Mag. rer. nat Dipl.-Ing. Simon Genser, BSc

Model-based Pre-Step Stabilization Method for Non-Iterative Co-Simulation

DOCTORAL THESIS
to achieve the university degree of
Doktor der technischen Wissenschaften
submitted to

Graz University of Technology

Supervisor

Univ.-Prof. Dipl.-Ing. Dr.techn. Daniel Watzenig

Institute of Automation and Control
Faculty of Electrical and Information Engineering

Graz, December 2020

STATUTORY DECLARATION

Abstract

As development processes should increase in efficiency, they need to become faster and cheaper, and therefore simulations are gaining in significance in this field. In todays industrial applications, different simulation software, and sometimes even hardware parts, have to be coupled accordingly, this special kind of simulation is called co-simulation. The Model-based Pre-Step Stabilization Method is a coupling method for the non-iterative co-simulation. The main goal of this thesis and the invented coupling method is to ensure a stable behaviour of co-simulations which are highly sensitive against coupling errors and are therefore especially challenging. The coupling method consists of three main steps: (1) approximating the monolithic result, by solving the so-called Error Differential Equation; (2) extrapolating the monolithic result one communication step into the future; (3) optimizing the input in such a way, that the co-simulated result fits the extrapolated monolithic one. All main steps are based on the so-called Interface Jacobian subsystem description, a surrogate model description based on the in- and outputs of the subsystems. These surrogate models can be provided by the subsystem themselves, or they can be approximated by the coupling method, therefore two different system identification methods can be utilized, the Recursive Least Squares and the Multivariable Output Error Stace Space approach. An optimal set of pre-defined parameters of the coupling method is determined by a sensitivity analysis, additionally the accuracy and stability are investigated at the classical co-simulation benchmark example, the dual mass oscillator. There it will be demonstrated, that the stability region according numerical stability is clearly enlarged by a factor between $[3.2, 8.2]$, compared to state of the art coupling methods. Also the accuracy of the co-simulation utilizing the Model-based Pre-Step Stabilization Method is improved, as larger communication step sizes lead to co-simulations with the same or an improved accuracy. Additionally the exponential and bounded-input bounded-output stability of the presented coupling method is derived. As evaluation the co-simulation of a helicopter and its control has been chosen, there it is shown, that the Model-based Pre-Step Stabilization Method improves the co-simulation clearly according stability and accuracy.

Keywords: co-simulation; Interface Jacobians; stabilization method; system identification

Kurzfassung

Die modellbasierte Pre-Step Stabilisierungsmethode ist eine Kopplungsmethode für die nicht-iterative Co-Simulation. Das Hauptziel der Kopplungsmethode ist die Stabilisierung herausfordernder Co-Simulationen. Die Kopplungsmethode besteht aus drei Hauptschritten: (1) Annäherung des monolithischen Ergebnisses durch Lösung der sogenannten Fehlerdifferentialgleichung; (2) Extrapolation des monolithischen Ergebnisses um einen Kommunikationsschritt in die Zukunft; (3) Optimierung des Inputs in der Weise, dass das co-simulierte Ergebnis mit dem extrapolierten monolithischen übereinstimmt. Alle Hauptschritte basieren auf der sogenannten Interface Jacobian Subsystembeschreibung, einer Ersatzmodellbeschreibung, die nur auf den Ein- und Ausgängen der Subsysteme basiert. Diese Ersatzmodelle können vom Subsystem selbst bereitgestellt werden, oder sie können durch die Kopplungsmethode approximiert werden. Dafür werden zwei verschiedene Systemidentifikationsmethoden verwendet, die rekursive Methode der kleinsten Quadrate und der Multivariable Output Error Stace Space Ansatz. Die optimalen Parameter der Kopplungsmethode werden durch eine Sensitivitätsanalyse bestimmt. Zusätzlich wird die Genauigkeit und Stabilität am klassischen Co-Simulations Benchmark-Beispiel, dem Zweimassen-Oszillator, untersucht. Dort wird gezeigt, dass der Stabilitätsbereich entsprechend der numerischen Stabilität im Vergleich mit anderen Kopplungsmethoden deutlich vergrößert ist. Zusätzlich wird auch die Genauigkeit der Co-Simulation unter Verwendung der modellbasierten Pre-Step-Stabilisierungsmethode verbessert. Des Weiteren wird die exponentielle und eingangs- und ausgangsbegrenzte Stabilität der vorgestellten Kopplungsmethode gezeigt. Zur Auswertung wird die Co-Simulation eines Hubschraubers und seiner Steuerung gewählt, es wird gezeigt, dass die modellbasierte Pre-Step-Stabilisierungsmethode die Co-Simulation hinsichtlich Stabilität und Genauigkeit deutlich verbessert.

Schlagwörter: Co-Simulation; Interface Jacobians; Stabilisierungsmethode; System Identifikation

Acknowledgments

This thesis was written at Virtual Vehicle Research GmbH in Graz, Austria. I would like to acknowledge the financial support of the COMET Competence Centers for Excellent Technologies from the Austrian Federal Ministry for Climate Action, the Austrian Federal Ministry for Digital and Economic Affairs, the Province of Styria (Dept. 12) and the Styrian Business Promotion Agency (SFG). The Austrian Research Promotion Agency (FFG) has been authorised for the programme management. Furthermore, I would like to express my thanks to the Institute of Automation and Control at Graz University of Technology.

Further, I would like to thank my colleagues at the Virtual Vehicle Research Center GmbH for the inspiring working environment. My special thanks go to Dr. Martin Benedikt, Dr. Georg Stettinger, Franz Holzinger and Dr. Johannes Rumetshofer for their support and countless discussions which enabled me to finish this thesis. Many thanks to my supervisors Dr. Daniel Watzenig and Dr. Martin Arnold for supporting me since the first day.

My love Petra and my son Finn deserve the greatest thanks as they supported and motivated me through the whole process of this thesis. Especially as both of you, cleared my head countless times and always brought a smile to my face.

Graz, December 2020

Simon Genser

Contents

1

Introduction

A key to master the challenge of faster and more efficient virtual development processes is the intelligent coupling of different simulation software. Especially of importance is the quality of the results, as high accurate results are required for a virtual validation. Therefore research and development in the field of co-simulation and especially in the part of the coupling strategies is mandatory.

1.1 Outlook

The main contribution of this thesis is to introduce a new coupling method for co-simulation, the so-called Model-based Pre-Step Stabilization Method. The first chapter will work as an introduction explaining co-simulation, what are the challenges and what are the different applications, especially focusing on the different coupling methods in co-simulation. Additionally the problem statement and the significance of this thesis will be given. The following chapter will deal with the invented method itself, the basic idea and its variations and options will be derived and discussed in full detail. Among other things, how to access the mandatory subsystem information and the workflow of the coupling method, will be stated and discussed in Chapter 2. The analysis of the method will be the topic of Chapter 3, there a well-known co-simulation benchmark example, the dual mass oscillator is used to perform a design of experiment to analyze the Model-based Pre-Step Stabilization Method according to its sensitivity against various parameters and different options. Additionally a stability analysis and accuracy analysis, according to the communication step size and the stiffness of the dual mass oscillator, will be carried out. In Chapter 4 the Model-based Pre-Step Stabilization Method is evaluated at the co-simulation of a helicopter and its control. The last chapter states a summary of the coupling method and its evaluation and concludes the thesis.

1.2 Background and Motivation

In today's industry there is the need to improve the development processes to keep the amount of time and money for the developing to a minimum. Therefore the concept of virtual development is getting more and more attention, this means that simulation gains in significance. Especially the coupling of different simulation tools is a crucial part, because typically an industrial simulation consists of more than one simulation software. A monolithic approach, which means modeling and simulating everything in one simulation tool is often not possible. For example, consider that the goal is to simulate a vehicle to determine the consumption according to a new control strategy of a hybrid engine, therefore at least the combustion engine, the electric motor, the control unit, the wheels, the chassis, the driver and the drivetrain have to be modeled. For all these parts exist tailored simulation software to model it in the appropriate degree of detail and typically there are different departments or even different companies involved. For all these reasons the coupling of different simulation software is required where it is not possible, or not wanted, to exchange the programming code or the models directly. Therefore the so-called weak coupling approach, see for more details [1], is needed because the different simulation software is transformed into so-called subsystems where the model and its solver are included, see Figure 1.1. The mandatory coupling between these subsystems is carried out by the exchange of data at certain time points, the so-called communication points, exclusively. This simulation of coupled simulation software is called co-simulation. It should be denoted that it is also possible to couple simulation software to real hardware parts, e.g. to an engine test bed or an electronic control unit, this is called real time co-simulation.

With the increasing relevance of co-simulation more and more industrial companies get attracted and therefore the need of a standardization is rising. The Functional Mock-Up Interface (FMI), see [2], is a standard for the subsystems of co-simulation, which is widely supported by a high number of simulation software. The utilization of a standard framework for the subsystems, so-called Functional Mock-Up Units (FMU), enables a much easier integration and therefore coupling between those subsystems. Additionally the look inside a subsystem is not required for the coupling and so the intellectual property rights of a certain subsystem can be protected by transforming the subsystems into an FMU. This is for sure an important factor that the FMI standard increased in significance because, so the companies can be sure that non of their top-secret models got unveiled. An FMU only needs signals as an input and provides the appropriate output signals, i.e. an FMU can be seen as a black box. In the upcoming version of FMI the output of a subsystem may contain the solver steps of the subsystem and not the value at the communication point exclusively, these quantities are called intermediate values.

Agile development processes, especially Continuous Integration and Continuous Development (CI/CD), are an additional application field of co-simulation. There the testing of different software components and their interaction is important,

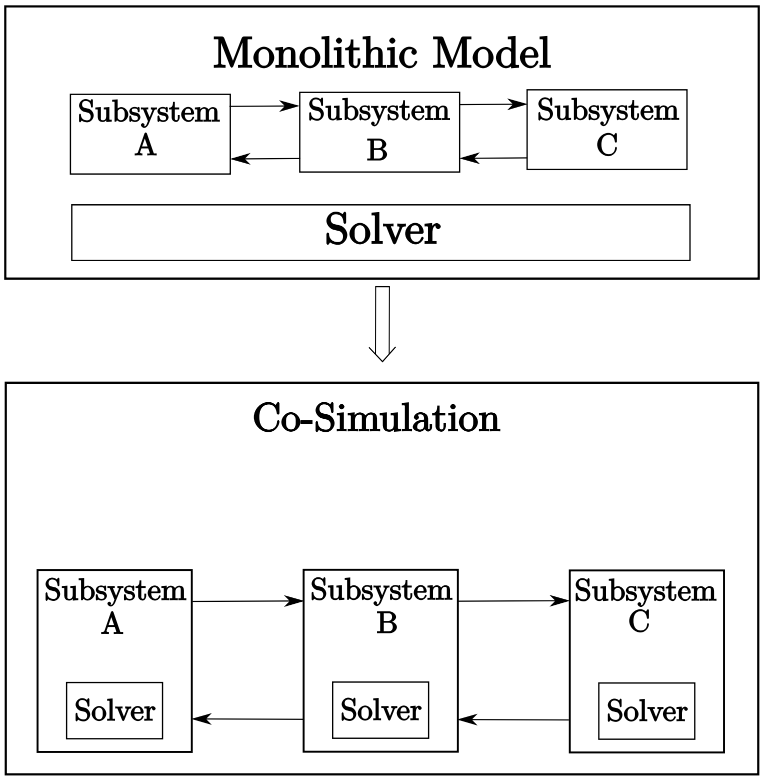


Figure 1.1: *The concept of co-simulation*

therefore the integration of the developed software components as subsystem for the co-simulation have to be automated. This means that the co-simulation is often used for this whole agile development process, because it ensures the correct functionality of the coupled and interacted software tools. In the last years cloud computing is getting more and more attention also for simulations. The basic idea is that parts of the simulation, pieces of software, are stored in the cloud and it is possible to run a simulation based on these parts from everywhere around the globe. Therefore the coupling of these different software parts have to be realized and this can be seen as a co-simulation, where the different subsystems are stored in a cloud. Also a mix of cloud and local co-simulation, some subsystems are stored in the cloud and others are locally available on a hard drive, might be of interest in the future. Co-simulation enables to combine and connect different simulation tools and software, which were otherwise not feasible to couple.

From a mathematical viewpoint co-simulation is an interesting opportunity to combine different numerical solvers. In a co-simulation, it is possible to adjust every subsystem with its perfectly tailored numerical solver. This means that systems with high dynamics can use sophisticated solvers and other subsystems with lower dynamics can use classical solvers, e.g. Runge-Kutta methods. Not only the type of the numerical solver also the step size of the solver can vary from subsystem to subsystem. One can be a classical fixed step solver and the other is a variable step size one, therefore it is possible to choose the solver accordingly to the needs of the subsystem. For cases where the dynamics in a subsystem are similar, the choice of the solver may not be problematic, but for the case that the dynamics vary significantly this can be challenging. Such systems are called stiff systems and typically the solver and its step size have to be chosen after the fast dynamic and therefore small step sizes and a high computational effort is required. For co-simulation stiff subsystems are additionally challenging, because these subsystems are sensitive according extrapolation and the induced coupling error, this can lead to weak results or even an instable co-simulation. Additional to the previously discussed case, also the coupling of subsystems which are significantly different in terms of dynamics may lead to challenges for the co-simulation, and therefore both cases are referred as stiff co-simulations.

1.3 Problem Statement and Significance

The contribution of this work is to derive, analyze and evaluate a stabilization method for co-simulation, namely the Model-based Pre-Step Stabilization Method. This coupling method is especially of importance for stiff co-simulations, as those co-simulations are especially sensitive against coupling errors and therefore it is particularly challenging to ensure a stable behaviour of the whole simulation. The state of the art to handle stiff problems is to decrease the communication step size accordingly to the fastest dynamic, this produces a high amount of communication and computational effort. By utilizing the introduced coupling method the need to decrease the communication step size significantly can be avoided, the stability region of the co-simulation is clearly enlarged, i.e. the utilization of large communication step sizes is possible. Additionally to the stability, also the accuracy of the results is increased and therefore the reliability of the co-simulation is enhanced.

1.4 Co-Simulation Coupling Methods – An Overview

The fundamental challenge in co-simulation is to cope with the coupling error, due to the extrapolations of the inputs of the subsystems. In the co-simulations which are considered in this thesis, the subsystems are always coupled in a closed loop manner, and therefore the extrapolation of at least one input is unavoidable. Different co-simulation approaches can be first divided into iterative and non-iterative, also called explicit and implicit, co-simulation coupling methods. Iterative methods repeat a communication step until the coupling error is under a certain threshold, therefore it must be possible to reset the whole subsystem, this is a restrictive assumption to the subsystems. Non-iterative methods calculate a communication step only once, there is no repeating or resetting of the subsystem needed. The terms explicit and implicit co-simulation methods arise from the analogy to numerical solvers of ordinary differential equations, e.g. the explicit and implicit Euler method, see e.g. [3]. Based on the exploited information for the coupling, co-simulation methods can be classified into signal-based and model-based approaches. Compared to signal-based coupling methods (sbc), which rely exclusively on the coupling signals, model-based

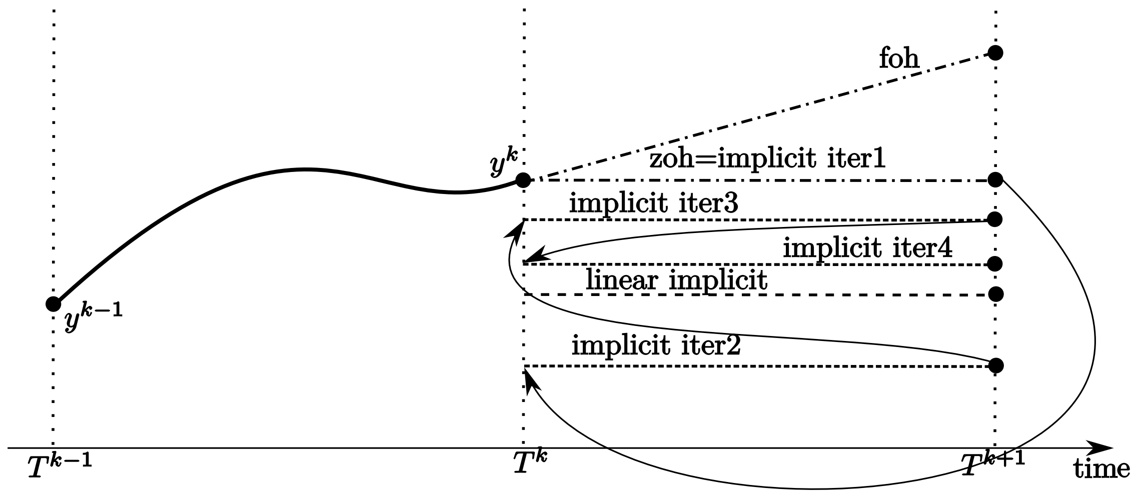


Figure 1.2: *Illustration of explicit, linear-implicit, and implicit extrapolation respectively coupling in co-simulation*

coupling methods (mbc) require additional information about the subsystems, non-differentiating whether the information of the subsystem is provided by the subsystem itself, or whether it is either approximated somehow. Figure 1.2 shows the differences between explicit or non-iterative (dashed-dotted lines) and implicit or iterative (dashed lines) co-simulation, the resetting of the subsystem required for the iterative coupling approach is displayed via the arrows. A mix of explicit and implicit approaches are the so-called linear implicit coupling methods, these methods are non-iterative but have benefits of the implicit approaches regarding accuracy and stability. Typically linear implicit methods are model-based approaches, as e.g. [4, 5] or the Model-based Pre-Step Stabilization Method presented in this work.

