \frac{\beta_1 - 1 - \omega^2 h^2 + \beta_2 \omega^2 h^2}{\beta_1 + \beta_2 \omega^2 h^2} \right| = \rho \left(\boldsymbol{H}_{\mathrm{JC}} \right)^2. \tag{4.73}$$

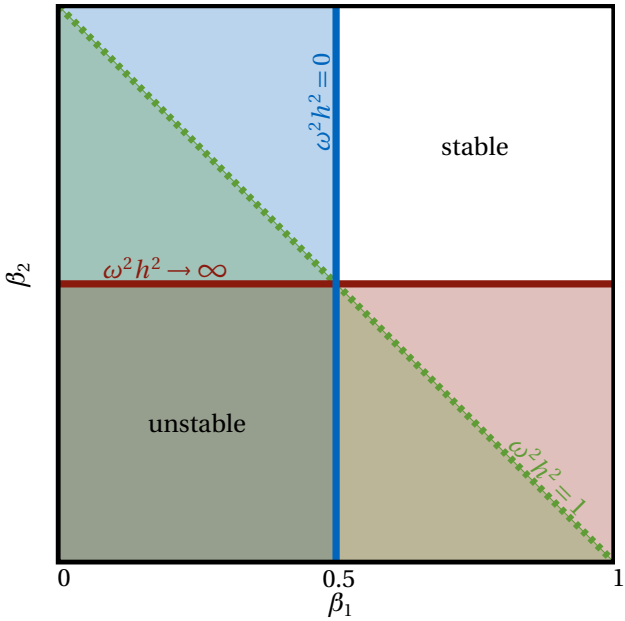


Figure 4.2: Stability limit graph

In Figure 4.2 the stability limits are shown for $\omega^2 h^2 = 0$, $\omega^2 h^2 = 1$ and $\omega^2 h^2 \to \infty$. As $\omega^2 h^2$ is increased the stability limit is rotated around $\{\beta_1 = 0.5, \beta_2 = 0.5\}$. The lower left part is always the unstable region. Thus, if $\{\beta_1 \in [0..0.5] \wedge \beta_2 \in [0..0.5]\}$ the parameter set is always unstable independent of the choice of $\omega^2 h^2$.

At the consistency bounds the stability is determined only by the mass ratio between domain 1 and domain 2. This is aligned with observations by Causin et al. [28] in fluid-structure interaction. This shows that IJCSA is the best choice as it will render the coupled solution of the model problem within one iteration independent of any system parameter as long as they are physical.

Note that the stability is not just determined by the choice of the communication pattern (e.g. Gauss-Seidel or Jacobi), but also the choice of the decomposition is decisive (see Section 3.4). It can be shown that the IJCSA is unconditionally stable for the model problem independent of the choice of the decomposition.

4.7 Examples

The following section presents different examples which illustrate the performance of the IJCSA in more detail. The most general form of the IJCSA is presented in Algorithm 4.4. By analyzing this algorithm three core capabilities of the subsystems can be identified. The most obvious capability of a "black-box" like subsystem is requested in line 4 which is a simple solution of one time step. This ability is indicated with `doSolve`. `doSolve` may indicate a method name in an implementation of the IJCSA. The core method of the IJCSA is given in line 10 where the subsystems need to deliver an interface Jacobian (`getInterfaceJacobian`). In line 6 the solution of all subsystems is performed. This can be done in an efficient way by using Approximation (4.20). Hence the subsystem method may be called `doApproximatedSolve`.

These three core capabilities of the subsystems are illustrated in the following examples. All presented examples are nonlinear as most co-simulation scenarios involve nonlinear problems.

4.7.1 Truss versus Truss Problem

IJCSA is used to solve a nonlinear transient structural mechanics problem. Within the problem, two truss systems are coupled. Each truss system is nonlinear due to the choice of the Hencky strain measure. The material law is chosen as St. Venant-Kirchhoff. A sketch of the problem setting is given in Figure 4.3.

The problem is similar to the one used in the stability Section 4.6. Therefore we have

$$Y_1 = \mathcal{S}_1\left(U_1\right), \tag{4.74}$$

$$Y_2 = \mathcal{S}_2\left(U_2\right), \tag{4.75}$$

and the interface relations are given by

$$\mathcal{I}_1\left(\mathcal{S}_2\left(U_2\right), U_1\right) = U_1 - Y_2 = 0, \tag{4.76}$$

$$\mathcal{I}_2\left(\mathcal{S}_1\left(U_1\right), U_2\right) = U_2 - Y_1 = 0. \tag{4.77}$$

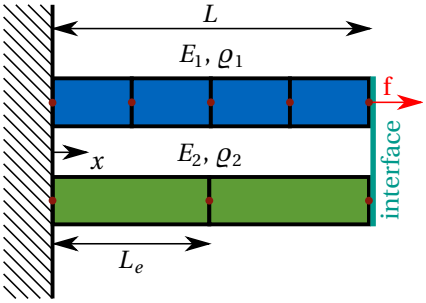


Figure 4.3: Setup of Truss versus Truss problem

The 1D Hencky Truss Element

We start with the quadratic internal energy functional for the truss

$$\Pi_{\text{internal}} = \frac{1}{2}\int_V E\,\varepsilon^2\,\mathrm{d}V, \tag{4.78}$$

where E is the Young's modulus and ε is the Hencky strain. The weak form of Equation (4.78) reads

$$\delta\Pi_{\text{internal}} = \int_V E\,\varepsilon\big(u(x)\big)\,\delta\varepsilon\big(u(x)\big)\,\mathrm{d}V. \tag{4.79}$$

This may be split over a sum of finite elements (n_{e} being the total number of elements)

$$\delta\Pi_{\text{internal}} = \sum_{\text{e}=1}^{n_{\text{e}}} \delta\Pi_{\text{e}_{\text{internal}}}. \tag{4.80}$$

If linear shape functions are used for the approximation of the displacement field $u(x)$, the discretized weak form of a one dimensional truss with Hencky strain measure is

$$\delta\Pi^{\text{h}}_{\text{e}_{\text{internal}}} = \underbrace{\begin{bmatrix} p_1(\boldsymbol{u}) \\ p_2(\boldsymbol{u}) \end{bmatrix}^{\top}}_{\boldsymbol{p}_{\text{e}}(\boldsymbol{u})} \begin{bmatrix} \delta u_1 \\ \delta u_2 \end{bmatrix}, \tag{4.81}$$

where $\boldsymbol{p}_\mathrm{e}(\boldsymbol{u})$ is called element internal force vector. Considering concentrated external forces only we can write

$$\delta \Pi^\mathrm{h} = \underbrace{\begin{bmatrix} p_1(\boldsymbol{u}) \\ p_2(\boldsymbol{u}) \end{bmatrix}^\top}_{\mathbf{p}_\mathrm{e}(\mathbf{u})} \begin{bmatrix} \delta u_1 \\ \delta u_2 \end{bmatrix} - \boldsymbol{f}_\mathrm{e}^\top \begin{bmatrix} \delta u_1 \\ \delta u_2 \end{bmatrix} = 0, \tag{4.82}$$

where the element external force vector is denoted by $\boldsymbol{f}_\mathrm{e}$. As this must be true for arbitrary test functions δu the nonlinear equation system to be solved can be written as

$$\widetilde{\boldsymbol{r}}_{\mathrm{subsys_e}}(\boldsymbol{u}) = \begin{bmatrix} p_1(\boldsymbol{u}) \\ p_2(\boldsymbol{u}) \end{bmatrix} - \boldsymbol{f}_\mathrm{e} = \mathbf{0}. \tag{4.83}$$

Dynamic Extension of the 1D Hencky Truss Element

In order to add dynamics effects to the nonlinear Residual (4.83), the D'Alembert forces must also be added. Thus the semi-discretized version of the dynamic nonlinear residual is

$$\boldsymbol{r}_\mathrm{subsys}(\boldsymbol{u}) = \boldsymbol{M}\ddot{\boldsymbol{u}} + \widetilde{\boldsymbol{r}}_\mathrm{subsys}(\boldsymbol{u}) = \boldsymbol{M}\ddot{\boldsymbol{u}} + \boldsymbol{p}(\boldsymbol{u}) - \boldsymbol{f} = \mathbf{0}. \tag{4.84}$$

In order to discretize the vector of accelerations $\ddot{\boldsymbol{u}}$ the BE operator is used. Thus Equation (4.84) can be written as

$$\boldsymbol{r}_\mathrm{subsys}\left(\boldsymbol{u}^{n+1}\right) = \boldsymbol{M}\frac{1}{h^2}\left(\boldsymbol{u}^{n+1} - 2\boldsymbol{u}^n + \boldsymbol{u}^{n-1}\right) + \boldsymbol{p}\left(\boldsymbol{u}^{n+1}\right) - \boldsymbol{f} = \mathbf{0}. \tag{4.85}$$

Solving the Nonlinear Equation System with the Newton Method

The iteration sequence for the Newton method is given by

$${}_{l+1}\boldsymbol{u} = {}_l\boldsymbol{u} - \mathcal{J}\left(\boldsymbol{r}_\mathrm{subsys}\left({}_l\boldsymbol{u}\right)\right)^{-1} \boldsymbol{r}_\mathrm{subsys}\left({}_l\boldsymbol{u}\right). \tag{4.86}$$

The Jacobian of the nonlinear residual vector $\mathcal{J}\left(\boldsymbol{r}_\mathrm{subsys}\left({}_l\boldsymbol{u}\right)\right)$ is given by

$$\mathcal{J}\left(\boldsymbol{r}_\mathrm{subsys}\left({}_l\boldsymbol{u}\right)\right) = \underbrace{\boldsymbol{M}\frac{1}{h^2} + \underbrace{\mathcal{J}\left(\boldsymbol{p}(\boldsymbol{u})\right)}_{\boldsymbol{K}(\boldsymbol{u})}}_{\boldsymbol{A}(\boldsymbol{u})}, \tag{4.87}$$

where $\boldsymbol{K}(\boldsymbol{u})$ is referred to as tangent stiffness matrix and given by

$$\boldsymbol{K}(\boldsymbol{u}) = \mathcal{A}_{\mathrm{e}=1}^{n_{\mathrm{e}}} \boldsymbol{K}_{\mathrm{e}}(\boldsymbol{u}). \tag{4.88}$$

Here $\mathcal{A}$ is the assembly operator and $\boldsymbol{K}_{\mathrm{e}}(\boldsymbol{u})$ is given by

$$\boldsymbol{K}_{\mathrm{e}}(\boldsymbol{u}) = \begin{bmatrix} \dfrac{\partial p_1(\boldsymbol{u})}{\partial u_1} & \dfrac{\partial p_1(\boldsymbol{u})}{\partial u_2} \\ \dfrac{\partial p_2(\boldsymbol{u})}{\partial u_1} & \dfrac{\partial p_2(\boldsymbol{u})}{\partial u_2} \end{bmatrix}. \tag{4.89}$$

The element system matrices and vectors can be summarized to

$$\boldsymbol{p}_{\mathrm{e}}(\boldsymbol{u}) = E L_{\mathrm{e}} \frac{\ln\left(\dfrac{L_{\mathrm{e}} + u_2 - u_1}{L_{\mathrm{e}}}\right)}{L_{\mathrm{e}} + u_2 - u_1} \begin{bmatrix} 1 \\ -1 \end{bmatrix}. \tag{4.90}$$

Where u_1 is the displacement at element node 1 and u_2 the displacement at element node 2. The element length is denoted by L_{e}. Form the internal force vector we get the stiffness matrix

$$\boldsymbol{K}_{\mathrm{e}}(\boldsymbol{u}) = E L_{\mathrm{e}} \frac{\ln\left(\dfrac{L_{\mathrm{e}} + u_2 - u_1}{L_{\mathrm{e}}}\right) - 1}{\left(L_{\mathrm{e}} + u_2 - u_1\right)^2} \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{4.91}$$

The lumped form of the mass matrix is given by

$$\boldsymbol{M}_{\mathrm{e}} = \varrho L_{\mathrm{e}} \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}. \tag{4.92}$$

Next, the algorithms inside the subsystems for the Truss versus Truss problem are described in detail. Thus the implementation of the IJCSA for the Truss versus Truss problem is illustrated.

Subsystem 1 $Y_1 = \mathcal{S}_1\left(U_1\right)$

Subsystem 1 expects an interface force as an input. The subsystem additionally applies a constant force f to the truss system at the interface node (see Figure 4.3). The nonlinear residual function for subsystem

Table 4.1: Input/Output quantities for subsystem 1

Subsystem	Input U	Output Y
	N	m
$\mathcal{S}_1$	$f_{\text{interface}}$	$u_{\text{interface}}$

1 is given by

$$\boldsymbol{r}_{\text{subsys}}\left(\boldsymbol{u}^{n+1}\right)=\boldsymbol{M}\frac{1}{h^2}\left(\boldsymbol{u}^{n+1}-2\boldsymbol{u}^{n}+\boldsymbol{u}^{n-1}\right)+\boldsymbol{p}\left(\boldsymbol{u}^{n+1}\right)-\boldsymbol{f}=\boldsymbol{0}. \tag{4.93}$$

Please also note that the vector of unknowns is sorted such that the last degree of freedom is the interface degree of freedom. The internal degrees of freedom, in the following, are referred to as state variables $\boldsymbol{X}_1=\begin{bmatrix}u_1 & u_2 & \cdots & u_{i-1}\end{bmatrix}^\top$. Therefore, Equation (4.93) becomes

$$\boldsymbol{r}_{\text{subsys}}\left(\boldsymbol{u}^{n+1}\right)=\boldsymbol{M}\frac{1}{h^2}\left(\underbrace{\begin{bmatrix}\boldsymbol{X}_1\\ u_{\text{interface}}=Y_1\end{bmatrix}}_{\boldsymbol{u}^{n+1}}-2\boldsymbol{u}^{n}+\boldsymbol{u}^{n-1}\right)+$$
$$\boldsymbol{p}\left(\boldsymbol{u}^{n+1}\right)-\underbrace{\begin{bmatrix}\boldsymbol{0}\\ f\end{bmatrix}}_{\boldsymbol{f}}+\underbrace{\begin{bmatrix}\boldsymbol{0}\\ f_{\text{interface}}=U_1\end{bmatrix}}_{\boldsymbol{f}_{\text{interface}}}=\boldsymbol{0}. \tag{4.94}$$

`doSolve` The `doSolve` function of subsystem 1 expects an interface force as input (U_1) and outputs the displacement of the interface degree of freedom (Y_1). In order to solve the system of nonlinear equations for every time step the function applies a local Newton method to Equation (4.94)

$$\mathcal{J}\left(\boldsymbol{r}_{\text{subsys}}\left({}_l\boldsymbol{u}^{n+1}\right)\right)\quad {}_{l+1}\Delta\boldsymbol{u}^{n+1}=-\boldsymbol{r}_{\text{subsys}}\left({}_l\boldsymbol{u}^{n+1}\right). \tag{4.95}$$

Note that l is the local Newton iteration counter and n is the time step counter.

`getInterfaceJacobian` This function computes the interface Jacobian and updates the internal Jacobian for the state variables. The basis for the following considerations is the so called global sensitivity equation (see Henrik [74] and Sobieszczanski-Sobieski [140]), which reads

$$\frac{\partial \boldsymbol{r}_{\text{subsys}}(\boldsymbol{u})}{\partial \boldsymbol{u}} \frac{\partial \boldsymbol{u}}{\partial \iota} = -\frac{\partial \boldsymbol{r}_{\text{subsys}}}{\partial \iota}. \tag{4.96}$$

In the case of the IJCSA ι is represented via the interface input variable. For subsystem 1 this is the interface force $f_{\text{interface}}$. Hence Equation (4.96) for subsystem 1 may be written as

$$\begin{aligned} \boldsymbol{A} \frac{\partial \boldsymbol{u}}{\partial f_{\text{interface}}} &= -\frac{\partial \boldsymbol{r}_{\text{subsys}}}{\partial f_{\text{interface}}}, \\ \frac{\partial \boldsymbol{u}}{\partial f_{\text{interface}}} &= -\boldsymbol{A}^{-1} \frac{\partial \boldsymbol{r}_{\text{subsys}}}{\partial f_{\text{interface}}}. \end{aligned} \tag{4.97}$$

Equation (4.97) reads in more detail

$$\begin{bmatrix} \dfrac{\partial \boldsymbol{X}_1}{\partial f_{\text{interface}}} \\ \dfrac{\partial u_{\text{interface}}}{\partial f_{\text{interface}}} \end{bmatrix} = -\boldsymbol{A}^{-1} \begin{bmatrix} \boldsymbol{0} \\ \\ 1 \end{bmatrix}.$$

Here

$$\frac{\partial \boldsymbol{X}_1}{\partial f_{\text{interface}}} = \begin{bmatrix} \dfrac{\partial u_1}{\partial f_{\text{interface}}} \\ \dfrac{\partial u_2}{\partial f_{\text{interface}}} \\ \vdots \\ \dfrac{\partial u_{i-1}}{\partial f_{\text{interface}}} \end{bmatrix}$$

and

$$\frac{\partial u_{\text{interface}}}{\partial f_{\text{interface}}} = \frac{\partial Y_1}{\partial U_1}.$$

Thus, a Jacobian extraction involves a linear solution of the system. However, if within `doSolve` a direct solver is used, the decomposition of $\boldsymbol{A}$ is reused for the Jacobian extraction.

`doApproximatedSolve` The `doApproximatedSolve` method provides an efficient approximation for the solution of the subsystem

$$^{m+1}\boldsymbol{X}_1 \approx {}^{m}\boldsymbol{X}_1 + \left|\frac{\partial \boldsymbol{X}_1}{\partial U_1}\right|^{m} {}^{m}\Delta U_1, \tag{4.98}$$

$$^{m+1}\boldsymbol{X}_1 \approx {}^{m}\boldsymbol{X}_1 + \left|\frac{\partial \boldsymbol{X}_1}{\partial f_{\text{interface}}}\right|^{m} {}^{m}\Delta f_{\text{interface}}, \tag{4.99}$$

$$^{m+1}Y_1 = g\left({}^{m+1}\boldsymbol{X}_1, {}^{m}U_1\right), \tag{4.100}$$

which does not involve a new integration in subsystem 1.

Note that the approximation is applied for the state variables only. Be aware that m is the iteration counter for the interface iterations. As the finite element method is used for the discretization of this problem and the trial and test functions are chosen to be linear, the g function is simply a nonlinear relation between the displacements of the nodes $n_e - 1$ and n_e.

Subsystem 2 $Y_2 = \mathcal{S}_2\left(U_2\right)$

In contrast to subsystem 1, subsystem 2 needs the interface displacement as input (see Table 4.2) . The nonlinear residual function for subsystem 2 is given by

$$\hat{\boldsymbol{r}}_{\text{subsys}}\left(\boldsymbol{u}^{n+1}, \boldsymbol{f}_{\text{interface}}\right) = \boldsymbol{M}\frac{1}{h^2}\left(\boldsymbol{u}^{n+1} - 2\boldsymbol{u}^{n} + \boldsymbol{u}^{n-1}\right) + \boldsymbol{p}\left(\boldsymbol{u}^{n+1}\right) - \boldsymbol{f}_{\text{interface}} = \boldsymbol{0}. \tag{4.101}$$

Table 4.2: Input/Output quantities for subsystem 2

Subsystem	Input U	Output Y
	N	m
$\mathcal{S}_2$	$u_{\text{interface}}$	$f_{\text{interface}}$

The internal degrees of freedom are in the following referred to as state variables $\boldsymbol{X}_2 = \begin{bmatrix} u_1 & u_2 & \cdots & u_{i-1} \end{bmatrix}^\top$ once more

$$\hat{\boldsymbol{r}}_{\text{subsys}}\left(\boldsymbol{u}^{n+1}, \boldsymbol{f}_{\text{interface}}\right) = \boldsymbol{M}\frac{1}{h^2}\left(\underbrace{\begin{bmatrix} \boldsymbol{X}_2 \\ u_{\text{interface}} = U_2 \end{bmatrix}}_{\mathbf{u}^{n+1}} - 2\boldsymbol{u}^n + \boldsymbol{u}^{n-1}\right) + \boldsymbol{p}\left(\boldsymbol{u}^{n+1}\right) - \underbrace{\begin{bmatrix} \mathbf{0} \\ f_{\text{interface}} = Y_2 \end{bmatrix}}_{\boldsymbol{f}_{\text{interface}}} = \mathbf{0}. \quad (4.102)$$

`doSolve` The `doSolve` function of subsystem 2 expects an interface displacement as an input (U_2) and outputs the interface force of the interface degree of freedom (Y_2). In order to solve the nonlinear equation system for every time step the function applies a local Newton method to Equation (4.102)

$$\mathcal{J}\left(\hat{\boldsymbol{r}}_{\text{subsys}}\left({}_l\hat{\boldsymbol{u}}^{n+1}\right)\right) \quad {}_{l+1}\Delta\hat{\boldsymbol{u}}^{n+1} = -\hat{\boldsymbol{r}}_{\text{subsys}}\left({}_l\hat{\boldsymbol{u}}^{n+1}\right). \quad (4.103)$$

Please note that $\boldsymbol{u}$ is replaced with $\hat{\boldsymbol{u}}$ because $u_{\text{interface}}$ is known now as it is an input quantity. Hence $\hat{\boldsymbol{u}}$ is given by

$$\hat{\boldsymbol{u}} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{i-1} \\ f_{\text{interface}} = Y_2 \end{bmatrix}. \tag{4.104}$$

`getInterfaceJacobian` Based on the Jacobian extraction procedure of subsystem 1 for subsystem 2 this is

$$\begin{aligned} \hat{\boldsymbol{A}} \frac{\partial \hat{\boldsymbol{u}}}{\partial u_{\text{interface}}} &= -\frac{\partial \hat{\boldsymbol{r}}_{\text{subsys}}}{\partial u_{\text{interface}}}, \\ \frac{\partial \hat{\boldsymbol{u}}}{\partial u_{\text{interface}}} &= -\hat{\boldsymbol{A}}^{-1} \frac{\partial \boldsymbol{r}_{\text{subsys}}}{\partial u_{\text{interface}}}. \end{aligned} \tag{4.105}$$

Equation (4.105) can be written as

$$\begin{bmatrix} \dfrac{\partial \boldsymbol{X}_2}{\partial u_{\text{interface}}} \\ \dfrac{\partial f_{\text{interface}}}{\partial u_{\text{interface}}} \end{bmatrix} = -\hat{\mathbf{A}}^{-1} \frac{\partial \hat{\boldsymbol{r}}_{\text{subsys}}}{\partial u_{\text{interface}}}.$$

Note that

$$\frac{\partial \boldsymbol{X}_2}{\partial u_{\text{interface}}} = \begin{bmatrix} \dfrac{\partial u_1}{\partial u_{\text{interface}}} \\ \dfrac{\partial u_2}{\partial u_{\text{interface}}} \\ \vdots \\ \dfrac{\partial u_{i-1}}{\partial u_{\text{interface}}} \end{bmatrix},$$

and

$$\frac{\partial f_{\text{interface}}}{\partial u_{\text{interface}}} = \frac{\partial Y_2}{\partial U_2}.$$

Furthermore, $\dfrac{\partial \boldsymbol{r}_{\text{subsys}}}{\partial u_{\text{interface}}}$ represents the last column of $\boldsymbol{A}$.

`doApproximatedSolve` The `doApproximatedSolve` for subsystem 2 is similar to subsystem 1, namely

$$ {}^{m+1}\boldsymbol{X}_2 \approx {}^{m}\boldsymbol{X}_2 + {}^{m}\left|\frac{\partial \boldsymbol{X}_2}{\partial U_2}\right. {}^{m}\Delta U_2, \tag{4.106} $$

$$ {}^{m+1}\boldsymbol{X}_2 \approx {}^{m}\boldsymbol{X}_2 + {}^{m}\left|\frac{\partial \boldsymbol{X}_2}{\partial u_{\text{interface}}}\right. {}^{m}\Delta u_{\text{interface}}, \tag{4.107} $$

$$ {}^{m+1}Y_2 = g\left({}^{m+1}\boldsymbol{X}_2, {}^{m}U_2\right). \tag{4.108} $$

Results

Numerical experiments are performed with the Truss versus Truss problem in order to demonstrate the performance of the IJCSA. Therefore, truss 1 is discretized with 20 elements and truss 2 with 10. The total length of the two truss systems is set to $L = 1\,\text{m}$. Initial conditions are $u(0) = 0\,\text{m}$ and $\dot{u}(0) = 0\,\text{m/s}$. The interface residual is considered as converged if

$$ \left\| {}^{m+1}\boldsymbol{r}^{n} \right\|_{\max} < 1.0 \cdot 10^{-10}. \tag{4.109} $$

A constant external load $f = 4.25\,\text{N}$ is applied inside subsystem 1. The local iteration convergence constraint for both subsystems is set to

$$ \left\| \boldsymbol{r}_{\text{subsys}} \right\|_{\max} < 1.0 \cdot 10^{-12}. \tag{4.110} $$

For the sake of clarity an absolute residual measure is used. As possible reference values for a relative residual measure, the applied force and the maximum interface displacement are close to one, the absolute residual measure does not restrict the outcome of the discussion in any way. Time discretization is done with a step size of $h = 0.01\,\text{s}$ and 200 time steps. To verify the coupled solution the simulations are compared with a monolithic solution performed with Abaqus. It turned out that the IJCSA solution matches all the digits of the Abaqus

solution. The densities are set to $\varrho_1 = 0.55\,\mathrm{kg/m^3}$ and $\varrho_2 = 0.5\,\mathrm{kg/m^3}$. The Young's moduli are $E_1 = 20\,\mathrm{N/m^2}$ and $E_2 = 50\,\mathrm{N/m^2}$. This parameter set is unstable with a standard Jacobi pattern when no relaxation is applied. Therefore constant under-relaxation $\alpha = 0.38$ is applied. The relaxation factor was determined by numerical experiments and seems to be optimal for this problem parameter set. However it is possible to use adaptive relaxation methods like Aitken. The Aitken relaxation method can be used in combination with the Gauss-Seidel pattern (see Joosten et al. [83]). Convergence for the Aitken relaxation method is only proven to be guaranteed if the residual is a scalar. The proof is available in Henrici [72].

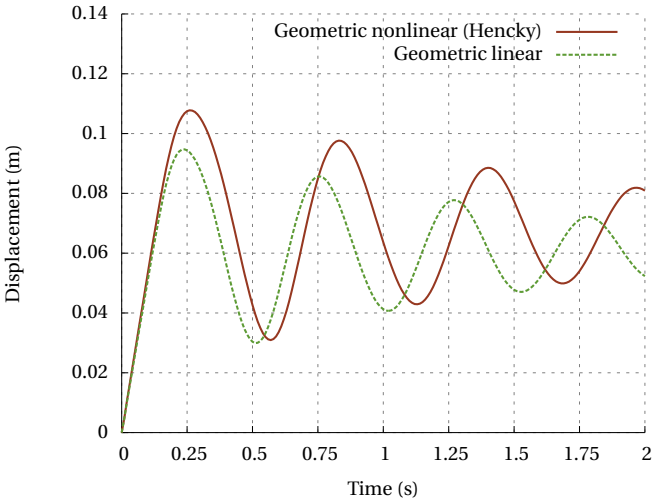


Figure 4.4: Numerical solution of the Truss versus Truss problem

Table 4.3 presents the number of local and interface iterations for different algorithms. Classical fixed-point techniques and various versions of the IJCSA are compared. The number of local iterations is the most important one for the runtime. Each local Newton iteration of subsystem 1 and 2 involves a solution of a linear equation system. For the Truss versus Truss problem this is by far the most time consuming part in terms of the wall-clock time. Hence this is the measure for the efficiency of the different co-simulation algorithms.

Table 4.3: Numerical results for the Truss versus Truss problem

Description	no. of local iterations for $\mathcal{S}_1$	no. of local iterations for $\mathcal{S}_2$	no. of local iterations for $\mathcal{S}_1\approx$	no. of local iterations for $\mathcal{S}_2\approx$	no. of interface iterations
Aitken relaxation with Gauss-Seidel pattern	4157	3720	0	0	1092
constant under-relaxation with Jacobi pattern	46388	44333	0	0	17446
basic IJCSA (Algorithm 4.3)	2526	2568	0	0	785
enhanced IJCSA (Algorithm 4.4)	771	807	1619	1178	789
basic IJCSA (Algorithm 4.3) with a fixed number of local Newton iterations for the solve step (1 local iteration)	1146	1146	0	0	1146
enhanced IJCSA (Algorithm 4.4) with a fixed number of local Newton iterations for the solve step (1 local iteration)	200	200	1262	932	666
enhanced IJCSA (algorithm with Jacobian extraction in the first iteration within each time step only (modified Newton)) (1 local iteration)	200	200	1854	1400	900

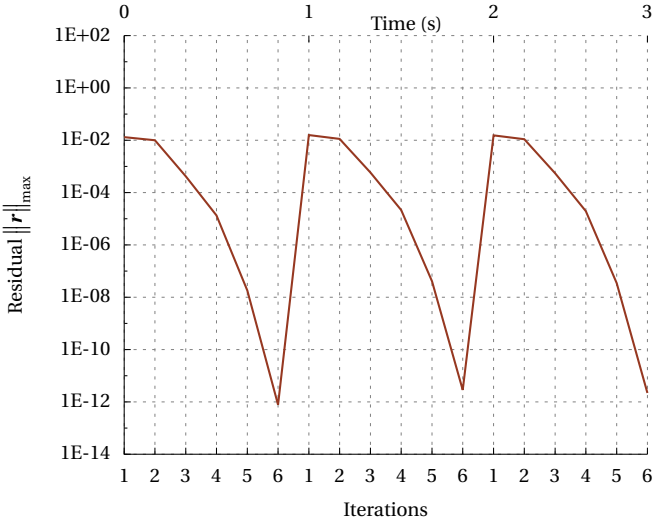
As the subsystems use local iterations to solve the nonlinear state equations an additional efficiency enhancement of the IJCSA may be done. This is to converge the local iteration and the interface (outer) iterations together. It is about three times faster than converging the local iterations for the Truss versus Truss co-simulation problem.

If the problem is "mildly" nonlinear it might also be of advantage to use a modified Newton strategy in which Jacobian extraction is done in the first iteration only within each time step. However modified Newton methods exhibit in general only a linear convergence rate (see Section 2.3).

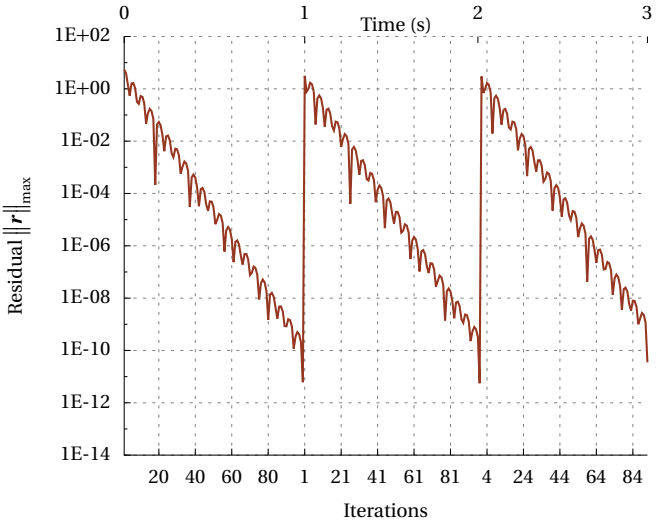
The numerical solution for the chosen parameter set is shown in Figure 4.4. The numerical dissipation of the BE time integrator is dominant. However this is the exact monolithic solution of the problem setting. Additionally, Figure 4.4 shows the geometrical linear solution of the Truss versus Truss problem. By comparing that to the geometrical nonlinear (Hencky) solution of the Truss versus Truss problem it is evident that the geometrical nonlinearity changes to structural response distinctly.

The fastest IJCSA version is about 18× faster than the Aitken relaxation for the Gauss-Seidel pattern (this also takes the extraction of the Jacobian into account). Moreover, Gauss-Seidel based patterns are no option for a general multi-code co-simulation scenario: they have the severe disadvantage of data flow dependency of all participating subsystems.

Figure 4.5(d) shows the evolution of the interface residual norm for the interface iterations needed to accomplish three time steps. The Aitken version shows a fairly good reduction of the interface residual. It needs six iterations to render the residual below $1 \cdot 10^{-10}$. The constant under-relaxation shows a linear convergence rate which leads to a large number of iterations (approx. 100 per time step). The basic version of the IJCSA needs the least number of iterations. The modified Newton starts with ten iterations and drops down to six iterations for the third time step. The modified Newton method is best when the problem is "mildly" nonlinear there it is more efficient than a full Newton method.

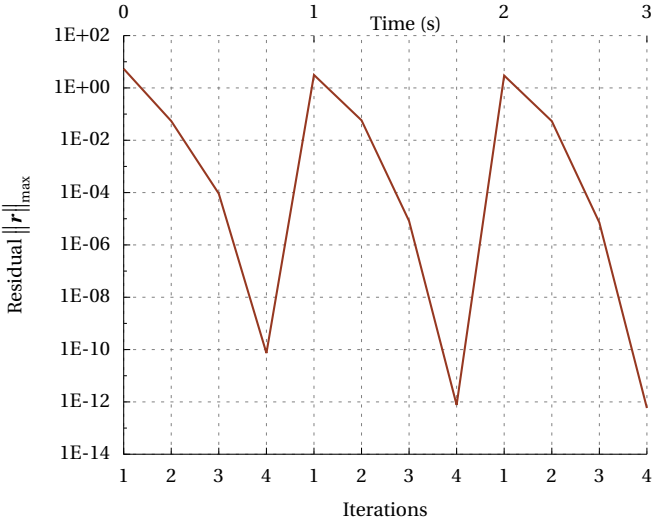


(a) Aitken relaxation with Gauss-Seidel pattern

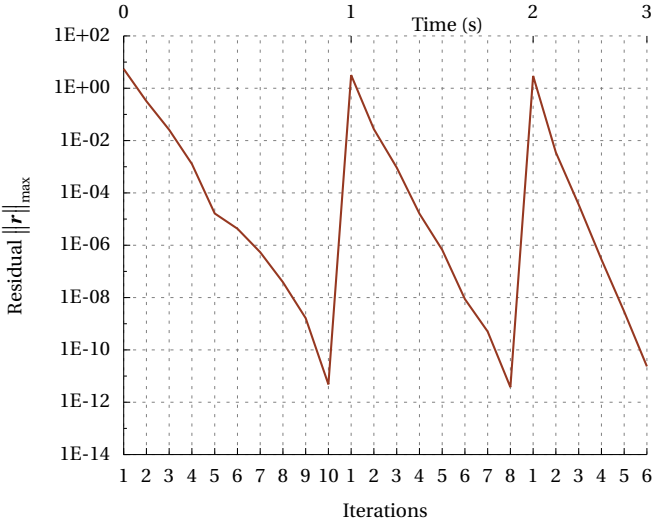


(b) Constant under-relaxation with Jacobi pattern

Figure 4.5: Convergence behavior of the Truss versus Truss problem for the first 3 time steps



(c) Basic IJCSA (Algorithm 4.3)



(d) Enhanced IJCSA (algorithm with Jacobian extraction in the first iteration within each time step only and one local Newton iteration per interface iteration)

Figure 4.5: Convergence behavior of the Truss versus Truss problem for the first 3 time steps

4.7.2 A Multi-Code Problem

Within this section an example problem which consists of five subsystems is discussed. The subsystems $\mathcal{S}_1$ and $\mathcal{S}_2$ are thereby modeled by a nonlinear operator described in Section 4.7.1. The spring-subsystem $\mathcal{S}_3$ is a simple linear spring model. The dashpot-subsystem $\mathcal{S}_4$ is integrated with the BE integrator; this subsystem is also linear. The last subsystem is a nonlinear ODE stemming from the simplification of the Navier-Stokes equations on moving grids; it is integrated with the trapezoidal rule (TR). The problem setup is illustrated in Figure 4.6.

The input/output quantities for the Multi-Code problem are presented in Table 4.4. Keeping in mind these input/output quantities

Table 4.4: Input/output quantities for the Multi-Code problem

Subsystem	Input U	Output Y
$\mathcal{S}_1$	f_1	u_1
$\mathcal{S}_2$	f_2	u_2
$\mathcal{S}_3$	f_3	u_3
$\mathcal{S}_4$	f_4	u_4
$\mathcal{S}_5$	u_5	f_5

the following interface operators and interface residuals are defined for this problem:

$$\mathcal{R}_1 = \mathcal{I}_1\left(U_1, U_2, U_3, U_4, Y_5\right) = U_1 + U_2 + \frac{1}{\lambda_3} U_3 + \frac{1}{\lambda_4} U_4 - Y_5 = 0 \tag{4.111}$$

$$\mathcal{R}_2 = \mathcal{I}_2\left(\mathcal{S}_1\left(U_1\right), S_2\left(U_2\right)\right) = Y_1 - Y_2 = 0 \tag{4.112}$$

$$\mathcal{R}_3 = \mathcal{I}_3\left(\mathcal{S}_1\left(U_1\right), S_3\left(U_3\right)\right) = Y_1 + \lambda_3 Y_3 = 0 \tag{4.113}$$

$$\mathcal{R}_4 = \mathcal{I}_4\left(\mathcal{S}_1\left(U_1\right), S_4\left(U_4\right)\right) = Y_1 + \lambda_4 Y_4 = 0 \tag{4.114}$$

$$\mathcal{R}_5 = \mathcal{I}_5\left(\mathcal{S}_1\left(U_1\right), S_5\left(U_5\right)\right) = Y_1 - U_5 = 0 \tag{4.115}$$

Hence, the global interface Jacobian matrix is given by

$$\begin{bmatrix} 1 & 1 & \frac{1}{\lambda_3} & \frac{1}{\lambda_4} & -\frac{\partial \mathcal{S}_5}{\partial U_5} \\ \frac{\partial \mathcal{S}_1}{\partial U_1} & -\frac{\partial \mathcal{S}_2}{\partial U_2} & 0 & 0 & 0 \\ \frac{\partial \mathcal{S}_1}{\partial U_1} & 0 & \lambda_3 \frac{\partial \mathcal{S}_3}{\partial U_3} & 0 & 0 \\ \frac{\partial \mathcal{S}_1}{\partial U_1} & 0 & 0 & \lambda_4 \frac{\partial \mathcal{S}_4}{\partial U_4} & 0 \\ \frac{\partial \mathcal{S}_1}{\partial U_1} & 0 & 0 & 0 & -1 \end{bmatrix}. \tag{4.116}$$

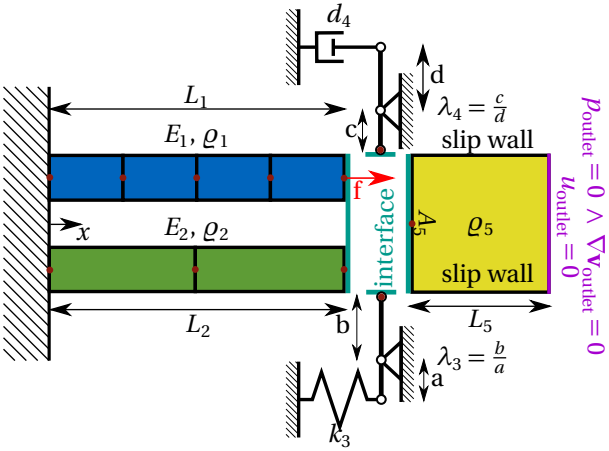


Figure 4.6: Setup of the Multi-Code problem

Subsystem 1 - Truss 1

This subsystem is defined in Section 4.7.1

Subsystem 2 - Truss 2

This subsystem is defined in Section 4.7.1

Subsystem 3 - Spring

A linear spring model is used in subsystem 3 which is given by

$$u_3^{n+1} = \frac{1}{k_3} f_3^{n+1}. \tag{4.117}$$

Hence the interface Jacobian for this subsystem is constant and given by

$$\frac{\partial \mathcal{S}_3}{\partial U_3} = \frac{1}{k_3}. \tag{4.118}$$

Subsystem 4 - Dashpot

In subsystem 4 a linear dashpot model is used

$$\dot{u}_4 d_4 = f_4, \tag{4.119}$$

which is integrated by using the BE integrator

$$u_4^{n+1} = u_4^n + \frac{h}{d_4} f_4^{n+1}. \tag{4.120}$$

The interface Jacobian for this subsystem is again constant. Hence, we have

$$\frac{\partial \mathcal{S}_4}{\partial U_4} = \frac{h}{d_4}. \tag{4.121}$$

Subsystem 5 - Idealized Piston

This section derives the governing equations for the Idealized Piston subsystem. As the fluid is modeled incompressible the starting point for the derivations are the Navier-Stokes equations for incompressible flow in combination with a linear material law (Stokes relations). These equations describe the flow on a fixed grid (Eulerian point of view). If the structure will be modeled in the Lagrangian framework it would cause a problem at the interface of fluid and structure. In order to cope with this problem the arbitrary Lagrangian-Eulerian (ALE) method is used. As the ALE method is a hybrid reference frame (mix-

ture of Eulerian and Lagrangian reference frame), the Navier-Stokes equations need to be changed as shown in Donea et al. [37] to

$$\frac{\partial v_i}{\partial x_i} = 0, \tag{4.122}$$

$$\frac{\partial v_i}{\partial t} + \left(v_j - v_j^{\text{grid}}\right)\frac{\partial v_i}{\partial x_j} = -\frac{1}{\varrho_5}\frac{\partial p}{\partial x_i} + \nu\frac{\partial^2 v_i}{\partial x_j^2}, \tag{4.123}$$

where ϱ_5 is the fluid density and ν is the kinematic viscosity. p is the most important quantity for the Idealized Piston subsystem as it represents the hydrostatic pressure. The flow velocity components are denoted by v_i. Furthermore, v_j^{grid} are the so called grid velocity components and $(v_j - v_j^{\text{grid}})$ the convective velocity components (note that grid displacement components are denoted via $u_j^{\text{grid}} = u_j$). By choosing an appropriate set of boundary conditions (see Figure 4.6) the fluid domain can be reduced to a one dimensional setting, hence the Equations (4.122) and (4.123) reduce to

$$\frac{\partial v_1}{\partial x_1} = 0, \tag{4.124}$$

$$\frac{\partial v_1}{\partial t} + \left(v_1 - v_1^{\text{grid}}\right)\frac{\partial v_1}{\partial x_1} = -\frac{1}{\varrho_5}\frac{\partial p}{\partial x_1} + \nu\frac{\partial^2 v_1}{\partial x_1^2}. \tag{4.125}$$

These equations can be reformulated to one integrable ordinary differential equation (ODE) for the pressure (insert conservation of mass Equation (4.124) in the conservation of momentum Equation (4.125)), namely

$$\frac{\partial v_1}{\partial t} = -\frac{1}{\varrho_5}\frac{\partial p}{\partial x_1} \quad \rightarrow \quad \frac{\partial p}{\partial x_1} = -a_1\varrho_5, \tag{4.126}$$

with a, being the acceleration in x-direction. In the subsequent derivations the index is dropped as only one dimension is present. Equation (4.126) can be solved by integration over the domain

$$\int \frac{\partial p}{\partial x}\,\mathrm{d}x = -\int a\varrho_5\,\mathrm{d}x, \tag{4.127}$$

$$p(x) = -a\varrho_5 x + C. \tag{4.128}$$

The integration constant C can be determined from the outlet boundary condition for the pressure $p(x)\big|_{x=L_5} = 0$. Please note that the

coordinate system in Figure 4.6 is fixed in space. As result one obtains the following linear function for the pressure distribution

$$p(x) = a\varrho_5\left(L_5 - x\right). \tag{4.129}$$

The ordinary differential nonlinear equation for subsystem 5 is given by

$$\ddot{u}_5\left(\varrho_5 \cdot A_5 \cdot L_5\right) - \ddot{u}_5 u_5\left(\varrho_5 \cdot A_5\right) = f_5, \tag{4.130}$$

with u_5 being the interface displacement and f_5 the interface force.

This equation is integrated with the help of the trapezoidal rule, which is a subset of the generalized-α method presented in Section 3.1.2. If the generalized-α parameters are set $\alpha_\mathrm{m} = 0$, $\alpha_\mathrm{f} = 0$, $\beta = 0.25$ and $\gamma = 0.5$ the trapezoidal rule is obtained. The trapezoidal rule approximates the velocities by

$$\dot{u}_5^{n+1} = \frac{2}{h}\left(u_5^{n+1} - u_5^n\right) - \dot{u}_5^n. \tag{4.131}$$

Thus the time discretized equation is given by

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & \left(L_5 - u_5^{n+1}\right)\dfrac{2\varrho_5 A_5}{h} & 0 \\ 0 & \dfrac{2}{h} & -1 \end{bmatrix} \begin{bmatrix} f_5^{n+1} \\ v_5^{n+1} \\ a_5^{n+1} \end{bmatrix} = \begin{bmatrix} \dfrac{2}{h} & 1 & 0 \\ 0 & \left(L_5 - u_5^{n+1}\right)\dfrac{2\varrho_5 A_5}{h} & \left(L_5 - u_5^{n+1}\right)\varrho_5 A_5 \\ 0 & \dfrac{2}{h} & 1 \end{bmatrix} \begin{bmatrix} u_5^n \\ v_5^n \\ a_5^n \end{bmatrix} - \begin{bmatrix} \dfrac{2}{h} u_5^{n+1} \\ 0 \\ 0 \end{bmatrix}. \tag{4.132}$$

With that the interface Jacobian of subsystem 5 is

$$\frac{\partial \mathcal{S}_5}{\partial U_5} = \frac{2\varrho_5 A_5}{h}\left(\frac{2}{h}u_5^n - \frac{4}{h}u_5^{n+1} + 2v_5^n + \frac{h}{2}a_5^n + \frac{2L_5}{h}\right). \tag{4.133}$$

Results

The spacial discretization is the same for subsystem 1 and 2 as in the previous example. The simulation time is 1 s, this results in 1 000 performed time steps. Two different settings are simulated (see Table 4.5).

Table 4.5: Parameter sets for the Multi-Code problem

Parameter	Values for setting 1	Values for setting 2	Unit
h	$1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$	s
f	12.25	12.25	N
E_1	20	20	$\mathrm{N/m^2}$
ϱ_1	0.55	0.55	$\mathrm{kg/m^3}$
L_1	1	1	m
E_2	50	50	$\mathrm{N/m^2}$
ϱ_2	0.5	0.5	$\mathrm{kg/m^3}$
L_2	1	1	m
k_3	500	500	N/m
λ_3	2	2	–
d_4	10	10	Ns/m
λ_4	2	2	–
A_5	0.1	0.1	$\mathrm{m^2}$
L_5	1	1	m
ϱ_5	$1 \cdot 10^{-10}$	2.75	$\mathrm{kg/m^3}$

As in setting 1 the fluid density ϱ_5 is almost zero the solution is verified with the monolithic solution of all structural subsystems (one to four) performed in Abaqus/Standard. For setting 2 the fluid density is increased and the added mass effect becomes dominate. For this highly nonlinear example the IJCSA needs in average 3.7 interface iterations. Please note that the interface residual tolerance of the previous example is used (see Equation (4.109)). If the modified Newton version of the IJCSA is used the interface iteration count rises to five iterations in average per time step. A comparison to classical

techniques is not possible as all of them failed to deliver a converged solution for this problem. The results are demonstrated in Figure 4.7.

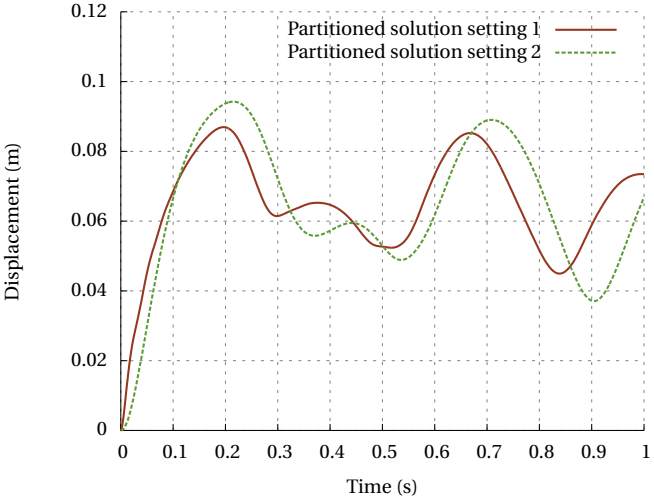


Figure 4.7: Numerical solution of the Multi-Code problem

This example demonstrates that the IJCSA can handle complex co-simulation scenarios in an efficient and accurate manner. It can handle different numerical time integrators within different subsystems.

Note that Figure 4.8 shows the absolute residual. In a general implementation it is better to use a relative residual. For the discussion of the Multi-Code problem there is no difference of using the absolute or relative residual as the applied force and the maximal interface displacement is close enough to one. Hence the use of a relative definition will not change the outcome of the discussion.

Figure 4.8 shows that for the Multi-Code problem the different residuals are not equally important. Here the residual associated with the interface forces $\mathcal{R}_1$ plays the dominate role. In a general co-simulation scenario this is always the case. The advantage of using a parallel (Jacobi) data flow is that this exposes all interface quantities in the interface residual vector. This is not the case if a Gauss-Seidel data flow is used e.g. it can be seen in the stability section that Gauss-Seidel

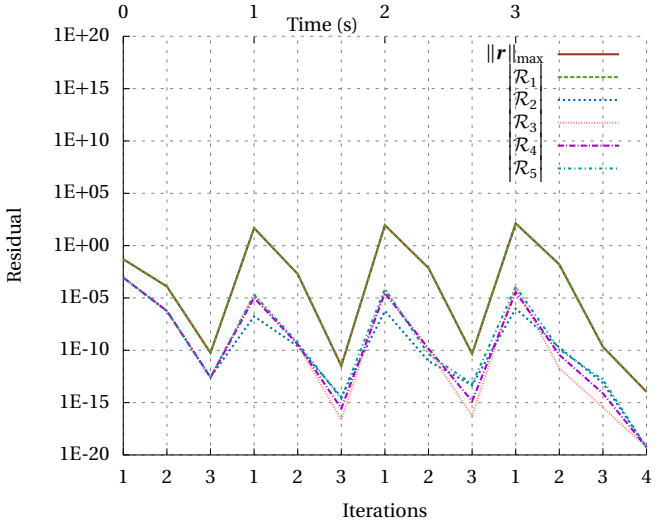


Figure 4.8: Residual evolution for the different components of the interface residual vector for setting 2 of the Multi-Code problem

removes the force from the interface residual. Hence the advantage of the IJCSA is that it controls all residuals which leads in general to more accurate results.

As it is known from Section 3.2 the mix of different time integrators may cause problems with the overall convergence of the solution with respect to the time step size, a convergence study for setting 2 of the problem is performed. Figure 4.9 shows that a first order of accuracy is achieved and thus the co-simulation is consistent.

4.7.3 BspK6

The BspK6 is a synthetic co-simulation example proposed by Bastian et al. [11]. The subsystems $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$ form a cycle within the graph of the block-diagram. A cycle is a path in a graph with the same node as start and end point. Additionally, the BspK6 example has discontinuities within the subsystems. Hence, it is a good supplement of the test examples for the IJCSA. It consist of four subsystems. The block-diagram of the BspK6 is shown in Figure 4.10. With the help of

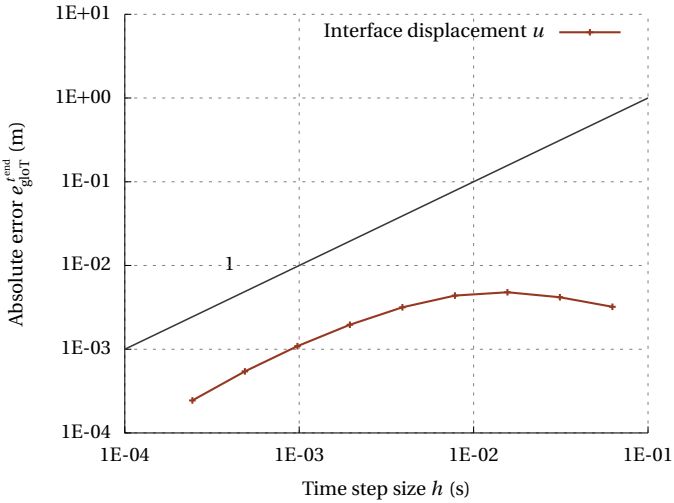


Figure 4.9: Absolute global error (2.80) of the interface displacement for setting 2 of the Multi-Code problem

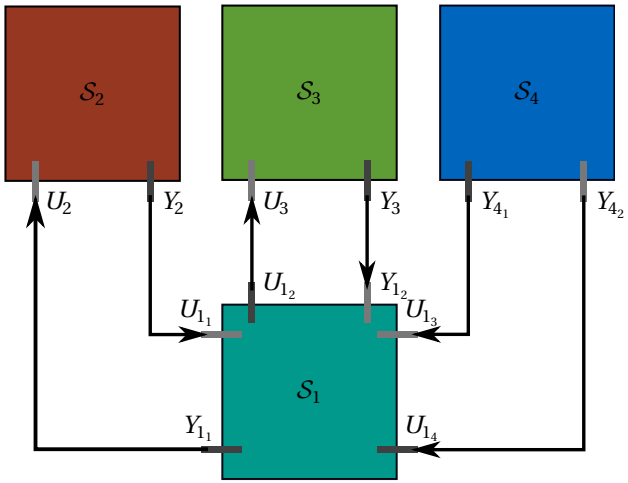


Figure 4.10: Block diagram that describes the BspK6 example

the block-diagram we can directly derive the global interface residual vector, which reads

$$\boldsymbol{r} = \begin{bmatrix} \mathcal{R}_1 \\ \mathcal{R}_2 \\ \mathcal{R}_3 \\ \mathcal{R}_4 \\ \mathcal{R}_5 \\ \mathcal{R}_6 \end{bmatrix} = \begin{bmatrix} U_{1_1} - Y_2 \\ U_{1_2} - Y_3 \\ U_{1_3} - Y_{4_1} \\ U_{1_4} - Y_{4_2} \\ U_2 - Y_{1_1} \\ U_3 - Y_{1_2} \end{bmatrix}. \tag{4.134}$$

Hence the global interface Jacobian matrix is defined by

$$\boldsymbol{J}_{\text{global}} = \begin{bmatrix} 1 & 0 & 0 & 0 & -\dfrac{\partial Y_2}{\partial U_2} & 0 \\ 0 & 1 & 0 & 0 & 0 & -\dfrac{\partial Y_3}{\partial U_3} \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ -\dfrac{\partial Y_{1_1}}{\partial U_{1_1}} & -\dfrac{\partial Y_{1_1}}{\partial U_{1_2}} & 0 & 0 & 1 & 0 \\ -\dfrac{\partial Y_{1_2}}{\partial U_{1_1}} & -\dfrac{\partial Y_{1_2}}{\partial U_{1_2}} & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.135}$$

Furthermore, the block-diagram shows that the subsystems $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$ form a cycle.

The BE method is used within all subsystems in order to compare the result with Bastian et al. [11].

Subsystem 1

The system is given by the following three equations:

$$\dot{X}_1 + 2X_1 - U_{1_1} - U_{1_2} = 0 \tag{4.136}$$

$$Y_{1_1} = g_{1_1}\left(U_{1_1}, X_1\right) = \begin{cases} 0 & \text{if } U_{1_4} = 1, \\ U_{1_1} - X_1 & \text{else.} \end{cases} \tag{4.137}$$

$$Y_{l_2} = g_{l_2}\left(U_{l_2}, X_1\right) = \begin{cases} 0 & \text{if } U_{l_3} = 1, \\ U_{l_2} - X_1 & \text{else.} \end{cases} \tag{4.138}$$

The BE discretized version of subsystem 1 reads:

$$X_1^{n+1} = \frac{U_{l_1}^{n+1} + U_{l_2}^{n+1} + \frac{X_1^n}{h}}{\frac{1}{h} + 2} \tag{4.139}$$

$$Y_{l_1}^{n+1} = \begin{cases} 0 & \text{if } U_{l_4}^{n+1} = 1, \\ U_{l_1}^{n+1} - X_1^{n+1} & \text{else.} \end{cases} \tag{4.140}$$

$$Y_{l_2}^{n+1} = \begin{cases} 0 & \text{if } U_{l_3}^{n+1} = 1, \\ U_{l_2}^{n+1} - X_1^{n+1} & \text{else.} \end{cases} \tag{4.141}$$

In the following the four interface Jacobian components of subsystem 1 are derived.

$$\frac{\partial Y_{l_1}}{\partial U_{l_1}} = \begin{cases} 0 & \text{if } U_{l_4} = 1, \\ \dfrac{\partial Y_{l_1}}{\partial X_1}\dfrac{\partial X_1}{\partial U_{l_1}} + \dfrac{\partial g_{l_1}}{\partial U_{l_1}} = 1 - \dfrac{1}{\frac{1}{h} + 2} & \text{else.} \end{cases} \tag{4.142}$$

$$\frac{\partial Y_{l_1}}{\partial U_{l_2}} = \begin{cases} 0 & \text{if } U_{l_4} = 1, \\ \dfrac{\partial Y_{l_1}}{\partial X_1}\dfrac{\partial X_1}{\partial U_{l_2}} + \dfrac{\partial g_{l_1}}{\partial U_{l_2}} = -\dfrac{1}{\frac{1}{h} + 2} & \text{else.} \end{cases} \tag{4.143}$$

$$\frac{\partial Y_{l_2}}{\partial U_{l_1}} = \begin{cases} 0 & \text{if } U_{l_3} = 1, \\ \dfrac{\partial Y_{l_2}}{\partial X_1}\dfrac{\partial X_1}{\partial U_{l_1}} + \dfrac{\partial g_{l_2}}{\partial U_{l_1}} = 1 - \dfrac{1}{\frac{1}{h} + 2} & \text{else.} \end{cases} \tag{4.144}$$

$$\frac{\partial Y_{l_2}}{\partial U_{l_2}} = \begin{cases} 0 & \text{if } U_{l_3} = 1, \\ \dfrac{\partial Y_{l_2}}{\partial X_1}\dfrac{\partial X_1}{\partial U_{l_2}} + \dfrac{\partial g_{l_2}}{\partial U_{l_2}} = -\dfrac{1}{\frac{1}{h} + 2} & \text{else.} \end{cases} \tag{4.145}$$

Subsystem 2

The system is given by the following two equations:

$$\frac{1}{2}\dot{X}_2 + X_2 + U_2 - \sin(3\pi t) = 0 \tag{4.146}$$

$$Y_2 = g_2\left(U_2, X_2\right) = X_2 + U_2\left(1000\sin\left(\frac{2\pi}{10}t\right) + 1001\right) \tag{4.147}$$

The BE discretized version of subsystem 2 reads

$$X_2^{n+1} = \frac{\frac{1}{2h}X_2^n - U_2^{n+1} + \sin\left(3\pi t^{n+1}\right)}{\frac{1}{2h}+1}, \tag{4.148}$$

$$Y_2^{n+1} = X_2^{n+1} + \left(1000\sin\left(\frac{2\pi}{10}t^{n+1}\right) + 1001\right)U_2^{n+1}. \tag{4.149}$$

The interface Jacobian of subsystem 2 is given by

$$\frac{\partial Y_2}{\partial U_2} = \frac{\partial Y_2}{\partial X_2}\frac{\partial X_2}{\partial U_2} + \frac{\partial g_2}{\partial U_2} = -\frac{1}{\frac{1}{2h}+1} + 1000\sin\left(\frac{2\pi}{10}t^{n+1}\right) + 1001. \tag{4.150}$$

Subsystem 3

The system is given by the following two equations:

$$\frac{1}{2}\dot{X}_3 + X_3 + U_3 - \sin(2\pi t) = 0 \tag{4.151}$$

$$Y_3 = g_3\left(U_3, X_3\right) = X_3 + U_3\left(-1000\sin\left(\frac{2\pi}{10}t\right) + 1001\right) \tag{4.152}$$

The BE discretized version of subsystem 3 reads

$$X_3^{n+1} = \frac{\frac{1}{2h}X_3^n - U_3^{n+1} + \sin\left(2\pi t^{n+1}\right)}{\frac{1}{2h}+1}, \tag{4.153}$$

$$Y_3^{n+1} = X_3^{n+1} + \left(-1000\sin\left(\frac{2\pi}{10}t^{n+1}\right) + 1001\right)U_3^{n+1}. \tag{4.154}$$

The interface Jacobian of subsystem 3 is given by

$$\frac{\partial Y_3}{\partial U_3} = \frac{\partial Y_3}{\partial X_3}\frac{\partial X_3}{\partial U_3} + \frac{\partial g_3}{\partial U_3} = -\frac{1}{\frac{1}{2h}+1} - 1000\sin\left(\frac{2\pi}{10}t^{n+1}\right) + 1001. \tag{4.155}$$

Subsystem 4

The system is given by the following two equations:

$$Y_{4_1} = \begin{cases} 1 & \text{if } \sin(\pi t) > \frac{1}{2}, \\ 0 & \text{else.} \end{cases} \tag{4.156}$$

$$Y_{4_2} = \begin{cases} 1 & \text{if } \sin(2\pi t) < -\frac{1}{2}, \\ 0 & \text{else.} \end{cases} \tag{4.157}$$

The discretized version of subsystem 4 reads

$$Y_{4_1} = \begin{cases} 1 & \text{if } \sin\left(\pi t^{n+1}\right) > \frac{1}{2}, \\ 0 & \text{else.} \end{cases} \tag{4.158}$$

$$Y_{4_2} = \begin{cases} 1 & \text{if } \sin\left(2\pi t^{n+1}\right) < -\frac{1}{2}, \\ 0 & \text{else.} \end{cases} \tag{4.159}$$

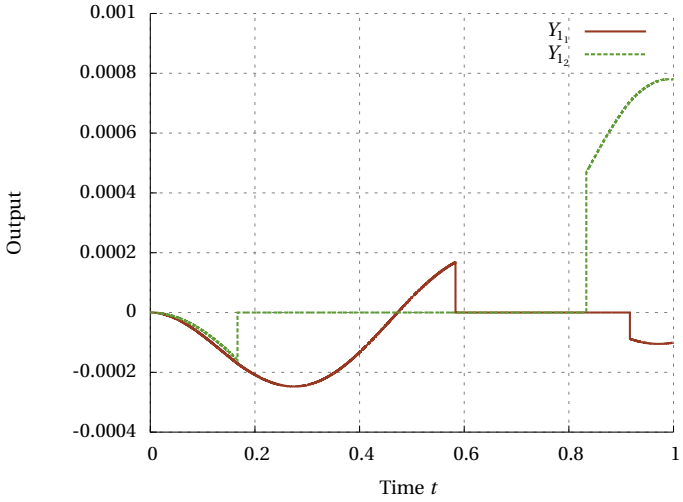
As subsystem 4 has no input quantities there is no interface Jacobian needed for this subsystem. The interface Jacobian of subsystem 4 is zero by definition.