The basic and easiest way to co-simulate various subsystems is to utilize the so-called zero-order-hold (ZOH) extrapolation, this is a non-iterative and signal-based coupling method, depicted in Figure 1.2. By this coupling approach the coupled signal is kept constant for the period of one communication step and therefore represents an extrapolation with a constant, respectively a polynomial extrapolation of order zero. Widely used is also the first-order-hold (FOH) extrapolation method, which is a polynomial extrapolation of order one, depicted in Figure 1.2. Theoretically a polynomial extrapolation of arbitrary order is possible but due to robustness and practical reasons, typically polynomial extrapolations up to a maximum order of two are common. The Nearly Energy Preserving Coupling Element (NEPCE), see [6, 7, 8] is a non-iterative and signal-based approach which compensates the coupling error in terms of energy. The lost energy of the extrapolation error is brought back into the subsystem in the next communication step, therefore this can be seen as a deferred correction approach. The NEPCE approach can be extended by a correction term if the subsystem includes a direct feed-through, see therefore [9]. This means that this extended NEPCE approach is a model-based coupling method because information about the subsystem is required.

Model-based coupling approaches can be differentiated into two further classes based on the requirement, whether the access to the states of the subsystems is mandatory or not. On the one hand the Linear-Implicit Stabilization Method in [4, 10] requires the access to the states of the subsystems. On the other hand the Model-based Corrector from [5] or the iterative coupling method Interface Jacobian based Co-Simulation Algorithm from [11, 12] does not require access to the states of the subsystem. The introduced Model-based Pre-Step Stabilization Method is a non-iterative coupling method which does not require access to the states of the subsystems. This is a mayor difference, because for simulation tools it can be hard to provide access to the states and the number of states is typically much higher than the number of coupling signals.

2 Model-based Pre-Step Stabilization Method

The Model-based Pre-Step Stabilization Method is a co-simulation coupling method especially designed to handle stiff problems by utilizing the information of the Interface Jacobians of the subsystems. The coupling method achieves improved stability and accuracy of the overall co-simulation.

This chapter will deal with the derivation of the Model-based Pre-Step Stabilization Method, which is based on the Interface Jacobian subsystem representation. A typical subsystem description is the state space description, see e.g. [13], and therefore the connection of the Interface Jacobians and state space representation will be discussed. The benefits and also the drawbacks of the Interface Jacobian representation will be in focus, too.
The Model-based Pre-Step Stabilization Method is composed of three main steps: (1) computing the approximated monolithic output, via solving the so-called Error Differential Equation; (2) performing the model-based extrapolation of the monolithic output one communication step into the future; (3) optimization of the input to minimize the deviation of the extrapolated monolithic output and the co-simulation result. The derivation of these steps will be carried out for the case of continuous-time and discrete-time subsystems.
An important question is how to access the crucial Interface Jacobians. There are two possible ways, the first one is that the subsystem itself provides the information, like it is mentioned in the Functional Mock-Up Interface Standard 2.0, see [2]. Unfortunately this option is in practice not supported by many simulation tools. The second way is to approximate the Interface Jacobians, based on the in- and outputs of a subsystem. This can be done by standard system identification methods, the focus in this work is drawn to two methods: (1) a subspace method, the so-called Multivariable Output Error State Space method (MOESP), for details see [14, 15, 16]; (2) the Recursive Least Squares (RLS) approximation method, see e.g. [17, 18].
The structure and the workflow of the coupling method is stated and discussed, there all required parameters and options of the coupling method are given. Special attention is drawn to the so-called Error-based Phase Check and the learning and switch

phase, which are important for the case of approximated Interface Jacobians.

2.1 Subsystem Representation

The basis for a model-based method is the description of the system, the model. For this work a linear time-variant representation has been chosen and therefore the most common description is the so-called state space representation

$$\dot{\boldsymbol{x}} = \boldsymbol{A}(t)\boldsymbol{x} + \boldsymbol{B}(t)\boldsymbol{u}, \tag{2.1}$$

$$\boldsymbol{y} = \boldsymbol{C}(t)\boldsymbol{x} + \boldsymbol{D}(t)\boldsymbol{u}, \tag{2.2}$$

for a continuous-time system. It should be denoted that this description is time-variant, i.e. that the system dynamics can change over time. Here $\boldsymbol{u}$ stands for a column vector of size m which denotes input of the system. The output is described by $\boldsymbol{y}$ a column vector of size n. Additionally to the in- and outputs there exists the so-called states $\boldsymbol{x}$, a multidimensional column vector, these are quantities which are utilized to model the physical processes, dependencies and connections inside the system. It should be mentioned that $\boldsymbol{y}, \boldsymbol{u}$ and $\boldsymbol{x}$ are time dependent. The state matrix $\boldsymbol{A}(t)$ describes the evolution of the states, $\boldsymbol{B}(t)$ is called the input matrix and connects the inputs with the states. In the output equation (2.2) the output matrix $\boldsymbol{C}(t)$ describes how the states influence the output and the matrix $\boldsymbol{D}(t)$ is called direct feed-through matrix, because it represents the instantaneous influence of the input to the output. It should be emphasized that the matrices $\boldsymbol{A}(t), \boldsymbol{B}(t), \boldsymbol{C}(t)$ and $\boldsymbol{D}(t)$ can vary over time, as the dependency of t denotes.

A time-discretization of (2.1) leads to the discrete-time state space representation

$$\boldsymbol{x}_{k+1} = \boldsymbol{\Phi}_{\boldsymbol{A}_k}\boldsymbol{x}_k + \boldsymbol{\Phi}_{\boldsymbol{B}_k}\boldsymbol{u}_k, \tag{2.3}$$

$$\boldsymbol{y}_k = \boldsymbol{C}_k\boldsymbol{x}_k + \boldsymbol{D}_k\boldsymbol{u}_k. \tag{2.4}$$

The connection to the continuous-time representation is the following

$$\boldsymbol{\Phi}_{\boldsymbol{A}_k} := e^{\boldsymbol{A}_k \Delta T}, \tag{2.5}$$

$$\boldsymbol{\Phi}_{\boldsymbol{B}_k} := \int_0^{\Delta T} e^{\boldsymbol{A}_k \tau}\boldsymbol{B}_k d\tau \tag{2.6}$$

where ΔT denotes a constant sampling rate and $\boldsymbol{A}_k := \boldsymbol{A}(\Delta T \cdot k), \boldsymbol{B}_k := \boldsymbol{B}(\Delta T \cdot k), \boldsymbol{C}_k := \boldsymbol{C}(\Delta T \cdot k)$ and $\boldsymbol{D}_k := \boldsymbol{D}(\Delta T \cdot k)$ for $k \in \mathbb{Z}$. The function $e^{\boldsymbol{A}}$ denotes the matrix exponential of the matrix $\boldsymbol{A}$, for more details see e.g. [19]. It should be mentioned that the continuous and the discrete-time representation are equally widespread. The benefit of the continuous-time description is that typically it is easier to model in continuous-time because most physical laws are given in a continuous-time manner. Therefore the utilization of a continuous-time description will be often the case if the modeling of a subsystem is of high importance and interest. The discrete-time description is better suited for the case of system identification, because most system identification techniques, and especially the two focused ones of this work, are identifying in a discrete-time manner. Therefore this representation is better suited if system identification is required.

For the case of co-simulation there are two major drawbacks of the state space representation. First, typically the subsystems are black boxes, so the only known quantities are the inputs and outputs of a subsystem and therefore the access to the states or the number of states is not possible. Although in FMI 2.0, see [2], there is the possibility that the subsystem itself provides the state space matrices, but unfortunately nearly none simulation software is supporting this feature. Second, if one gets access to the state matrix $\boldsymbol{A}$, there will be the issue that, typically the size of $\boldsymbol{A}$ will be much greater than the number of in- or outputs. This would lead to a much higher computational effort of the whole model-based coupling method, which would result in a significant slow down of the overall co-simulation, so this case is from minor interest. All in all this means, that a subsystem description including states is not preferable. Therefore the non-linear, input-output based description of the subsystem S_i is

$$\dot{\boldsymbol{y}}_i = \boldsymbol{S}_i(\boldsymbol{y}_i, \boldsymbol{u}_i, \dot{\boldsymbol{u}}_i). \tag{2.7}$$

The basis of the coupling method is the linearized input-output based representation

$$\dot{\boldsymbol{y}}_i = \frac{\partial \boldsymbol{S}_i}{\partial \boldsymbol{y_i}} \boldsymbol{y_i} + \frac{\partial \boldsymbol{S}_i}{\partial \boldsymbol{u_i}} \boldsymbol{u_i} + \frac{\partial \boldsymbol{S}_i}{\partial \dot{\boldsymbol{u}}_i} \dot{\boldsymbol{u}}_i. \tag{2.8}$$

The matrices $\frac{\partial \boldsymbol{S}_i}{\partial \boldsymbol{y}_i}, \frac{\partial \boldsymbol{S}_i}{\partial \boldsymbol{u}_i}, \frac{\partial \boldsymbol{S}_i}{\partial \dot{\boldsymbol{u}}_i}$ are called the partial derivatives or the Interface Jacobians of the subsystem $\boldsymbol{S}_i$. The concept of Interface Jacobians avoids the usage of states and is based on in- and outputs of a subsystem exclusively. The Interface Jacobians have already been used for iterative co-simulation in [11, 12, 20]. The main aim of this work is to utilize the Interface Jacobians for non-iterative co-simulation, for previous publications of this coupling method see [21, 22, 23].

2.1.1 Continuous-Time Version

The continuous-time representation of a subsystem $\boldsymbol{S}_i$ by Interface Jacobians is stated in (2.8). Here it should be denoted that in general this description is time-variant, i.e. the matrices $\frac{\partial \boldsymbol{S}_i}{\partial \boldsymbol{y}}, \frac{\partial \boldsymbol{S}_i}{\partial \boldsymbol{u}}, \frac{\partial \boldsymbol{S}_i}{\partial \dot{\boldsymbol{u}}}$ are time dependent and can change their values over time. It should be denoted that a necessary assumption is that the in- and outputs of a subsystem are continuous differentiable, which means e.g. that hybrid systems are not considered. As subsystems with a relative degree > 1 would lead to $\frac{\partial \boldsymbol{S}_i}{\partial \boldsymbol{u_i}} = \frac{\partial \boldsymbol{S}_i}{\partial \dot{\boldsymbol{u}}_i} = 0$, the linearized subsystem description (2.8) is designed for handling subsystems with a relative degree $\leqslant 1$. As the co-simulation of subsystems with an relative degree > 1 is not challenging, only subsystems with a relative degree $\leqslant 1$ are considered. The relative degree of a subsystem is defined as the maximum of the relative degree of every output of the subsystem, see [24].

For academic examples and for general academic reasons, the connection of the state space representation in (2.1) and (2.2) and the Interface Jacobian subsystem description in (2.8) for the continuous-time case is of high interest and will be further discussed in the following. Time differentiation of the output equation of $\boldsymbol{S}_i$

$$\boldsymbol{y}_i = \boldsymbol{C}_i(t)\boldsymbol{x}_i + \boldsymbol{D}_i(t)\boldsymbol{u}_i \tag{2.9}$$

is leading to

$$\dot{\boldsymbol{y}}_i = \boldsymbol{C}_i(t)\dot{\boldsymbol{x}}_i + \boldsymbol{D}_i(t)\dot{\boldsymbol{u}}_i.$$

In this equation the two terms $\dot{\boldsymbol{C}}_i(t)\boldsymbol{x}$ and $\dot{\boldsymbol{D}}_i(t)\boldsymbol{u}$, which have their origin in the product rule of differentiation, are neglected because only system matrices which slowly vary in time are considered, therefore $\dot{\boldsymbol{C}}_i(t)\boldsymbol{x} \approx \dot{\boldsymbol{D}}_i(t)\boldsymbol{u} \approx \boldsymbol{0}$ is assumed. Inserting the state equation (2.1) in the equation above results in

$$\dot{\boldsymbol{y}}_i = \boldsymbol{C}_i(t)\left(\boldsymbol{A}_i(t)\boldsymbol{x}_i + \boldsymbol{B}_i(t)\boldsymbol{u}_i\right) + \boldsymbol{D}_i(t)\dot{\boldsymbol{u}}_i.$$

From rearranging follows

$$\dot{\boldsymbol{y}}_i = \boldsymbol{C}_i(t)\boldsymbol{A}_i(t)\boldsymbol{x}_i + \boldsymbol{C}_i(t)\boldsymbol{B}_i(t)\boldsymbol{u}_i + \boldsymbol{D}_i(t)\dot{\boldsymbol{u}}_i. \tag{2.10}$$

Transforming the output equation (2.9) into

$$\boldsymbol{x}_i = \boldsymbol{C}_i^{-1}(t)\left(\boldsymbol{y}_i - \boldsymbol{D}_i(t)\boldsymbol{u}_i\right) \tag{2.11}$$

and inserting it into (2.10), is leading to

$$\dot{\boldsymbol{y}}_i = \boldsymbol{C}_i(t)\big(\boldsymbol{A}_i(t)\left[\boldsymbol{C}_i^{-1}(t)\left(\boldsymbol{y}_i - \boldsymbol{D}_i(t)\boldsymbol{u}_i\right)\right] + \boldsymbol{B}_i(t)\boldsymbol{u}_i\big) + \boldsymbol{D}_i(t)\dot{\boldsymbol{u}}_i.$$

Rearranging results in

$$\begin{aligned}\dot{\boldsymbol{y}}_i = {} & \boldsymbol{C}_i(t)\boldsymbol{A}_i(t)\boldsymbol{C}_i^{-1}(t)\boldsymbol{y}_i + \\ & \big(-\boldsymbol{C}_i(t)\boldsymbol{A}_i(t)\boldsymbol{C}_i^{-1}(t)\boldsymbol{D}_i(t) + \boldsymbol{C}_i(t)\boldsymbol{B}_i(t)\big)\boldsymbol{u}_i + \boldsymbol{D}_i(t)\dot{\boldsymbol{u}}_i.\end{aligned}$$

Comparing this equation with (2.8) leads to

$$\frac{\partial \boldsymbol{S}_i}{\partial \boldsymbol{y}_i} = \boldsymbol{C}_i(t)\boldsymbol{A}_i(t)\boldsymbol{C}_i^{-1}(t), \tag{2.12}$$

$$\frac{\partial \boldsymbol{S}_i}{\partial \boldsymbol{u}_i} = -\boldsymbol{C}_i(t)\boldsymbol{A}_i(t)\boldsymbol{C}_i^{-1}(t)\boldsymbol{D}_i(t) + \boldsymbol{C}_i(t)\boldsymbol{B}_i(t), \tag{2.13}$$

$$\frac{\partial \boldsymbol{S}_i}{\partial \dot{\boldsymbol{u}}_i} = \boldsymbol{D}_i(t). \tag{2.14}$$

This shows the connection of the Interface Jacobian subsystem description and the state space representation.