Results

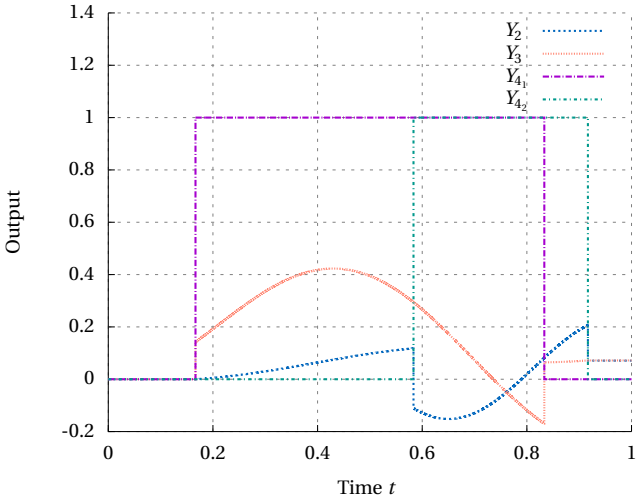
The IJCSA is used to solve the BspK6 example with a time step size h of $1 \cdot 10^{-4}$. This is the same time step size as in Bastian et al. [11].

The IJCSA needs in average two interface iterations in order to converge the global interface residual vector to machine precision. Even when the discontinuity is introduced the IJCSA converges the interface residual to machine precision within three interface iteration.

Figure 4.11 shows the converged results of all five output quantities of the BspK6, the results match the one from Bastian et al. [11] exactly.



(a) Y_{1_1} and Y_{1_2}



(b) Y_2, Y_3, Y_{4_1} and Y_{4_2}

Figure 4.11: Numerical solution of the BspK6 example

4.8 Conclusion

The presented co-simulation algorithm (IJCSA) is based on the idea of interface Jacobian information for stabilization. With the help of the stability analysis it is shown that the algorithm outperforms classical fixed point techniques based on the Jacobi and Gauss-Seidel approaches. In the example section numerical investigations are performed where the IJCSA is compared to classical co-simulation algorithms which include Aitken relaxation for the Gauss-Seidel pattern. The IJCSA is for all examples by far more stable and efficient than the classical techniques. Its design allows for parallel execution of all the involved subsystems. Furthermore, performance enhancements are discussed and investigated with different nonlinear examples. The IJCSA can cope with different time integrators in the subsystems. Cycles within the graph (block-diagram) can be handled because the co-simulation is formulated in residual form. Furthermore, even algebraic loops can be handled because of the residual-based formulation of the IJCSA (see Appendix A) . The IJCSA presents an efficient, accurate and robust method for solving co-simulation scenarios.

Mathematics is the language in which God has written the universe.

Galileo Galilei

CHAPTER 5

APPLICATION EXAMPLES

After presenting the theory and the concept of co-simulation in Chapter 3, a new algorithm for co-simulation was presented in Chapter 4. The algorithm was tested with demonstrator examples. Within this chapter the focus is on the validation of a co-simulation framework for application relevant examples.

Moreover it should be shown how the modularity of co-simulation can be used to combine existing simulation tools to a powerful multiphysical environment. Within this chapter the structural mechanics solvers Abaqus/Standard and Carat++, the computational fluid dynamics solver OpenFOAM, signal solvers implemented in C and Python and control units implemented in C are coupled together by using the *E*nhanced *M*ulti*P*hysics *I*nterface *R*esearch *E*ngine (*EMPIRE* [40]) and the SIMULIA Co-Simulation Engine (CSE).

The IJCSA is used in order to solve the coupled problem. In case where the subsystems are not able to provide their individual interface Jacobian, it is approximated with techniques introduced in the previous chapters. The focus of the application examples is the co-simulation of fields and signals where open- and closed-loop control is present. Within this chapter the examples have fluid-structure inter-

action in common. The FSI simulation are coupled to various different signal and control units.

The chapter shows also the benefits of modern co-simulation techniques. It demonstrates how these methods can be applied to a fully coupled emergency brake maneuver of a wind turbine. All shown examples were either verified with simulations or validated with experiments.

5.1 Turek Benchmark

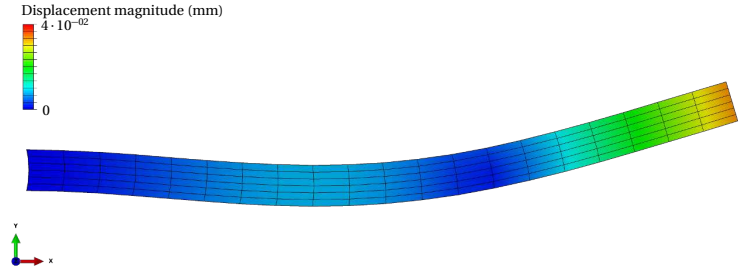
Before the discussion of fluid-structure-signal interaction is entered the pure FSI simulation should be verified. A well accepted verification case in FSI is the FSI3 benchmark proposed by Turek et al. [148]. The FSI3 benchmark is a rather difficult test case as it has a 1:1 density ratio between fluid and structure which results in a comparable dominant interaction with respect to both fields. In other words an imperfection on the fluid pressure has a significant impact on the displacement on the structure and vice versa.

Six 8-node incompatible mode elements through the thickness of the beam are used. The element formulation is based on Simo et al. [138]. The boundary conditions are set such that plane strain conditions are met. The structure uses the Hilber-Hughes-Taylor method for time integration which is a subset of the generalized-α discussed in Chapter 3.

In OpenFOAM a mesh study was performed, the final mesh has $\approx 50\,000$ cells. All schemes are set to second order accuracy and the time integration method is set to BDF2.

5.1.1 Co-Simulation

The problem is decomposed in a Dirichlet/Neumann manner. The interface Jacobian for the IJCSA method is set such that it results in a constant under-relaxation method. It has been shown for the scalar case in Algorithm 4.2 that a constant under-relaxation method can be derived from the IJCSA by setting the global interface Jacobian to be the identity matrix if the individual subsystem residuals have a certain structure. The same is shown in the following for the Turek benchmark problem (vector case).



(a) Displacement field

(b) ϱ-normalized pressure field

Figure 5.1: Turek benchmark FSI3 displacement and ϱ-normalized pressure at 10 s

The CFD subsystem OpenFOAM can be abbreviated by

$$\mathcal{S}_1\left({}^m\boldsymbol{U}_1^n\right) = {}^m\boldsymbol{Y}_1^{n+1}, \tag{5.1}$$

where the input ${}^m\boldsymbol{U}_1^n$ is a displacement field and the output ${}^m\boldsymbol{Y}_1^{n+1}$ a force field. For the CSM subsystem the opposite is true, namely

$$\mathcal{S}_2\left({}^m\boldsymbol{U}_2^n\right) = {}^m\boldsymbol{Y}_2^{n+1}. \tag{5.2}$$

Here the input ${}^m\boldsymbol{U}_2^n$ is the force field and the output ${}^m\boldsymbol{Y}_2^{n+1}$ a displacement field. As already noted this is called Dirichlet/Neumann decomposition. For this decomposition the individual interface residuals are of type

$${}^m\boldsymbol{r}^n = \begin{bmatrix} {}^m\mathcal{R}_1^n \\ {}^m\mathcal{R}_2^n \end{bmatrix} = \begin{bmatrix} {}^m\boldsymbol{U}_1^n - {}^m\boldsymbol{Y}_2^{n+1} \\ {}^m\boldsymbol{U}_2^n - {}^m\boldsymbol{Y}_1^{n+1} \end{bmatrix}. \tag{5.3}$$

Note that due to the non-machting grids at the fluid-structure interface the vectors $\boldsymbol{U}_1^n$ and $\boldsymbol{Y}_2^{n+1}$ have different number of elements. The same problem arises for $\boldsymbol{U}_2^n$ and $\boldsymbol{Y}_1^{n+1}$. This problem can be solved by using a mapping operation such that the one vector is projected into a space which renders two equal sized vectors. Within this work an energy conserving mortar method is used to solve the arising mapping problems (see Boer et al. [19] and Puso et al. [119]). If the mapping operator $\mathcal{L}$ is discretized this results in a rectangular matrix $\boldsymbol{L}$. If the mapping procedure is added to Equation (5.3) this reads

$$ {}^m\boldsymbol{r}^n = \begin{bmatrix} {}^m\boldsymbol{\mathcal{R}}_1^n \\ {}^m\boldsymbol{\mathcal{R}}_2^n \end{bmatrix} = \begin{bmatrix} {}^m\boldsymbol{U}_1^n - \boldsymbol{L}\,{}^m\boldsymbol{Y}_2^{n+1} \\ {}^m\boldsymbol{U}_2^n - \boldsymbol{L}^\top\,{}^m\boldsymbol{Y}_1^{n+1} \end{bmatrix}. \tag{5.4} $$

In order to keep the notation more readable this is not explicitly written in the following discussions.

Equation (5.3) results in a block diagram as depicted in Figure 5.2. The input/output quantities for the Turek problem are summarized in Table 5.1. Moreover based on the interface residual definition the global interface Jacobian is also defined by

$$ {}^m\boldsymbol{J}_{\text{global}}^n = \begin{bmatrix} \boldsymbol{I} & -\dfrac{\partial\,\boldsymbol{Y}_2}{\partial\,\boldsymbol{U}_2} \\ -\dfrac{\partial\,\boldsymbol{Y}_1}{\partial\,\boldsymbol{U}_1} & \boldsymbol{I} \end{bmatrix}. \tag{5.5} $$

Chapter 2 derives the update rule for the constant under-relaxation method as

$$ {}^{m+1}\boldsymbol{x} = {}^m\boldsymbol{x} - \alpha\,{}^m\boldsymbol{r}. \tag{5.6} $$

Here $\boldsymbol{x}$ is the global input vector as they are the unknowns. Furthermore the update rule of the IJCSA (Algorithm 4.3) is defined by

$$ {}^{m+1}\boldsymbol{x}^n = {}^m\boldsymbol{x}^n - {}^m\boldsymbol{J}_{\text{global}}{}^{-1}\,{}^m\boldsymbol{r}. \tag{5.7} $$

It is evident that Equation (5.6) and Equation (5.7) are equal for the case that

$$ {}^m\boldsymbol{J}_{\text{global}} = \frac{1}{\alpha}\boldsymbol{I}, \tag{5.8} $$

in Equation (5.7). For the Turek benchmark problem the global interface Jacobian of the IJCSA is set such that it corresponds to a constant under-relaxation method with $\alpha = 0.05$ by using Equation (5.6) and Equation (5.7). Please note that the IJCSA with the crude interface Jacobian approximation performs worse than the GS Aitken method for the Turek FSI3 example. This is aligned with theory because it has been shown in Chapter 2 that the Aitken method is an adaptive advancement of the constant under-relaxation method. This example is used as a demonstrator, as the Turek FSI3 problem is still solvable by using a crude approximation for the interface Jacobian and accepted as a challenging FSI example this shows that the Multi-Code problem presented in Chapter 4 is even more challenging from the co-simulation point of view. As the Multi-Code problem is not solvable with the Aitken method nor with the constant under-relaxation method.

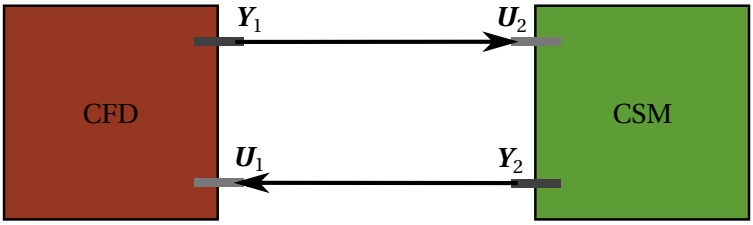


Figure 5.2: Block diagram for fluid-structure interaction

Table 5.1: Description of input and output quantities for the Turek benchmark

Symbol	Description	Unit
$\boldsymbol{U}_1$	Input to CFD displacement field	m
$\boldsymbol{Y}_1$	Output of CFD force field	N
$\boldsymbol{U}_2$	Input to CSM force field	N
$\boldsymbol{Y}_2$	Output of CSM displacement field	m

A GS Aitken solution where the forces are not present as an interface residual should be compared to a IJCSA solution where forces and displacements are presented as interface residuals in a fair manner. Therefore it is necessary to limit the interface residual check for the IJCSA solution to the displacements only. Both the GS Aitken solution and the IJCSA solution are converged until the interface residual vector for the displacements meets

$$\left\| \mathcal{R}_1 \right\|_\epsilon < 1.0 \cdot 10^{-4} \text{mm}. \tag{5.9}$$

5.1.2 Results

Both the Aitken and the IJCSA render the same solution which is shown in the Figures 5.3 to 5.4 and summarized in Table 5.2.

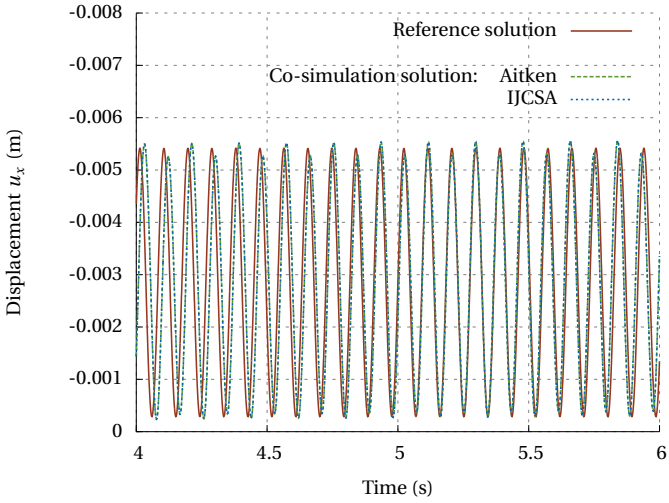


Figure 5.3: FSI3 benchmark displacement u_x over time

In order to illustrate how the pressure and the displacement look like a contour plot of the pressure and the displacement field is given in Figure 5.1.

The presented results are from a simulation where Abaqus/Standard is coupled to OpenFOAM. The results show a very good agree-

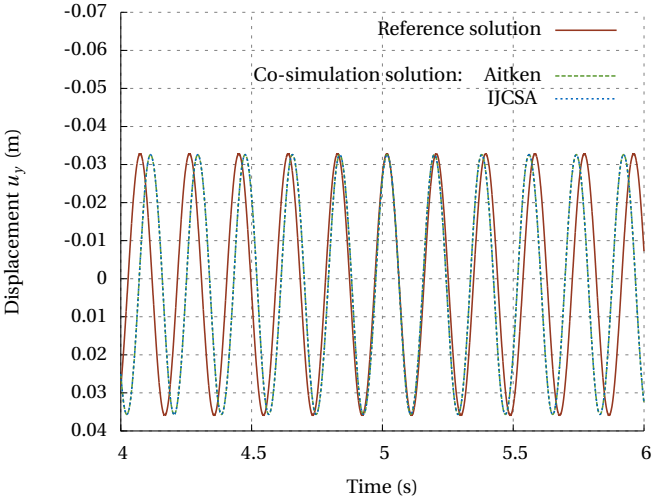


Figure 5.4: FSI3 benchmark displacement u_y over time

Table 5.2: Results for FSI3 with time step $h = 1 \cdot 10^{-3}$ s for amplitude and frequency

Symbol	Reference Turek et al. [148]	Co-Simulation	Rel. deviation
	mm {1/s}	mm {1/s}	% {%}
u_x	−2.69 ± 2.53 {10.9}	−2.79 ± 2.51 {11.050}	0.83 {1.37}
u_y	1.48 ± 34.38 { 5.3}	1.52 ± 34.13 { 5.525}	0.74 {4.24}

ment with the provided reference data. The deviation in the amplitude is less than 1 %, the deviation in frequency is less than 5 %.

In order to show the modularity of the co-simulation the FSI3 benchmark was also simulated by using an in-house FEM package called CARAT. The fluid model could be reused without any modifications. On top of the reusability of the individual subsystem models, here a commercial and an in-house FEM package are coupled to OpenFOAM. This demonstrates that with co-simulation it is possible to

replace the simulation tool of one individual subsystem and still be able to reuse all the other subsystem models without any modification.

5.2 Oscillating Cylinder

Within this section the transition from FSI to fluid-structure-signal interaction is performed by using an oscillating cylinder as validation example. The setting of the oscillating cylinder example is show in Figure 5.5. A rigid cylinder with mass m is mounted on a spring-damper system. The cylinder is excited by the vortex shedding of a laminar flow. An actuator can move the root point of the spring Ψ.

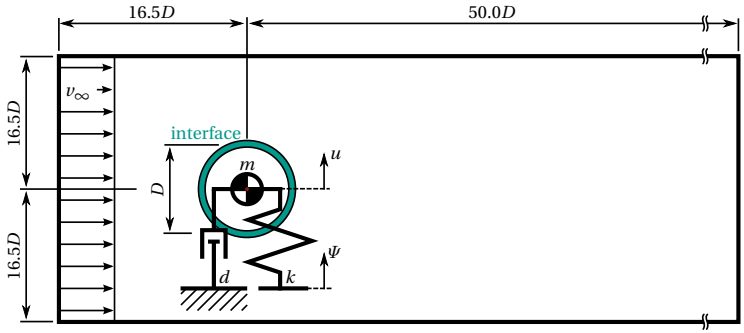


Figure 5.5: Setting of the oscillating cylinder example

The example is used in order to demonstrate closed-loop control for a fully coupled fluid-structure-signal example. As a first step the pure CFD simulation where the cylinder is not moving is validated.

5.2.1 CFD Validation

The laminar flow past a cylinder is a well documented benchmark case in literature. In the following the drag coefficient, lift coefficient and Strouhal number are compared to various references. The OpenFOAM and StarCCM+ simulation used the same block structured mesh with $\approx 70\,000$ cells. Second order schemes are used for time and spatial discretization. The time step size is set to $1 \cdot 10^{-4}$ s. The mesh size and the time step size are the result of a performed convergence study.

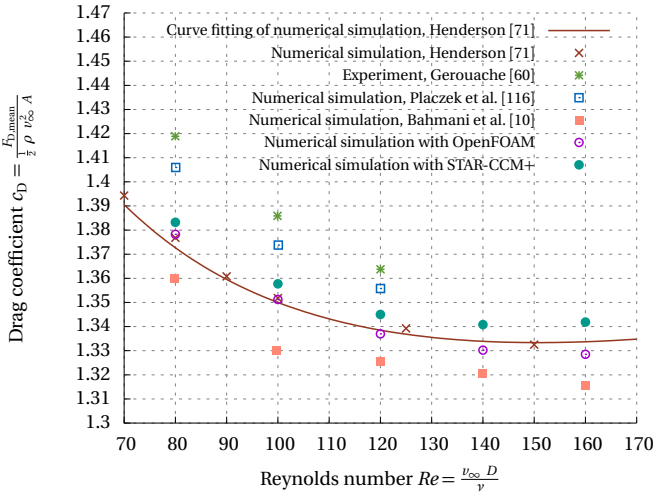


Figure 5.6: Drag coefficient of different measurements and simulations

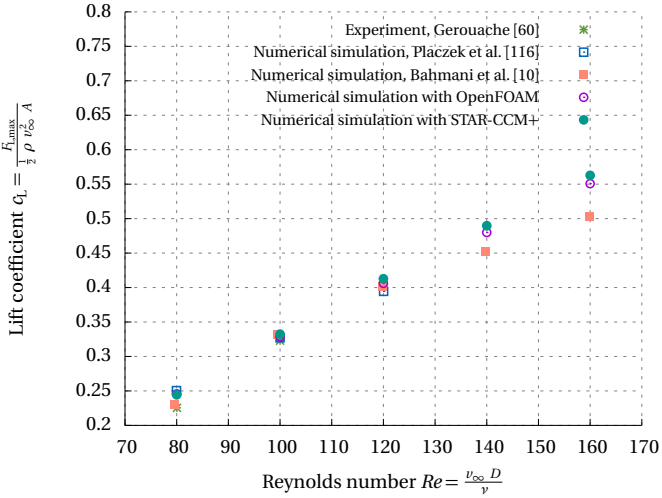


Figure 5.7: Lift coefficient of different measurements and simulations

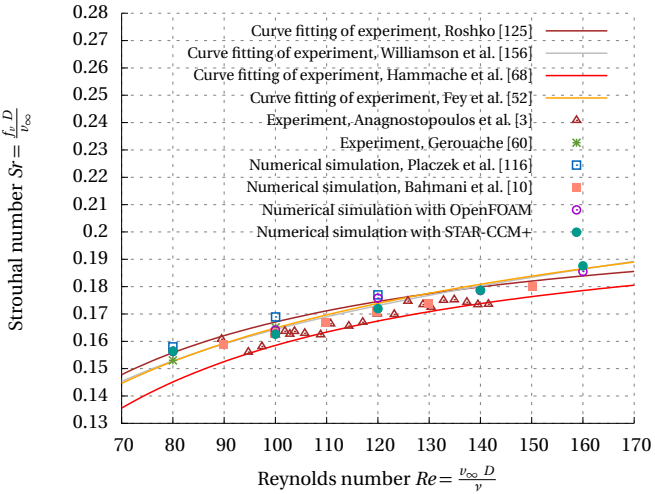


Figure 5.8: Strouhal number of different measurements and simulations

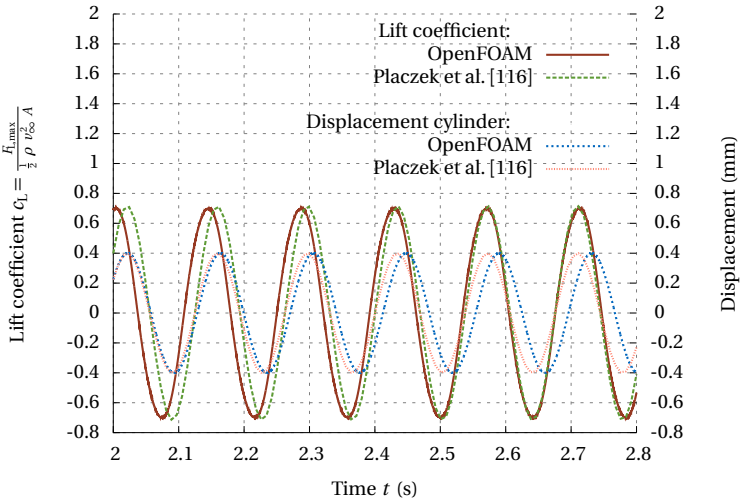


Figure 5.9: Time varying lift coefficient forced oscillating cylinder

The drag and the lift coefficient show good agreement with the values provided by literature (see Figure 5.6 and Figure 5.7 respectively). Moreover the Strouhal number is also in good agreement with literature values as shown in Figure 5.8. With these results the CFD is validated. The next evolution step of the example is to verify the forced motion of the cylinder which is done in the following.

5.2.2 Forced Oscillation Validation

Within the forced motion validation the motion of the cylinder is prescribed as a time-varying boundary condition via a separate subsystem. The resulting lift coefficients are compared to simulation results provided by Placzek et al. [116].

The oscillation frequency of the cylinder is chosen to be 110 % of the vortex shedding frequency for Reynolds number 100 as this is according to Placzek et al. [116].

The Strouhal number for a Reynolds number of 100 can be found in Figure 5.8. With the help of the Strouhal number the oscillation frequency can be found by

$$\begin{aligned} Sr(Re = 100) = 0.16406 &= \frac{f_{Sr} D}{v_\infty}, \\ f_{Sr} &= 6.4085\,\text{Hz}. \end{aligned} \tag{5.10}$$

The amplitude of the oscillating motion is set to 0.4 mm accordingly to Placzek et al. [116]. Hence, the function for the oscillating motion of the cylinder is given by

$$\begin{aligned} u(t) &= 0.4 \sin(2\pi\ 6.4085\,\text{Hz}\ 1.1\ t), \\ u(t) &= 0.4 \sin(44.2930\,^{\text{rad}}/_{\text{s}}\ t). \end{aligned} \tag{5.11}$$

The comparison of obtained values and the ones of Placzek et al. [116] for the lift coefficient and the oscillation frequency are presented in Figure 5.9 and Table 5.3. The results show a good agreement with values provided by Placzek et al. [116] (deviation less than 3 %).

5.2.3 Fluid-Structure with Closed-Loop Control

With the validated CFD results the fully coupled fluid-structure-signal interaction is approached. The spring-damper model has properties

Table 5.3: Results for forced oscillating case with time step size $h = 1 \cdot 10^{-4}$ s amplitude and frequency

Symbol	Reference Placzek et al. [116]	Co-Simulation	Rel. deviation
	mm {1/s}	mm {1/s}	% {%}
u_y	0.00 ± 0.40 {7.26}	0.00 ± 0.40 {7.05}	0.00 {2.90}
	− {1/s}	− {1/s}	% {%}
c_L	0.00 ± 0.71 {7.26}	0.00 ± 0.70 {7.05}	1.40 {2.90}

Table 5.4: System parameters for the spring-damper system of the oscillating cylinder problem

Property	Symbol	Value	Unit
Mass	m	0.035 75	kg
Damping coefficient	d	0.004 3	N/s
Spring stiffness	k	69.48	N/m
Damped eigenfrequency	ω_d	7.016 35	Hz

according to Anagnostopoulos et al. [3], which are summarized in Table 5.4.

The spring-damper system can be described by

$$m\ddot{u} + d\dot{u} + ku = f + k\Psi. \tag{5.12}$$

Here f is an external force coming from the fluid. In OpenFOAM the trapezoidal rule is used for time integration. Hence, it is also used for the discretization of Equation (5.12), which results in

$$\begin{bmatrix} \frac{2}{h} & -1 & 0 \\ k & \frac{2m}{h}+d & 0 \\ 0 & \frac{2}{h} & -1 \end{bmatrix} \begin{bmatrix} u^{n+1} \\ v^{n+1} \\ a^{n+1} \end{bmatrix} = \begin{bmatrix} \frac{2}{h} & 1 & 0 \\ 0 & \frac{2m}{h} & m \\ 0 & \frac{2}{h} & 1 \end{bmatrix} \begin{bmatrix} u^{n} \\ v^{n} \\ a^{n} \end{bmatrix} + \begin{bmatrix} 0 \\ f^{n+1}+k\Psi^{n+1} \\ 0 \end{bmatrix}. \tag{5.13}$$

The Reynolds number is set to 108.83. This value is in the so called lock-in region of the fluid-structure interaction where the vortex shedding frequency is synchronizing with the oscillation frequency of the cylinder. This lock-in frequency is slightly smaller than the damped eigenfrequency of the spring-damper system. The lock-in frequency for the lift coefficient and the cylinder oscillation is 6.990 Hz for the simulation. This behavior is also observed in the measurement of Anagnostopoulos et al. [3] where the coupled frequency is measured to be 6.995 Hz.

Proportional-Integral-Derivative Controller

A proportional-integral-derivative controller (PID controller) is used for the fluid-structure-signal interaction simulation. This is a closed loop feedback mechanism widely used in industrial control systems. The PID controller calculates an error value as the difference between a measured process variable and a prescribed trajectory value. The controller continuously attempts to minimize the error by manipulating the control variable. The continuous control law for the PID controller is

$$\Psi(t) = K_{\mathrm{P}}\, e(t) + K_{\mathrm{I}} \int_0^t e(\tau)\,\mathrm{d}\tau + K_{\mathrm{D}}\, \dot{e}(t), \tag{5.14}$$

(see Friedland [55], Liptak [96], Preumont [118], and Unbehauen [149]). The constants K_{P}, K_{I} and K_{D} are user inputs, they are determined

within a preprocessing step. The continuous error $e(t)$ is the deviation of the measured output quantity with respect to the desired trajectory value. The discretized control law for the PID controller is

$$\Psi^{n+1} = K_\mathrm{P} X_\mathrm{P}^{n+1} + K_\mathrm{I} X_\mathrm{I}^{n+1} + K_\mathrm{D} X_\mathrm{D}^{n+1}, \tag{5.15}$$

with

$$X_\mathrm{P}^{n+1} = e^{n+1}, \tag{5.16}$$

$$X_\mathrm{I}^{n+1} = \frac{h}{2}\left(e^{n+1} + e^{n}\right) + X_\mathrm{I}^{n}, \tag{5.17}$$

$$X_\mathrm{D}^{n+1} = \frac{1}{h}\left(e^{n+1} - e^{n}\right). \tag{5.18}$$

Equation (5.17) can be derived by using TR time integration and Equation (5.18) by using BE time integration. This combination renders a stable and robust discretization.