The inversion of the output matrix $\boldsymbol{C}(t)$ in (2.11) should be discussed in more detail. The following is partly published in [25]. The basic assumption for a sensible application of the Model-based Pre-Step Stabilization Method is, that the outputs $y_j, j = 1, \ldots, n(i)$ of the subsystem $\boldsymbol{S}_i$ are linear independent, this means

$$c_1 y_1 + \ldots c_n y_n = 0 \quad \Rightarrow c_1 = \ldots = c_n = 0.$$

This characterization is equivalent to the fact that none y_i can be combined from the others. This means, that every output consists unique information and no output signal is redundant. To discuss the inversion of $\boldsymbol{C}(t)$, the case differentiation according the number of states x and the number of outputs n, is helpful:
$\underline{x < n}$: This case is not of interest, as a subsystem with less states than outputs is not sensible from the modeling point of view. Additionally only due to the influence of the direct feed-through, it is possible to get n linear independent outputs from only x states.
$\underline{x = n}$: For the case that the number of outputs is the same as the number of states the matrix $\boldsymbol{C}(t)$ is a square matrix. So there is still the question, if a singular matrix $\boldsymbol{C}(t)$ is possible. Due to the assumption that all n outputs are linear independent also x states have to be linear independent and therefore the mapping between these have to be unique, this means that $\boldsymbol{C}(t)$ is regular. If the influence of the direct feed-through matrix $\boldsymbol{D}(t)$ is not negligible the case can occur that $\boldsymbol{C}(t)$ is singular. That would mean, that for the crucial inversion of $\boldsymbol{C}(t)$ the application of pseudo inverse is needed, herein the preferred one is the so-called Moore-Penrose pseudo inverse, see for more Details [26].
$\underline{x > n}$: As the number of states is greater than the number of outputs not all dynamics of the states can be displayed in the outputs and therefore the inversion of $\boldsymbol{C}(t)$, utilizing the pseudo inverse, is always canceling some behaviour of the subsystem out. Especially for the case of strong connected states, this can lead to problems with the Interface Jacobian subsystem description.
The mandatory inversion of $\boldsymbol{C}(t)$ has major influence to the choice of certain parameters for the system identification methods, more about this in Section 2.3.

2.1.2 Discrete-Time Version

The discrete-time Interface Jacobian subsystem description

$$\boldsymbol{y}_i^{k+1} = \frac{\partial \boldsymbol{S}_{d_i}^k}{\partial \boldsymbol{y}_i^k} \boldsymbol{y}_i^k + \frac{\partial \boldsymbol{S}_{d_i}^k}{\partial \boldsymbol{u}_i^k} \boldsymbol{u}_i^k + \frac{\partial \boldsymbol{S}_{d_i}^k}{\partial \boldsymbol{u}_i^{k+1}} \boldsymbol{u}_i^{k+1}, \tag{2.15}$$

is the basis for the Model-based Stabilization Method for the case of approximated Interface Jacobians, because system identification methods typically generate discrete-time models. To keep the notation and following derivation as simple as possible, the communication step size ΔT is assumed to be constant for the whole simulation and the subscript i for the subsystem $\boldsymbol{S}_i$ is omitted. In this work the discrete-time state space description is the following

$$\boldsymbol{x}^{k+1} = \boldsymbol{\Phi}_{\boldsymbol{A}}^k \boldsymbol{x}^k + \boldsymbol{\Phi}_{\boldsymbol{B}}^k \boldsymbol{u}^{k+1}, \tag{2.16}$$

$$\boldsymbol{y}^{k+1} = \boldsymbol{C}^k \boldsymbol{x}^{k+1} + \boldsymbol{D}^k \boldsymbol{u}^{k+1}. \tag{2.17}$$

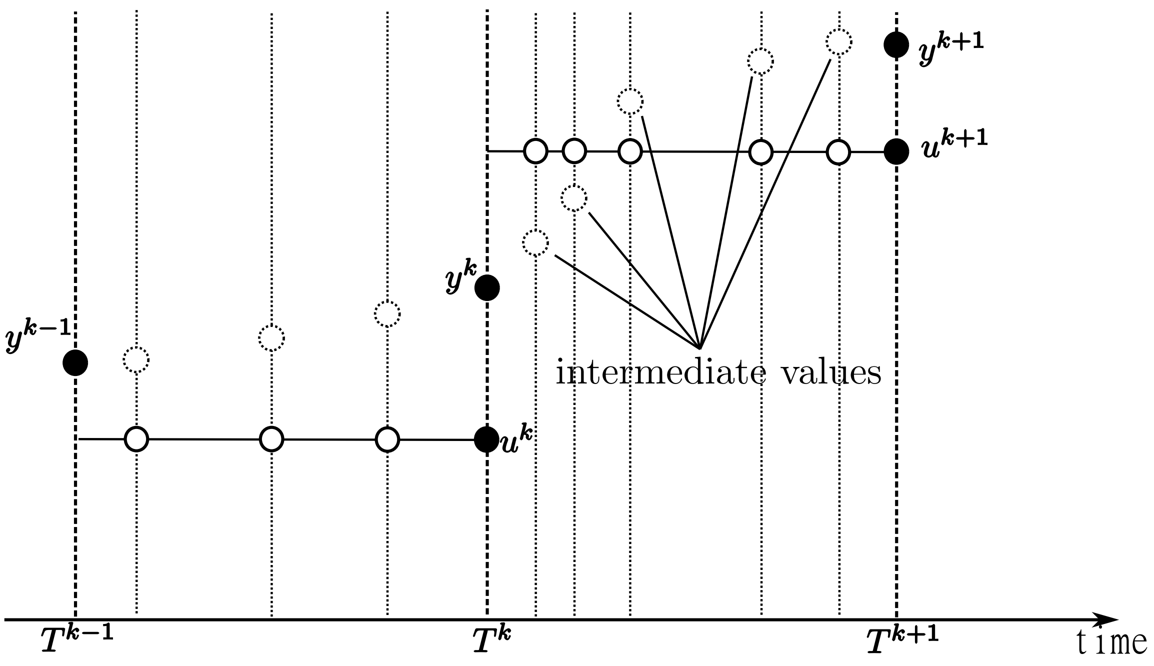


Figure 2.1: *Connection between input and output of a discrete-time subsystem for co-simulation*

The matrices $\boldsymbol{\Phi}_{\boldsymbol{A}}^k, \boldsymbol{\Phi}_{\boldsymbol{B}}^k, \boldsymbol{C}^k$ and $\boldsymbol{D}^k$ are denoted with the subscript k because the matrices are approximated on information, i.e. in- and outputs, until the communication point $T^k := k\Delta T$. In (2.16) it is clear to see, that the input $\boldsymbol{u}^{k+1}$ influences $\boldsymbol{x}^{k+1}$ and as depicted in Figure 2.1 and stated in (2.17) the output $\boldsymbol{y}^{k+1}$ is influenced by $\boldsymbol{u}^{k+1}$, too. This means that at T^k the input $\boldsymbol{u}$ is extrapolated to the communication point T^{k+1}, resulting in $\boldsymbol{u}^{k+1}$. Then the subsystem, i.e. the subsystem solver, is interpolating the input at the appropriate time points between T^k and T^{k+1}, how the input is interpolated is mostly out of the power of the co-simulation. The subsystem solver decides the number and step size of the interpolation of $\boldsymbol{u}$, also the utilization of a variable step size solver is possible and common. This is important because in this way the subsystems are typically integrated in co-simulation platforms, see therefore [27, 28]. Although this state space description is in contrast to the classical discrete-time state space representation in (2.3) and (2.4). If another co-simulation platform integrates the subsystems in a different way, one has to change this subsystem description appropriately and so the Model-based Stabilization Method can be utilized for this platform, too.
In the following the connection of the discrete-time Interface Jacobian subsystem description (2.15) and the discrete-time state space description in (2.16) and (2.17) is derived. Inserting (2.16) in (2.17) is leading to

$$\boldsymbol{y}^{k+1} = \boldsymbol{C}^k \left(\boldsymbol{\Phi}_{\boldsymbol{A}}^k \boldsymbol{x}^k + \boldsymbol{\Phi}_{\boldsymbol{B}}^k \boldsymbol{u}^{k+1}\right) + \boldsymbol{D}^k \boldsymbol{u}^{k+1}.$$

Rearranging (2.17), with a time shift of one communication step back, results in

$$\boldsymbol{x}^k = \left(\boldsymbol{C}^{k-1}\right)^{-1} \left[\boldsymbol{y}^k - \boldsymbol{D}^{k-1}\boldsymbol{u}^k\right].$$

Substituting this into the equation above is leading to

$$\boldsymbol{y}^{k+1} = \boldsymbol{C}^k \left(\boldsymbol{\Phi}_{\boldsymbol{A}}^k \left(\boldsymbol{C}^{k-1}\right)^{-1} \left[\boldsymbol{y}^k - \boldsymbol{D}^{k-1}\boldsymbol{u}^k\right] + \boldsymbol{\Phi}_{\boldsymbol{B}}^k \boldsymbol{u}_{k+1} \right) + \boldsymbol{D}^k \boldsymbol{u}^{k+1}.$$

Transforming this equation in

$$\begin{aligned} \boldsymbol{y}^{k+1} = & \boldsymbol{C}^k \boldsymbol{\Phi}_{\boldsymbol{A}}^k \left(\boldsymbol{C}^{k-1}\right)^{-1} \boldsymbol{y}^k - \boldsymbol{C}^k \boldsymbol{\Phi}_{\boldsymbol{A}}^k \left(\boldsymbol{C}^{k-1}\right)^{-1} \boldsymbol{D}^{k-1}\boldsymbol{u}^k + \dots \\ & \dots + \left(\boldsymbol{C}^k \boldsymbol{\Phi}_{\boldsymbol{B}}^k + \boldsymbol{D}^k\right) \boldsymbol{u}^{k+1} \end{aligned}$$

and comparing this with (2.15) results in

$$\frac{\partial \boldsymbol{S}_d^k}{\partial \boldsymbol{y}^k} = \boldsymbol{C}^k \boldsymbol{\Phi}_{\boldsymbol{A}}^k \left(\boldsymbol{C}^{k-1}\right)^{-1}, \tag{2.18}$$

$$\frac{\partial \boldsymbol{S}_d^k}{\partial \boldsymbol{u}^k} = -\boldsymbol{C}^k \boldsymbol{\Phi}_{\boldsymbol{A}}^k \left(\boldsymbol{C}^{k-1}\right)^{-1} \boldsymbol{D}^{k-1}, \tag{2.19}$$

$$\frac{\partial \boldsymbol{S}_d^k}{\partial \boldsymbol{u}^{k+1}} = \boldsymbol{C}^k \boldsymbol{\Phi}_{\boldsymbol{B}}^k + \boldsymbol{D}^k. \tag{2.20}$$

The inversion of $\boldsymbol{C}^{k-1}$ can be analogously discussed to the continuous-time case, see therefore Section 2.1.1. In (2.20) is it obvious that a direct feed-through $\boldsymbol{D}^k$ has the same influence as $\boldsymbol{C}^k \boldsymbol{\Phi}_{\boldsymbol{B}}^k$. This means that for this subsystem representation there is no difference between a direct feed-through and high dynamics of the input which influences the output through the states, which is compatible with Fig. 2.1.

2.2 Three Main Steps

The core of the Model-based Pre-Step Stabilization Method are the following three main steps:
(1) approximation of the monolithic results, utilizing the Error Differential Equation;
(2) performing a model-based extrapolation of the monolithic results one communication step into the future;
(3) optimizing the inputs in such a way that the gap between the extrapolated monolithic result and the co-simulated resulted is minimal.
In the following, these three steps are derived and discussed in detail for the continuous- and discrete-time version of the coupling method. This section is mainly published in [23]. The following assumptions are made to keep the notation of the derivation as simple as possible:

- the overall co-simulation consists of two fully coupled subsystems (i.e. $u_1 = y_2$, $u_2 = y_1$ at the communication points);
- the inputs and outputs u_i, y_i for $i = 1, 2$ are scalar signals;
- the communication step size ΔT and the subsystem solver step size δT are fixed for the whole computation time and equal for both subsystems;
- the inputs and outputs u_i, y_i for $i = 1, 2$ are continuous differentiable signals in time;

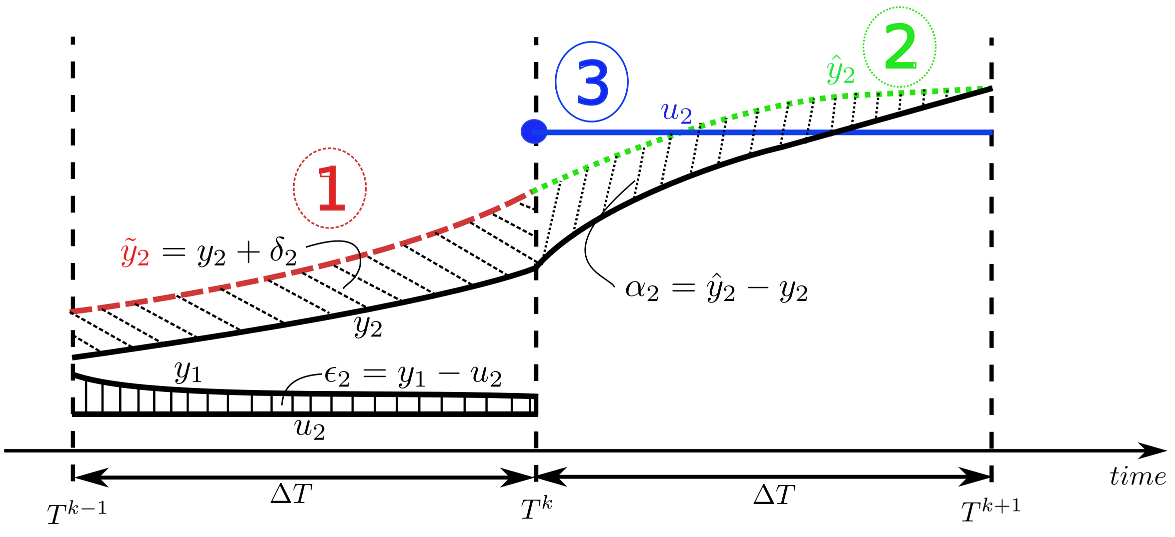


Figure 2.2: *Illustration of the three main steps of the Model-based Pre-Step Stabilization Method, from the viewpoint of subsystem S_2, all subsystems have simulated until the communication point T^k.*

- to ensure zero-stability, see [1, 29], the co-simulation does not consist of an algebraic loop, i.e. at least one of the direct feed-through terms of the subsystems is zero;
- the co-simulation scheduling is parallel, that means all inputs have to be extrapolated in every communication step.

2.2.1 Continuous-Time Version

The continuous-time version of the Model-based Pre-Step Stabilization Method consists of two ordinary differential equations respectively initial value problems and one optimization problem. The user-defined parameters and their influence to the results according accuracy and stability will be discussed in Chapter 3. The idea of the method is in Figure 2.2, from the viewpoint of the subsystem S_2, depicted. Subsystem S_1 and S_2 have simulated until the communication point T^k. In step one, the monolithic result $\tilde{y}_2$ is approximated over the last communication step. Based on that, a model-based extrapolation over the next communication step is performed in step two, resulting in $\hat{y}_2$. Step three contains the optimization of the input u_2 in such a way that the difference α_2 between y_2 and $\hat{y}_2$ is minimized.

Step 1: Approximate the Monolithic Solution

For approximation of the monolithic solution $\tilde{y}_i$ over the last communication step $T^{k-1} \rightarrow T^k$ of a subsystem, see step 1 in Figure 2.2, the so-called Error Differential

Equation is utilized, which represents an ordinary, vector-valued differential equation[1]. Its derivation is based on the subsystem description (2.8) and the following two cases:

- Ideal coupling of the two subsystems:

$$\dot{\tilde{y}}_1 = \frac{\partial S_1}{\partial y_1}\tilde{y}_1 + \frac{\partial S_1}{\partial u_1}\tilde{y}_2 + \frac{\partial S_1}{\partial \dot{u}_1}\dot{\tilde{y}}_2 \tag{2.21}$$

$$\dot{\tilde{y}}_2 = \frac{\partial S_2}{\partial y_2}\tilde{y}_2 + \frac{\partial S_2}{\partial u_2}\tilde{y}_1 + \frac{\partial S_2}{\partial \dot{u}_2}\dot{\tilde{y}}_1 \tag{2.22}$$

- Coupling by co-simulation:

$$\dot{y}_1 = \frac{\partial S_1}{\partial y_1}y_1 + \frac{\partial S_1}{\partial u_1}u_1 + \frac{\partial S_1}{\partial \dot{u}_1}\dot{u}_1 \tag{2.23}$$

$$\dot{y}_2 = \frac{\partial S_2}{\partial y_2}y_2 + \frac{\partial S_2}{\partial u_2}u_2 + \frac{\partial S_2}{\partial \dot{u}_2}\dot{u}_2 \tag{2.24}$$

Comparing (2.21) with (2.23) results in

$$\dot{\tilde{y}}_1 - \frac{\partial S_1}{\partial y_1}\tilde{y}_1 + \frac{\partial S_1}{\partial u_1}\tilde{y}_2 + \frac{\partial S_1}{\partial \dot{u}_1}\dot{\tilde{y}}_2 = \dot{y}_1 - \frac{\partial S_1}{\partial y_1}y_1 + \frac{\partial S_1}{\partial u_1}u_1 + \frac{\partial S_1}{\partial \dot{u}_1}\dot{u}_1. \tag{2.25}$$

Inserting of the definition

$$\delta_i := \tilde{y}_i - y_i, \text{ for } i = 1, 2, \tag{2.26}$$

representing the gap between the monolithic output $\tilde{y}_i$ and the computed output y_i from the subsystem, see Figure 2.2, in (2.25) is resulting in

$$\frac{\partial S_1}{\partial y_1}\tilde{y}_1 + \frac{\partial S_1}{\partial u_1}\tilde{y}_2 + \frac{\partial S_1}{\partial \dot{u}_1}\dot{\tilde{y}}_2 - \frac{\partial S_1}{\partial y_1}\cdot y_1 - \frac{\partial S_1}{\partial u_1}u_1 - \frac{\partial S_1}{\partial \dot{u}_1}\dot{u}_1 = \underbrace{\dot{\tilde{y}}_1 - \dot{y}_1}_{\dot{\delta}_1}. \tag{2.27}$$

Combining the extrapolation error[2] $\epsilon_2 := y_1 - u_2$ with $\delta_1 = \tilde{y}_1 - y_1$ is leading to

$$\tilde{y}_1 = u_2 + \epsilon_2 + \delta_1. \tag{2.28}$$

Rearranging (2.26) for $i = 2$ is resulting in

$$\tilde{y}_2 = y_2 + \delta_2. \tag{2.29}$$

Rearranging the extrapolation errors $\epsilon_2 = y_1 - u_2$ and $\epsilon_1 := y_2 - u_1$ is leading to

$$y_1 = u_2 + \epsilon_2, \tag{2.30}$$

$$u_1 = y_2 - \epsilon_1. \tag{2.31}$$

[1]The size of the differential equation is based on the overall number of outputs of all subsystems, for this derivation, due to the assumptions, it is fixed to two.

[2]The terms ϵ_1 and ϵ_2 are denoted as extrapolation errors because they describe the deviation of the coupled signals, $u_1 = y_2$ and $u_2 = y_1$ at the communication points, over a communication step, see Figure 2.2.

Inserting (2.28) - (2.31) in (2.27) results in

$$\begin{aligned}\dot{\delta}_1 = \frac{\partial S_1}{\partial y_1}\left(u_2+\epsilon_2+\delta_1\right) + \frac{\partial S_1}{\partial u_1}\left(y_2+\delta_2\right) + \frac{\partial S_1}{\partial \dot{u}_1}(\dot{y}_2+\dot{\delta}_2) \\ \ldots - \frac{\partial S_1}{\partial y_1}\left(u_2+\epsilon_2\right) - \frac{\partial S_1}{\partial u_1}\left(y_2-\epsilon_1\right) - \frac{\partial S_1}{\partial \dot{u}_1}\left(\dot{y}_2-\dot{\epsilon}_1\right).\end{aligned}$$

From algebraic rearrangements follows the final ordinary Error Differential Equation

$$\dot{\delta}_1 = \frac{\partial S_1}{\partial y_1}\delta_1 + \frac{\partial S_1}{\partial u_1}(\delta_2+\epsilon_1) + \frac{\partial S_1}{\partial \dot{u}_1}(\dot{\delta}_2+\dot{\epsilon}_1). \tag{2.32}$$

Via symmetry for (2.22) and (2.24) follows

$$\dot{\delta}_2 = \frac{\partial S_2}{\partial y_2}\delta_2 + \frac{\partial S_2}{\partial u_2}(\delta_1+\epsilon_2) + \frac{\partial S_2}{\partial \dot{u}_2}(\dot{\delta}_1+\dot{\epsilon}_2). \tag{2.33}$$

The fact that the two ordinary differential equations above are coupled motivates to write them in vector and matrix notation

$$\begin{aligned}\underbrace{\begin{pmatrix}\dot{\delta}_1\\ \dot{\delta}_2\end{pmatrix}}_{=:\dot{\boldsymbol{\delta}}} = \underbrace{\begin{pmatrix}\frac{\partial S_1}{\partial y_1} & \frac{\partial S_1}{\partial u_1}\\ \frac{\partial S_2}{\partial u_2} & \frac{\partial S_2}{\partial y_2}\end{pmatrix}}_{=:\tilde{\boldsymbol{A}}} \underbrace{\begin{pmatrix}\delta_1\\ \delta_2\end{pmatrix}}_{=:\boldsymbol{\delta}} + \underbrace{\begin{pmatrix}\frac{\partial S_1}{\partial u_1} & 0\\ 0 & \frac{\partial S_2}{\partial u_2}\end{pmatrix}}_{=:\tilde{\boldsymbol{B}}} \underbrace{\begin{pmatrix}\epsilon_1\\ \epsilon_2\end{pmatrix}}_{=:\boldsymbol{\epsilon}} \ldots \\ \ldots + \underbrace{\begin{pmatrix}0 & \frac{\partial S_1}{\partial \dot{u}_1}\\ \frac{\partial S_2}{\partial \dot{u}_2} & 0\end{pmatrix}}_{=:\tilde{\boldsymbol{C}}} \underbrace{\begin{pmatrix}\dot{\delta}_1\\ \dot{\delta}_2\end{pmatrix}}_{=:\dot{\boldsymbol{\delta}}} + \underbrace{\begin{pmatrix}\frac{\partial S_1}{\partial \dot{u}_1} & 0\\ 0 & \frac{\partial S_2}{\partial \dot{u}_2}\end{pmatrix}}_{=:\tilde{\boldsymbol{D}}} \underbrace{\begin{pmatrix}\dot{\epsilon}_1\\ \dot{\epsilon}_2\end{pmatrix}}_{=:\dot{\boldsymbol{\epsilon}}}.\end{aligned}$$

Rearranging this equation leads to

$$(\boldsymbol{I}-\tilde{\boldsymbol{C}})\dot{\boldsymbol{\delta}} = \tilde{\boldsymbol{A}}\boldsymbol{\delta} + \tilde{\boldsymbol{B}}\boldsymbol{\epsilon} + \tilde{\boldsymbol{D}}\dot{\boldsymbol{\epsilon}},$$

where $\boldsymbol{I}$ denotes the identity matrix from appropriate size. The final Error Differential Equation results in

$$\dot{\boldsymbol{\delta}} = (\boldsymbol{I}-\tilde{\boldsymbol{C}})^{-1}\left[\tilde{\boldsymbol{A}}\boldsymbol{\delta} + \tilde{\boldsymbol{B}}\boldsymbol{\epsilon} + \tilde{\boldsymbol{D}}\dot{\boldsymbol{\epsilon}}\right]. \tag{2.34}$$

It should be mentioned that $(\boldsymbol{I}-\tilde{\boldsymbol{C}})$ is regular because at least one of $\frac{\partial S_1}{\partial \dot{u}_1}$ or $\frac{\partial S_2}{\partial \dot{u}_2}$ are zero, due to (2.14) and the assumption of zero-stability.

Note: Due to the definition of $\delta_i = \tilde{y}_i - y_i$ and the assumption that y_i is continuous it follows that δ_i is continuous as well. This leads obviously to the initial conditions for $\boldsymbol{\delta} = (\delta_1, \delta_2)^T$, combining this with the equation (2.34) above, leads to an initial value problem for the computation of $\boldsymbol{\delta}$ in $[T^{k-1}, T^k]$, the monolithic solution $\tilde{y}_i$ can be computed via

$$\tilde{y}_i = \delta_i + y_i \text{ for } i = 1, 2. \tag{2.35}$$

The numerical solving of (2.34) requires an appropriate numerical solver for ordinary differential equations, like Runge-Kutta methods, see e.g. [13]. For a satisfying performance of the overall co-simulation coupling method, the parameters of the solver, especially the step size, have to be chosen appropriately.

The error analysis for co-simulations is complicated as there are different sources for the coupling error, e.g. the extrapolation of the inputs, time delay due to the coupling or aliasing[3] of the coupling signals. In the further the focus is drawn to the coupling error originating from aliasing effects. It is clear that such an aliasing in the signal can only be detected if the subsystem solver is utilizing smaller time steps and these intermediate values have to be available for the co-simulation. To handle this effect so-called anti-aliasing filters have been developed e.g. [8]. There the lost energy of the aliasing coupling error is brought back into the system with one step delay. Due to the fact that the coupling error ϵ_i for $i = 1, 2$ is the input for the Error Differential Equation, the computing of the monolithic output $\tilde{y}_i$ has additionally an anti-aliasing effect to the overall co-simulation. It should be pointed out that the aliasing coupling error is recovered with no delay. Summarizing this means that, if intermediate values are available, the utilization of the Error Differential Equation is compensating the extrapolation error and the effect of aliasing.

Step 2: Model-based Extrapolation

The second step of the coupling method is to perform a model-based extrapolation of the approximated monolithic output over the next communication step ΔT, depicted as step 2 in Figure 2.2. The model-based extrapolation of the overall system is the basis for the pre-step behaviour of the coupling method. The ordinary differential equation for the extrapolation is based on the assumption of ideal coupling between the subsystems and the subsystem description

$$\dot{\hat{y}}_i = \frac{\partial \boldsymbol{S}_i}{\partial y_i}\hat{y}_i + \frac{\partial \boldsymbol{S}_i}{\partial u_i}u_i + \frac{\partial \boldsymbol{S}_i}{\partial \dot{u}_i}\dot{u}_i \text{ for } i = 1, 2. \tag{2.36}$$

The assumption of ideal coupling between the subsystems

$$\begin{aligned}\hat{y}_1 &= u_2, \\ \hat{y}_2 &= u_1,\end{aligned}$$

is combined with equation (2.36), which results in

$$\underbrace{\begin{pmatrix}\dot{\hat{y}}_1 \\ \dot{\hat{y}}_2\end{pmatrix}}_{=:\dot{\hat{\boldsymbol{y}}}} = \underbrace{\begin{pmatrix}\frac{\partial S_1}{\partial y_1} & \frac{\partial S_1}{\partial u_1} \\ \frac{\partial S_2}{\partial u_2} & \frac{\partial S_2}{\partial y_2}\end{pmatrix}}_{=:\hat{\boldsymbol{A}}} \underbrace{\begin{pmatrix}\hat{y}_1 \\ \hat{y}_2\end{pmatrix}}_{=:\hat{\boldsymbol{y}}} + \underbrace{\begin{pmatrix}0 & \frac{\partial S_1}{\partial \dot{u}_1} \\ \frac{\partial S_2}{\partial \dot{u}_2} & 0\end{pmatrix}}_{=:\hat{\boldsymbol{B}}} \underbrace{\begin{pmatrix}\dot{\hat{y}}_1 \\ \dot{\hat{y}}_2\end{pmatrix}}_{=:\dot{\hat{\boldsymbol{y}}}}.$$

[3]Note: Aliasing of a signal means that due to the sampling rate no all important parts can be captured, therefore reducing of the sampling rate is typically recommended. In the field of co-simulation aliasing means, that the signal can not be displayed correctly, by the values at the communication points only, i.e. between two communication points the signal is varying more than linear, e.g. oscillating.

Rearranging this equation results in the final differential equation[4]

$$\dot{\hat{\boldsymbol{y}}} = (\boldsymbol{I} - \hat{\boldsymbol{B}})^{-1}\hat{\boldsymbol{A}}\hat{\boldsymbol{y}}. \tag{2.37}$$

Note: The initial condition for (2.37) is defined from the solution of the Error Differential Equation, i.e. $\hat{y}_i(T^k) = \tilde{y}_i(T^k)$ for the extrapolation over $[T^k, T^{k+1}]$, leading to an initial value problem for every communication step.
As in step one, the final ordinary differential equation in (2.37) has to be solved by an appropriate numerical solver, like the Runge-Kutta methods. Like in the case of the Error Differential Equation, the parameters of the numerical solver have to be chosen in such a way, that the accuracy of the numerical solution of (2.37) is sufficient. Otherwise the performance of the presented co-simulation coupling method could decrease drastically.

Step 3: Pre-Step Input Optimization

The Pre-step Input Optimization is the third main step of the Model-based Pre-Step Stabilization Method and can be computed for every subsystem on their own, i.e. the optimization of the input of a subsystem is independent from other subsystems and can therefore be computed in parallel. Hence the subscript i is avoided in the following derivation. The input optimization is based on

$$\alpha(t) := y(t) - \hat{y}(t),$$

which displays the gap between the co-simulated result $y(t)$ and the extrapolated monolithic output $\hat{y}(t)$. The optimization problem

$$\int_{T^k}^{T^{k+1}} |\alpha(\tau)| d\tau \longrightarrow 0 \tag{2.38}$$

is formulated in such a way that the gap between $y(t)$ and $\hat{y}(t)$ is minimized over the next communication step ΔT. The optimization problem in (2.38) can be solved by applying different numerical integration schemes, see [13], here the focus is on the Right Riemann sum

$$\int_a^b f(t)dt = (b-a)\, f(b) \tag{2.39}$$

and the Trapezoidal rule

$$\int_a^b f(t)dt = \frac{b-a}{2}\left(f(b) + f(a)\right).$$

Applying the Right Riemann sum (2.39) to (2.38) results in

$$\left(y(T^{k+1}) - \hat{y}(T^{k+1})\right) \Delta T \longrightarrow 0, \tag{2.40}$$

[4] $(\boldsymbol{I} - \hat{\boldsymbol{B}})$ is regular because of the zero-stability assumption, analogous to $(\boldsymbol{I} - \tilde{\boldsymbol{C}})$ before.

the absolute value $|\cdot|$ is here not necessary, because $y(T^{k+1}) - \hat{y}(T^{k+1})$ is a scalar value. Utilizing the Trapezoidal rule leads to

$$\big(y(T^{k+1}) - \hat{y}(T^{k+1}) + y(T^k) - \hat{y}(T^k)\big)\,\frac{\Delta T}{2} \longrightarrow 0.$$