With all the necessary equations at hand the block diagram of the fluid-structure-signal interaction of the oscillating cylinder example is given according to Figure 5.10. Within Table 5.5 all output and input quantities are defined and summarized. With the help of the block

Table 5.5: Description of input and output quantities for the fluid-structure-signal interaction

Symbol	Description	Unit
U_1	Input to CFD displacement of cylinder u	m
Y_1	Output of CFD force on cylinder f	N
U_{2_1}	Input to CSM force on cylinder f	N
U_{2_2}	Input to CSM root point displacement of cylinder Ψ	m
Y_2	Output of CSM displacement of cylinder u	m
U_3	Input to Controller displacement of cylinder u	m
Y_3	Output of Controller root point displacement of cylinder Ψ	m

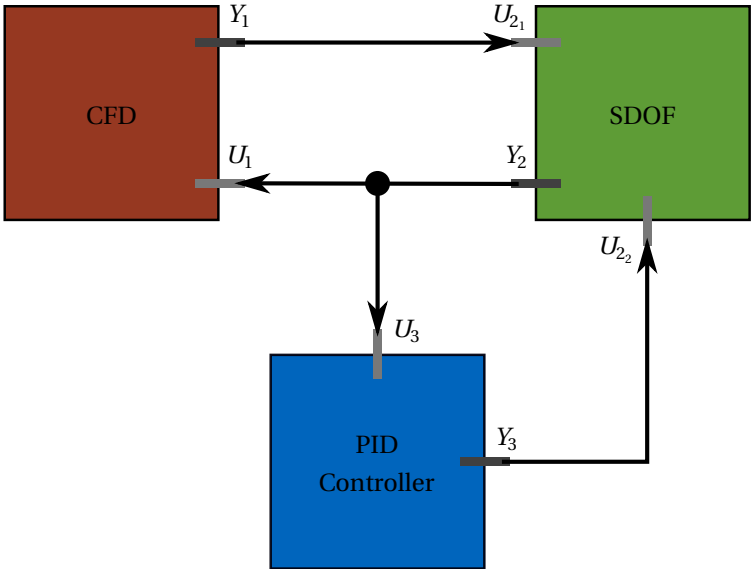


Figure 5.10: Block diagram for oscillating cylinder with PID controller

diagram the interface residual equations are also set to

$$\boldsymbol{r} = \begin{bmatrix} \mathcal{R}_1 \\ \mathcal{R}_2 \\ \mathcal{R}_3 \\ \mathcal{R}_4 \end{bmatrix} = \begin{bmatrix} U_1 - Y_2 \\ U_{2_1} - Y_1 \\ U_{2_2} - Y_3 \\ U_3 - Y_2 \end{bmatrix} = \begin{bmatrix} U_1 - \mathcal{S}_2\left(U_{2_1}, U_{2_2}\right) \\ U_{2_1} - \mathcal{S}_1\left(U_1\right) \\ U_{2_2} - \mathcal{S}_3\left(U_3\right) \\ U_3 - \mathcal{S}_2\left(U_{2_1}, U_{2_2}\right) \end{bmatrix}. \tag{5.19}$$

This results in

$$\boldsymbol{J}_{\text{global}} = \begin{bmatrix} 1 & -\frac{\partial Y_2}{\partial U_{2_1}} & -\frac{\partial Y_2}{\partial U_{2_2}} & 0 \\ -\frac{\partial Y_1}{\partial U_1} & 1 & 0 & 0 \\ 0 & 0 & 1 & -\frac{\partial Y_3}{\partial U_3} \\ 0 & -\frac{\partial Y_2}{\partial U_{2_1}} & -\frac{\partial Y_2}{\partial U_{2_2}} & 1 \end{bmatrix}, \tag{5.20}$$

being the assembled interface Jacobian matrix. The secant variant of the IJCSA (Algorithm 4.5) is used in order to solve the interface residual system. The secant approximation is only used for the interface Jacobian of the CFD, namely

$$\frac{\partial Y_1}{\partial U_1} \approx \frac{{}^{m}Y_1^{n+1} - {}^{m-1}Y_1^{n+1}}{{}^{m}U_1^{n} - {}^{m-1}U_1^{n}}, \tag{5.21}$$

as this is hard to compute exactly. All other needed interface Jacobians can be computed straight forward from Equations (5.13) and (5.15). The procedure is the same as presented in Chapter 4. In average 4 interface iterations where necessary to converge $\| \boldsymbol{r} \|_\epsilon$ below $1 \cdot 10^{-8}$.

The PID controller constants are set to $K_\mathrm{P} = 0.02$, $K_\mathrm{I} = 0.02$ and $K_\mathrm{D} = 0.01$. This results in a robust transient behavior which means that overshoots are small. If the objective of the PID controller is to suppress the cylinder motion the error e is equal to the displacement of the cylinder u.

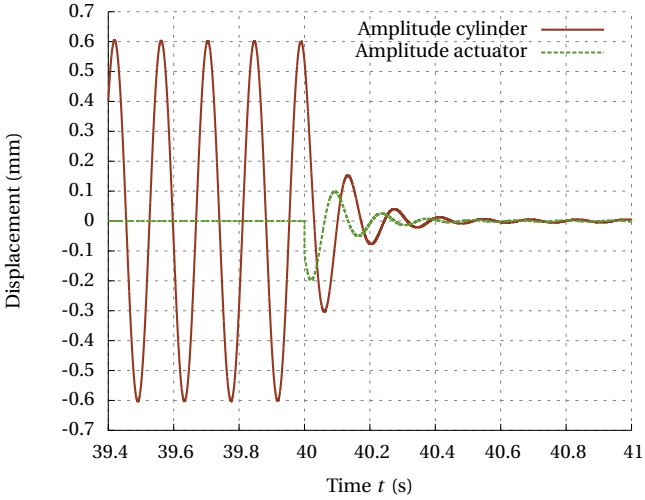


Figure 5.11: PID controller for case with $Re = 108.83$

Figure 5.11 depicts the impact of active closed-loop control for the oscillating cylinder example. Firstly, the controller is switched

off which results in an oscillating motion of the cylinder as the lock-in effect is present for $Re = 108.83$. At time 40 s the PID controller is activated. The PID controller reduces the motion of the cylinder by orders of magnitudes as shown in Figure 5.11.

5.2.4 Conclusion

Within this section a validated example shows how a fully coupled fluid-structure interaction with closed-loop control can be handled with the IJCSA. Moreover, the IJCSA allows to fulfill the interface residual equation numerically exact. Therefore, the controller can be treated in a continuous manner. It means that the delay which occurs if loosely coupled co-simulation approaches are deployed is not present.

These delays between the controller and the other subsystems may cause problems with stability of the closed-loop system. These problems are especially present when the hardware control unit has a much higher sampling frequency as the time sampling frequency of the numerical model (large time step). For this situation the delay of the real control unit is much smaller than for the one in the simulation in case the simulation is done in a loose manner.

5.3 NREL Phase VI Wind Turbine

The last and most complicated application example should demonstrate how the IJCSA performs in a large scale co-simulation involving fields and signals. Within this example a wind turbine is analyzed by using the co-simulation approach. The interaction between the fluid, the flexible blades, the generator and the control unit is taken into account.

The example is presented in a hierarchical manner where the complexity is gradually increased. First the pure CFD is validated and afterwards more complexity is added. Again this shows the modularity of co-simulation.

In order to be able to validate the simulation an experiment needs to be the basis for the simulation. The NREL Phase VI experiment is a good experiment for validation purposes as it is a full scale wind tunnel experiment. This means that the needed boundary conditions for the simulation are known. A major advantage for validation is the

elimination of stochastic atmospheric wind as the wind tunnel can deliver a constant inlet velocity profile.

Furthermore, this experiment is a very well documented case and investigated by other research groups, namely Anjuri [4], Dam et al. [32], Hsu et al. [76, 77], Li et al. [93], Lindenburg [95], McTavish et al. [100], Mo et al. [105, 106], Potsdam et al. [117], Sezer-Uzol et al. [133], Sørensen et al. [141], Tongchitpakdee et al. [146], Wang et al. [152], Yelmule et al. [160], and Zahle et al. [161]. A lot of different numerical methods were validated by using the data of the experiment. This gives a good basis as a lot of simulation data is available. However, none of the given references takes into account the interaction of the fluid, the flexible blades, the generator and the control unit at the same time.

In the following a comprehensive description of the experiment is given.

5.3.1 Experiment

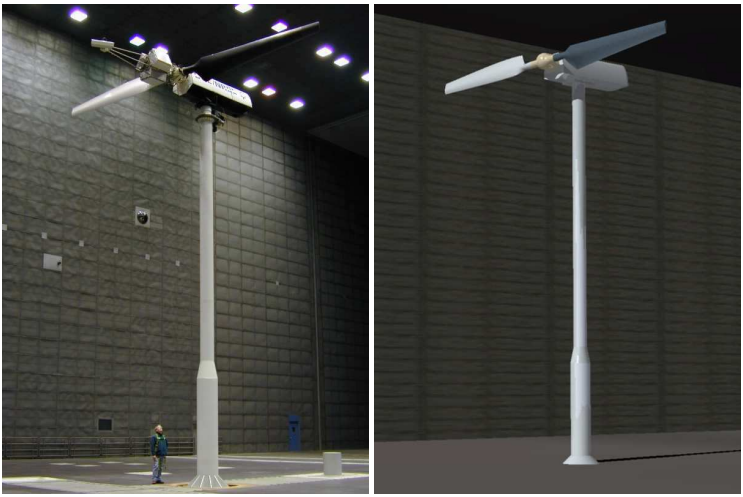
The Unsteady Aerodynamics Experiment, initially named "Combined Experiment", was started in 1987 by the National Renewable Energy Laboratory (NREL) to provide detailed information on the full-scale 3D aerodynamic behavior of wind turbines. Within this series detailed wind tunnel experiments were performed on a full-scale machine (Phase VI).

(a) Wind tunnel aerial view (b) Six wind tunnel fans

Figure 5.12: NASA wind tunnel photos from *NREL 10-m Wind Turbine Testing in NASA Ames* [110]

The wind tunnel (see Figure 5.12) is located in the NASA Ames Research Center in Moffett Field (Silicon Valley), California. Six fans with a total power consumption of 104 MW produce the inlet velocity for the 24.4 m by 36.6 m test section. The achieved turbulence intensity in streamwise direction is less than 0.5 % Simms et al. [137].

The testing wind turbine (see Figure 5.13) is the NREL Phase VI, a modified Grumman Windstream 33 with full-span pitch control and a power rating of 20 kW. It has 2 blades, with NREL S809 tapered and twisted blade profile. The rotor has 10.058 m in diameter while hub height is 12.192 m.



(a) Experiment photo from *NREL 10-m Wind Turbine Testing in NASA Ames* [110]

(b) CAD model

Figure 5.13: NREL Phase VI wind turbine real model and virtual model

The wind turbine details can be found in Hand et al. [69]. From this data a CAD model was made as the basis for all simulation models (see Figure 5.13(b)). As reference for the pitch angle the chord line of the tip airfoil lying in the rotor plane is defined as zero see Mo et al. [105]. To achieve the desired pitch angle of 3° all values in the definition table of Hand et al. [69] are corrected by an offset of +1.814°. The simulated test sequence "S" normally uses the standard tip out of

the three available tip adapters. The CAD model is cut off at a radius of 5.029 m, which corresponds to the total length including standard tip. This results in sharp edges which simplifies the mesh generation.

The first simulation model which is derived from this geometry is the CFD model which is described in the following section.

5.3.2 CFD Model

With OpenFOAM a body-fitted finite-volume discretization for the fluid is chosen. As this means that the mesh is fixed and needs to follow the blade surfaces a sliding mesh interface is needed in order to allow for rotation and pitching motion of the blades while the tower is present.

In OpenFOAM the sliding mesh interface is called Arbitrary Mesh Interface (AMI) (see *OpenFOAM* [111]). Furthermore, the arising non-matching grid problem is solved via local Galerkin projection method, which is presented in Farrell et al. [46].

The overall mesh topology is shown in Figure 5.14, there four separate mesh parts can be identified. The non-moving outer part covers the wind tunnel, the tower and the nacelle. It encloses the second part, which is following the rotating motion of the blades around the global y-axis, indicated by the angle of rotation ω. This rotor mesh part again encloses two more mesh parts which are directly attached to the blade surfaces. These innermost mesh parts provide the possibility of pitching around the global x-axis. On top of the rigid body rotations elastic deformations of both blades can be handled by Laplacian based mesh deformation inside both blade mesh parts only. For more information see Section 5.3.4.

The block structured outer mesh part has the same dimensions as the wind tunnel and consists of ≈ 3.7 million cells with an edge length starting from 180 mm immediately downstream of the wind turbine and increases to 800 mm towards the outlet. The rotor region containing ≈ 0.8 million cells is realized as hybrid mesh. It consists of block structured and tetrahedral meshed parts. A pure block structured version was tested without gaining any benefit in accuracy but increasing the cell count and computation time. Therefore, the hybrid version was chosen.

The mesh of the inner cylinders which embeds the blades is transitioned from the blade shoulder (Figure 5.15(a)) to the blade tip (Figure 5.15(b)). 100 cells in chord-wise direction and 240 in radial di-

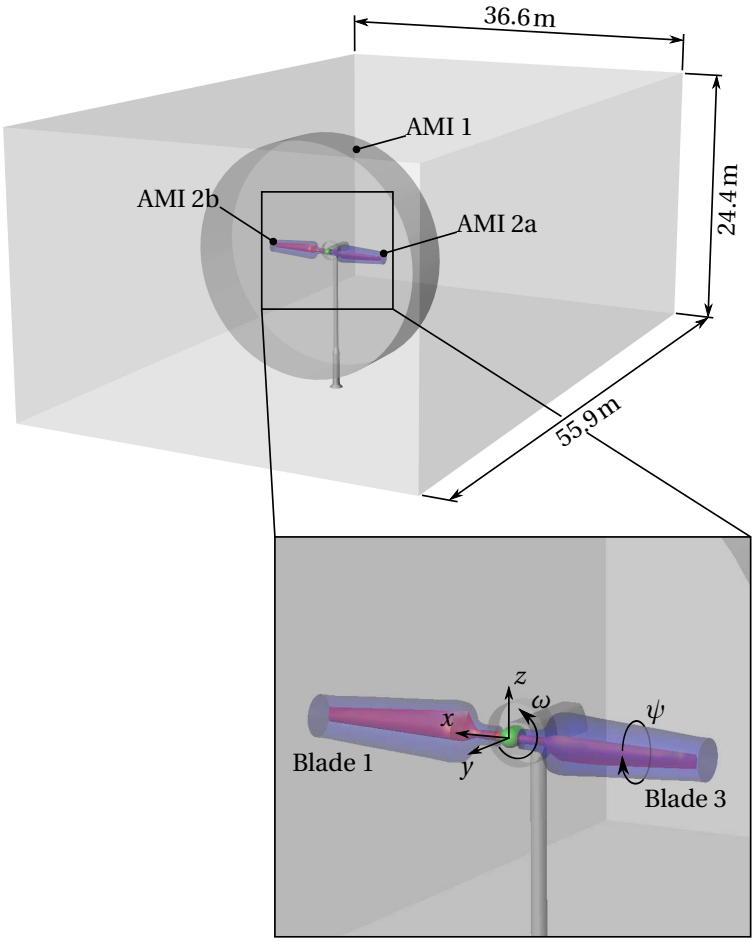
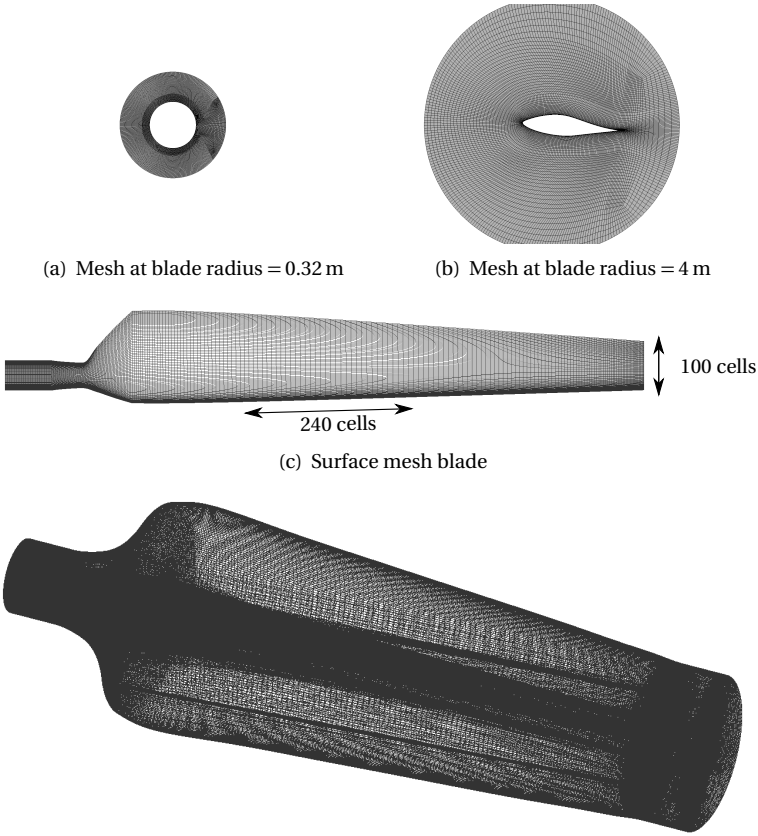


Figure 5.14: CFD mesh parts for NREL Phase VI wind turbine

rection render ≈ 2.6 million cells per blade region. Due to the CFL limitation the mesh was coarsened and aligned to the main flow direction at specific critical positions, where high velocities are expected. The first cell height at the blade surfaces is set to 0.4 mm. Hence, for the S0700000 case (inlet velocity is 7 m/s) an average dimensionless wall distance $y^+ \approx 7$ and a maximum dimensionless wall distance

$y^+ \approx 12$ are achieved. The maximal normal geometrical growth rate of the mesh is 1.4.



(a) Mesh at blade radius = 0.32 m

(b) Mesh at blade radius = 4 m

(c) Surface mesh blade

(d) Entire mesh around blade

Figure 5.15: CFD mesh of the inner cylinder around the blades

The boundary conditions for the CFD model are all of type wall except for the inlet and outlet. Here a velocity inlet (5 m/s or 7 m/s) and a pressure outlet (0 Pa) are set. The turbulence is modeled via the URANS k-ω-SST model introduced by Menter [102] in combination with high Reynolds wall functions. The boundary conditions for k and ω are set such that a turbulence intensity of 0.5 % is reached. The

segregated solution procedure called PIMPLE in combination with ALE mesh handling is used. PIMPLE is a variant of the famous PISO pressure-correction algorithm by Issa [81]. For time integration the BDF2 method is used and spatial schemes are set to be second order accurate. The time step is set to $1 \cdot 10^{-3}$ s for all simulations.

For the validation of the CFD model two measurement sequences are chosen. They are called S0500000 and S0700000. The parameters are summarized in Table 5.6.

Table 5.6: Properties of NREL Phase VI experiment for test sequences S0500000 and S0700000

Parameter	Sequence S0500000	Sequence S0700000	Unit
Inlet velocity	5	7	$^{m}/_{s}$
Cone angle	0.0	0.0	$^{\circ}$
Yaw angle	0.0	0.0	$^{\circ}$
Blade tip pitch angle	3.0	3.0	$^{\circ}$
Angular velocity	432	432	$^{\circ}/_{s}$
Air density	1.23	1.23	$^{kg}/_{m^3}$
Kinematic viscosity of air	$1.46 \cdot 10^{-5}$	$1.46 \cdot 10^{-5}$	$^{m^2}/_{s}$
Turbulence inlet intensity	0.5	0.5	%

During these two test sequences the flow is fully attached on the surface of the blades, which is also observed by Dam et al. [32] and Sørensen et al. [141].

Results & Validation

During the experiment pressure coefficients were evaluated. Therefore, pressure probes have been integrated into the blade surfaces. Multiple of these pressure probes were clustered at the five specific sections of the blade as shown in Figure 5.16.

For the validation, the first step was to post-process the raw measurement data in order to compute characteristic values such as mean, standard deviation and extrema of the measurement over time. All these values are computed out of measurement data which was recorded over 30 s at 15 625 samples.

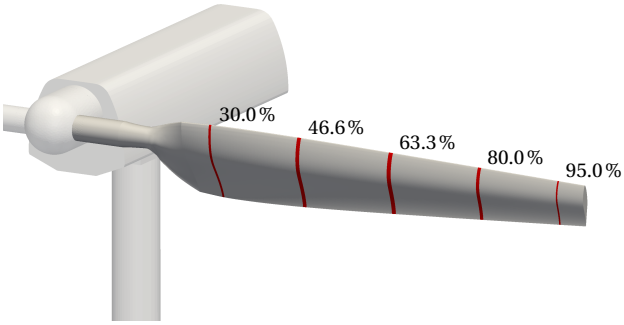


Figure 5.16: Measurement sections for pressure coefficients at different locations in % blade radius

The same procedure was applied to the simulation. However, here seven full revolutions of the rotor are the basis for the averaging procedure. These seven revolutions of the turbine, which is equivalent to 5.8333 s, are recorded after the flow is fully developed. The averaging over five and seven revolutions gave the same results, so seven revolutions are more than sufficient. Also the measurement data was averaged over 5.8333 s and the results were compared to values which resulted from the averaging over 30 s. There was almost (less than 0.1 %) no difference between these two averaging time spans. The seven revolutions of the simulation render 5 833 sample points in time for each individual boundary face.

The computation of the standard deviation σ is done by

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(p_i - \overline{p}\right)^2}, \text{ where } \overline{p} = \frac{1}{N}\sum_{i=1}^{N} p_i. \tag{5.22}$$

Here N is the total number of samples and p_i is the pressure value for a specific sample i at a specific probe location. $\overline{p}$ represents the time averages pressure value at a specific probe location. For the computation of the pressure coefficients the maximum of all time averaged pressure values at one specific measurement section is needed, it is denoted with $p_{\max}$.

The results of the validation of the pressure coefficients for series S0500000 and series S0700000 are depicted in Figure 5.18 and Fig-

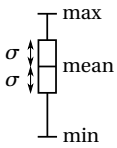
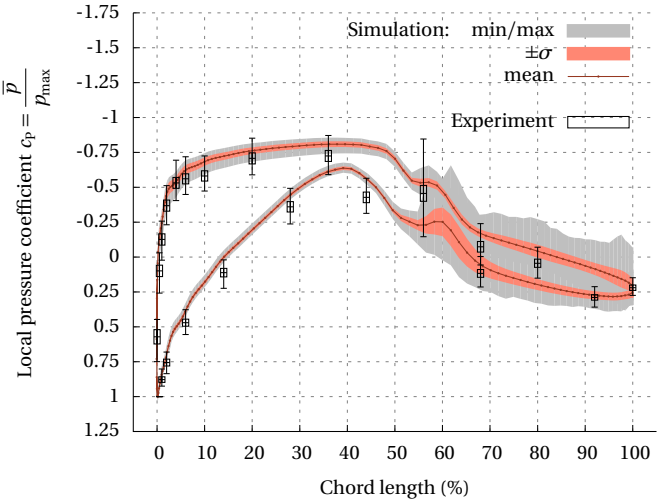


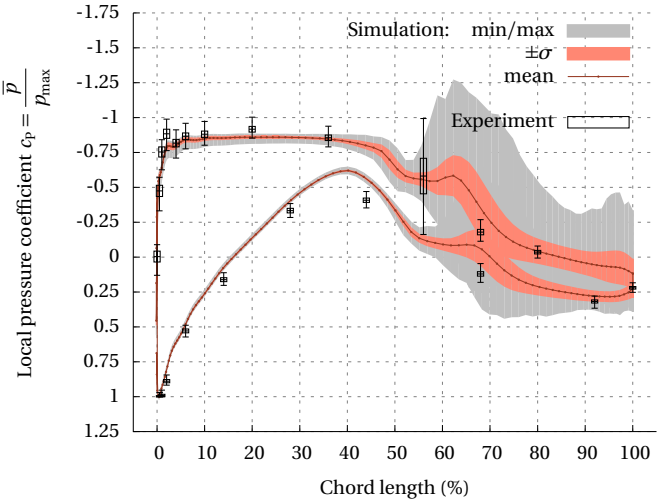
Figure 5.17: Legend of experiment box and whisker plots for pressure coefficients

ure 5.20 respectively. The meaning of the whiskers is illustrated in Figure 5.17.

Additionally, Figure 5.19 and Figure 5.21 show the development of the low-speed-shaft-torque (LSSTQ) in comparison to the averaged experimental data. A fast Fourier transform (FFT) analysis of the LSSTQ simulation data renders a peak at 2.4 Hz, this corresponds to the double frequency of the rotor rotations. Hence, it is caused by the blades passing the tower.

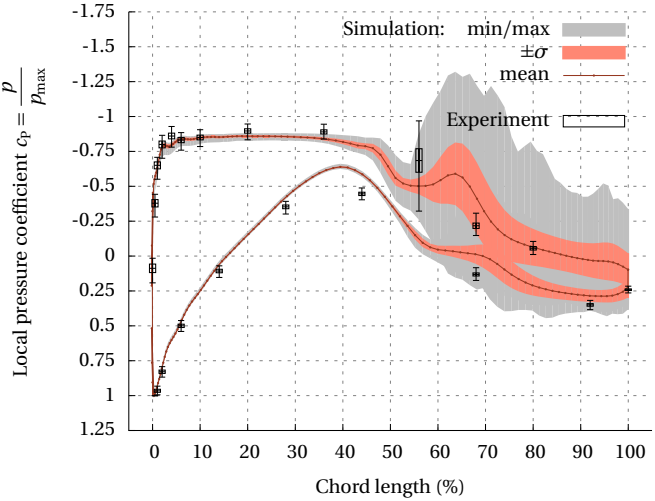


(a) Cut at 30 % blade radius

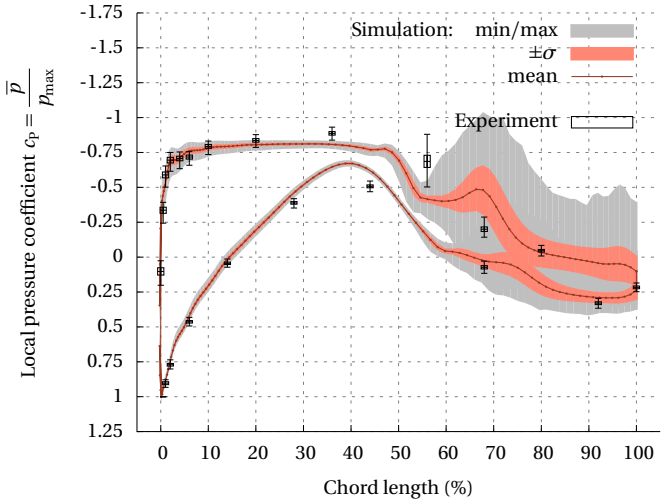


(b) Cut at 46.6 % blade radius

Figure 5.18: Pressure coefficients for NREL case S0500000

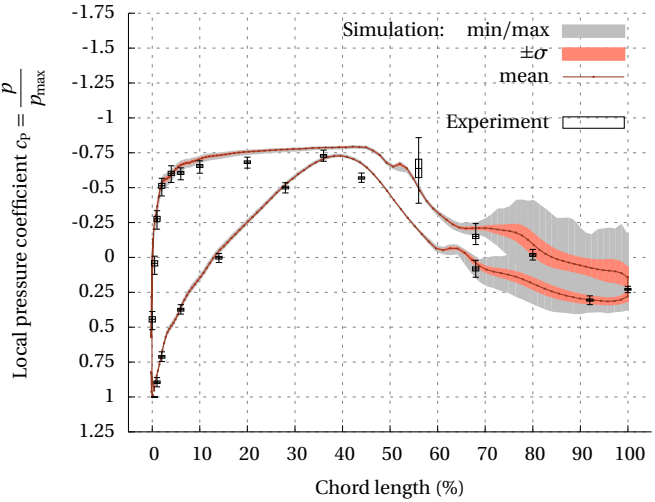


(c) Cut at 63.3 % blade radius



(d) Cut at 80 % blade radius

Figure 5.18: Pressure coefficients for NREL case S0500000



(e) Cut at 95 % blade radius

Figure 5.18: Pressure coefficients for NREL case S0500000

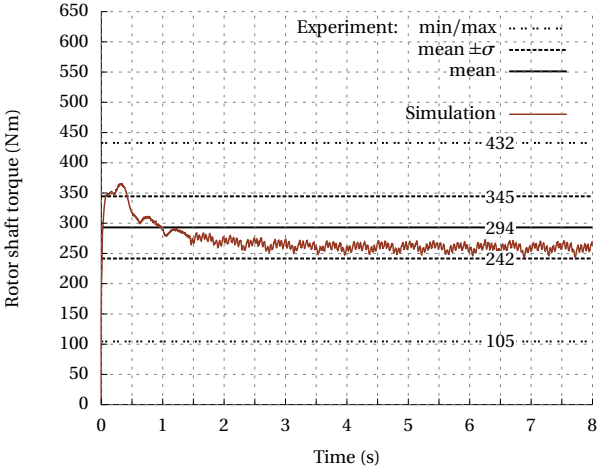
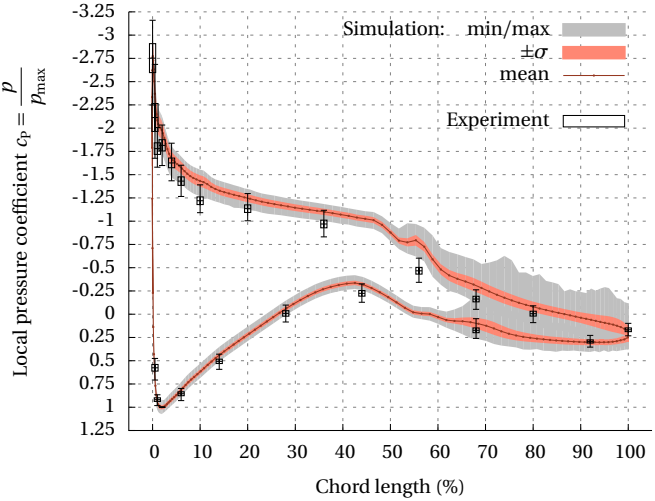
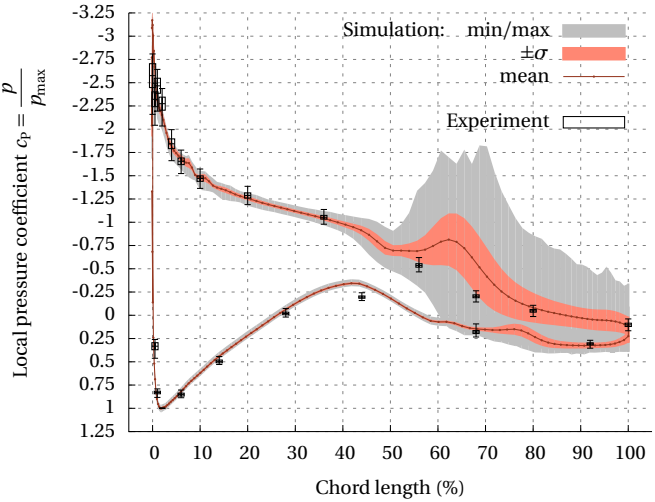