Due to the fact that $\hat{y}(T^{k+1}), \hat{y}(T^k)$ and $y(T^k)$ are known values the choice of the numerical integration rule does not change the idea and structure for the further derivation in a crucial way. This means that also other integration schemes, like the Midpoint rule can be easily utilized. To keep the derivation understandable and the notation clear the focus is drawn to the discretized optimization problem in (2.40), it can be written as

$$y(T^{k+1}) \overset{!}{=} \hat{y}(T^{k+1}). \tag{2.41}$$

The goal of the optimization is to compute the input $u(t)$ and therefore the connection between $y(t)$ and the input $u(t)$ has to be elaborated. To describe this relation a close look on the system description (2.8) and standard theory of ordinary differential equations, e.g. [13] is needed. The analytical solution of (2.8) is described through

$$y(t) = e^{\frac{\partial S}{\partial y}(t-T^k)}y(T^k) + \int_{T^k}^{t} e^{\frac{\partial S}{\partial y}(t-\tau)}\frac{\partial S}{\partial u}u(\tau)\,d\tau + \int_{T^k}^{t} e^{\frac{\partial S}{\partial y}(t-\tau)}\frac{\partial S}{\partial \dot{u}}\dot{u}(\tau)\,d\tau, \tag{2.42}$$

where $t \in [T^k, T^{k+1}]$ for $k = 0, 1, \ldots$ with T^0 denotes the beginning of the co-simulation and $y(T^0)$ the initial condition. The derivative $\dot{u}$ denotes the time derivative, which is computed piecewise for every communication step. Due to the appearance of $\dot{u}$ in (2.42) the choice of piecewise constant basic functions is not favorable because u would not be differentiable. That is the reason why piecewise linear basic functions, they are at least weak differentiable, are utilized to describe the input u in the following, i.e. the preferred approach for $u(t)$ is

$$u(t) = u_0^k + s^k t \text{ for } t \in [T^k, T^{k+1}], \tag{2.43}$$

where $u_0^k = y(T^k)$, with y representing the output from the coupled subsystem. The slope of u in the communication step $T^k \to T^{k+1}$ is denoted with s^k. Inserting this approach in (2.42) and evaluating the equation for $t = T^{k+1}$ leads to

$$\begin{aligned} y(T^{k+1}) =& e^{\frac{\partial S}{\partial y}(T^{k+1}-T^k)}y(T^k) + \int_{T^k}^{T^{k+1}} e^{\frac{\partial S}{\partial y}(T^{k+1}-\tau)}\frac{\partial S}{\partial u}\ldots \\ & \ldots(u_0^k + s^k\tau)\,d\tau + \int_{T^k}^{T^{k+1}} e^{\frac{\partial S}{\partial y}(T^{k+1}-\tau)}\frac{\partial S}{\partial \dot{u}}s^k\,d\tau. \end{aligned}$$

Rearranging the last equation is leading to

$$\begin{aligned} y^{k+1} = e^{\frac{\partial S}{\partial y}\Delta T}y^k + u_0^k \int_{T^k}^{T^{k+1}} e^{\frac{\partial S}{\partial y}(T^{k+1}-\tau)}\frac{\partial S}{\partial u}\,d\tau + \ldots \\ s^k \int_{T^k}^{T^{k+1}} e^{\frac{\partial S}{\partial y}(T^{k+1}-\tau)}\frac{\partial S}{\partial u}\tau\,d\tau + s^k \int_{T^k}^{T^{k+1}} e^{\frac{\partial S}{\partial y}(T^{k+1}-\tau)}\frac{\partial S}{\partial \dot{u}}\,d\tau, \end{aligned}$$

with

$$\phi_A := e^{\frac{\partial S}{\partial y}\Delta T}, \tag{2.44}$$

$$\phi_B := \int_{T^k}^{T^{k+1}} e^{\frac{\partial S}{\partial y}(T^{k+1}-\tau)} \frac{\partial S}{\partial u}\, d\tau, \tag{2.45}$$

$$\phi_C := \int_{T^k}^{T^{k+1}} e^{\frac{\partial S}{\partial y}(T^{k+1}-\tau)} \frac{\partial S}{\partial u}\tau\, d\tau, \tag{2.46}$$

$$\phi_D := \int_{T^k}^{T^{k+1}} e^{\frac{\partial S}{\partial y}(T^{k+1}-\tau)} \frac{\partial S}{\partial \dot{u}}\, d\tau, \tag{2.47}$$

it is possible to write

$$y^{k+1} = \phi_A y^k + u_0^k \phi_B + s^k\left(\phi_C + \phi_D\right). \tag{2.48}$$

Inserting (2.48) into (2.41) results in

$$\hat{y}^{k+1} = \phi_A y^k + \phi_B u_0^k + s^k\left(\phi_C + \phi_D\right). \tag{2.49}$$

As it is the case, that y^k and u_0^k is known from the previous communication step and $\hat{y}^{k+1}$ is given due to the model-based extrapolation it is now possible to determine s^k through

$$s^k = \left(\phi_C + \phi_D\right)^{-1}\left(\hat{y}^{k+1} - \phi_A y^k - \phi_B u_0^k\right) \tag{2.50}$$

and therefore the input over the next communication step is fully determined. A closer look on (2.49) directly implies that for $\phi_C + \phi_D = 0$, which represents the singular case, the choice of s^k has no influence, so one can set $s^k = 0$. Therefore the computation of s^k in (2.50) is well-defined.
If one is interested in utilizing piecewise constant functions for the input, one choice is to use the energy equivalent piecewise constant input u_{const}. It is defined as

$$\int_{T^k}^{T^{k+1}} u(\tau) d\tau = \Delta T u_{const},$$

which means that over the communication step the energy, the integral of the function, is the same for input u and $u_{const.}$, therefore they are equivalent according to their energy.

Summarizing all three main steps, one can say that in step one and in step two, the ordinary differential equations (2.34) and (2.37) are taken mutual couplings between the subsystems into account. Therefore is the, one step into the future extrapolated, monolithic output, an satisfying approximation of the monolithic output. One can say it displays the optimal value for the co-simulated output of the next communication step. Based on this quantity the minimization of α will be achieved, this means the Model-based Pre-Step Stabilization Method is choosing the input in such a way, that the error over the future communication step is minimized. This means the method is compensating an extrapolated coupling error and is therefore working in pre-step manner. This one-step look into the future is the reason why the method is called pre-step. Combining this pre-step behaviour with considering the mutual coupling between the subsytems is the reason why the method is stabilizing the overall co-simulation.

2.2.2 Discrete-Time Version

A discrete-time version of the Model-based Pre-Step Stabilization Method is especially useful for the case that the Interface Jacobians have to be approximated because this is leading to the discrete-time subsystem description. A possible choice would also be to transform the identified discrete-time subsystem description into a continuous-time one, but this approach is not preferred mainly due to the following three reasons:
(1) A transformation from the discrete-time to the continuous-time is a numerical computation and therefore contains additional errors and so the quality of the approximation decreases, see therefore e.g. [30].
(2) A major benefit of the discrete-time version of the coupling method is, that the two differential equations, Step 1 and Step 2, from the continuous-time version are algebraic equations in the discrete-time version. This means less computational effort solving it and even more accurate results of the equation because the discretization error of numerical integration methods is omitted.
(3) The transformation from a discrete-time to the continuos-time representation is computational expensive because it has to be performed in every communication point.

Due to those reasons a discrete-time version of the coupling method is sensible because the direct utilization of the approximated Interface Jacobians is possible. The discrete-time Model-based Pre-Step Stabilization Method consists of the same three main steps as the continuous-time version. The difference lies in the fact that the underlying subsystem description is in discrete-time (2.15) instead of continuous-time (2.8).

Step 1: Error Differential Equation

The Error Differential Equation approximates the monolithic output based on the extrapolation error and the discrete-time Interface Jacobians. The idea and structure of the derivation is similar to the one of the continuous-time case, starting with the following cases:
Ideal coupling of the subsystems:

$$\tilde{y}_1^k = \frac{\partial S_{d_1}}{\partial y_1^{k-1}} \tilde{y}_1^{k-1} + \frac{\partial S_{d_1}}{\partial u_1^{k-1}} \tilde{y}_2^{k-1} + \frac{\partial S_{d_1}}{\partial u_1^k} \tilde{y}_2^k, \tag{2.51}$$

$$\tilde{y}_2^k = \frac{\partial S_{d_2}}{\partial y_2^{k-1}} \tilde{y}_2^{k-1} + \frac{\partial S_{d_2}}{\partial u_2^{k-1}} \tilde{y}_1^{k-1} + \frac{\partial S_{d_2}}{\partial u_2^k} \tilde{y}_1^k. \tag{2.52}$$

Co-simulation coupling of the subsystems:

$$y_1^k = \frac{\partial S_{d_1}}{\partial y_1^{k-1}} y_1^{k-1} + \frac{\partial S_{d_1}}{\partial u_1^{k-1}} u_1^{k-1} + \frac{\partial S_{d_1}}{\partial u_1^k} u_1^k, \tag{2.53}$$

$$y_2^k = \frac{\partial S_{d_2}}{\partial y_2^{k-1}} y_2^{k-1} + \frac{\partial S_{d_2}}{\partial u_2^{k-1}} u^{k-1} + \frac{\partial S_{d_2}}{\partial u_2^k} u_2^k. \tag{2.54}$$

Comparing (2.51) and (2.53) results in

$$\begin{aligned}\tilde{y}_1^k - \frac{\partial S_{d_1}}{\partial y_1^{k-1}}\tilde{y}_1^{k-1} + \frac{\partial S_{d_1}}{\partial u_1^{k-1}}\tilde{y}_2^{k-1} + \frac{\partial S_{d_1}}{\partial u_1^{k}}\tilde{y}_2^{k} &= \dots \\ \dots y_1^k - \frac{\partial S_{d_1}}{\partial y_1^{k-1}}y_1^{k-1} + \frac{\partial S_{d_1}}{\partial u_1^{k-1}}u_1^{k-1} + \frac{\partial S_{d_1}}{\partial u_1^{k}}u_1^{k}.\end{aligned} \tag{2.55}$$

The discrete-time version of (2.26) and (2.28) - (2.31) are

$$\begin{aligned}\delta_i^k &= \tilde{y}_i^k - y_i^k \text{ for } i = 1,2,\\ \tilde{y}_1^{k-1} &= u_2^{k-1} + \epsilon_2^{k-1} + \delta_1^{k-1},\\ \tilde{y}_2^{k-1} &= y_2^{k-1} + \delta_2^{k-1},\\ y_1^{k-1} &= u_2^{k-1} + \epsilon_2^{k-1},\\ u_1^{k-1} &= y_2^{k-1} - \epsilon_1^{k-1}.\end{aligned}$$

Rearranging (2.55) and utilizing the quantities above is leading to

$$\delta_1^k = \frac{\partial S_{d_1}}{\partial y_1^{k-1}}\delta_1^{k-1} + \frac{\partial S_{d_1}}{\partial u_1^{k-1}}\left(\epsilon_1^{k-1} + \delta_2^{k-1}\right) + \frac{\partial S_{d_1}}{\partial u_1^{k}}\left(\epsilon_1^{k} + \delta_2^{k}\right).$$

From (2.52) and (2.54) follows analogously

$$\delta_2^k = \frac{\partial S_{d_2}}{\partial y_2^{k-1}}\delta_2^{k-1} + \frac{\partial S_{d_2}}{\partial u_2^{k-1}}\left(\epsilon_2^{k-1} + \delta_1^{k-1}\right) + \frac{\partial S_{d_2}}{\partial u_2^{k}}\left(\epsilon_2^{k} + \delta_1^{k}\right). \tag{2.56}$$

Combining the above equations and rearranging them results in the discrete-time Error Differential Equation[5]

$$\boldsymbol{\delta}^k = (\boldsymbol{I} - \tilde{\boldsymbol{C}}_d^k)^{-1}\left[\tilde{\boldsymbol{A}}_d^k\boldsymbol{\delta}^{k-1} + \tilde{\boldsymbol{B}}_d^k\boldsymbol{\epsilon}^{k-1} + \tilde{\boldsymbol{D}}_d^k\boldsymbol{\epsilon}^k\right], \tag{2.57}$$

where

$$\begin{aligned}\boldsymbol{\delta}^k &= \begin{pmatrix}\delta_1^k\\ \delta_2^k\end{pmatrix},\ \tilde{\boldsymbol{A}}_d^k = \begin{pmatrix}\frac{\partial S_{d_1}}{\partial y_1^{k-1}} & \frac{\partial S_{d_1}}{\partial u_1^{k-1}}\\ \frac{\partial S_{d_2}}{\partial u_2^{k-1}} & \frac{\partial S_{d_2}}{\partial y_2^{k-1}}\end{pmatrix},\ \tilde{\boldsymbol{B}}_d^k = \begin{pmatrix}\frac{\partial S_{d_1}}{\partial u_1^{k-1}} & 0\\ 0 & \frac{\partial S_{d_2}}{\partial u_2^{k-1}}\end{pmatrix},\\ \boldsymbol{\epsilon}^k &= \begin{pmatrix}\epsilon_1^k\\ \epsilon_2^k\end{pmatrix},\ \tilde{\boldsymbol{C}}_d^k = \begin{pmatrix}0 & \frac{\partial S_{d_1}}{\partial u_1^{k}}\\ \frac{\partial S_{d_2}}{\partial u_2^{k}} & 0\end{pmatrix},\ \tilde{\boldsymbol{D}}_d^k = \begin{pmatrix}\frac{\partial S_{d_1}}{\partial u_1^{k}} & 0\\ 0 & \frac{\partial S_{d_2}}{\partial u_2^{k}}\end{pmatrix}.\end{aligned}$$

It should be noted that the discrete-time Error Differential Equation is an algebraic equation and therefore the solution can be computed with no additional effort.
If intermediate values of the subsystem are available there are two ways of improving the accuracy of the discrete-time Error Differential Equation:

[5]The matrix $(\boldsymbol{I} - \tilde{\boldsymbol{C}}_d^k)$ is regular due to the assumption of zero-stability, analog to the continuous-time case. It should be mentioned that the name Error Differential Equation is kept although the equation is a difference equation instead of a differential equation.

(1) First one can determine the extrapolation error based on the intermediate values and then interpolate the quantities $\epsilon^k, \epsilon^{k+1}$ energy equivalent, i.e.

$$\epsilon_{cor.} = \frac{\sum_j \epsilon_j}{\Delta T} - \left(\epsilon^k + \epsilon^{k+1}\right) 0.5, \tag{2.58}$$

$$\epsilon^k = \epsilon^k + \epsilon_{cor.}, \tag{2.59}$$

$$\epsilon^{k+1} = \epsilon^{k+1} + \epsilon_{cor.}, \tag{2.60}$$

where j denotes the index representing the intermediate values.
(2) The second option exploiting the intermediate values is, to transform the Interface Jacobians to a smaller step size, therefore one has to interpolate all intermediate values to this step size, and solving the algebraic equation (2.57) recursive with the chosen smaller step size. This option needs a higher computational effort and is therefore not always recommended.

Step 2: Model-based Extrapolation

The second step of the discrete-time Model-based Pre-Step Stabilization Method is to perform an extrapolation of the approximated monolithic output $\tilde{y}^k$, computed in Step 1. Analog to the continuous-time case the derivation starts with the case of ideal coupling:

$$\hat{y}_1^{k+1} = \frac{\partial S_{d_1}}{\partial y_1^k}\hat{y}_1^k + \frac{\partial S_{d_1}}{\partial u_1^k}\hat{y}_2^k + \frac{\partial S_{d_1}}{\partial u_1^{k+1}}\hat{y}_2^{k+1}, \tag{2.61}$$

$$\hat{y}_2^{k+1} = \frac{\partial S_{d_2}}{\partial y_2^k}\hat{y}_2^k + \frac{\partial S_{d_2}}{\partial u_2^k}\hat{y}_1^k + \frac{\partial S_{d_2}}{\partial u_2^{k+1}}\hat{y}_1^{k+1}. \tag{2.62}$$

Combining both equations above and rearranging them results in[6]

$$\hat{\boldsymbol{y}}^{k+1} = (\boldsymbol{I} - \hat{\boldsymbol{B}}^k)^{-1}\hat{\boldsymbol{A}}^k\hat{\boldsymbol{y}}^k. \tag{2.63}$$

with

$$\hat{\boldsymbol{A}}^k := \begin{pmatrix} \frac{\partial S_{d_1}}{\partial y_1^k} & \frac{\partial S_{d_1}}{\partial u_1^k} \\ \frac{\partial S_{d_2}}{\partial u_2^k} & \frac{\partial S_{d_2}}{\partial y_2^k} \end{pmatrix},$$

$$\hat{\boldsymbol{B}}^k := \begin{pmatrix} 0 & \frac{\partial S_{d_1}}{\partial u_1^{k+1}} \\ \frac{\partial S_{d_2}}{\partial u_2^{k+1}} & 0 \end{pmatrix},$$

$$\hat{\boldsymbol{y}}^{\boldsymbol{k}} := \begin{pmatrix} \hat{y}_1^k \\ \hat{y}_2^k \end{pmatrix}.$$

Analog to the continuous-time version, the initial condition is defined as $\hat{\boldsymbol{y}}^{\boldsymbol{k}} = \tilde{\boldsymbol{y}}^{\boldsymbol{k}}$, which displays the fact that the approximated monolithic result is extrapolated one communication step into the future.
Due to the fact that there is no approximation error while solving an algebraic equation, there is no need and it is not possible to improve the accuracy of (2.63).

[6]The matrix $(\boldsymbol{I}-\hat{\boldsymbol{B}}^k)$ is regular due to the assumption of zero-stability, analog to the continuous-time case.

Step 3: Input Optimization

The input optimization for the discrete-time Model-based Pre-Step Stabilization Method can be computed for every subsystem S_i independent from the others, therefore the subindex i is ommitted in this section. Based on the discrete-time subsystem description (2.15) and the extrapolated monolithic output $\hat{y}^{k+1}$ the input u^{k+1} is optimized. The optimization problem is stated as

$$\hat{y}^{k+1} \overset{!}{=} y^{k+1}$$

combining this with (2.15), results in

$$\hat{y}^{k+1} = \frac{\partial S_d^k}{\partial y^k} y^k + \frac{\partial S_d^k}{\partial u^k} u^k + \frac{\partial S_d^k}{\partial u^{k+1}} u^{k+1}.$$

The quantities $\hat{y}^{k+1}, y^k$ and u^k are known values and therefore the equation is rearranged into

$$\frac{\partial S_d^k}{\partial u^{k+1}} u^{k+1} = \hat{y}^{k+1} - \frac{\partial S_d^k}{\partial y^k} y^k - \frac{\partial S_d^k}{\partial u^k} u^k. \tag{2.64}$$

The right side of the equation above is known and therefore this equation represents a linear system of equations. The solution can be determined with

$$u^{k+1} = \left(\frac{\partial S_d}{\partial u^{k+1}}\right)^{-1} \left[\hat{y}^{k+1} - \frac{\partial S_d}{\partial y^k} y^k - \frac{\partial S_d}{\partial u^k} u^k\right]. \tag{2.65}$$

Due to the assumptions, the in- and outputs are scalar signals exclusively, $\frac{\partial S_d}{\partial u^{k+1}}$ is a scalar value and therefore the case that $\frac{\partial S_d}{\partial u^{k+1}} = 0$ can be omitted, because the optimized input would not have any influence to the system and this case is not of interest. The case of multidimensional in- and output signals will be further discussed in the next section.

2.2.3 Generalization for Multidimensional In- and Outputs

For the derivation of the main steps of the continuous- and discrete-time method it is assumed that the in- and outputs of the subsystems are scalar signals exclusively. For Step 1, the Error Differential Equation, and Step 2, the Model-based Extrapolation, the generalization for an arbitrary number of in- and outputs is straightforward. For Step 3, the Input Optimization, the generalization for an arbitrary number of in- and outputs of a subsystem will be discussed afterwards. In the following the discrete-time version will be discussed in detail, because for the continuous-time version it is analog. The need to pay attention to the number of in- and outputs can be directly seen at (2.65), because there the matrix $\frac{\partial S_d^k}{\partial u^{k+1}}$ is inverted. The size of the matrix is $n \times m$, where n denotes the number of outputs and m denotes the number of inputs. The classical inversion is only possible for the case that $n = m$ and the matrix is regular. That means three remaining cases have to discussed, $n < m$, $n = m$ with a singular

matrix and $m > n$.
For $n < m$ the linear system of equations in (2.64) is an overdetermined system, i.e. there are more equations than variables and therefore no solution exists, therefore a least squares approach is used to solve the system of equations. The least squares solution can be computed by solving the normal equations, which can be determined by multiplying (2.64) from the left with $\left(\frac{\partial S_d^k}{\partial \boldsymbol{u}^{k+1}}\right)^T$ resulting in

$$\left(\frac{\partial \boldsymbol{S}_d}{\partial \boldsymbol{u}^{k+1}}\right)^T \frac{\partial \boldsymbol{S}_d}{\partial \boldsymbol{u}^{k+1}} \boldsymbol{u}^{k+1} = \left(\frac{\partial \boldsymbol{S}_d}{\partial \boldsymbol{u}^{k+1}}\right)^T \left(\hat{\boldsymbol{y}}^{k+1} - \frac{\partial \boldsymbol{S}_d}{\partial \boldsymbol{y}^k}\boldsymbol{y}^k - \frac{\partial \boldsymbol{S}_d}{\partial \boldsymbol{u}^k}\boldsymbol{u}^k\right),$$

for more details about the normal equations see e.g. [31]. Therefore the least squares solution of (2.64) is

$$\boldsymbol{u}^{k+1} = \left[\left(\frac{\partial \boldsymbol{S}_d^k}{\partial \boldsymbol{u}^{k+1}}\right)^T \frac{\partial \boldsymbol{S}_d^k}{\partial \boldsymbol{u}^{k+1}}\right]^{-1} \left(\frac{\partial \boldsymbol{S}_d^k}{\partial \boldsymbol{u}^{k+1}}\right)^T \left(\hat{\boldsymbol{y}}^{k+1} - \frac{\partial \boldsymbol{S}_d^k}{\partial \boldsymbol{y}^k}\boldsymbol{y}^k - \frac{\partial \boldsymbol{S}_d^k}{\partial \boldsymbol{u}^k}\boldsymbol{u}^k\right).$$

Here it should be denoted that the matrix

$$\left(\frac{\partial \boldsymbol{S}_d^k}{\partial \boldsymbol{u}^{k+1}}\right)^+ := \left[\left(\frac{\partial \boldsymbol{S}_d^k}{\partial \boldsymbol{u}^{k+1}}\right)^T \frac{\partial \boldsymbol{S}_d^k}{\partial \boldsymbol{u}^{k+1}}\right]^{-1} \left(\frac{\partial \boldsymbol{S}_d^k}{\partial \boldsymbol{u}^{k+1}}\right)^T$$

is called the Moore-Penrose pseudo inverse of $\frac{\partial \boldsymbol{S}_d^k}{\partial \boldsymbol{u}^{k+1}}$, for more details of the pseudo inverse in general, see e.g. [32].
For $n = m$ it should not occur, that $\frac{\partial \boldsymbol{S}_d^k}{\partial \boldsymbol{u}^{k+1}}$ is singular because the coupled signals are all linear independent and therefore they all inherit non-redundant information and therefore is $\frac{\partial \boldsymbol{S}_d^k}{\partial \boldsymbol{u}^{k+1}}$ regular. Just in case that due to approximations or numerical errors $\frac{\partial \boldsymbol{S}_d^k}{\partial \boldsymbol{u}^{k+1}}$ gets singular the pseudo inverse should be utilized to solve the linear system of equations (2.64).
For $m > n$, which means there are more inputs than outputs, the linear system of equations in (2.64) is underdetermined, which means there are more variables than equations. Therefore additional conditions can be fulfilled, for the case of co-simulation the minimizing of the extrapolation error is a sensible claim. This means that the linear system of equations is restructured to the following optimization problem

$$\min_{\boldsymbol{u}^{k+1}} \left(\hat{\boldsymbol{y}}^{k+1} - \boldsymbol{u}^{k+1}\right)^2 \tag{2.66}$$

$$s.t. \frac{\partial \boldsymbol{S}_d}{\partial \boldsymbol{u}^{k+1}}\boldsymbol{u}^{k+1} = \underbrace{\hat{\boldsymbol{y}}^{k+1} - \frac{\partial \boldsymbol{S}_d}{\partial \boldsymbol{y}^k}\boldsymbol{y}^k - \frac{\partial \boldsymbol{S}_d}{\partial \boldsymbol{u}^k}\boldsymbol{u}^k}_{:=\boldsymbol{rhs}=const.}. \tag{2.67}$$

Here it should be denoted that $\hat{\boldsymbol{y}}^{k+1}$ stands for the appropriate extrapolated outputs from the coupled subsystem, so that $\hat{\boldsymbol{y}}^{k+1} - \boldsymbol{u}^{k+1}$ describes the extrapolation error. This optimization problem will be solved by the theory of Lagrange multipliers, for more details see e.g. [33]. The Lagrange function L is defined as

$$L(\boldsymbol{u}^{k+1}, \boldsymbol{\lambda}) = \left(\hat{\boldsymbol{y}}^{k+1} - \boldsymbol{u}^{k+1}\right)^2 + \boldsymbol{\lambda}\left(\frac{\partial \boldsymbol{S}_d}{\partial \boldsymbol{u}^{k+1}}\boldsymbol{u}^{k+1} - \boldsymbol{rhs}\right),$$

where $\boldsymbol{\lambda} := (\lambda_1, \ldots, \lambda_n)^T$ denotes the Lagrange multipliers. To determine the optimal solution $\boldsymbol{u}^{k+1}$ the following linear system of equations have to be solved

$$\frac{\partial L}{\partial \boldsymbol{u}^{k+1}} \overset{!}{=} \mathbf{0},$$
$$\frac{\partial L}{\partial \boldsymbol{\lambda}} \overset{!}{=} \mathbf{0}.$$

Differentiation L with respect to $\boldsymbol{u}^{k+1}$ results in

$$\frac{\partial L}{\partial \boldsymbol{u}^{k+1}} = -2\left(\hat{\boldsymbol{y}}^{k+1} - \boldsymbol{u}^{k+1}\right) + \left(\frac{\partial \boldsymbol{S}_d}{\partial \boldsymbol{u}^{k+1}}\boldsymbol{\lambda}\right)^T \overset{!}{=} \mathbf{0} \tag{2.68}$$

and differentiating L with respect to $\boldsymbol{\lambda}$ is leading to

$$\frac{\partial L}{\partial \boldsymbol{\lambda}} = \frac{\partial \boldsymbol{S}_d}{\partial \boldsymbol{u}^{k+1}}\boldsymbol{u}^{k+1} - \boldsymbol{rhs} \overset{!}{=} \mathbf{0}. \tag{2.69}$$

Rearranging (2.68) results in

$$\boldsymbol{u}^{k+1} = \hat{\boldsymbol{y}}^{k+1} - 0.5\left(\frac{\partial \boldsymbol{S}_d}{\partial \boldsymbol{u}^{k+1}}\right)^T \boldsymbol{\lambda}, \tag{2.70}$$

inserting this in (2.69) leads to

$$\frac{\partial \boldsymbol{S}_d}{\partial \boldsymbol{u}^{k+1}}\left[\hat{\boldsymbol{y}}^{k+1} - 0.5\left(\frac{\partial \boldsymbol{S}_d}{\partial \boldsymbol{u}^{k+1}}\right)^T \boldsymbol{\lambda}\right] = \boldsymbol{rhs}.$$

Algebraic rearranging results in

$$\left[\frac{\partial \boldsymbol{S}_d}{\partial \boldsymbol{u}^{k+1}}\left(\frac{\partial \boldsymbol{S}_d}{\partial \boldsymbol{u}^{k+1}}\right)^T\right]\boldsymbol{\lambda} = 2\frac{\partial \boldsymbol{S}_d}{\partial \boldsymbol{u}^{k+1}}\hat{\boldsymbol{y}}^{k+1} - 2\boldsymbol{rhs}.$$

Finally $\boldsymbol{\lambda}$ can be determined by solving

$$\boldsymbol{\lambda} = \left[\frac{\partial \boldsymbol{S}_d}{\partial \boldsymbol{u}^{k+1}}\left(\frac{\partial \boldsymbol{S}_d}{\partial \boldsymbol{u}^{k+1}}\right)^T\right]^{-1}\left[\frac{\partial \boldsymbol{S}_d}{\partial \boldsymbol{u}^{k+1}}\hat{\boldsymbol{y}}^{k+1} - 2\boldsymbol{rhs}\right]. \tag{2.71}$$

Combining this equation with (2.70) results in the solution of the optimization problem (2.71), that means $\boldsymbol{u}^{k+1}$ can be computed in such a way that the extrapolation error is minimized additionally.

2.3 Approximation of the Interface Jacobians

The Interface Jacobians are the basis for the coupling method and therefore the question how to access them is fundamental. The Functional Mock-Up Interface Standard 2.0 is providing the function *fmi2GetDirectionalDerivative*, by this the required partial derivatives of the subsystem are accessible, unfortunately these functionality is

supported by almost none simulation tool. Therefore the Interface Jacobians have to be approximated, the utilization of system identification methods is unavoidable. This work is focusing on two different system identification methods: First the so-called Multivariable Output Error State Space method or MOESP method, which is a specific version of the so-called subspace methods. Second a Recursive Least Square (RLS) approach is utilized for approximating the Interface Jacobians. As all system identification methods, these two also require a so-called learning phase, this means it is not possible to determine sensible Interface Jacobians in the first steps. This means that at the beginning of every co-simulation there is a certain time frame where a signal-based coupling approach (ZOH or FOH) has to be applied because the Interface Jacobians are not available, more about this so-called learning phase in Section 2.4.3.
Both methods require constant sampling rates and therefore the communication step size is restricted to be constant through the whole co-simulation. All subsystem solvers are fixed step solvers with the same solver step size, so the number of possible intermediate values is in every communication step and for every subsystem is the same. The identified Interface Jacobians are, for both methods, in discrete-time description. In every communication point the system identification has to be performed and is therefore an online identification, i.e. it happens simultaneously to the co-simulation. To keep the notation simple and clear, the subindex i for the subsystem $\boldsymbol{S}_i$ is omitted in every quantity because all considerations are valid for each subsystem independent of the others.

2.3.1 Multivariable Output Error State Space Method

The MOESP method is based on the inputs $\boldsymbol{u}$ and the outputs $\boldsymbol{y}$ of a subsystem exclusively. Therefore it is well suited for this situation because typically the subsystems are black boxes and so there is no additional information available. This section will derive the MOESP method, similar as in [5], for more details see [14, 15, 16]. As the system identification will be applied online, the inputs $\boldsymbol{u}^0, \ldots, \boldsymbol{u}^k$ and the outputs $\boldsymbol{y}^0, \ldots, \boldsymbol{y}^k$ of the subsystem are available, assuming that the actual communication point is T^k.
The MOESP method will compute the linear time-invariant discrete-time state space matrices $\boldsymbol{\Phi_A}, \boldsymbol{\Phi_B}, \boldsymbol{C}$ and $\boldsymbol{D}$, which represents the state space description

$$\begin{aligned}\boldsymbol{x}_{k+1} &= \boldsymbol{\Phi_A}\boldsymbol{x}_k + \boldsymbol{\Phi_B}\boldsymbol{u}_k,\\ \boldsymbol{y}_k &= \boldsymbol{C}\boldsymbol{x}_k + \boldsymbol{D}\boldsymbol{u}_k.\end{aligned}$$

Starting r time steps back

$$\begin{aligned}\boldsymbol{x}_{k-r+2} &= \boldsymbol{\Phi_A}\boldsymbol{x}_{k-r+1} + \boldsymbol{\Phi_B}\boldsymbol{u}_{k-r+1},\\ \boldsymbol{y}_{k-r+1} &= \boldsymbol{C}\boldsymbol{x}_{k-r+1} + \boldsymbol{D}\boldsymbol{u}_{k-r+1}\end{aligned}$$

and r-times recursive insertion until index k is reached, results in

$$\begin{aligned}
\boldsymbol{x}_{k-r+2} &= \boldsymbol{\Phi_A} \boldsymbol{x}_{k-r+1} + \boldsymbol{\Phi_B} \boldsymbol{u}_{k-r+1},\\
\boldsymbol{x}_{k-r+3} &= \boldsymbol{\Phi_A} \boldsymbol{x}_{k-r+2} + \boldsymbol{\Phi_B} \boldsymbol{u}_{k-r+2},\\
&= \boldsymbol{\Phi_A}^2 \boldsymbol{x}_{k-r+1} + \boldsymbol{\Phi_A}\boldsymbol{\Phi_B} \boldsymbol{u}_{k-r+1} + \boldsymbol{\Phi_B} \boldsymbol{u}_{k-r+2},\\
&\vdots\\
\boldsymbol{x}_{k} &= \boldsymbol{\Phi_A}^{r-1} \boldsymbol{x}_{k-r+1} + \boldsymbol{\Phi_A}^{r-2}\boldsymbol{\Phi_B} \boldsymbol{u}_{k-r+1} + \ldots + \boldsymbol{\Phi_A}\boldsymbol{\Phi_B} \boldsymbol{u}_{k-1} + \boldsymbol{\Phi_B} \boldsymbol{u}_{k},\\
\boldsymbol{y}_{k-r+1} &= \boldsymbol{C} \boldsymbol{x}_{k-r+1} + \boldsymbol{D} \boldsymbol{u}_{k-r+1},\\
\boldsymbol{y}_{k-r+2} &= \boldsymbol{C} \boldsymbol{x}_{k-r+2} + \boldsymbol{D} \boldsymbol{u}_{k-r+2},\\
&= \boldsymbol{C}\boldsymbol{\Phi_A} \boldsymbol{x}_{k-r+1} + \boldsymbol{C}\boldsymbol{\Phi_B} \boldsymbol{u}_{k-r+1} + \boldsymbol{D} \boldsymbol{u}_{k-r+2},\\
&\vdots\\
\boldsymbol{y}_{k} &= \boldsymbol{C}\boldsymbol{\Phi_A}^{r-1} \boldsymbol{x}_{k-r+1} + \boldsymbol{C}\boldsymbol{\Phi_A}^{r-2}\boldsymbol{\Phi_B} \boldsymbol{u}_{k-r+1} + \ldots + \boldsymbol{C}\boldsymbol{B}\boldsymbol{\Phi_A}\boldsymbol{\Phi_B} \boldsymbol{u}_{k-1} + \boldsymbol{D} \boldsymbol{u}_{k}.
\end{aligned}$$