Figure 5.19: Low speed shaft torque S0500000

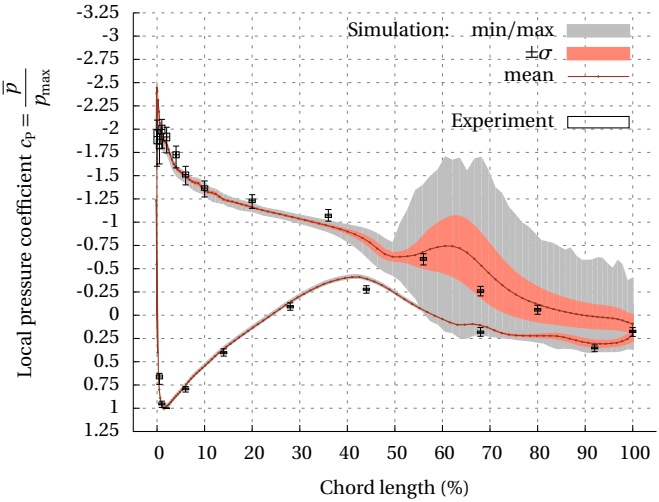


(a) Cut at 30 % blade radius

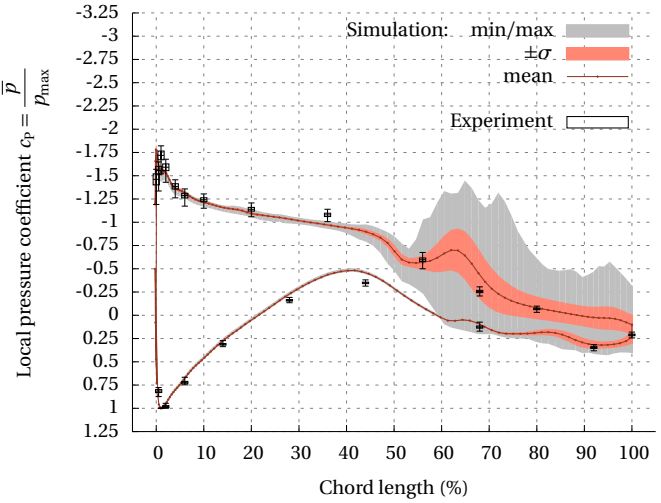


(b) Cut at 46.6 % blade radius

Figure 5.20: Pressure coefficients for NREL case S0700000

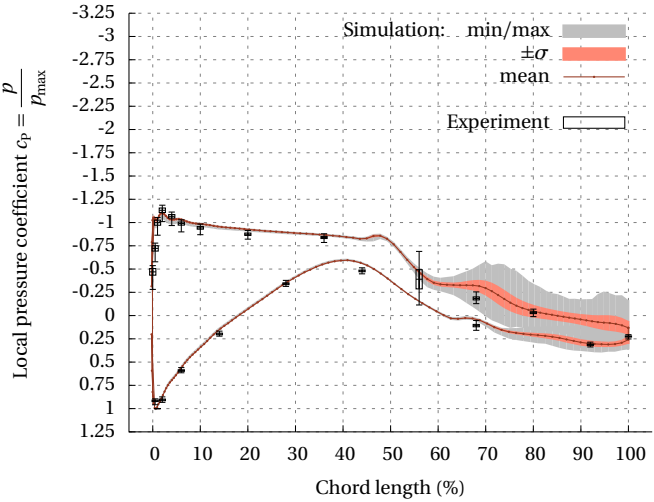


(c) Cut at 63.3 % blade radius



(d) Cut at 80 % blade radius

Figure 5.20: Pressure coefficients for NREL case S0700000



(e) Cut at 95 % blade radius

Figure 5.20: Pressure coefficients for NREL case S0700000

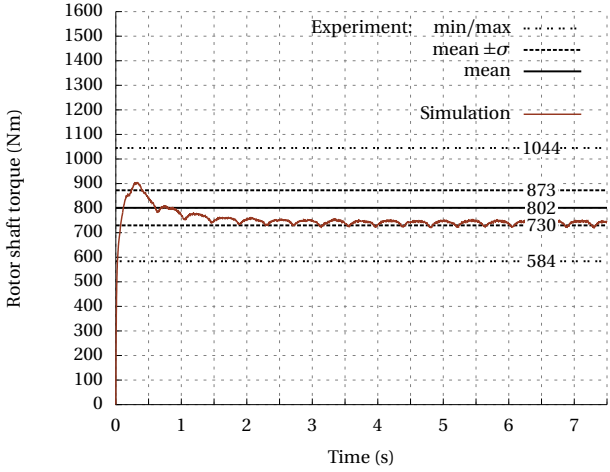


Figure 5.21: Low speed shaft torque S0700000

For the pressure coefficients and low speed shaft torque good agreement between simulation and measurement is achieved. In order to get more insight into the flow field the surface streamlines are presented. Figure 5.22 shows a clear three dimensional flow pattern towards the trailing edge of the blade.

(a) High pressure side

(b) Low pressure side

Figure 5.22: Surface streamlines of blade 3 for S0700000 at 7.5 s

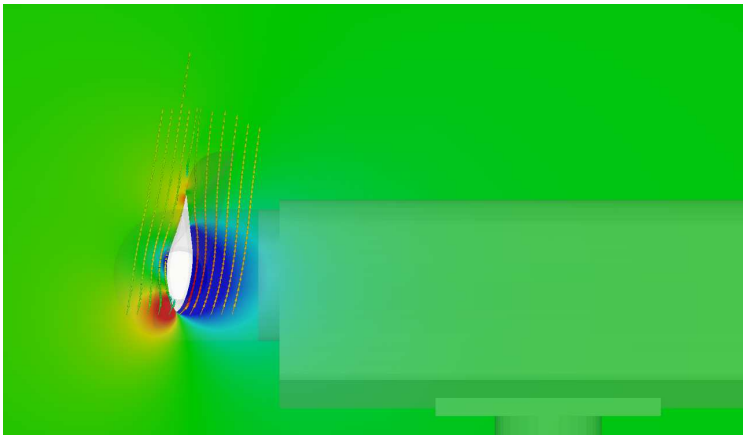


Figure 5.23: Pressure contour plot at 5.00 s including streamlines at 80% radius of blade 1 of sequence S0700000

The flow condition for wind turbines are exceptionally challenging. This is due to the varying twist angle, chord length and effective

velocity over the blade radius. The effective velocity is the resultant of the angular and upstream velocity (see Figure 5.24). Figure 5.25 illustrates the change of the effective velocity and the local Reynolds number with respect to the blade radius.

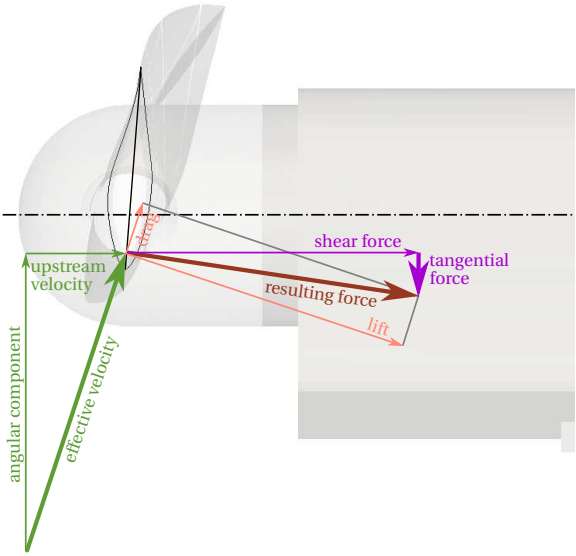


Figure 5.24: Velocity and force components of a horizontal axis wind turbine

Finally, Figure 5.23 gives insight into the pressure field of the flow field for the rotating turbine. The streamlines depicted in Figure 5.23 indicate the fully attached flow conditions.

5.3.3 CSM Model

After validating the CFD model, the CSM model for the flexible blades is validated in this section. The composite structure of the blades is described in detailed in Van Dusen [150]. The aerodynamic, dynamic and dead loads are carried by a carbon fiber D-spar that tapers in thickness from root to tip, shown in Figures 5.26 to 5.27. The non-load-carrying skin is fiberglass. A 3.175 mm thick honeycomb core was included aft of the spar dam for added skin stiffness.

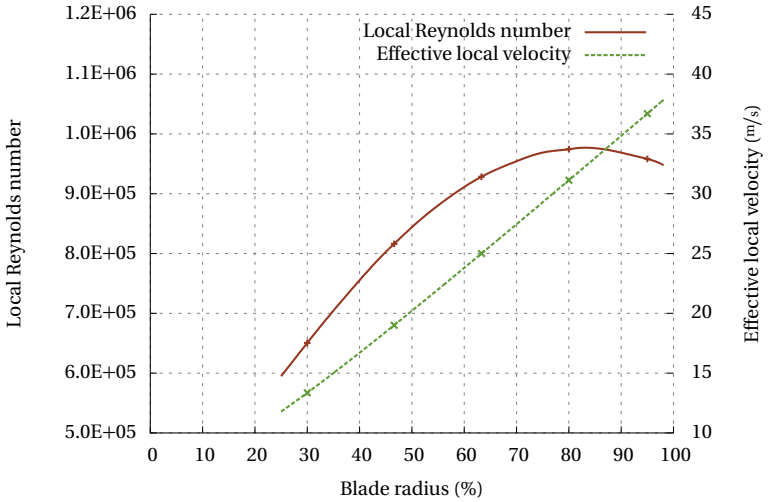


Figure 5.25: Local Reynolds number and effective local velocity for sequence S0700000

Classical laminate theory (see Daniel et al. [33]) is used in order to model the structure where all properties are given in Van Dusen [150]. The measurement instrumentation and paint is modeled by using non-structural mass. The root of the blades is clamped by using a kinematic coupling which constrains all the root DOFs of the blades to one central node. This node has 6 DOFs, namely 3 displacements and 3 finite rotations. Each blade is discretized with ≈ 60 000 fully integrated finite-membrane-strain shell elements (S3 and S4). A nonlinear kinematic formulation is chosen in order to handle the finite rotations. In Abaqus/Standard the Hilber-Hughes-Taylor method is set for the time integration.

Results & Validation

For validation an eigenfrequency analysis is performed and compared to the measurement data from Hand et al. [69, p. 79]. A comparison between measurement and simulation can be found in Table 5.7.

The relative deviation between simulation and measurement is below 10 % except for the first edge-wise eigenfrequency. The cor-

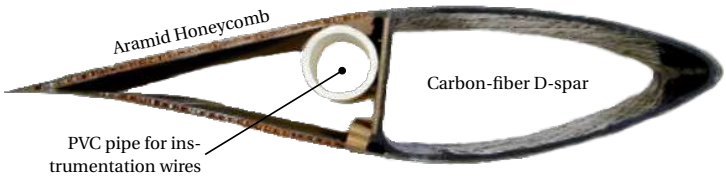


Figure 5.26: Photograph of blade section presented in Hand et al. [69, p. 77]

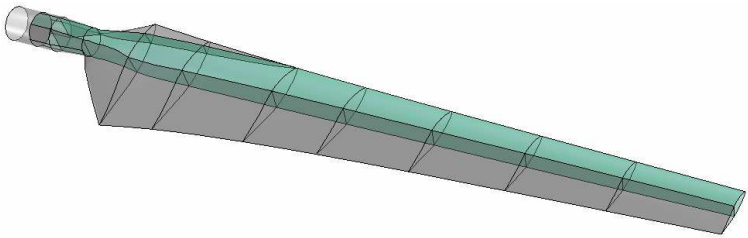


Figure 5.27: Outer and inner geometry of the blade (green D-spar structure)

Table 5.7: Structural properties for blade 1 & blade 3

Property	Measured value	Simulation value	Unit	Relative error %
Blade mass with standard tip	60.2	60.12	kg	0.13
Blade center of gravity with standard tip	2.266	2.287	m	0.92
First flap-wise eigenfrequency	7.313	7.901	Hz	8.04
Second flap-wise eigenfrequency	30.062	30.981	Hz	3.05
First edge-wise eigenfrequency	9.062	12.740	Hz	40.58

responding eigenvector of the first edge-wise eigenfrequency (see Figure 5.28) shows that the stiffness for the hub adapter is decisive for this eigenfrequency. As no detailed geometry information is available for the adapter it is modeled as rigid, therefore the eigenfrequency is higher than the measurement. However, this eigenfrequency is not decisive for the dynamic behavior of the turbine as the fluid load mainly excites the flap-wise eigenfrequencies.

The displacement contour plot of the gravity loaded turbine blades is depicted in Figure 5.29. Under this loading the maximal tip displacement is 2.3 mm.

5.3.4 Handling Deformations & Rotations

Since both field-type subsystem models (CFD and CSM) are validated so far the coupling issues are discussed in the following. As the used CFD approach is based on ALE (see Section 5.3.2) special care needs to be taken when the displacements of the rotor are applied to the CFD solver.

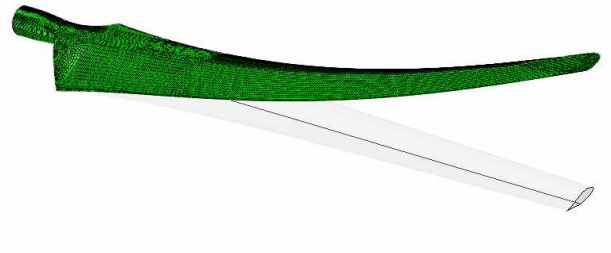
Typically the structural solver is outputting a displacement field with respect to the original frame of reference (also called initial configuration see Donea et al. [37]). The total displacement field with respect to the original frame is denoted with $\boldsymbol{u}_{\text{tot}}$.

OpenFOAM expects the mesh deformation also with respect to the initial mesh configuration. However, due to the boundary fitted approach the structural displacement field needs to be decomposed in a rigid body part stemming from the rotation (main rotation and pitch) and a part stemming from the deformation of the blades. Hence, we have

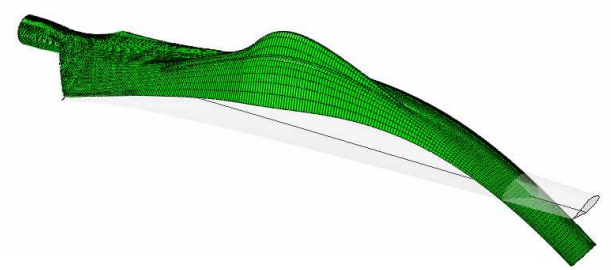
$$\boldsymbol{u}_{\text{tot}} = \boldsymbol{u}_{\text{rot}} + \boldsymbol{u}_{\text{ptc}} + \boldsymbol{u}_{\text{def}}. \tag{5.23}$$

Here $\boldsymbol{u}_{\text{rot}}$ is a rigid body rotation around the y-axis of the turbine, so it represents the main rotor motion of the turbine. $\boldsymbol{u}_{\text{ptc}}$ is also a rigid body rotation representing the pitching of blade 1. Hence, $\boldsymbol{u}_{\text{ptc}}$ is a rotation around the x-axis of the turbine. Last but not least, $\boldsymbol{u}_{\text{def}}$ is the displacement field stemming from the elastic/plastic deformation of the blades.

It is possible to represent the total deformation of the turbine with one global rotation vector representing $\boldsymbol{u}_{\text{rot}} + \boldsymbol{u}_{\text{ptc}}$ and the global deformation vector $\boldsymbol{u}_{\text{def}}$.



(a) First flap-wise eigenmode



(b) Second flap-wise eigenmode

(c) First edge-wise eigenmode

Figure 5.28: Eigenmode shapes of CSM model

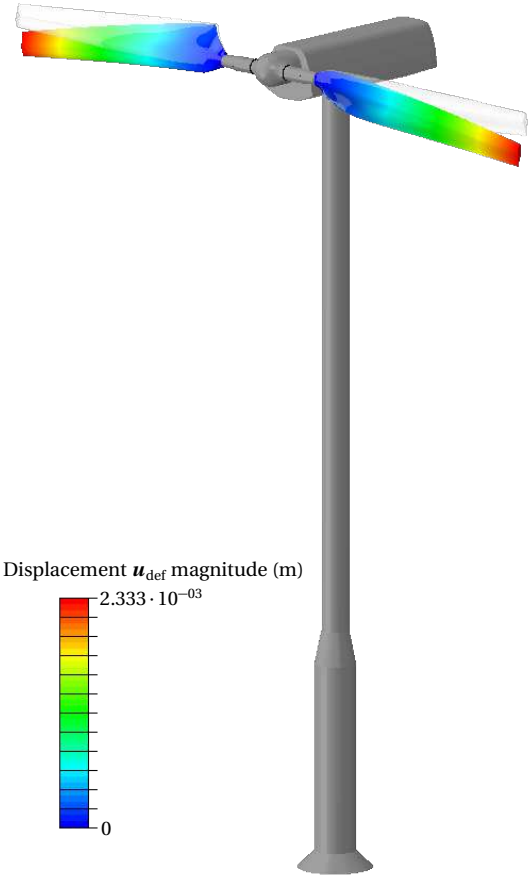


Figure 5.29: Displacement contour plot of gravity loaded turbine; deformation scaled by 200

The rigid body rotation of the pitching around the x-axis is applied to mesh part AMI 2a. It is assumed that the pitch angle of blade 3 is the negative of the one of blade 1. Hence, the negative rigid body rotation around the x-axis is applied to blade 3. The main rigid body rotation around the y-axis is applied to the mesh parts AMI 1, AMI 2a and AMI 2b. The remaining displacement vector $\boldsymbol{u}_{\text{def}}$ is diffused via the Laplacian mesh motion approach, which is described in Jasak

et al. [82], from the blades surfaces into the corresponding mesh parts AMI 2a and AMI 2b.

In the following it is shown how to compute $\boldsymbol{u}_{\text{rot}}$, $\boldsymbol{u}_{\text{ptc}}$ and $\boldsymbol{u}_{\text{def}}$ from a given field of the structural solver $\boldsymbol{u}_{\text{tot}}$.

Dealing with Finite Rotations

In order to be able to apply the main rotation and the pitch rotation at the same time the non-commutative property of finite rotations needs to be taken into account (see Altmann [2] and Woernle [158]).

A transformation matrix which rotates the blades first around the x-axis and then around the y-axis is given by

$$\boldsymbol{R} = \begin{bmatrix} \cos(\omega) & \sin(\psi)\sin(\omega) & \cos(\psi)\sin(\omega) \\ 0 & \cos(\psi) & -\sin(\psi) \\ -\sin(\omega) & \sin(\psi)\cos(\omega) & \cos(\psi)\cos(\omega) \end{bmatrix}. \tag{5.24}$$

Where ψ is the pitch angle around the x-axis and ω being the main rotation angle around the y-axis. If needed this rotation matrix can be transformed in a global rotation vector by converting it to quaternions, which is discussed in Woernle [158, p. 83].

If we assume that ψ and ω are known we can write

$$\boldsymbol{u}_{\text{def}} = \boldsymbol{u}_{\text{tot}} - \boldsymbol{u}_{\text{rot}} - \boldsymbol{u}_{\text{ptc}}. \tag{5.25}$$

This can be reformulated by using Equation (5.24) to

$$\boldsymbol{u}_{\text{def}} = \boldsymbol{u}_{\text{tot}} - \boldsymbol{R}\,\boldsymbol{u}_{\text{tot}}. \tag{5.26}$$

Note that the coupling is done with angular velocities. The velocities are integrated by the time integrator used by the individual subsystem. Hence, the transformations are performed on the basis of angles again. This is done in order to avoid high acceleration oscillations as shown in Section 3.2.

5.3.5 Fluid-Structure Interaction Model

Similar to the pure CFD case a prescribed constant angular velocity of $\dot{\omega} = 432\,^{\circ}/\text{s}$ is used for the CFD and the CSM model. Inside the CSM solver the rigid part of the deformation is removed and only $\boldsymbol{u}_{\text{def}}$ is

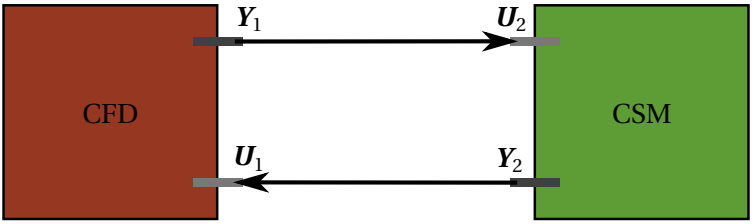


Figure 5.30: Block diagram for fluid-structure interaction

Table 5.8: Description of input and output quantities for the fluid-structure interaction model

Symbol	Description	Unit
$\boldsymbol{U}_1$	Input to CFD displacement field $\boldsymbol{u}_{\text{def}}$	m
$\boldsymbol{Y}_1$	Output of CFD force field	N
$\boldsymbol{U}_2$	Input to CSM force field	N
$\boldsymbol{Y}_2$	Output of CSM displacement field $\boldsymbol{u}_{\text{def}}$	m

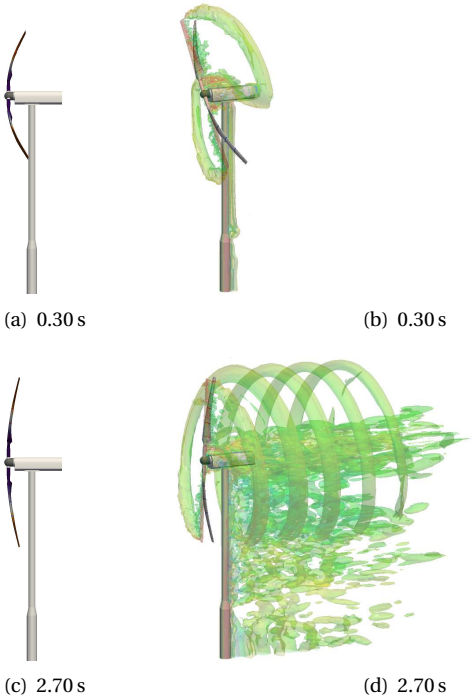
exchanged with the CFD solver. Therefore, the block diagram for the fluid-structure interaction case can be stated as shown in Figure 5.30.

With the help of Table 5.8 and the block diagram (see Figure 5.30) the global interface residual vector can be defined by

$$\boldsymbol{r} = \begin{bmatrix} \mathcal{R}_1 \\ \mathcal{R}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{U}_1 - \boldsymbol{Y}_2 \\ \boldsymbol{U}_2 - \boldsymbol{Y}_1 \end{bmatrix}. \tag{5.27}$$

Co-Simulation

The IJCSA is deployed in order to minimize the interface residual vector defined in Equation (5.27). Similar to Section 5.1 the global interface Jacobian is approximated by the identity matrix. In contrast to the Turek example of Section 5.1 no under-relaxation is applied $\alpha = 1$ as the interaction between fluid and structure is not as severe as



(a) 0.30 s (b) 0.30 s

(c) 2.70 s (d) 2.70 s

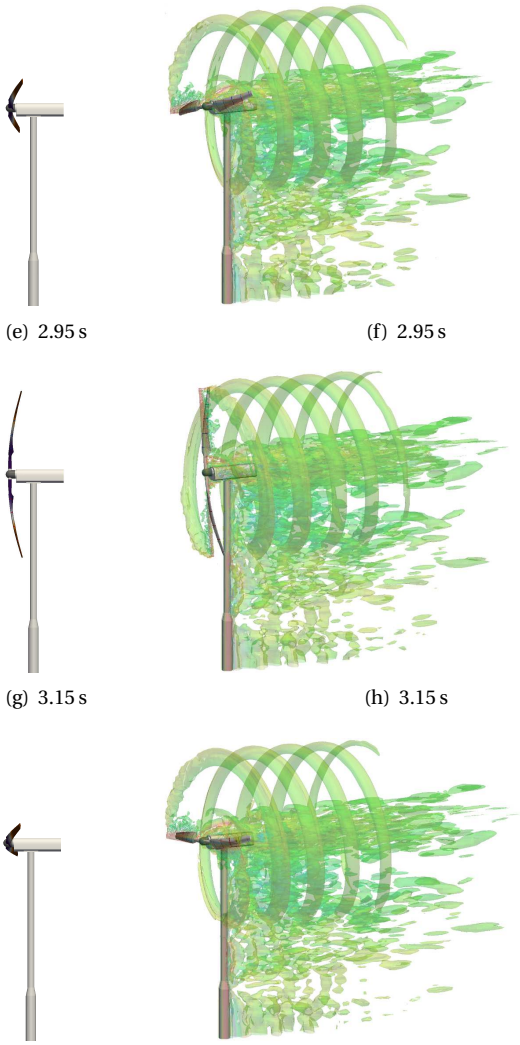
Figure 5.31: Q-criterion isosurface colored by velocity magnitude for the FSI case, where $\boldsymbol{u}_{\text{def}}$ is scaled by factor of 70

for the Turek example. Essentially, this means that the two interface Jacobian blocks $\partial \boldsymbol{Y}_1 / \partial \boldsymbol{U}_1$ and $\partial \boldsymbol{Y}_2 / \partial \boldsymbol{U}_2$ are set to zero. Despite the fact that this seems to be a crude approximation for the interface Jacobian only two interface iterations were necessary to achieve

$$\left\| \boldsymbol{\mathcal{R}}_1 \right\|_\epsilon < 1.0 \cdot 10^{-5} \text{m}. \tag{5.28}$$

Results

A time series of this simulation is illustrated in Figure 5.31. There the flow field is visualized by the Q-criterion according to Hunt et al. [78]. Note that an overview of different vortex identification schemes can be found in Chakraborty et al. [29].



(e) 2.95 s (f) 2.95 s

(g) 3.15 s (h) 3.15 s

(i) 3.35 s (j) 3.35 s

Figure 5.31: Q-criterion isosurface colored by velocity magnitude for the FSI case, where $\boldsymbol{u}_{\text{def}}$ is scaled by factor of 70

The fluid-structure interaction simulation was performed with sequence S0700000 only, as this case gives higher loads on the blades as the inlet velocity is higher than for sequence S0500000.

The low speed shaft torque is compared to the pure CFD case in Figure 5.32. The impact of including the fluid-structure interaction for the case where the angular velocity is prescribed is rather small. This results in a maximum blade tip displacement in flow direction of approximately 1 cm as shown in Figure 5.32. The blade-tower interaction is nicely illustrated by Figure 5.32 as the lowest frequency of the tip displacement curve spectra is 1.2 Hz.

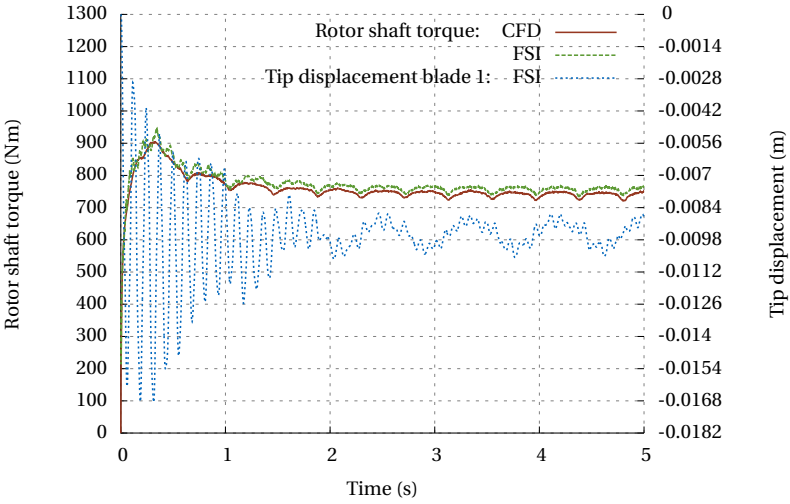


Figure 5.32: Low speed shaft torque S0700000 including FSI with tip displacement in flow direction of blade 1

5.3.6 Emergency Brake Maneuver

In the previous section a FSI co-simulation was presented. However, within this simulation an important part of physics is missing as the angular velocity is prescribed and is not determined by the interaction of the fluid with the blades and the generator.

This defect is removed in the following. In order to focus the discussion the blade flexibility is removed within this section again. The FSI for constant angular velocity renders small displacements and stresses. However, for other load situations of the turbine this changes drastically. One of these cases is an aerodynamic emergency brake maneuver where the wind turbine is brought to standstill by pitching the blades in stall conditions.

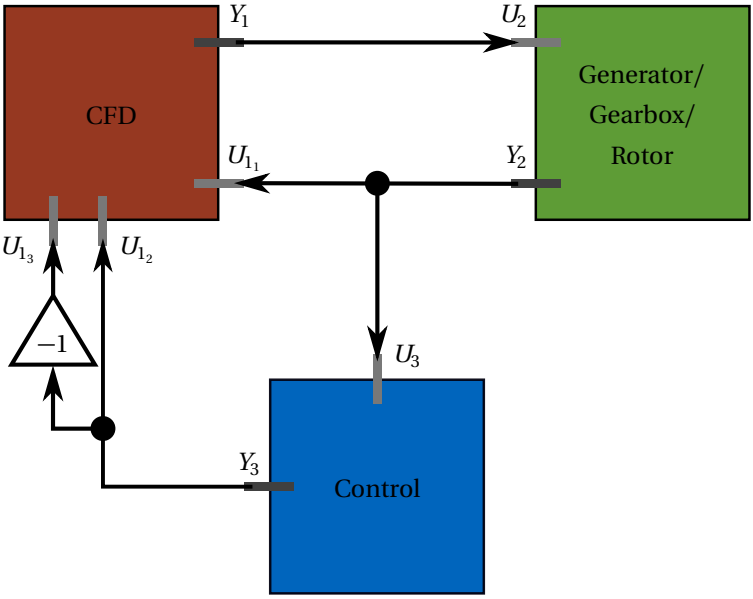


Figure 5.33: Block diagram for emergency brake maneuver

In order to simulate a physically accurate emergency brake maneuver of the NREL Phase VI wind turbine three subsystems are needed, the CFD, a generator/gearbox/rotor subsystem and an open-loop control unit.