Rewriting these equations above results in

$$\begin{aligned}
\underbrace{\begin{pmatrix} \boldsymbol{y}_{k-r+1} \\ \boldsymbol{y}_{k-r+2} \\ \boldsymbol{y}_{k-r+3} \\ \vdots \\ \boldsymbol{y}_{k} \end{pmatrix}}_{:=\boldsymbol{y}_{k-r+1|k}} &= \underbrace{\begin{pmatrix} \boldsymbol{C} \\ \boldsymbol{C}\boldsymbol{\Phi_A} \\ \boldsymbol{C}\boldsymbol{\Phi_A}^2 \\ \vdots \\ \boldsymbol{C}\boldsymbol{\Phi_A}^{r-1} \end{pmatrix}}_{:=\boldsymbol{\Gamma}_r} \boldsymbol{x}_{k-r+1} + \ldots \\
&+ \underbrace{\begin{pmatrix} \boldsymbol{D} & & & & \\ \boldsymbol{C}\boldsymbol{\Phi_B} & \boldsymbol{D} & & & \\ \boldsymbol{C}\boldsymbol{\Phi_A}\boldsymbol{\Phi_B} & \boldsymbol{C}\boldsymbol{\Phi_B} & \boldsymbol{D} & & \\ \vdots & \ddots & \ddots & \ddots & \\ \boldsymbol{C}\boldsymbol{\Phi_A}^{r-2}\boldsymbol{\Phi_B} & \ldots & \boldsymbol{C}\boldsymbol{\Phi_A}\boldsymbol{\Phi_B} & \boldsymbol{C}\boldsymbol{\Phi_B} & \boldsymbol{D} \end{pmatrix}}_{:=\boldsymbol{\Psi}_r} \underbrace{\begin{pmatrix} \boldsymbol{u}_{k-r+1} \\ \boldsymbol{u}_{k-r+2} \\ \boldsymbol{u}_{k-r+3} \\ \vdots \\ \boldsymbol{u}_{k} \end{pmatrix}}_{:=\boldsymbol{u}_{k-r+1|k}},
\end{aligned} \tag{2.72}$$

where $\boldsymbol{\Gamma}_r$ stands for the extended observability matrix, $\boldsymbol{\Psi}_r$ for a Toeplitz matrix and $\boldsymbol{y}_{k-r+1|k}$ and $\boldsymbol{u}_{k-r+1|k}$ for the out- respectively inputs of the last r time steps, from index $k-r+1$ to index k. The Block Hankel matrice composed of the outputs is defined as

$$\begin{aligned}
\boldsymbol{Y}_{k-r+1|k} := &\begin{pmatrix} \boldsymbol{y}_{k-r-N+2|k-N+1} & \boldsymbol{y}_{k-r-N+3|k-N+2} & \cdots & \boldsymbol{y}_{k-r+1|k} \end{pmatrix} = \ldots \\
= &\begin{pmatrix} \boldsymbol{y}_{k-r-N+2} & \boldsymbol{y}_{k-r-N+3} & \cdots & \boldsymbol{y}_{k-r+1} \\ \boldsymbol{y}_{k-r-N+3} & \boldsymbol{y}_{k-r-N+4} & \cdots & \boldsymbol{y}_{k-r+2} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{y}_{k-N+1} & \boldsymbol{y}_{k-N+2} & \cdots & \boldsymbol{y}_{k} \end{pmatrix}.
\end{aligned} \tag{2.73}$$

Analog the Block Hankel matrix composed of the inputs is defined as

$$\begin{aligned}\boldsymbol{U}_{k-r+1|k} :&= \begin{pmatrix}\boldsymbol{u}_{k-r-N+2|k-N+1} & \boldsymbol{u}_{k-r-N+3|k-N+2} & \dots & \boldsymbol{u}_{k-r+1|k}\end{pmatrix} = \dots \\ &= \begin{pmatrix}\boldsymbol{u}_{k-r-N+2} & \boldsymbol{u}_{k-r-N+3} & \dots & \boldsymbol{u}_{k-r+1} \\ \boldsymbol{u}_{k-r-N+3} & \boldsymbol{u}_{k-r-N+4} & \dots & \boldsymbol{u}_{k-r+2} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{u}_{k-N+1} & \boldsymbol{u}_{k-N+2} & \dots & \boldsymbol{u}_{k}\end{pmatrix}.\end{aligned} \tag{2.74}$$

Here N stands for the window size, i.e. the number of past samples which are taken into account for the system identification, typically $N >> r$. Taken these two Hankel matrices it is possible to write

$$\boldsymbol{Y}_{k-r+1|k} = \boldsymbol{\Gamma}_r \boldsymbol{X}_k + \boldsymbol{\Psi}_r \boldsymbol{U}_{k-r+1|k}, \tag{2.75}$$

where $\boldsymbol{X}_k := \begin{pmatrix}\boldsymbol{x}_{k-r-N+2} & \boldsymbol{x}_{k-r-N+3} & \dots & \boldsymbol{x}_{k-r+1}\end{pmatrix}$. Equ. (2.75) can be written as

$$\begin{pmatrix}\boldsymbol{U}_{k-r+1|k} \\ \boldsymbol{Y}_{k-r+1|k}\end{pmatrix} = \begin{pmatrix}\boldsymbol{I} & \boldsymbol{O} \\ \boldsymbol{\Psi}_r & \boldsymbol{\Gamma}_r\end{pmatrix} \begin{pmatrix}\boldsymbol{U}_{k-r+1|k} \\ \boldsymbol{X}_k\end{pmatrix}. \tag{2.76}$$

Combining this with the classical LQ-decomposition, see e.g. [3],

$$\begin{pmatrix}\boldsymbol{U}_{k-r+1|k} \\ \boldsymbol{Y}_{k-r+1|k}\end{pmatrix} = \begin{pmatrix}\boldsymbol{L}_{11} & 0 \\ \boldsymbol{L}_{21} & \boldsymbol{L}_{22}\end{pmatrix} \begin{pmatrix}\boldsymbol{Q}_1^T \\ \boldsymbol{Q}_2^T\end{pmatrix}, \tag{2.77}$$

where $\boldsymbol{Q}_1^T$ and $\boldsymbol{Q}_2^T$ are orthogonal and $\boldsymbol{L}_{11}$ and $\boldsymbol{L}_{22}$ are lower triangular matrices and inserting in (2.76) the upper block row $\boldsymbol{U}_{k-r+1|k} = \boldsymbol{L}_{11}\boldsymbol{Q}_1^T$, into the lower block results in

$$\boldsymbol{Y}_{k-r+1|k} = \boldsymbol{\Psi}_r \boldsymbol{L}_{11} \boldsymbol{Q}_1^T + \boldsymbol{\Gamma}_r \boldsymbol{X}_k. \tag{2.78}$$

Rearranging of (2.77) leads to

$$\boldsymbol{Y}_{k-r+1|k} = \boldsymbol{L}_{21}\boldsymbol{Q}_1^T + \boldsymbol{L}_{22}\boldsymbol{Q}_2^T, \tag{2.79}$$

in contrast to (2.78) here the sums are orthogonal, and therefore the summands of (2.78) and (2.79) can not be equal. Multiplying (2.78) and (2.79) from the right side with $\boldsymbol{Q}_2$ leads to

$$\boldsymbol{Y}_{k-r+1|k}\boldsymbol{Q}_2 = \boldsymbol{\Psi}_r \boldsymbol{L}_{11} \underbrace{\boldsymbol{Q}_1^T \boldsymbol{Q}_2}_{=\boldsymbol{0}} + \boldsymbol{\Gamma}_r \boldsymbol{X}_k \boldsymbol{Q}_2, \tag{2.80}$$

$$\boldsymbol{Y}_{k-r+1|k}\boldsymbol{Q}_2 = \boldsymbol{L}_{21} \underbrace{\boldsymbol{Q}_1^T \boldsymbol{Q}_2}_{=\boldsymbol{0}} + \boldsymbol{L}_{22} \underbrace{\boldsymbol{Q}_2^T \boldsymbol{Q}_2}_{=\boldsymbol{I}}. \tag{2.81}$$

To approximate the discrete-time matrices $\boldsymbol{\Phi}_{\boldsymbol{A}}$ and $\boldsymbol{C}$ the row space of $\boldsymbol{\Gamma}_r$ has to be extracted and therefore a singular value decomposition, see e.g. [31], is applied to (2.81), resulting in

$$\boldsymbol{L}_{22} = \begin{pmatrix}\boldsymbol{U}_s \\ \boldsymbol{U}_n\end{pmatrix} \begin{pmatrix}\boldsymbol{S}_s & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S}_n\end{pmatrix} \begin{pmatrix}\boldsymbol{V}_s^T \\ \boldsymbol{V}_n^T\end{pmatrix}$$

Here $\boldsymbol{S}_n$ denotes the singular values which are non-significant, i.e. they are below a certain threshold and therefore we can assume $\boldsymbol{S}_n = \boldsymbol{0}$, this leads to

$$\boldsymbol{L}_{22} = \boldsymbol{U}_s \boldsymbol{S}_s \boldsymbol{V}_s^T = \boldsymbol{\Gamma}_r \boldsymbol{X}_k \boldsymbol{Q}_2. \tag{2.82}$$

Therefore the extended observability matrix $\boldsymbol{\Gamma}_r$ can be stated as follows, with the desired order and the corresponding row space

$$\boldsymbol{\Gamma}_r = \boldsymbol{U}_s \boldsymbol{S}_s^{1/2}. \tag{2.83}$$

From the definition of the extended observability matrix $\boldsymbol{\Gamma}_r$ in (2.72) one can see that $\boldsymbol{C}$ can be determined from the first block row of $\boldsymbol{\Gamma}_r$. To identify $\boldsymbol{\Phi_A}$ the following linear system has to be solved

$$\underline{\boldsymbol{\Gamma}_r} \boldsymbol{\Phi_A} = \overline{\boldsymbol{\Gamma}_r} \tag{2.84}$$

where $\underline{\boldsymbol{\Gamma}_r}$ stands for $\boldsymbol{\Gamma}_r$ without the last block row and $\overline{\boldsymbol{\Gamma}_r}$ denotes $\boldsymbol{\Gamma}_r$ without the first block row. The system is overdetermined and therefore a solution should be computed by using the least square approach. To determine the matrices $\boldsymbol{\Phi_B}$ and $\boldsymbol{D}$ the multiplication of (2.78) and (2.79) from the left with $\boldsymbol{U}_n^T$ and from the right with $\boldsymbol{Q}_1$

$$\begin{aligned}
\boldsymbol{U}_n^T \boldsymbol{Y}_{k-r+1|k} \boldsymbol{Q}_1 &= \boldsymbol{U}_n^T \boldsymbol{\Psi}_r \boldsymbol{L}_{11} \underbrace{\boldsymbol{Q}_1^T \boldsymbol{Q}_1}_{=\boldsymbol{I}} + \underbrace{\boldsymbol{U}_n^T \boldsymbol{\Gamma}_r \boldsymbol{X}_k \boldsymbol{Q}_1}_{=\boldsymbol{0}} \\
\boldsymbol{U}_n^T \boldsymbol{Y}_{k-r+1|k} \boldsymbol{Q}_1 &= \boldsymbol{U}_n^T \boldsymbol{L}_{21} \underbrace{\boldsymbol{Q}_1^T \boldsymbol{Q}_1}_{=\boldsymbol{I}} + \underbrace{\boldsymbol{U}_n^T \boldsymbol{L}_{22} \boldsymbol{Q}_2^T \boldsymbol{Q}_1}_{=\boldsymbol{0}}
\end{aligned}$$

is required. The term $\boldsymbol{U}_n^T \boldsymbol{\Gamma}_r \boldsymbol{X}_k \boldsymbol{Q}_1 = \boldsymbol{0}$ and $\boldsymbol{U}_n^T \boldsymbol{L}_{22} \boldsymbol{Q}_2^T \boldsymbol{Q}_1 = \boldsymbol{0}$ holds because of (2.82) with $\boldsymbol{U}_n^T \boldsymbol{U}_s = \boldsymbol{0}$. Combining the above equations results in

$$\boldsymbol{U}_n^T \boldsymbol{\Psi}_r \boldsymbol{L}_{11} = \boldsymbol{U}_n^T \boldsymbol{L}_{21} \tag{2.85}$$

which represents a overdetermined linear system of equations to compute $\boldsymbol{\Phi_B}$ and $\boldsymbol{D}$, therefore the least squares approach should be applied to solve it.

Summarizing the MOESP method in the following steps:

1. Set the window size N.
2. Determine the In/Output data matrices $\boldsymbol{Y}_{k-r+1|k}$ in (2.73) and $\boldsymbol{U}_{k-r+1|k}$ in (2.74).
3. Compute the LQ decomposition of $\begin{pmatrix}\boldsymbol{Y}_{k-r+1|k} & \boldsymbol{U}_{k-r+1|k}\end{pmatrix}^T$ in (2.77).
4. Compute the singular value decomposition of $\boldsymbol{L}_{22}$ in (2.82) and determine or choose the order of the identified model.
5. Evaluate (2.83) and determine so the extended observability matrix $\boldsymbol{\Gamma}_r$.
6. Extract $\boldsymbol{C}$ from $\boldsymbol{\Gamma}_r$ and solve (2.84) to determine $\boldsymbol{\Phi_A}$.
7. Solve the linear system of equations in (2.85) and determine $\boldsymbol{\Phi_B}$ and $\boldsymbol{D}$.

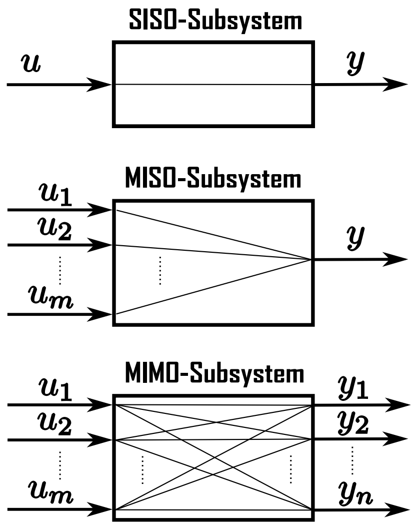


Figure 2.3: *Dependencies of in- and output by SISO, MISO and MIMO systems*

The MOESP method will be applied in every communication step and the used samples are collected by a moving window of size N. Therefore a time-invariant system is available in every communication step leading to a time-variant subsystem description. Here it should be mentioned that, the larger the moving window is, i.e. choose N large, the slower the adaption of a change in the subsystem dynamics is.
The identified state space matrices $\boldsymbol{\Phi_A}, \boldsymbol{\Phi_B}, \boldsymbol{C}$ and $\boldsymbol{D}$ have to be transformed in the Interface Jacobian representation, this can be done by utilizing the equations (2.18) - (2.20). Here it should be emphasized that the transformation rules are based on time-variant subsystems and therefore an ignoring of the time shift in the transformation rules can lead to poor results.

2.3.2 Recursive Least Squares

The Recursive Least Squares (RLS) approach is widely known and often used as a method for system identification, more details e.g. [17, 18]. In this work the RLS will be utilized to identify the time-variant discrete-time Interface Jacobian subsystem representation in (2.15). The following section will explain the identification process starting with the classical single input single output (SISO) RLS, over a RLS for multiple inputs and single outputs systems (MISO) to a MIMO (multiple input multiple output) RLS. Then the transformation to a state space model[7] utilizing the Gilbert Realization, see e.g. [34], is discussed and finally this state space model is transformed to the desired Interface Jacobian subsystem representation. The differences between the SISO, MISO and MIMO subsystems are depicted in Fig. 2.3, there it is obvious that every input may influence every output, which is directly connected to the number of to be identified parameters.