Table 5.9: Description of input and output quantities for the emergency brake maneuver

Symbol	Description	Unit
U_{1_1}	Input to CFD angular velocity of rotor around y-axis $\dot{\omega}$	$^\circ/s$
U_{1_2}	Input to CFD angular pitching velocity around x-axis for blade 1 $\dot{\psi}$	$^\circ/s$
U_{1_3}	Input to CFD angular pitching velocity around x-axis for blade 3 $-\dot{\psi}$	$^\circ/s$
Y_1	Output of CFD aerodynamic torque around y-axis b_{CFD}	Nm
U_2	Input to Generator/Rotor aerodynamic torque around y-axis b_{CFD}	Nm
Y_2	Output of Generator/Rotor angular velocity of rotor around y-axis $\dot{\omega}$	$^\circ/s$
U_3	Input to Control Unit angular velocity of rotor around y-axis $\dot{\omega}$	$^\circ/s$
Y_3	Output of Control Unit angular pitching velocity around x-axis for blade 1 $\dot{\psi}$	$^\circ/s$

With the help of Table 5.9 and the block diagram (see Figure 5.33) the global interface residual vector can be defined by

$$\boldsymbol{r} = \begin{bmatrix} \mathcal{R}_1 \\ \mathcal{R}_2 \\ \mathcal{R}_3 \\ \mathcal{R}_4 \\ \mathcal{R}_5 \end{bmatrix} = \begin{bmatrix} U_{1_1} - Y_2 \\ U_{1_2} - Y_3 \\ U_{1_3} + Y_3 \\ U_2 - Y_1 \\ U_3 - Y_2 \end{bmatrix}. \tag{5.29}$$

Generator/Gearbox/Rotor Model

The generator/gearbox/rotor subsystem mainly models the mass moment of inertia, friction and electrical behavior of the power train, the generator and the rotor. Hence the following linear ordinary dif-

ferential equation can be used as a simplified model for this system:

$$J\frac{\mathrm{d}\dot{\omega}}{\mathrm{d}t} + D\dot{\omega} = b_{\mathrm{CFD}} \tag{5.30}$$

$\dot{\omega}$ is the angular velocity and J the mass moment of inertia. D represents the generator as it removes energy from the system. The torque, which acts on the rotor around the horizontal y-axis is denoted by b_{CFD}. This subsystem is integrated in time via the BDF2 method, the same method as used in the CFD subsystem which results in

$$\dot{\omega}^{n+1} = \frac{4J}{3J+2hD}\dot{\omega}^{n} - \frac{J}{3J+2hD}\dot{\omega}^{n-1} + \frac{2h}{3J+2hD}b_{\mathrm{CFD}}^{n+1}. \tag{5.31}$$

Here the time step size is denoted by h. The input U_2 of the generator subsystem is given by b_{CFD} and its output Y_2 is given by $\dot{\omega}^{n+1}$. Hence, the interface Jacobian of the generator subsystem is given by

$$\frac{\partial Y_2}{\partial U_2} = \frac{\partial \dot{\omega}^{n+1}}{\partial b_{\mathrm{CFD}}^{n+1}} = \frac{2h}{3J+2hD}. \tag{5.32}$$

In order to show the potential of the IJCSA the measured mass moment of inertia of the turbine is reduced by a factor of ten. This results in a highly sensitive behavior with respect to the pitch angle of the turbine. The mass moment of inertia is set to $J = 111\,\mathrm{kg}\cdot\mathrm{m}^2$ and the angular damping coefficient to $D = 10.62\,\mathrm{Nm}\cdot\mathrm{s}$.

Control Unit

The open-loop control unit observes the angular velocity $\dot{\omega}$ and triggers an emergency brake maneuver, when $\dot{\omega}$ exceeds $500^\circ/\mathrm{s}$. The control unit takes also care of the startup procedure of the turbine. It gradually pitches the blades to operation conditions (pitch angle $\psi = 0^\circ$). The pitch angle over time is depicted in Figure 5.37 and a zoomed view is shown in Figure 5.38. Note that $\psi = 0^\circ$ corresponds to the geometrical position of the blades of the pure CFD case.

Co-Simulation

Similar to Section 5.2.3 the secant version of the IJCSA is used in order to solve the co-simulation scenario.

Note that the interface residuals $\mathcal{R}_2$, $\mathcal{R}_3$, $\mathcal{R}_5 \equiv 0$ by definition due to the open-loop control, as its output Y_3 does not depend on the input U_3. The input is only used for triggering the emergency brake maneuver. Therefore all interface Jacobian parts associated with the open-loop control unit are zero.

With this knowledge the reduced global interface residual vector can be formulated to

$$\boldsymbol{r} = \begin{bmatrix} \mathcal{R}_1 \\ \mathcal{R}_4 \end{bmatrix} = \begin{bmatrix} U_{1_1} - Y_2 \\ U_2 - Y_1 \end{bmatrix} = \begin{bmatrix} U_{1_1} - \mathcal{S}_2\left(U_2\right) \\ U_2 - \mathcal{S}_1\left(U_{1_1}\right) \end{bmatrix}, \tag{5.33}$$

where $\mathcal{S}_1$ denotes OpenFOAM and $\mathcal{S}_2$ the generator/gearbox/rotor subsystem.

According to the reduced interface residual vector the global interface Jacobian is evaluated to

$$\mathbf{J}_{\text{global}} = \begin{bmatrix} \frac{\partial \mathcal{R}_1}{\partial U_{1_1}} & \frac{\partial \mathcal{R}_1}{\partial U_2} \\ \frac{\partial \mathcal{R}_4}{\partial U_{1_1}} & \frac{\partial \mathcal{R}_4}{\partial U_2} \end{bmatrix} = \begin{bmatrix} 1 & -\frac{\partial Y_2}{\partial U_2} \\ -\frac{\partial Y_1}{\partial U_{1_1}} & 1 \end{bmatrix}. \tag{5.34}$$

The secant version of the IJCSA is used in order to solve the co-simulation scenario. The interface Jacobian part of the discretized generator/gearbox/rotor model (see Equation (5.31)) is provided by the subsystem. It takes the constant value

$$\frac{\partial Y_2}{\partial U_2} = \frac{2h}{3J + 2hD}, \tag{5.35}$$

according to Equation (5.32). The time-variant interface Jacobian part of OpenFOAM is approximated by using the secant method

$$\frac{\partial Y_1}{\partial U_{1_1}} \approx \frac{{}^{m}Y_1^{n+1} - {}^{m-1}Y_1^{n+1}}{{}^{m}U_{1_1}^{n} - {}^{m-1}U_{1_1}^{n}}, \tag{5.36}$$

(see line 11 of Algorithm 4.5).

Hence, the global interface Jacobian is evaluated during the co-simulation as

$$\mathbf{J}_{\text{global}} = \begin{bmatrix} \frac{\partial \mathcal{R}_1}{\partial U_{1_1}} & \frac{\partial \mathcal{R}_1}{\partial U_2} \\ \frac{\partial \mathcal{R}_4}{\partial U_{1_1}} & \frac{\partial \mathcal{R}_4}{\partial U_2} \end{bmatrix} \approx \begin{bmatrix} 1 & -\frac{2h}{3J+2hD} \\ -\frac{{}^{m}Y_1^{n+1} - {}^{m-1}Y_1^{n+1}}{{}^{m}U_{1_1}^{n} - {}^{m-1}U_{1_1}^{n}} & 1 \end{bmatrix}. \quad (5.37)$$

Figure 5.37 and Figure 5.38 show the secant approximation of the time-variant interface Jacobian component of OpenFOAM during the emergency brake maneuver.

Moreover, as this example is stable also for loosely coupled co-simulation with first-order hold extrapolation, an accuracy comparison is done. Loosely coupled means that one does not iterate within the time step ($m_{\text{end}} = 0$).

Results

Figure 5.34 and Figure 5.35 illustrate the absolute error in velocity and torque respectively if loosely coupled co-simulation with first-order hold extrapolation is carried out. The absolute error is defined as the difference between the loosely coupled co-simulation solution and the IJCSA solution, as the IJCSA states the correct solution, provided that the interface residual is zero. The overlay of the velocity and torque for loosely coupled and IJCSA coupled is demonstrated in Figure 5.36. The Figures show that the error of the loosely coupled co-simulation of the emergency brake maneuver is up to 10 % for the torque which is significant, as the peak loads are decisive for the design of such a machine.

The interface Jacobian of OpenFOAM which is estimated by the secant version of the IJCSA on the fly is visualized within Figure 5.37 and Figure 5.38. They show a noisy behavior, especially in the last part of the simulation (see Figure 5.38) where the turbine has already almost stopped. This raises the question how is the convergence behavior of the IJCSA for such conditions. The IJCSA takes at most five inter-

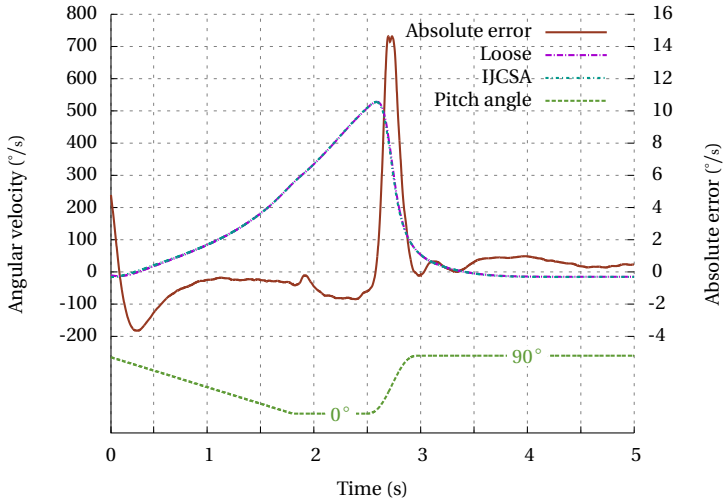


Figure 5.34: Angular velocity of loose coupling and IJCSA with absolute error for the emergency brake maneuver of S0700000

face iteration in order to converge the interface velocity to machine precision. This shows the great potential of the IJCSA.

Figure 5.39 depicts the convergence behavior during startup, Figure 5.40 during the brake maneuver and Figure 5.41 after the brake maneuver.

These kinds of co-simulations can be used not only to optimize for higher durability and performance, but they can also be used for the proper design of the control units for wind turbines. As the control unit takes care for startup of the turbine, it is crucial that pitching is not done too fast otherwise the angular velocity of the rotor is too slow which results in stall conditions of the blades as the effective angle of attack becomes too high.

Figure 5.42 shows the situation of a too fast pitching with respect to the angular velocity of the rotor. These situations can be investigated via the co-simulation.

Finally, a time series of the emergency brake maneuver is shown in Figure 5.43 where the vorticity magnitude isosurface is used to visualize the flow field.

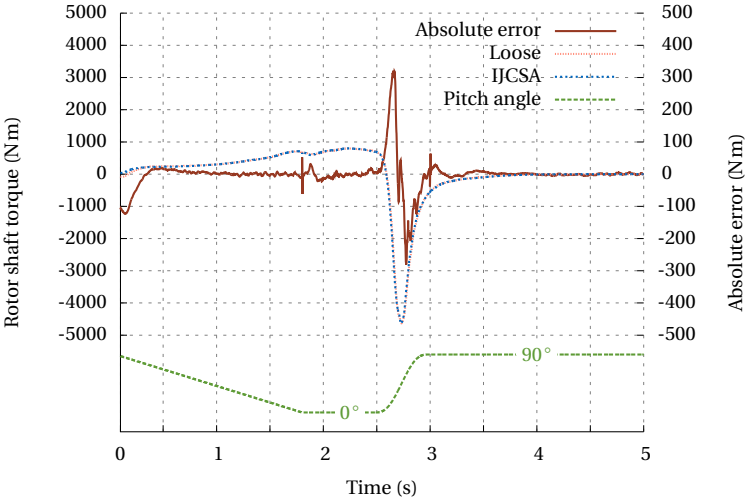


Figure 5.35: Rotor shaft torque of loose coupling and IJCSA with absolute error for the emergency brake maneuver of S0700000

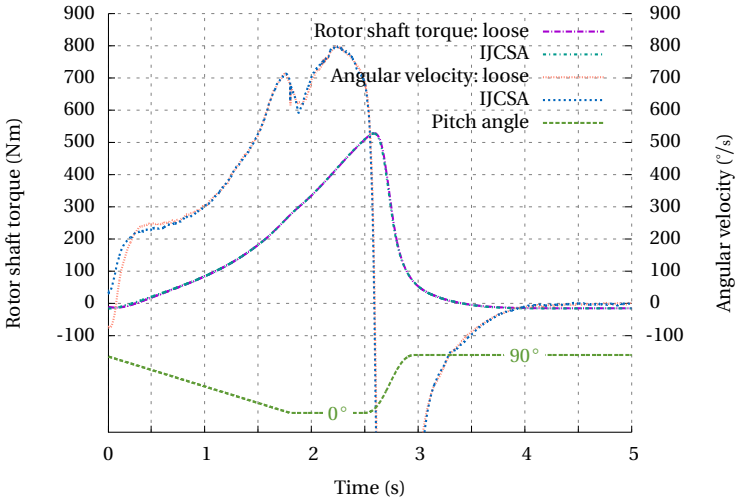


Figure 5.36: Rotor shaft torque and angular velocity of loose coupling and IJCSA for the emergency brake maneuver of S0700000

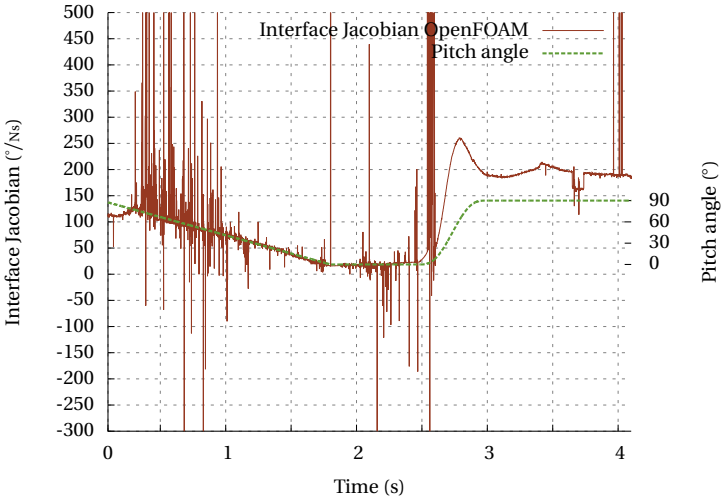


Figure 5.37: Interface Jacobian component of OpenFOAM vs. pitch angle for the emergency brake maneuver of S0700000

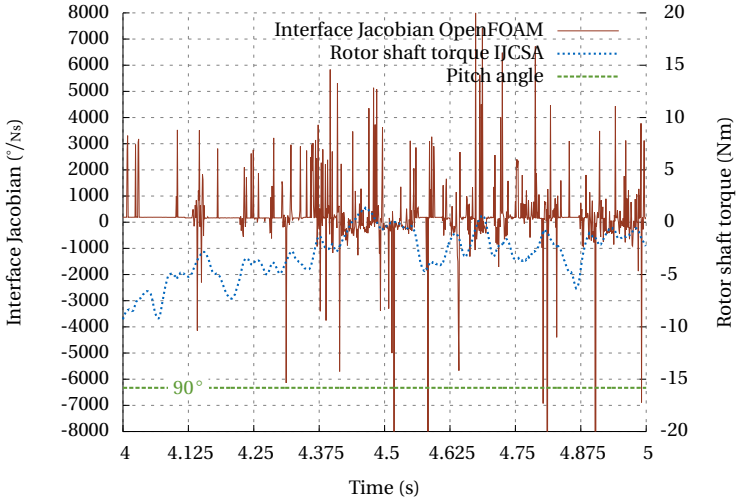


Figure 5.38: Interface Jacobian component of OpenFOAM vs. pitch angle for the emergency brake maneuver of S0700000 (zoomed)

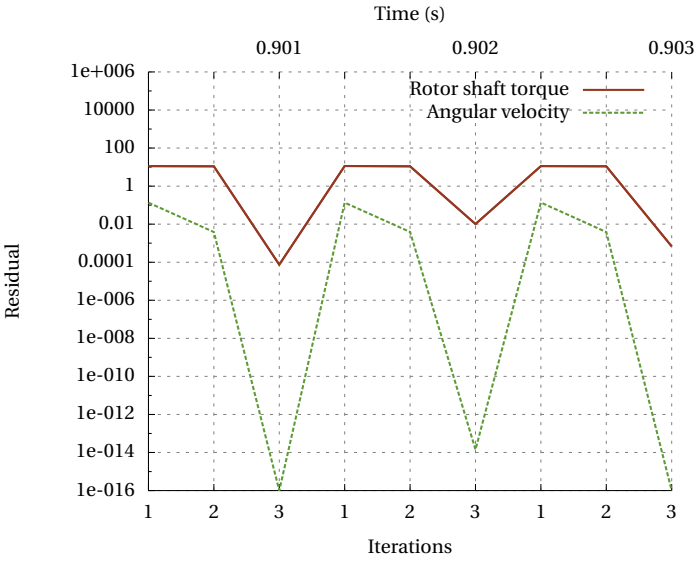


Figure 5.39: Interface residuals during startup

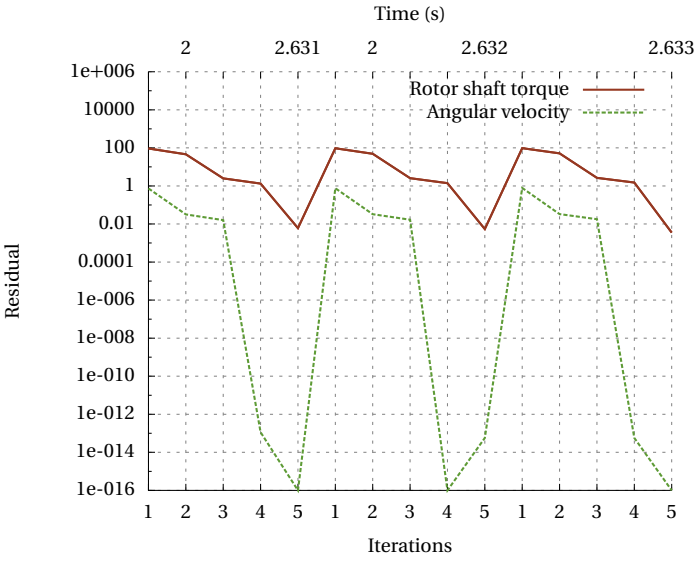


Figure 5.40: Interface residuals during emergency braking

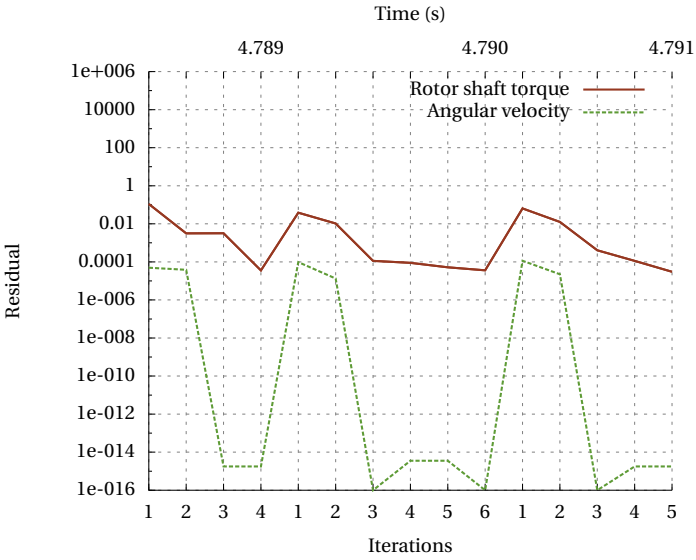


Figure 5.41: Interface residuals after emergency braking

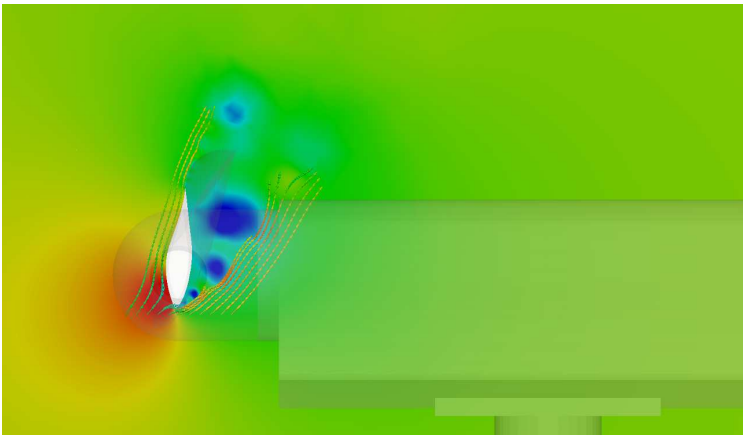
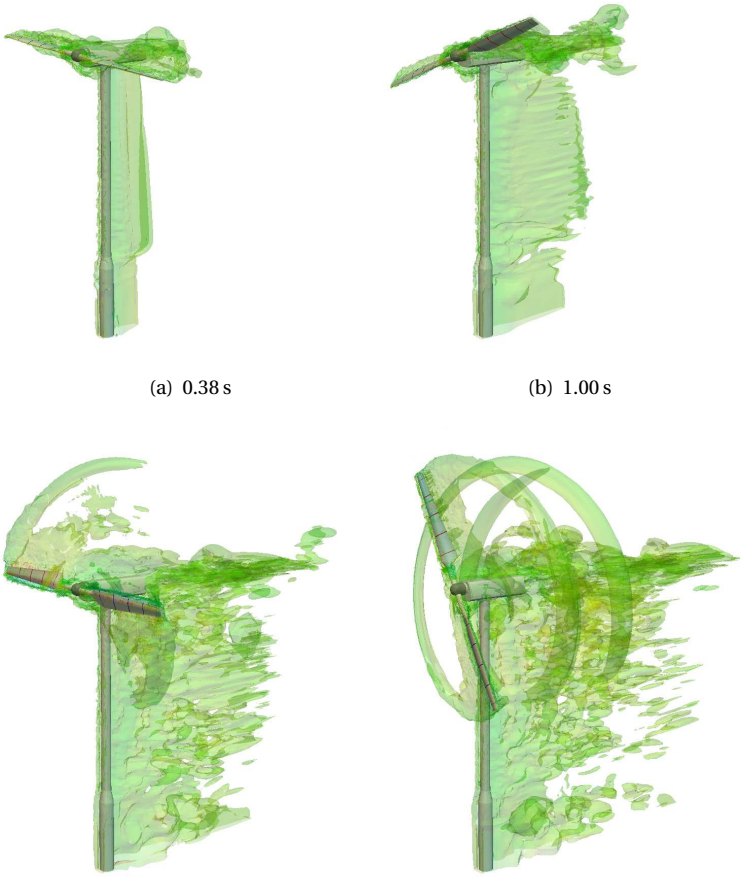


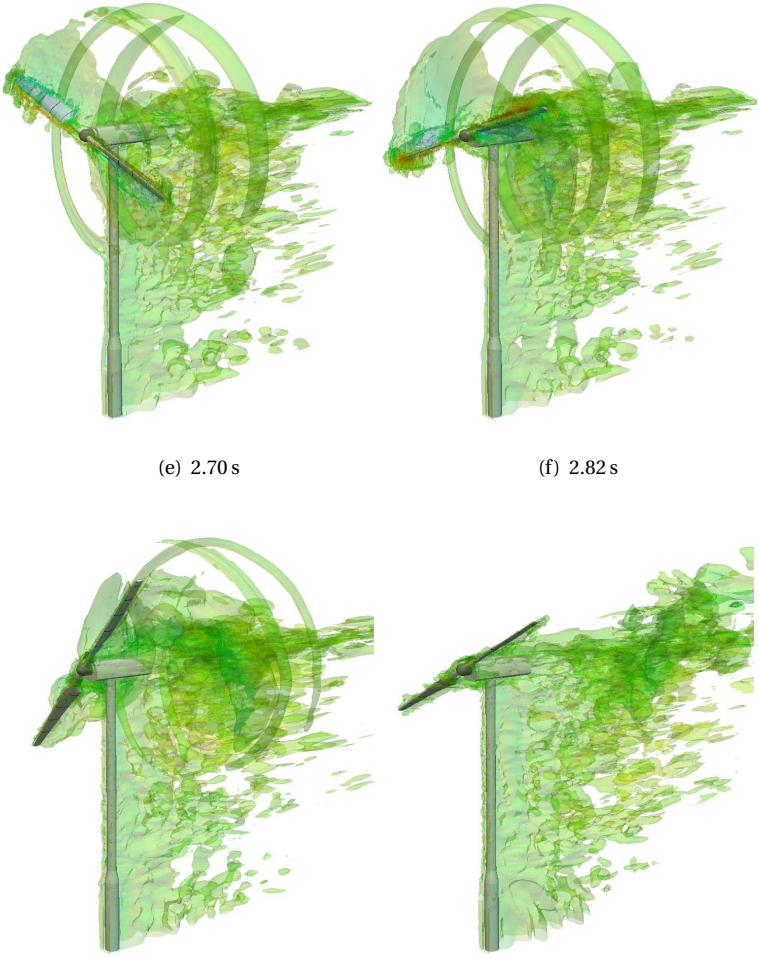
Figure 5.42: Pressure contour plot at 2.79 s including streamlines at 80% radius of blade 1 for a wrong designed control unit



(a) 0.38 s (b) 1.00 s

(c) 1.90 s (d) 2.62 s

Figure 5.43: Time series of vorticity magnitude isosurface for emergency brake maneuver colored by the velocity magnitude



(e) 2.70 s (f) 2.82 s

(g) 3.32 s (h) 5.00 s

Figure 5.43: Time series of vorticity magnitude isosurface for emergency brake maneuver colored by the velocity magnitude

5.3.7 Emergency Brake Maneuver with Flexible Blades

Within this section the FSI and the emergency brake maneuver are combined within one co-simulation. This results in a block diagram which is shown in Figure 5.44.

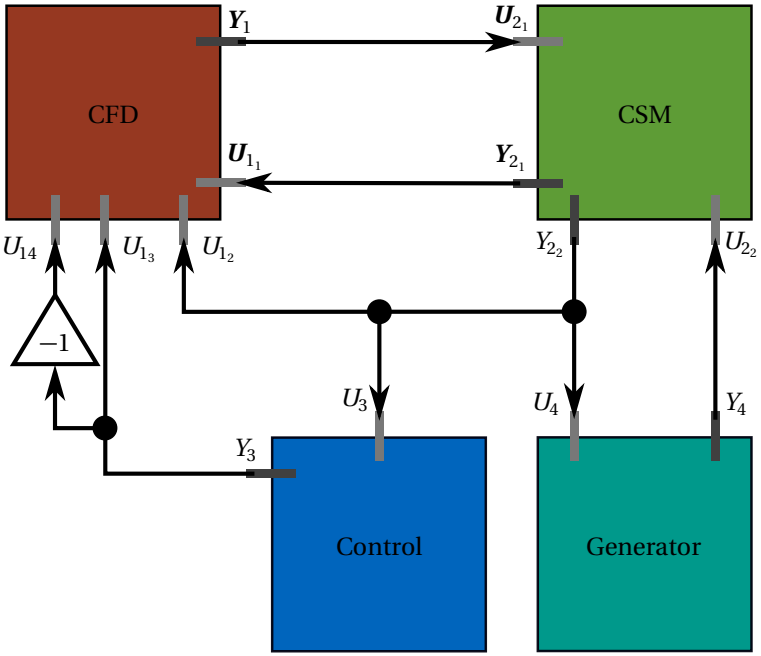


Figure 5.44: Block diagram for emergency brake maneuver with flexible blades

The corresponding input and output quantities are specified in Table 5.10.

Besides the reused subsystems CFD, CSM and control unit it is necessary to change the subsystem of the generator unit. This is due to the fact that the mass moment of inertia of the rotor is no longer needed to be modeled by this subsystem as it is already taken care of by the CSM model.

Generator/Gearbox Model

Hence, this new generator/gearbox subsystem has smaller mass moment of inertia. Moreover, it is also equipped with the possibility to work as an electric motor during the startup of the turbine. The ODE for the generator/gearbox subsystem reads

$$J\frac{\mathrm{d}\dot{\omega}}{\mathrm{d}t} + D\,\dot{\omega} = b_{\mathrm{CSM}} + b_{\mathrm{motor}}. \tag{5.38}$$

Table 5.10: Description of input and output quantities for the emergency brake maneuver with flexible blades

Symbol	Description	Unit
$\boldsymbol{U}_{1_1}$	Input to CFD displacement field $\boldsymbol{u}_{\mathrm{def}}$	m
U_{1_2}	Input to CFD angular velocity of rotor around y-axis $\dot{\omega}$	$^\circ/\mathrm{s}$
U_{1_3}	Input to CFD angular pitching velocity around x-axis for blade 1 $\dot{\psi}$	$^\circ/\mathrm{s}$
U_{1_4}	Input to CFD angular pitching velocity around x-axis for blade 3 $-\dot{\psi}$	$^\circ/\mathrm{s}$
$\boldsymbol{Y}_1$	Output of CFD force field	N
$\boldsymbol{U}_{2_1}$	Input to CSM force field	N
U_{2_2}	Input to CSM torque induced by generator b_{CSM}	Nm
$\boldsymbol{Y}_{2_1}$	Output of CSM displacement field $\boldsymbol{u}_{\mathrm{def}}$	m
Y_{2_2}	Output of CSM angular velocity of rotor around y-axis $\dot{\omega}$	$^\circ/\mathrm{s}$
U_3	Input to Control Unit angular velocity of rotor around y-axis $\dot{\omega}$	$^\circ/\mathrm{s}$
Y_3	Output of Control Unit angular pitching velocity around x-axis for blade 1 $\dot{\psi}$	$^\circ/\mathrm{s}$
U_4	Input to Generator angular velocity of rotor around y-axis $\dot{\omega}$	$^\circ/\mathrm{s}$
Y_4	Output of Generator torque induced by generator b_{CSM}	Nm

In contrast to Section 5.3.6 the mass moment of inertia is set to $J = 161\,\mathrm{kg}\cdot\mathrm{m}^2$ and the angular damping coefficient to $D = 100.1\,\mathrm{Nm}\cdot\mathrm{s}$.

Co-Simulation

With the help of Table 5.10 and the block diagram (see Figure 5.44) the global interface residual vector can be defined by

$$\boldsymbol{r} = \begin{bmatrix} \boldsymbol{\mathcal{R}}_1 \\ \mathcal{R}_1 \\ \mathcal{R}_2 \\ \mathcal{R}_3 \\ \boldsymbol{\mathcal{R}}_4 \\ \mathcal{R}_5 \\ \mathcal{R}_6 \\ \mathcal{R}_7 \end{bmatrix} = \begin{bmatrix} \boldsymbol{U}_{1_1} - \boldsymbol{Y}_{2_1} \\ U_{1_2} - Y_{2_2} \\ U_{1_3} - Y_3 \\ U_{1_4} + Y_3 \\ \boldsymbol{U}_{2_1} - \boldsymbol{Y}_1 \\ U_{2_2} - Y_4 \\ U_{3_1} - Y_{2_2} \\ U_4 - Y_{2_2} \end{bmatrix}. \tag{5.39}$$

Again, the IJCSA is deployed in order to minimize the interface residual vector. The individual interface Jacobian entries and convergence criteria of each subsystem are set accordingly to Section 5.3.5 and Section 5.3.6. Therefore the overall interface iteration count is the same as for the emergency brake maneuver of Section 5.3.6.

Results

The overall co-simulation is best explained with the help of Figure 5.45. The first thing to note is that the generator acts as a motor during the first 1.2 s of the simulation.

Within this period the motor delivers 5 000 Nm. At 4 s the control unit issues the emergency brake maneuver by pitching the blades. Towards the end of the simulation a high frequency oscillation in the shaft torque can be observed. By analyzing the CSM model it is evident that this particular setup triggers a flutter phenomena. This is also visual in the time series of the CSM model in Figure 5.46. Due to the control of the interface residual a numerical instability of the coupling can be ruled out.

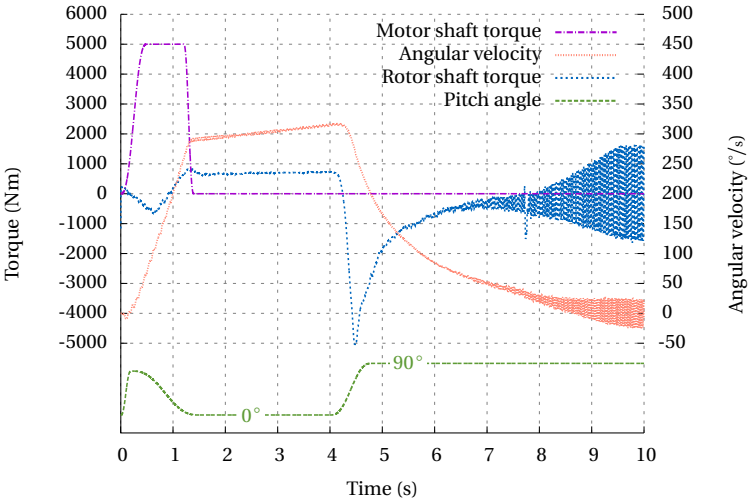
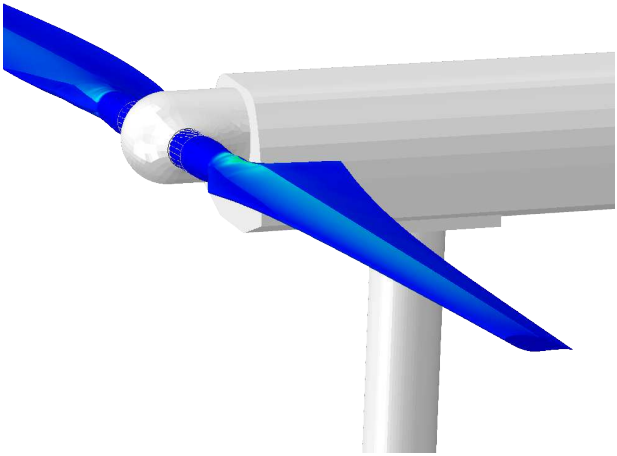
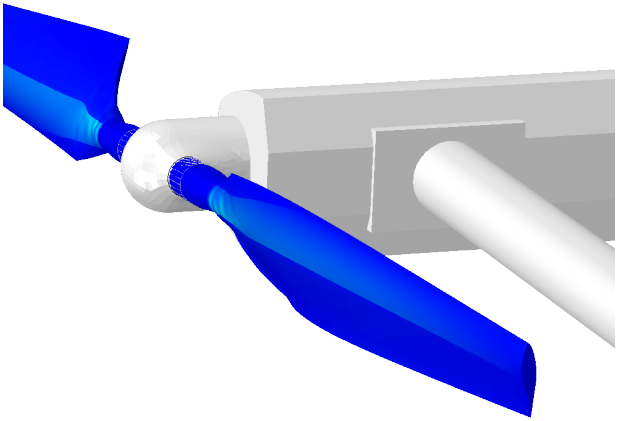


Figure 5.45: Torque and velocity for emergency brake maneuver with flexible blades

Last but not least a time series of the Q-criterion is available within Figure 5.47 which presents a summary of the overall NREL Phase VI co-simulation where the IJCSA was used. Furthermore the physical correct energy extraction from the flow field due to the wind turbine was taken into account.

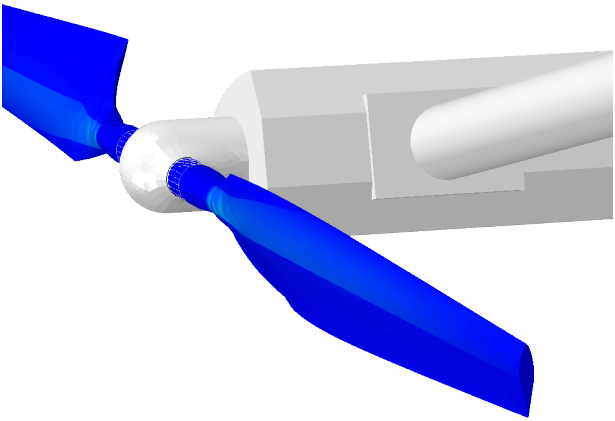


(a) 0.39 s



(b) 1.73 s

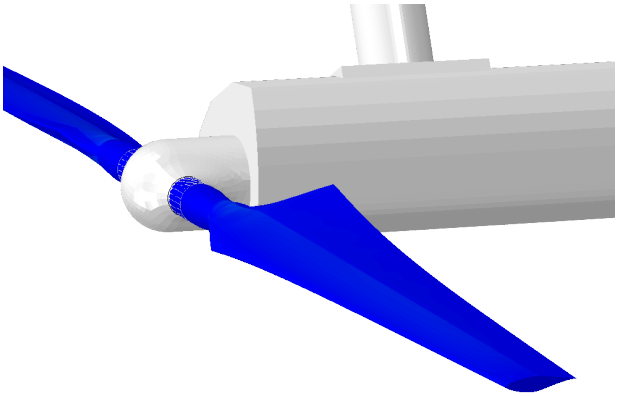
Figure 5.46: von Mises stress on outer composite layer



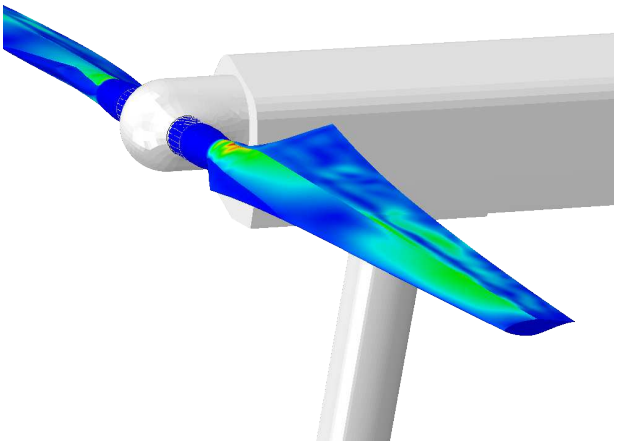
(c) 4.03 s

(d) 4.37 s

Figure 5.46: von Mises stress on outer composite layer



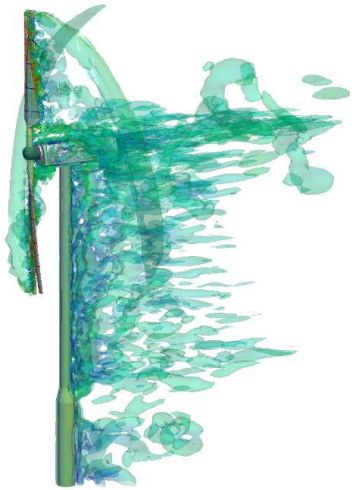
(e) 5.08 s



(f) 10.00 s

Figure 5.46: von Mises stress on outer composite layer

(a) 0.39 s



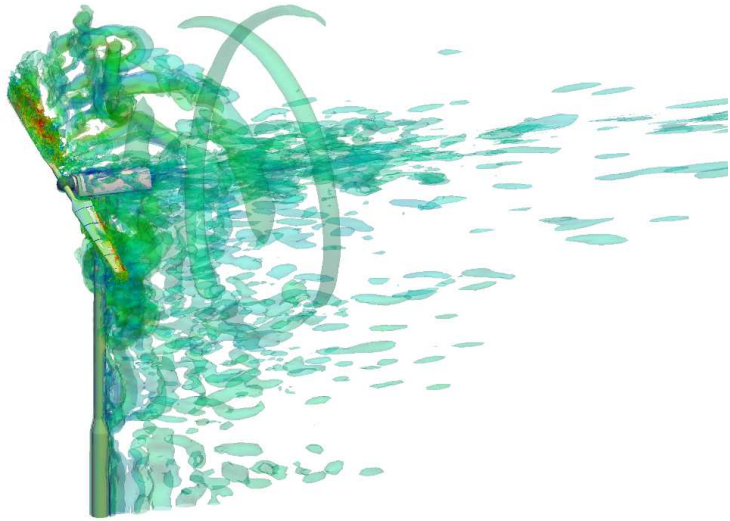
(b) 1.73 s

Figure 5.47: Q-criterion isosurface colored by velocity magnitude where $\boldsymbol{u}_{\text{def}}$ is scaled by factor of ten

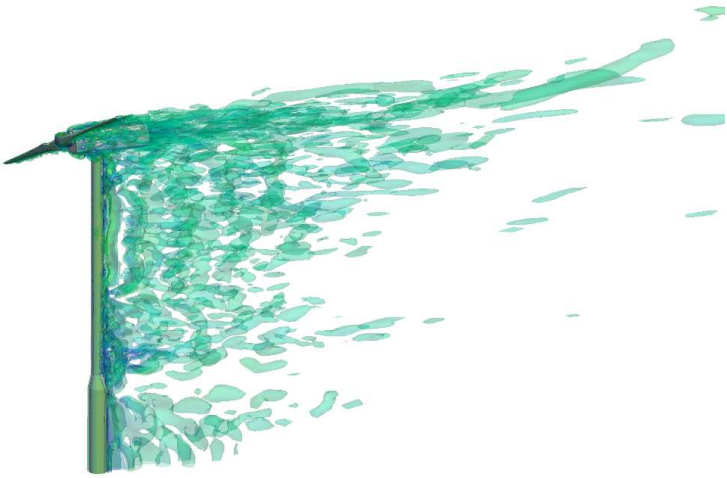
(c) 4.03 s

(d) 4.37 s

Figure 5.47: Q-criterion isosurface colored by velocity magnitude where $\boldsymbol{u}_{\text{def}}$ is scaled by factor of ten



(e) 5.08 s



(f) 10.00 s

Figure 5.47: Q-criterion isosurface colored by velocity magnitude where $\boldsymbol{u}_{\text{def}}$ is scaled by factor of ten

Learning never exhausts the mind.

Leonardo da Vinci

CHAPTER 6

CONCLUSION AND OUTLOOK

Within this work a hybrid algorithm for the solution of coupled problems is presented and benchmarked. It is hybrid in the sense of combining the advantages from the monolithic and the co-simulation approach, such as allowing for the (re)use of well-established and specialized simulation software in a modular fashion, the possibility to combine different fidelity models at all stages of the product design process and this is achieved while maintaining stability and accuracy of the solution.

The presented algorithm is based on the stabilization by interface Jacobian information, therefore it is called Interface Jacobian-based Co-Simulation Algorithm (IJCSA). Moreover, the introduced approach handles the co-simulation involving an arbitrary number of fields and signals. Due to the fact that the IJCSA is based on the residual form it handles algebraic loops in a natural manner. Furthermore, the individual simulators can run in parallel without flow dependency reducing the wall-clock time of the simulation, since the subsystems use the Jacobi pattern as a communication model.

After the theoretical derivation of the IJCSA its mathematical stability properties are analyzed and compared to classical co-simulation approaches. This clearly shows that the IJCSA is the superior choice

in terms of stability and also efficiency. Furthermore, academic test cases are used to demonstrate a variety of different use cases of the IJCSA, as there are nonlinear cases, algebraic loops, discontinuities and the handling of difficult cases with respect to stability. In addition it is possible to add any (non)linear interface constraint between the subsystems while maintaining the full modularity of co-simulation, since there is no need to modify the subsystems by any means.

Last but not least, industrial relevant use cases proof the applicability of the IJCSA. The examples add the interaction of different signals, open and closed-loop control to common fluid-structure interaction. In order to achieve that a number of different open-source and commercial software tools are coupled. At the end the emergency brake maneuver of a wind turbine is simulated and validated in order to hopefully avoid the headline "Storm caused wind turbine fire"[1] in future.

The IJCSA is so far tested for the case where all subsystems run at the same time step size and exchange information after every time step. To relax that constraint subcycling is needed. This is especially beneficial for co-simulation scenarios which involve the coupling of explicit and implicit time integrators, since then subcycling of the explicit subsystems is needed to maintain an overall coupling step size which is in the same range as the time step of the implicit integrated subsystems. For the cases of structure-structure interaction this is already discussed in Gravouil et al. [64].

The ability to perform subcycling with the IJCSA and the a-priori knowledge of the convergence order in case of mixed time integrators would be an interesting and an application relevant extension of the presented work.

[1] `http://www.bbc.co.uk/news/uk-16115139` British Broadcasting Corporation [21]

Appendices

APPENDIX A

ALGEBRAIC LOOPS

> An algebraic loop in a Simulink model occurs when a signal loop exists with only direct feedthrough blocks within the loop. Direct feedthrough means that the block output depends on the value of an input port; the value of the input directly controls the value of the output. Non-direct-feedthrough blocks maintain a State variable.

This is the definition of algebraic loop according to *Simulink User Guide R2014a* [139]. In the following this definition should be used in order to illustrate how the IJCSA can cope with algebraic loops. Similar to the definition given in *Simulink User Guide R2014a* [139] the example in Figure A.1 is constructed. For the example we define two simple subsystems. The first one is given by

$$\mathcal{S}_1\left(U_1\right) = \sin\left(U_1\right). \tag{A.1}$$

The second subsystem if given by

$$\mathcal{S}_2\left(U_2\right) = \cos\left(U_2\right). \tag{A.2}$$

An analysis of the block diagram shown in Figure A.1 renders the following two interface constraint equations:

$$\mathcal{I}_1\left(Y_2, U_1\right) = Y_2 - U_1 = 0 \tag{A.3}$$

$$\mathcal{I}_2\left(Y_1, U_2\right) = Y_1 - U_2 = U_2 \tag{A.4}$$

The latter equation is derived from the cyan block in the middle of Figure A.1 which is causing the algebraic loop. The IJCSA is used to

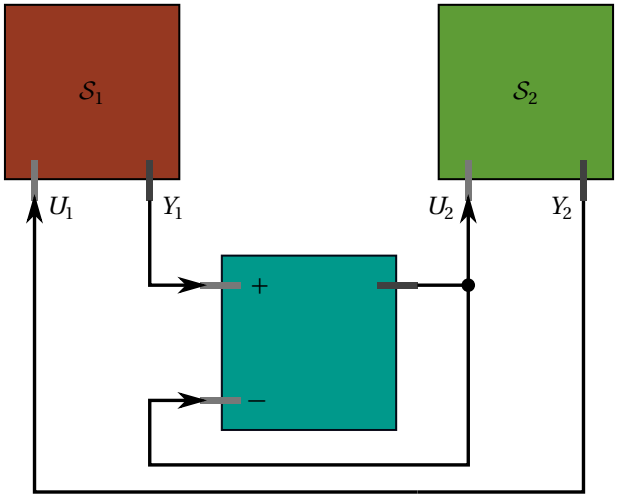


Figure A.1: Block diagram that describes the algebraic loop example

solve the interface constraint Equations (A.3) and (A.4). Based on these equations the interface residual vector is given by

$$\boldsymbol{r} = \begin{bmatrix} \mathcal{R}_1 \\ \mathcal{R}_2 \end{bmatrix} = \begin{bmatrix} Y_2 - U_1 \\ Y_1 - U_2 - U_2 \end{bmatrix}. \tag{A.5}$$

Hence the global interface Jacobian matrix is defined by

$$\boldsymbol{J}_{\text{global}} = \begin{bmatrix} -1 & -\sin\left(U_2\right) \\ \cos\left(U_1\right) & -2 \end{bmatrix}. \tag{A.6}$$

The algebraic loop example is solving equation

$$\sin\Big(\cos\big(U_2\big)\Big) - U_2 = U_2, \tag{A.7}$$

which has $3.983\,194\,523\,366\,732 \cdot 10^{-01}$ as solution.

This example demonstrates that the IJCSA can handle algebraic loops without special treatment.

APPENDIX B

ASPECTS OF CO-SIMULATION SOFTWARE REALIZATION

Within this work the algorithmic and mathematical aspects of co-simulation are discussed. Another interesting point of view of co-simulation is the implementation point of view. A client-server model fits the needs of a general co-simulation scenario, as it allows to run the overall co-simulation on a heterogeneous architecture. The server can for instance use a state machine approach as shown by Tripakis et al. [147] to control all participating subsystems.

The Co-Simulation Engine (CSE) from SIMULIA Dassault Systèmes and MpCCI from Fraunhofer SCAI are two prominent commercially available client-server-based approaches.

An open-source software called *E*nhanced *M*ulti*P*hysics *I*nterface *R*esearch *E*ngine (*EMPIRE* [40]) was developed as a part of this thesis. In EMPIRE the server is called Emperor and coordinates the co-simulation via a xml-based input file. For more information and documentation please visit the website.

For large scale co-simulations it is important that the server can communicate with clients (subsystems) in an efficient way by utilizing the high-performance computing interconnects as InfiniBand. In

order that client-server-based programs run on such interconnects it takes a considerable implementation effort. The Message Passing Interface (MPI) introduced by the Message Passing Interface Forum [103] provides a way that client-server-based approaches can be built on top of the MPI standard as shown in Gropp et al. [66] and Latham et al. [90]. Especially, for research purposes (see Schlüter et al. [129]) this is a good approach as most MPI implementation support high-performance computing interconnects in an efficient way for instance see *Intel MPI Library for Linux* OS* [79]. However, the use of MPI poses the constraint that all clients and the server need to run on the same architecture.

In order that the server can connect to an arbitrary number of clients at any time different threads need to be used inside the server application. An example using MPI-2 and OpenMPI is provided in Listing B.1 for the server and in Listing B.2 for the client.

B.1 Listing: server.c

```
// C99
// Start program: mpirun -np 1 server
#include <mpi.h>
#include <omp.h>
#include <stdio.h>
#include <stdbool.h>
#include <unistd.h> // needed for sleep() on POSIX system

#define MAX_DATA 100
int main( int argc, char **argv )
{
    int providedThreadSupport;
    bool terminateListening = false;
    char portName[MPI_MAX_PORT_NAME];
    MPI_Init_thread(&argc, &argv, MPI_THREAD_MULTIPLE, &
        providedThreadSupport);
    if (MPI_THREAD_MULTIPLE != providedThreadSupport) {
        printf( "Requested MPI thread support is not guaranteed.\n
            ");
    }
    MPI_Open_port(MPI_INFO_NULL, portName);
    printf("Server available at port:%s\n", portName);
    #pragma omp parallel num_threads(2) shared(portName,
        terminateListening)
    {
        // Use OpemMP section construct for function parallelism
        #pragma omp sections
        {
            #pragma omp section
            {
            // Do some work
            sleep(15);
            // Connect to yourself in order to terminate listening
            terminateListening = true;
            MPI_Comm dummy;
```

```
            MPI_Comm_connect(portName, MPI_INFO_NULL, 0,
                 MPI_COMM_WORLD, &dummy);
            printf("Server is connected to itself.\n");
            MPI_Comm_disconnect(&dummy);
            printf("Server is disconnected.\n");
            MPI_Close_port(portName);
            }
            #pragma omp section
            {
            // Listening section
            while (1) {
                MPI_Comm interClient = MPI_COMM_NULL;
                MPI_Comm_accept(portName, MPI_INFO_NULL, 0,
                     MPI_COMM_WORLD, &interClient);
                if (terminateListening == true) {
                    break;
                }
                MPI_Status status;
                char clientName[MAX_DATA];
                MPI_Recv(clientName, MAX_DATA, MPI_CHAR,
                     MPI_ANY_SOURCE, MPI_ANY_TAG, interClient, &
                     status);
                printf("Client is connected with name: %s\n",
                     clientName);
                MPI_Comm_disconnect(&interClient);
                printf("Client is disconnected.\n");
            }
            }
        } // End of sections
    } // End of parallel section
    MPI_Finalize();
    return (0);
}
```

B.2 Listing: client.c

```
// C99
// Start program: mpirun -np 2 client
#include <mpi.h>
#include <stdio.h>

#define MAX_DATA 100
int main( int argc, char **argv )
{
    int isMpiInitCalledByClient;
    int myRank;
    char portName[MPI_MAX_PORT_NAME];
    char clientName[MAX_DATA];
    MPI_Initialized(&isMpiInitCalledByClient);
    if (!isMpiInitCalledByClient){
        MPI_Init(&argc, &argv);
    }
    MPI_Comm_rank(MPI_COMM_WORLD, &myRank);
    if (myRank == 0) {
        printf("Please provide server port:");
        scanf("%99s", &portName[0]);
        printf("Please provide client name:");
        scanf("%99s", &clientName[0]);
    }
    MPI_Comm server;
```

```
26      MPI_Comm_connect(portName, MPI_INFO_NULL, 0, MPI_COMM_WORLD, &
            server );
        if (myRank == 0) {
28          MPI_Send(clientName, MAX_DATA, MPI_CHAR, 0, 0, server);
        }
30      MPI_Comm_disconnect(&server);
        MPI_Finalize();
32      return (0);
}
```

BIBLIOGRAPHY

[1] A. C. Aitken. "On Bernoulli's numerical solution of algebraic equations." In: *Proceedings of the Royal Society of Edinburgh*. Vol. 46. 1926, pp. 289–305.

[2] S. Altmann. *Rotations, Quaternions, and Double Groups*. Dover books on mathematics. Dover Publications, 2005. ISBN: 9780486445182.

[3] P. Anagnostopoulos and P. Bearman. "Response characteristics of a vortex-excited cylinder at low reynolds numbers." In: *Journal of Fluids and Structures* 6.1 (1992), pp. 39–50. DOI: 10.1016/0889-9746(92)90054-7.

[4] E. V. Anjuri. "Comparison of Experimental results with CFD for NREL Phase VI Rotor with Tip Plate." In: *International Journal of Renewable Energy Research (IJRER)* 2.4 (2012), pp. 556–563.

[5] U. Ascher and L. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Society for Industrial and Applied Mathematics, 1998. ISBN: 9780898714128.

[6] S. Aubrun, S. Loyer, P. Hancock, and P. Hayden. "Wind turbine wake properties: Comparison between a non-rotating simplified wind turbine model and a rotating model." In: *Journal of Wind Engineering and Industrial Aerodynamics* 120 (2013), pp. 1–8. DOI: 10.1016/j.jweia.2013.06.007.

[7] E. Ayachour. "A fast implementation for GMRES method." In: *Journal of Computational and Applied Mathematics* 159.2 (2003), pp. 269–283. DOI: 10.1016/S0377-0427(03)00534-X.

[8] S. Badia, F. Nobile, and C. Vergara. “Robin–Robin preconditioned Krylov methods for fluid–structure interaction problems.” In: *Computer Methods in Applied Mechanics and Engineering* 198.33–36 (2009), pp. 2768–2784. DOI: 10.1016/j.cma.2009.04.004.

[9] S. Badia, F. Nobile, and C. Vergara. “Robin-Robin preconditioned Krylov methods for fluid-structure interaction problems.” In: *Computer Methods In Applied Mechanics And Engineering* 198.33-36 (2009), pp. 2768–2784. DOI: 10.1016/j.cma.2009.04.004.

[10] M. Bahmani and M. Akbari. “Response characteristics of a vortex-excited circular cylinder in laminar flow.” In: *Journal of Mechanical Science and Technology* 25.1 (2011), pp. 125–133. DOI: 10.1007/s12206-010-1021-0.

[11] J. Bastian, C. Clauß, S. Wolf, and P. Schneider. “Master for co-simulation using FMI.” In: *8th International Modelica Conference. Dresden*. 2011, pp. 115–120.

[12] Y. Bazilevs, M.-C. Hsu, I. Akkerman, S. Wright, K. Takizawa, B. Henicke, T. Spielman, and T. E. Tezduyar. “3D simulation of wind turbine rotors at full scale. Part I: Geometry modeling and aerodynamics.” In: *International Journal for Numerical Methods in Fluids* 65.1 (2011), pp. 207–235. DOI: 10.1002/fld.2400.

[13] Y. Bazilevs, M.-C. Hsu, J. Kiendl, R. Wüchner, and K.-U. Bletzinger. “3D simulation of wind turbine rotors at full scale. Part II: Fluid-structure interaction modeling with composite blades.” In: *International Journal for Numerical Methods in Fluids* 65.1 (2011), pp. 236–253. DOI: 10.1002/fld.2454.

[14] T. Belytschko and T. Hughes. *Computational methods for transient analysis*. Computational methods in mechanics. North-Holland, 1983. ISBN: 9780444864796.

[15] T. Belytschko, W. Liu, and B. Moran. *Nonlinear finite elements for continua and structures*. John Wiley & Sons, 2000. ISBN: 9780471987734.

[16] M. Benedikt, H. Stippel, and D. Watzenig. “An adaptive coupling methodology for fast time-domain distributed heterogeneous co-simulation.” In: *SAE Technical Paper* (2010). DOI: 10.4271/2010-01-0649.

[17] M. Benzi, R. Kouhia, and M. Tůma. “An assessment of some preconditioning techniques in shell problems.” In: *Communications in Numerical Methods in Engineering* 14.10 (1998), pp. 897–906. DOI: 10.1002/(SICI)1099-0887(1998100)14:10<897::AID-CNM196>3.0.CO;2-L.

[18] R. Bisplinghoff, H. Ashley, and R. Halfman. *Aeroelasticity*. Dover Books on Aeronautical Engineering Series. Dover Publications, 1996. ISBN: 9780486691893.

[19] A. de Boer, A. van Zuijlen, and H. Bijl. “Comparison of conservative and consistent approaches for the coupling of non-matching meshes.” In: *Computer Methods in Applied Mechanics and Engineering* 197.49–50 (2008), pp. 4284–4297. DOI: 10.1016/j.cma.2008.05.001.

[20] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN: 9780521833783.

[21] British Broadcasting Corporation. *Storm caused wind turbine fire*. Dec. 9, 2011. URL: http://www.bbc.co.uk/news/uk-16115139.

[22] P. Brown and Y. Saad. “Hybrid Krylov Methods for Nonlinear Systems of Equations.” In: *SIAM Journal on Scientific and Statistical Computing* 11.3 (1990), pp. 450–481. DOI: 10.1137/0911026.

[23] C. G. Broyden et al. “A class of methods for solving nonlinear simultaneous equations.” In: *Mathematics of Computation* 19.92 (1965), pp. 577–593.

[24] K. Burg, H. Haf, F. Wille, and A. Meister. *Partielle Differentialgleichungen und Funktionalanalytische Grundlagen: Höhere Mathematik Für Ingenieure, Naturwissenschaftler und Mathematiker*. Vieweg Studium. Vieweg Verlag, Friedr, & Sohn Verlagsgesellschaft mbH, 2010. ISBN: 9783834896841.

[25] M. Busch and B. Schweizer. “An explicit approach for controlling the macro-step size of co-simulation methods.” In: *Proc. of The 7th European Nonlinear Dynamics Conference, Rome, Italy*. 2011.

[26] M. Busch and B. Schweizer. “Coupled simulation of multibody and finite element systems: an efficient and robust semi-implicit coupling approach.” In: *Archive of Applied Mechanics* 82.6 (2012), pp. 723–741. DOI: 10.1007/s00419-011-0586-0.

[27] M. Busch and B. Schweizer. “Explicit and Implicit Solver Coupling: Stability Analysis Based on an Eight-Parameter Test Model.” In: *Proceedings in Applied Mathematics and Mechanics* 10.1 (2010), pp. 61–62. DOI: 10.1002/pamm.201010023.

[28] P. Causin, J.-F. Gerbeau, and F. Nobile. “Added-mass effect in the design of partitioned algorithms for fluid–structure problems.” In: *Computer methods in applied mechanics and engineering* 194.42 (2005), pp. 4506–4527. DOI: 10.1016/j.cma.2004.12.005.

[29] P. Chakraborty, S. Balachandar, and R. J. Adrian. “On the relationships between local vortex identification schemes.” In: *Journal of Fluid Mechanics* 535 (2005), pp. 189–214. DOI: 10.1017/S0022112005004726.

[30] A. Chorin. “Numerical solution of the Navier-Stokes equations.” In: *Mathematics of Computation* 22.104 (1968), pp. 745–762. DOI: 10.1090/S0025-5718-1968-0242392-2.

[31] J. Chung and G. M. Hulbert. “A Time Integration Algorithm for Structural Dynamics With Improved Numerical Dissipation: The Generalized-α Method.” In: 60.2 (1993), pp. 371–375. DOI: 10.1115/1.2900803.

[32] C. van Dam, D. D. Chao, and D. E. Berg. *CFD analysis of rotating two-bladed flatback wind turbine rotor*. Sandia National Laboratories, 2008.

[33] I. Daniel and O. Ishai. *Engineering Mechanics of Composites Materials*. Engineering mechanics of composite materials. Oxford University Press, Incorporated, 2006. ISBN: 9780195150971.

[34] J. Degroote. “Development of algorithms for the partitioned simulation of strongly coupled fluid-structure interaction problems.” PhD thesis. Belgium: Ghent University, 2010. ISBN: 9789085783442.

[35] W. G. Dettmer and D. Perić. “A new staggered scheme for fluid–structure interaction.” In: *International Journal for Numerical Methods in Engineering* 93.1 (2013), pp. 1–22. DOI: 10.1002/nme.4370.

[36] P. Deuflhard. *Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms*. Springer Series in Computational Mathematics. Springer, 2011. ISBN: 9783642238987.

[37] J. Donea, A. Huerta, J.-P. Ponthot, and A. Rodríguez-Ferran. “Arbitrary Lagrangian–Eulerian Methods.” In: *Encyclopedia of Computational Mechanics*. John Wiley & Sons, Ltd, 2004. ISBN: 9780470091357. DOI: 10.1002/0470091355.ecm009.

[38] M. R. Dörfel. “Fluid-structure interaction: a differential-algebraic approach and acceleration techniques for strong coupling.” PhD thesis. Germany: Technische Universität München, 2011.

[39] M. Dörfel and B. Simeon. “Fluid-Structure Interaction: Acceleration of Strong Coupling by Preconditioning of the Fixed-Point Iteration.” English. In: *Numerical Mathematics and Advanced Applications 2011*. Ed. by A. Cangiani, R. L. Davidchack, E. Georgoulis, A. N. Gorban, J. Levesley, and M. V. Tretyakov. Springer Berlin Heidelberg, 2013, pp. 741–749. ISBN: 978-3-642-33133-6. DOI: 10.1007/978-3-642-33134-3_78.

[40] *EMPIRE*. Aug. 2014. URL: http://empire-multiphysics.com.

[41] G. Falkovich. *Fluid Mechanics: A Short Course for Physicists*. Cambridge University Press, 2011. ISBN: 9781139497510.

[42] C. Farhat and M. Lesoinne. “Two efficient staggered algorithms for the serial and parallel solution of three-dimensional nonlinear transient aeroelastic problems.” In: *Computer Methods in Applied Mechanics and Engineering* 182.3–4 (2000), pp. 499–515. DOI: 10.1016/S0045-7825(99)00206-6.

[43] C. Farhat, M. Lesoinne, and P. L. Tallec. “Load and motion transfer algorithms for fluid/structure interaction problems with non-matching discrete interfaces: Momentum and energy conservation, optimal discretization and application to aeroelasticity.” In: *Computer Methods in Applied Mechanics and Engineering* 157.1–2 (1998), pp. 95–114. DOI: 10.1016/S0045-7825(97)00216-8.

[44] C. Farhat, M. Lesoinne, and N. Maman. “Mixed explicit/implicit time integration of coupled aeroelastic problems: Three-field formulation, geometric conservation and distributed solution.” In: *International Journal for Numerical Methods in Fluids* 21.10 (1995), pp. 807–835. DOI: 10.1002/fld.1650211004.

[45] C. Farhat, K. G. van der Zee, and P. Geuzaine. “Provably second-order time-accurate loosely-coupled solution algorithms for transient nonlinear computational aeroelasticity.” In: *Computer Methods in Applied Mechanics and Engineering* 195.17–18 (2006). Fluid-Structure Interaction, pp. 1973–2001. DOI: 10.1016/j.cma.2004.11.031.

[46] P. Farrell and J. Maddison. “Conservative interpolation between volume meshes by local Galerkin projection.” In: *Computer Methods in Applied Mechanics and Engineering* 200.1–4 (2011), pp. 89–100. DOI: 10.1016/j.cma.2010.07.015.

[47] C. A. Felippa and K. C. Park. “Computational Aspects of Time Integration Procedures in Structural Dynamics—Part 1: Implementation.” In: 45.3 (1978), pp. 595–602. DOI: 10.1115/1.3424368.

[48] C. Felippa and K. Park. “Staggered transient analysis procedures for coupled mechanical systems: Formulation.” In: *Computer Methods in Applied Mechanics and Engineering* 24.1 (1980), pp. 61–111. DOI: 10.1016/0045-7825(80)90040-7.

[49] C. Felippa and K. Park. “Synthesis tools for structural dynamics and partitioned analysis of coupled systems.” In: *NATO advanced research workshop*. 2004, pp. 50–111.

[50] C. A. Felippa. “Procedures for computer analysis of large nonlinear structural systems.” In: *Large Engineering Systems* (1976), pp. 60–101.

[51] C. A. Felippa, K. Park, and C. Farhat. “Partitioned analysis of coupled mechanical systems.” In: *Computer methods in applied mechanics and engineering* 190.24 (2001), pp. 3247–3270. DOI: 10.1016/S0045-7825(00)00391-1.

[52] U. Fey, M. König, and H. Eckelmann. “A new Strouhal–Reynolds-number relationship for the circular cylinder in the range 47<Re<2e5.” In: *Physics of Fluids* 10.7 (1998), pp. 1547–1549. DOI: 10.1063/1.869675.

[53] *FMI standard*. Aug. 2014. URL: http://www.fmi-standard.org.

[54] T. Frey, M. Bossert, and N. Fliege. *Signal- und Systemtheorie*. Vieweg + Teubner Studium. Vieweg Verlag, Friedr, & Sohn Verlagsgesellschaft mbH, 2008. ISBN: 9783834892928.

[55] B. Friedland. *Control system design: an introduction to state-space methods*. McGraw-Hill series in electrical engineering: Control theory. McGraw-Hill, 1986. ISBN: 9780070224414.

[56] M. Friedrich. “Parallel Co-Simulation for Mechatronic Systems.” Dissertation. Germany: Technische Universität München, 2011.

[57] R. Gasch and J. Twele. *Wind Power Plants: Fundamentals, Design, Construction and Operation*. Electrical Engineering. Springer, 2011. ISBN: 9783642229381.

[58] A. Gasmi, M. Sprague, J. Jonkman, and W. Jones. *Numerical Stability and Accuracy of Temporally Coupled Multi-Physics Modules in Wind-Turbine CAE Tools*. Tech. rep. National Renewable Energy Laboratory Colorado, USA, 2013.

[59] M. Geimer, T. Krüger, and P. Linsel. “Co-Simulation, gekoppelte Simulation oder Simulationskopplung? Ein Versuch der Begriffsvereinheitlichung.” In: *O+P Zeitschrift für Fluidtechnik-Aktorik, Steuerelektronik und Sensorik* 50 (2006), pp. 572–576.

[60] M. Gerouache. “Etude numérique de l’instabilité de bénard-karman derrière un cylindre fix ou en mouvement pérodique. dynamique de l’écoulement et advection chaotique.” PhD thesis. France: Ecole Polytechnique de l’Université de Nantes, 2000.

[61] P. Giguere and M. S. Selig. *Design of a tapered and twisted blade for the NREL combined experiment rotor*. Tech. rep. National Renewable Energy Laboratory Colorado, USA, 1999.

[62] D. Goldberg. “What Every Computer Scientist Should Know About Floating-point Arithmetic.” In: *ACM Computing Surveys* 23.1 (1991), pp. 5–48. DOI: 10.1145/103162.103163.

[63] D. Goldberg. “What Every Computer Scientist Should Know About Floating-point Arithmetic.” In: *ACM Comput. Surv.* 23.1 (1991), pp. 5–48. DOI: 10.1145/103162.103163.

[64] A. Gravouil and A. Combescure. “Multi-time-step explicit–implicit method for non-linear structural dynamics.” In: *International Journal for Numerical Methods in Engineering* 50.1 (2001), pp. 199–225. DOI: 10.1002/1097-0207(20010110)50:1<199::AID-NME132>3.0.CO;2-A.

[65] J. Gribbin. *Q is for Quantum: Particle Physics from A-Z*. Universities Press (India) Pvt. Limited, 1998. ISBN: 9788173712432.

[66] W. Gropp, E. Lusk, and R. Thakur. *Using MPI-2: Advanced Features of the Message-passing Interface*. Scientific and engineering computation Bd. 2. Globe Pequot Press, 1999. ISBN: 9780762728206.

[67] M. H. Gutknecht and S. Röllin. “The Chebyshev iteration revisited.” In: *Parallel Computing* 28.2 (2002), pp. 263–283. DOI: 10.1016/S0167-8191(01)00139-9.

[68] M. Hammache and M. Gharib. “An experimental study of the parallel and oblique vortex shedding from circular cylinders.” In: *Journal of Fluid Mechanics* 232 (1991), pp. 567–590. DOI: 10.1017/S0022112091003804.

[69] M. M. Hand, D. Simms, L. Fingersh, D. Jager, J. Cotrell, S. Schreck, and S. Larwood. *Unsteady Aerodynamics Experiment Phase VI: Wind Tunnel Test Configurations and Available Data Campaigns*. Tech. rep. National Renewable Energy Laboratory Colorado, USA, 2001.

[70] M. Hanke-Bourgeois. *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens*. Vieweg+Teubner Verlag, 2008. ISBN: 9783834807083.

[71] R. D. Henderson. "Details of the drag curve near the onset of vortex shedding." In: *Physics of Fluids* 7.9 (1995), pp. 2102–2104. DOI: 10.1063/1.868459.

[72] P. Henrici. *Elements of numerical analysis*. Wiley international edition. Wiley, 1964.

[73] P. Henrici. *Error propagation for difference methods*. SIAM series in applied mathematics. Robert E. Krieger Pub. Co.(Huntington, NY), 1977. ISBN: 9780882754482.

[74] M. Henrik. "Analysis and Optimization for Fluid-Structure Interaction Problems." PhD thesis. Denmark: Aalborg University, 2002.

[75] H. M. Hilber, T. J. R. Hughes, and R. L. Taylor. "Improved numerical dissipation for time integration algorithms in structural dynamics." In: *Earthquake Engineering & Structural Dynamics* 5.3 (1977), pp. 283–292. DOI: 10.1002/eqe.4290050306.

[76] M.-C. Hsu, I. Akkerman, and Y. Bazilevs. "Wind turbine aerodynamics using ALE-VMS: validation and the role of weakly enforced boundary conditions." English. In: *Computational Mechanics* 50.4 (2012), pp. 499–511. DOI: 10.1007/s00466-012-0686-x.

[77] M.-C. Hsu and Y. Bazilevs. "Fluid—structure interaction modeling of wind turbines: simulating the full machine." In: *Computational Mechanics* 50.6 (2012), pp. 821–833. DOI: 10.1007/s00466-012-0772-0.

[78] J. C. Hunt, A. Wray, and P. Moin. "Eddies, streams, and convergence zones in turbulent flows." In: *Studying Turbulence Using Numerical Simulation Databases, 2*. Vol. 1. 1988, pp. 193–208.

[79] *Intel MPI Library for Linux* OS*. Version 4.1. Intel. 2014.

[80] B. M. Irons and R. C. Tuck. "A version of the Aitken accelerator for computer iteration." In: *International Journal for Numerical Methods in Engineering* 1.3 (1969), pp. 275–277. DOI: 10.1002/nme.1620010306.

[81] R. Issa. "Solution of the implicitly discretised fluid flow equations by operator-splitting." In: *Journal of Computational Physics* 62.1 (1986), pp. 40–65. DOI: 10.1016/0021-9991(86)90099-9.

[82] H. Jasak and Z. Tukovic. "Automatic mesh motion for the unstructured finite volume method." In: *Transactions of FAMENA* 30.2 (2006), pp. 1–20.

[83] M. M. Joosten, W. G. Dettmer, and D. Perić. "Analysis of the block Gauss–Seidel solution procedure for a strongly coupled model problem with reference to fluid–structure interaction." In: *International Journal for Numerical Methods in Engineering* 78.7 (2009), pp. 757–778. DOI: 10.1002/nme.2503.

[84] M. M. Joosten, W. G. Dettmer, and D. Perić. "On the temporal stability and accuracy of coupled problems with reference to fluid–structure interaction." In: *International Journal for Numerical Methods in Fluids* 64.10-12 (2010), pp. 1363–1378. DOI: 10.1002/fld.2333.

[85] K. Kamran, R. Rossi, E. Oñate, and S. Idelsohn. "A compressible Lagrangian framework for the simulation of the underwater implosion of large air bubbles." In: *Computer Methods in Applied Mechanics and Engineering* 255 (2013), pp. 210–225. DOI: 10.1016/j.cma.2012.11.018.

[86] C. Kassiotis. *Which strategy to move the mesh in the Computational Fluid Dynamic code OpenFOAM*. Tech. rep. 2008.

[87] T. Kerkhoven and Y. Saad. "On acceleration methods for coupled nonlinear elliptic systems." English. In: *Numerische Mathematik* 60.1 (1991), pp. 525–548. DOI: 10.1007/BF01385735.

[88] D. Knoll and D. Keyes. “Jacobian-free Newton–Krylov methods: a survey of approaches and applications.” In: *Journal of Computational Physics* 193.2 (2004), pp. 357–397. DOI: 10.1016/j.jcp.2003.08.010.

[89] U. Küttler and W. A. Wall. “Fixed-point fluid–structure interaction solvers with dynamic relaxation.” English. In: *Computational Mechanics* 43.1 (2008), pp. 61–72. DOI: 10.1007/s00466-008-0255-5.

[90] R. Latham, R. B. Ross, and R. Thakur. “Can MPI Be Used for Persistent Parallel Services?” In: *PVM/MPI*. Ed. by B. Mohr, J. L. Träff, J. Worringen, and J. Dongarra. Vol. 4192. Lecture Notes in Computer Science. Springer, 2006, pp. 275–284. ISBN: 3-540-39110-X.

[91] C. Lerch. “Towards a Turbulent Fluid-Structure-Signal Co-Simulation: Validate with the NREL 10m Wind Turbine Testing in NASA Ames Wind Tunnel.” 2013.

[92] M. Lesoinne and C. Farhat. “Geometric conservation laws for flow problems with moving boundaries and deformable meshes, and their impact on aeroelastic computations.” In: *Computer Methods in Applied Mechanics and Engineering* 134.1–2 (1996), pp. 71–90. DOI: 10.1016/0045-7825(96)01028-6.

[93] Y. Li, K.-J. Paik, T. Xing, and P. M. Carrica. “Dynamic overset CFD simulations of wind turbine aerodynamics.” In: *Renewable Energy* 37.1 (2012), pp. 285–298. DOI: 10.1016/j.renene.2011.06.029.

[94] Z. Li, A. Combescure, and F. Leboeuf. “Coupling of finite volume and finite element subdomains using different time integrators.” In: *International Journal for Numerical Methods in Fluids* 72.12 (2013), pp. 1286–1306. DOI: 10.1002/fld.3786.

[95] C. Lindenburg. “Investigation into rotor blade aerodynamics.” In: *Netherlands Society for Energy and the Environment, Paper ECN-C-03-025* (2003).

[96] B. Liptak. *Instrument Engineers’ Handbook,(Volume 2) Third Edition: Process Control*. Instrument Engineers’ Handbook. Taylor & Francis, 1995. ISBN: 9780801982422.

[97] A. J. Macleod. “Acceleration of vector sequences by multi-dimensional Δ^2 methods.” In: *Communications in Applied Numerical Methods* 2.4 (1986), pp. 385–392. DOI: 10.1002/cnm.1630020409.

[98] N. Mahjoubi, A. Gravouil, and A. Combescure. “Coupling subdomains with heterogeneous time integrators and incompatible time steps.” English. In: *Computational Mechanics* 44.6 (2009), pp. 825–843. DOI: 10.1007/s00466-009-0413-4.

[99] D. Manolakis and V. Ingle. *Applied Digital Signal Processing: Theory and Practice*. Cambridge University Press, 2011. ISBN: 9781139495738.

[100] S. McTavish, D. Feszty, and F. Nitzsche. “Aeroelastic simulations of the NREL Phase VI wind turbine using a discrete vortex method coupled with a nonlinear beam model.” In: *Proceedings of the European Wind Energy Conference*. 2009.

[101] F. R. Menter, R. Langtry, and S. Völker. “Transition Modelling for General Purpose CFD Codes.” In: *Flow, Turbulence and Combustion* 77.1 (2006), pp. 277–303. DOI: 10.1007/s10494-006-9047-1.

[102] F. R. Menter. “Two-equation eddy-viscosity turbulence models for engineering applications.” In: *AIAA journal* 32.8 (1994), pp. 1598–1605.

[103] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard Version 3.0*. 2012.

[104] U. Miekkala and O. Nevanlinna. “Convergence of Dynamic Iteration Methods for Initial Value Problems.” In: *SIAM Journal on Scientific and Statistical Computing* 8.4 (1987), pp. 459–482. DOI: 10.1137/0908046.

[105] J.-O. Mo, A. Choudhry, M. Arjomandi, and Y.-H. Lee. “Large eddy simulation of the wind turbine wake characteristics in the numerical wind tunnel model.” In: *Journal of Wind Engineering and Industrial Aerodynamics* 112 (2013), pp. 11–24. DOI: 10.1016/j.jweia.2012.09.002.

[106] J.-O. Mo and Y.-H. Lee. “CFD Investigation on the aerodynamic characteristics of a small-sized wind turbine of NREL PHASE VI operating with a stall-regulated method.” In: *Journal of Mechanical Science and Technology* 26.1 (2012), pp. 81–92. DOI: 10.1007/s12206-011-1014-7.

[107] P. Moin. *Fundamentals of Engineering Numerical Analysis*. Cambridge University Press, 2010. ISBN: 9780521711234.

[108] *Navier Stokes Equations*. Jan. 2014. URL: http://www.cfd-online.com/Wiki/Navier-Stokes_equations.

[109] G. D. Nayer, A. Kalmbach, M. Breuer, S. Sicklinger, and R. Wüchner. “Flow past a cylinder with a flexible splitter plate: A complementary experimental–numerical investigation and a new FSI test case (FSI-PfS-1a).” In: *Computers & Fluids* 99 (2014), pp. 18–43. DOI: 10.1016/j.compfluid.2014.04.020.

[110] *NREL 10-m Wind Turbine Testing in NASA Ames*. Jan. 2014. URL: https://wind.nrel.gov/amestest.

[111] *OpenFOAM*. Aug. 2014. URL: http://openfoam.org.

[112] K. C. Park and C. A. Felippa. “Computational Aspects of Time Integration Procedures in Structural Dynamics—Part 2: Error Propagation.” In: 45.3 (1978), pp. 603–611. DOI: 10.1115/1.3424369.

[113] T. Peacock and E. Bradley. “Going with (or Against) the Flow.” In: *Science* 320.5881 (2008), pp. 1302–1303. DOI: 10.1126/science.1153479.

[114] S. Piperno and C. Farhat. “Partitioned procedures for the transient solution of coupled aeroelastic problems – Part II: energy transfer analysis and three-dimensional applications.” In: *Computer Methods in Applied Mechanics and Engineering* 190.24–25 (2001). Advances in Computational Methods for Fluid-Structure Interaction, pp. 3147–3170. DOI: 10.1016/S0045-7825(00)00386-8.

[115] S. Piperno, C. Farhat, and B. Larrouturou. “Partitioned procedures for the transient solution of coupled aroelastic problems Part I: Model problem, theory and two-dimensional application.” In: *Computer Methods in Applied Mechanics and*

Engineering 124.1–2 (1995), pp. 79–112. DOI: 10.1016/0045-7825(95)92707-9.

[116] A. Placzek, J.-F. Sigrist, and A. Hamdouni. "Numerical simulation of an oscillating cylinder in a cross-flow at low Reynolds number: Forced and free oscillations." In: *Computers & Fluids* 38.1 (2009), pp. 80–100. DOI: 10.1016/j.compfluid.2008.01.007.

[117] M. A. Potsdam and D. J. Mavriplis. "Unstructured mesh CFD aerodynamic analysis of the NREL Phase VI rotor." In: *AIAA paper* 1221 (2009), p. 2009.

[118] A. Preumont. *Vibration Control of Active Structures: An Introduction*. Solid Mechanics and Its Applications. Springer, 2011. ISBN: 9789400720336.

[119] M. A. Puso and T. A. Laursen. "A mortar segment-to-segment frictional contact method for large deformations." In: *Computer Methods in Applied Mechanics and Engineering* 193.45–47 (2004), pp. 4891–4913. DOI: 10.1016/j.cma.2004.06.001.

[120] A. Quarteroni and A. Valli. *Domain Decomposition Methods for Partial Differential Equations*. Numerical mathematics and scientific computation. Clarendon Press, 1999. ISBN: 9780198501787.

[121] J. Reid, I. of Mathematics, and I. Applications. *Large sparse sets of linear equations: proceedings of the Oxford conference of the Institute of Mathematics and Its Applications held in April, 1970*. Academic P., 1971. ISBN: 9780125861502.

[122] *Richardson extrapolation*. Mar. 2014. URL: http://de.wikipedia.org/wiki/Richardson-Extrapolation.

[123] L. F. Richardson and J. A. Gaunt. "The deferred approach to the limit. Part I. Single lattice. Part II. Interpenetrating lattices." In: *Philosophical Transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character* (1927), pp. 299–361. DOI: 10.1098/rsta.1927.00080.

[124] L. F. Richardson. “The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam.” In: *Royal Society of London Philosophical Transactions Series A* 210 (1911), pp. 307–357.

[125] A. Roshko. *On the Development of Turbulent Wakes from Vortex Streets. Report 1191*. Tech. rep. California Institute of Technology, 1954.

[126] P. Ryzhakov, R. Rossi, S. Idelsohn, and E. Oñate. “A monolithic Lagrangian approach for fluid–structure interaction problems.” English. In: *Computational Mechanics* 46.6 (2010), pp. 883–899. DOI: 10.1007/s00466-010-0522-0.

[127] Y. Saad. *Iterative Methods for Sparse Linear Systems: Second Edition*. Society for Industrial and Applied Mathematics, 2003. ISBN: 9780898715347.

[128] Y. Saad and M. H. Schultz. “GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems.” In: *SIAM J. Sci. Stat. Comput.* 7.3 (1986), pp. 856–869. DOI: 10.1137/0907058.

[129] J. Schlüter, X. Wu, E. vd Weide, S. Hahn, M. Herrmann, J. J. Alonso, and H. Pitsch. “A python approach to multi-code simulations: Chimps.” In: *Annual Research Briefs* (2005), pp. 97–110.

[130] S. Schulte. “Modulare und hierarchische Simulation gekoppelter Probleme.” PhD thesis. Germany: Technische Universität München, 1998.

[131] I. Schur. “On Power Series Which are Bounded in the Interior of the Unit Circle. I.” English. In: *I. Schur Methods in Operator Theory and Signal Processing*. Ed. by I. Gohberg. Vol. 18. Operator Theory: Advances and Applications. Birkhäuser Basel, 1986, pp. 31–59. ISBN: 978-3-0348-5484-9. DOI: 10.1007/978-3-0348-5483-2_3.

[132] H. Schwarz and N. Köckler. *Numerische Mathematik*. Lehrbuch Mathematik. Teubner B.G. GmbH, 2004. ISBN: 9783519429609.

[133] N. Sezer-Uzol and L. N. Long. “3-D time-accurate CFD simulations of wind turbine rotor flow fields.” In: *AIAA paper* 394 (2006), pp. 1–23.

[134] S. Sicklinger, V. Belsky, B. Engelmann, H. Elmqvist, H. Olsson, R. Wüchner, and K.-U. Bletzinger. “Interface Jacobian-based Co-Simulation.” In: *International Journal for Numerical Methods in Engineering* 98.6 (2014), pp. 418–444. DOI: 10.1002/nme.4637.

[135] S. Sicklinger. “Formulation and object-oriented implementation of a nonlinear node-to-surface mechanical contact algorithm.” Germany: Technische Universität München, 2010.

[136] D. A. Simms, M. M. Hand, L. J. Fingersh, and D. W. Jager. *Unsteady aerodynamics experiment phases II-IV test configurations and available data campaigns*. Tech. rep. 1999.

[137] D. A. Simms, S. Schreck, M. Hand, and L. Fingersh. *NREL unsteady aerodynamics experiment in the NASA-Ames wind tunnel: a comparison of predictions to measurements*. Tech. rep. National Renewable Energy Laboratory Colorado, USA, 2001.

[138] J. C. Simo and M. S. Rifai. “A class of mixed assumed strain methods and the method of incompatible modes.” In: *International Journal for Numerical Methods in Engineering* 29.8 (1990), pp. 1595–1638. DOI: 10.1002/nme.1620290802.

[139] *Simulink User Guide R2014a*. MathWorks, 2014.

[140] J. Sobieszczanski-Sobieski. “Sensitivity of complex, internally coupled systems.” In: *AIAA journal* 28.1 (1990), pp. 153–160.

[141] N. N. Sørensen, J. A. Michelsen, and S. Schreck. “Navier–Stokes predictions of the NREL phase VI rotor in the NASA Ames 80 ft x 120 ft wind tunnel.” In: *Wind Energy* 5.2–3 (2002), pp. 151–169. DOI: 10.1002/we.64.

[142] D. Sternel, M. Schäfer, M. Heck, and S. Yigit. “Efficiency and accuracy of fluid-structure interaction simulations using an implicit partitioned approach.” In: *Computational Mechanics* 43.1 (2008), pp. 103–113. DOI: 10.1007/s00466-008-0278-y.

[143] G. Strang. *Computational Science and Engineering*. Wellesley-Cambridge Press, 2007. ISBN: 9780961408817.

[144] G. Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 2003. ISBN: 9780961408893.

[145] *Taylor's Theorem*. Jan. 2014. URL: http://mathworld.wolfram.com/TaylorsTheorem.html.

[146] C. Tongchitpakdee, S. Benjanirat, and L. N. Sankar. "Numerical Simulation of the Aerodynamics of Horizontal Axis Wind Turbines under Yawed Flow Conditions." In: *Journal of Solar Energy Engineering* 127.4 (2005), pp. 464–474. DOI: 10.1115/1.2035705.

[147] S. Tripakis and D. Broman. *Bridging the Semantic Gap Between Heterogeneous Modeling Formalisms and FMI*. Tech. rep. UCB/EECS-2014-30. EECS Department, University of California, Berkeley, Apr. 2014.

[148] S. Turek and J. Hron. "Proposal for Numerical Benchmarking of Fluid-Structure Interaction between an Elastic Object and Laminar Incompressible Flow." In: *Fluid-Structure Interaction*. Ed. by H.-J. Bungartz and M. Schäfer. Vol. 53. Lecture Notes in Computational Science and Engineering. Springer Berlin Heidelberg, 2006, pp. 371–385. ISBN: 978-3-540-34595-4. DOI: 10.1007/3-540-34596-5_15.

[149] H. Unbehauen. *Regelungstechnik. 1. Klassische Verfahren zur Analyse und Synthese linearer kontinuierlicher Regelsysteme, Fuzzy-Regelsysteme*. Regelungstechnik. Vieweg, 2005. ISBN: 9783528213329.

[150] T. Van Dusen. *Unsteady Aerodynamics Experiment Phase VI: Wind Tunnel Test Configurations and Available Data Campaigns*. Tech. rep. Composite Engineering 277 Baker Avenue Concord, MA 01742-2115, USA, 2002.

[151] W. A. Wall1, A. Gerstenberger, P. Gamnitzer, C. Förster, and E. Ramm. "Large Deformation Fluid-Structure Interaction – Advances in ALE Methods and New Fixed Grid Approaches." In: *Fluid-Structure Interaction*. Ed. by H.-J. Bungartz and M. Schäfer. Vol. 53. Lecture Notes in Computational Science and Engineering. Springer Berlin Heidelberg, 2006, pp. 195–232. ISBN: 978-3-540-34595-4. DOI: 10.1007/3-540-34596-5_9.

[152] Q. Wang, H. Zhou, and D. Wan. “Numerical simulation of wind turbine blade-tower interaction.” In: *Journal of Marine Science and Application* 11.3 (2012), pp. 321–327. DOI: 10.1007/s11804-012-1139-9.

[153] T. Wang. “A brief review on wind turbine aerodynamics.” In: *Theoretical and Applied Mechanics Letters* 2.6, 062001 (2012). DOI: 10.1063/2.1206201.

[154] D. Werner. *Funktionalanalysis*. Springer-Lehrbuch. Springer, 2011. ISBN: 9783642210167.

[155] J. Wilkinson. *Rounding Errors in Algebraic Processes*. Dover books on advanced mathematics. Dover, 1994. ISBN: 9780486679990.

[156] C. Williamson and A. Roshko. “Vortex formation in the wake of an oscillating cylinder.” In: *Journal of Fluids and Structures* 2.4 (1988), pp. 355–381. DOI: 10.1016/S0889-9746(88)90058-8.

[157] E. L. Wilson. “The static condensation algorithm.” In: *International Journal for Numerical Methods in Engineering* 8.1 (1974), pp. 198–203. DOI: 10.1002/nme.1620080115.

[158] C. Woernle. *Mehrkörpersysteme*. Springer, 2011. ISBN: 9783642159824.

[159] P. Wriggers. *Nonlinear Finite Element Methods*. Vol. 4. Springer, 2008. ISBN: 9783540710004.

[160] M. M. Yelmule and E. A. VSJ. “CFD predictions of NREL Phase VI Rotor Experiments in NASA/AMES Wind tunnel.” In: *International Journal of Renewable Energy Research (IJRER)* 3.2 (2013), pp. 261–269.

[161] F. Zahle, N. N. Sørensen, and J. Johansen. “Wind turbine rotor-tower interaction using an incompressible overset grid method.” In: *Wind Energy* 12.6 (2009), pp. 594–619. DOI: 10.1002/we.327.

[162] F. Zhang. *The Schur Complement and Its Applications*. Numerical Methods and Algorithms. Springer, 2006. ISBN: 9780387242736.

[163] X. Zhou and K. K. Tamma. “Design, analysis, and synthesis of generalized single step single solve and optimal algorithms for structural dynamics.” In: *International Journal for Numerical Methods in Engineering* 59.5 (2004), pp. 597–668. DOI: 10.1002/nme.873.

[164] O. Zienkiewicz. “Coupled problems and their numerical solution.” In: *Numerical Methods in Coupled Systems*. Ed. by R. Lewis, P. Bettess, and E. Hinton. Wiley & Sons, Chichester, 1984, pp. 35–68.

[165] O. Zienkiewicz and K. Morgan. *Finite elements and approximation*. Dover books on engineering. Dover Publications, 2006. ISBN: 9780486453019.

[166] O. Zienkiewicz and R. Taylor. *The Finite Element Method for Solid and Structural Mechanics*. The element method set. Elsevier Butterworth-Heinemann, 2005. ISBN: 9780750664318.

[167] O. Zienkiewicz and R. Taylor. *The Finite Element Method for Solid and Structural Mechanics*. The element method set. Elsevier Butterworth-Heinemann, 2013. ISBN: 9780080951362.

[168] O. Zienkiewicz, R. Taylor, and P. Nithiarasu. *The Finite Element Method for Fluid Dynamics*. The element method set. Elsevier Butterworth-Heinemann, 2005. ISBN: 9780080455594.

[169] O. Zienkiewicz, R. Taylor, and P. Nithiarasu. *The Finite Element Method for Fluid Dynamics*. The element method set. Elsevier Butterworth-Heinemann, 2013. ISBN: 9780080951379.

[170] O. Zienkiewicz, R. Taylor, and J. Zhu. *The Finite Element Method: Its Basis and Fundamentals*. The element method set. Elsevier Butterworth-Heinemann, 2005. ISBN: 9780080472775.

[171] O. Zienkiewicz, R. Taylor, and J. Zhu. *The Finite Element Method: Its Basis and Fundamentals*. The element method set. Elsevier Butterworth-Heinemann, 2013. ISBN: 9780080951355.