[7]It should be mentioned that with a RLS approach a direct feed-through term $\boldsymbol{D}$ can not be identified and therefore it is always set to zero.

The RLS for a SISO system is the basis of the identification process, it is a standard method and can be found in many books or papers, e.g. [17, 18]. A SISO subsystem, where y denotes the scalar output and u stands for the scalar input, can by described by the following difference equation

$$y^k = a_1^k y^{k-1} + \ldots + a_N^k y^{k-N} + b_1^k u^{k-1} + \ldots + b_M^k u^{k-M}. \tag{2.86}$$

Here k denotes the discrete time index, N and M stands for the number of considered past samples of the output and input and the order of the difference equation is the maximum of N and M. The SISO RLS approximates the parameters $a_1^k, \ldots a_N^k$ and $b_1^k, \ldots b_M^k$ in the least square sense, i.e. minimizing the quadratic error. The following pseudo code steps describes the SISO RLS:

```
1: $\boldsymbol{g} = \boldsymbol{P}\boldsymbol{\phi}\left(\lambda + \boldsymbol{\phi}^T\boldsymbol{P}\boldsymbol{\phi}\right)^{-1}$
2: $\boldsymbol{p} = \boldsymbol{p} + \boldsymbol{g}\left(y^k_{measure} - \boldsymbol{\phi}^T\boldsymbol{p}\right)$
3: $\boldsymbol{P} = \left(\boldsymbol{P} - \boldsymbol{g}\boldsymbol{\phi}^T\boldsymbol{P}\lambda^{-1}\right)$
```

$\boldsymbol{g}$ denotes the vector of the gain, $\boldsymbol{P}$ stands for the covariance matrix, $\boldsymbol{p} := \left[a_1^k, \ldots, a_N^k, b_1^k, \ldots, b_M^k\right]^T$ describes the vector of the to be identified parameters, $\boldsymbol{\phi} := \left[y^{k-1}, \ldots, y^{k-N}, u^{k-1}, \ldots, u^{k-M}\right]^T$ denotes the data vector containing the appropriate in- and output samples and λ is called the forgetting factor. In the first line of the RLS the gain is computed. The second line describes the update of the identified parameter $\boldsymbol{p}$, here it is clear to see, that the parameters are only adapted if the measurement $y^k_{measure}$ and the solution based on the identified model $y^k = \boldsymbol{\phi}^T\boldsymbol{p}$ does not fit. In the third line the covariance matrix $\boldsymbol{P}$ is updated. The initial value of the matrix $\boldsymbol{P}$, with a size of $(M+N) \times (M+N)$ has to be chosen by the user. A detailed derivation of the RLS can be found, e.g. [17, 18]. Due to the adaption of the parameters the result of the identification is a time-variant system description.
The first step to generalize the SISO RLS, is to apply the same idea to MISO systems, there the following difference equation is the basis

$$\begin{aligned} y^k =& a_1^k y^{k-1} + \ldots + a_N^k y^{k-N} + b_{11}^k u_1^{k-1} + \ldots + b_{1M}^k u_1^{k-M} + \\ & b_{21}^k u_2^{k-1} + \ldots + b_{M2}^k u_2^{k-M} + \ldots + b_{m1}^k u_m^{k-1} + \ldots + b_{Mm}^k u_m^{k-M}, \end{aligned} \tag{2.87}$$

where m denotes the number of inputs. The difference to the SISO system is that, one output depends on m inputs. The SISO RLS can be easily adapted to a MISO RLS, by changing the definitions of $\boldsymbol{\phi}$ and $\boldsymbol{p}$ to

$$\begin{aligned} \boldsymbol{p} :=& \left[a_1^k, \ldots, a_N^k, b_{11}^k, \ldots, b_{1M}^k, b_{21}^k, \ldots, b_{M2}^k, b_{m1}^k, \ldots, b_{Mm}^k\right]^T, \\ \boldsymbol{\phi} :=& \left[y^{k-1}, \ldots, y^{k-N}, u_1^{k-1}, \ldots, u_1^{k-M}, u_2^{k-1}, \ldots, u_2^{k-M}, u_m^{k-1}, \ldots, u_m^{k-M}\right]^T. \end{aligned}$$

According the parametrization, only the size of the covariance matrix $\boldsymbol{P}$ is changed to $(N + Mm) \times (N + Mm)$.
To identify subsystems containing n outputs and m inputs, so-called MIMO systems, the MISO RLS has to be applied to every output. This means that the MIMO RLS is basically a MISO RLS for every output of the subsystem, the MISO RLS is n times,

independent from each other, applied, see also Fig. 2.3. The difference equations of a MIMO system are

$$\begin{aligned}
y_1^k =& a_{11}^k y_1^{k-1} + \ldots + a_{1N}^k y_1^{k-N} + b_{111}^k u_1^{k-1} + \ldots + b_{11M}^k u_1^{k-M} + \\
& b_{121}^k u_2^{k-1} + \ldots + b_{1M2}^k u_2^{k-M} + \ldots + b_{1m1}^k u_m^{k-1} + \ldots + b_{1Mm}^k u_m^{k-M}, \\
y_2^k =& a_{21}^k y_2^{k-1} + \ldots + a_{2N}^k y_2^{k-N} + b_{211}^k u_1^{k-1} + \ldots + b_{21M}^k u_1^{k-M} + \\
& b_{221}^k u_2^{k-1} + \ldots + b_{2M2}^k u_2^{k-M} + \ldots + b_{2m1}^k u_m^{k-1} + \ldots + b_{2Mm}^k u_m^{k-M}, \\
&\vdots \\
y_n^k =& a_{n1}^k y_n^{k-1} + \ldots + a_{nN}^k y_n^{k-N} + b_{n11}^k u_1^{k-1} + \ldots + b_{n1M}^k u_1^{k-M} + \\
& b_{n21}^k u_2^{k-1} + \ldots + b_{nM2}^k u_2^{k-M} + \ldots + b_{nm1}^k u_m^{k-1} + \ldots + b_{nMm}^k u_m^{k-M}.
\end{aligned}$$

and the parameters $a_{11}^k, \ldots, a_{nN}^k$ and $b_{111}^k, \ldots, b_{nMm}^k$ are identified with the MIMO RLS.

Difference equations are equivalent to discrete-time transfer functions $H(z)$, which contain the elements

$$\boldsymbol{H}_{ij}^k(z) = \frac{y_i(z)}{u_j(z)} = \frac{b_{ij1}^k z^{-1} + \ldots + b_{ijM}^k z^{-M}}{1 + a_{i1}^k z^{-1} + \ldots + a_{iN}^k z^{-N}} \tag{2.88}$$

for all $i = 1, \ldots, n$ and $j = 1, \ldots, m$. Based on this MIMO transfer function $\boldsymbol{H}(z)$ a state space model can be realized. The transformation from a state space model to a transfer function is unique and is described by

$$\boldsymbol{H}^k(z) = \boldsymbol{C}^k \left(z\boldsymbol{I} - \boldsymbol{\Phi_A}^k\right)^{-1} \boldsymbol{\Phi_B}^k + \boldsymbol{D}^k. \tag{2.89}$$

The reverse way, from the transfer function to the state space model is not necessarily unique and more complicated to determine. The so-called minimal realization of $\boldsymbol{H}(z)$ is desired, because it describes $\boldsymbol{H}(z)$ with the least possible number of states, with the Gilberts Realization, see e.g. [34], it is possible to determine such a minimal realization. The first step of Gilberts Realization is to determine the partial fraction expansion for every $\boldsymbol{H}_{ij}(z)$, resulting in

$$\boldsymbol{H}^k(z) = \sum_{i=1}^{\overline{n}} \frac{R_i^k}{z - z_i}, \tag{2.90}$$

where z_i denotes the $\overline{n}$ poles of the transfer function. The $n \times m$ matrices $\boldsymbol{R}_i$ can be determined as the residues of z_i

$$\boldsymbol{R}_i^k = \lim_{z \to z_i} (z - z_i)^k \boldsymbol{H}(z).$$

With $n_i := \text{rank}(\boldsymbol{R}_i)$ the state matrix $\boldsymbol{\Phi_A}^k$ in Gilberts Realization is determined as

$$\boldsymbol{\Phi_A}^k = \begin{pmatrix} z_1 \boldsymbol{I}_{n_1} & & \\ & \ddots & \\ & & z_{\overline{n}} \boldsymbol{I}_{n_{\overline{n}}} \end{pmatrix}. \tag{2.91}$$

This means that the number of states is $\sum_{i=1}^{\overline{n}} n_i$. Comparing (2.89),(2.90) and (2.91) results in the fact that the matrices R_i can be written as

$$\boldsymbol{R}_i = \boldsymbol{C}_i \boldsymbol{\Phi}_{\boldsymbol{B}i}$$

where $\boldsymbol{C}_i$ is a $n \times n_i$ matrix and $\boldsymbol{\Phi}_{\boldsymbol{B}i}$ a $n_i \times m$ matrix is, leading to

$$\boldsymbol{\Phi}_{\boldsymbol{B}} = \begin{pmatrix} \boldsymbol{\Phi}_{\boldsymbol{B}1} \\ \boldsymbol{\Phi}_{\boldsymbol{B}2} \\ \vdots \\ \boldsymbol{\Phi}_{\boldsymbol{B}\overline{n}} \end{pmatrix},$$
$$\boldsymbol{C} = (\boldsymbol{C}_1 \boldsymbol{C}_2 \dots \boldsymbol{C}_{\overline{n}})\,.$$

Here it should be denoted that choice of $\boldsymbol{\Phi}_{\boldsymbol{B}i}$ and $\boldsymbol{C}_i$ is not unique, therefore the minimal realization as a discrete-time state space model is not unique either.
To determine the discrete-time Interface Jacobian subsystem representation in (2.15) the utilization of the transformation rules (2.18) - (2.20) is mandatory. As mentioned in the previous section there is a time shift in the transformation rules which have to be taken into account.

Summarizing the approximation of the Interface Jacobian subsystem representation utilizing the RLS as system identification method:

1. Utilize the MIMO RLS for approximating the parameters of the MIMO transfer function respectively difference equation.
2. Apply Gilberts Realization to transform the MIMO transfer function to a minimal state space model.
3. Transform the state space model to the desired Interface Jacobian representation.

How to choose the forgetting factor λ and the parameters N and M will be discussed in Chapter 3.

2.3.3 Quality Assessment

Due to the fact that system identification always contains errors the assessement of the quality of the identified Interface Jacobians is essential. Initially the reasons for insufficient system approximations are discussed, the main reasons for the approximation errors of system identification can be classified in two groups. First, requirements according the subsystem and its structure and second, the persistence of excitation of the input, see for more details e.g. [17, 35].
Due to the fact that the subsystems, are black-boxes there is no information about the structure or the behavior of the subsystem available. In the following the three main reasons for approximation errors in system identification are stated:

- non-linearities in the subsystem,
- changes in the dynamics of the subsystem,
- high number of states in the subsystem, i.e. more states in the subsystem than in the surrogate model.

All these three sources of approximations errors can be handled by the fact that the system identification is time-variant and therefore the identified linear subsystems can adapt themselves in every communication step. For example, a time-invariant subsystem containing more, typically much more, states than the surrogate model is approximated by a time-variant surrogate model with fewer states and so the fewer number of states in the surrogate model is compensated by a time-variant approximation. It is clear that, a change in the dynamics of the subsystem can only be taken into account in delayed manner in the surrogate model. Therefore it is important to parametrize the system identification in such a way that the delay and the approximation error is minimal, more details about this parametrization and its influence in Chapter 3.
The second source of insufficient results of the system identification is that the input of the subsystem is not appropriately. This means that with this input the subsystem is not enough stimulated, so that the output contains enough information about the behavior of the system. To specify the requirements of the input more precisely, the term of a *persistent excited input* is introduced, the following definition and more details can be found in [17]:

Definition 1. *A signal $u[k]$ is called persistently exciting of order n if for all k there exists an integer m such that*

$$\rho_1 \boldsymbol{I}_n < \sum_{i=k}^{k+m} \boldsymbol{\phi}(i,n)\boldsymbol{\phi}(i,n)^T < \rho_2 \boldsymbol{I}_n,$$

where $\boldsymbol{I}_n$ denotes the $n \times n$ identity matrix, $0 < \rho_1 < \rho_2$ are real scalars and the regression vector is defined as

$$\boldsymbol{\phi}(i,n)^T = \begin{pmatrix} u[k-1] & u[k-2] & \dots & u[k-n] \end{pmatrix}.$$

According to [17] the number of distinct spectral lines are equivalent to the order of persistence excitation of a signal. This means that, for example a sinusoid signal containing of n different frequency components is persistently exciting of order $2n$. To determine the persistence excitation order of a signal, a frequency analysis of the signal is required, for more details about this computation see [36].
There are different reasons for insufficient persistent excited signals, first the so-called oversampling, this means that the sampling rate, in the case of co-simulation, the communication step size, is too small. At first this sounds not logic because typically the smaller the sampling rate the better the signal is displayed, but for the case of system identification it is important that the utilized samples hold enough dynamic components and therefore the step size should not be too small. How to compute and determine the ideal step size for system identification is explained in detail in [37]. The second reason is that the signal may be in a phase where there is not much dynamic in the signal, e.g. an idle phase of an engine.

Whether the MOESP or the RLS method is utilized the same quality assessment approach can be utilized, because it is based on the Interface Jacobian subsystem description in (2.15). The quality assessement is implemented directly after the system identification, that means before the three main steps of the Model-based Pre-Step Stabilization Method are executed, more about the workflow of the coupling method in general in Section 2.4. The quality assessement is based on the difference of the value $\boldsymbol{y}^k$, simulated by the subsystem itself, and the value

$$\overline{\boldsymbol{y}}^k = \frac{\partial \boldsymbol{S}_d^k}{\partial \boldsymbol{y}^k}\boldsymbol{y}^{k-1} + \frac{\partial \boldsymbol{S}_d^k}{\partial \boldsymbol{u}^k}\boldsymbol{u}^{k-1} + \frac{\partial \boldsymbol{S}_d^k}{\partial \boldsymbol{u}^{k+1}}\boldsymbol{u}^k,$$

which represents the output of the linear, identified subsystem $\boldsymbol{S}_d$. It should be emphasized that $\boldsymbol{y}^k$ and $\overline{\boldsymbol{y}}^k$ are computed with the same inputs $\boldsymbol{u}^k$ and $\boldsymbol{u}^{k+1}$. Therefore the difference between $\boldsymbol{y}^k$ and $\overline{\boldsymbol{y}}^k$ is exclusively due to the error of the system identification method, the MOESP or RLS method. The measurement of the difference between $\boldsymbol{y}^k$ and $\overline{\boldsymbol{y}}^k$ is essential to grade the quality of the system identification, therefore the following normalized error measure is introduced

$$\epsilon_{SI,rel}^k := \frac{1}{n}\sum_{i=1}^{n} \frac{|\boldsymbol{y}^k[i] - \overline{\boldsymbol{y}}^k[i]|}{\max\limits_{j=1,\ldots,k}(\boldsymbol{y}^j[i]) - \min\limits_{j=1,\ldots,k}(\boldsymbol{y}^j[i]) + \epsilon}, \tag{2.92}$$

where n denotes the number of scalar outputs of a subsystem and $\boldsymbol{y}^k[i]$ denotes the i-th entry of the output at the coupling point T^k. The idea of definition of $\epsilon_{SI,rel}^k$ above is to normalize the absolute error of every scalar output. This normalization is done by the peak-to-peak value or range of the past samples of the scalar output, determined by the term $\max\limits_{j=1,\ldots,k}(\boldsymbol{y}^j[i]) - \min\limits_{j=1,\ldots,k}(\boldsymbol{y}^j[i])$. The additional term ϵ is added to the range value for the case of constant or nearly constant signals, because so the singular case $\max\limits_{j=1,\ldots,k}(\boldsymbol{y}^j[i]) - \min\limits_{j=1,\ldots,k}(\boldsymbol{y}^j[i]) \approx 0$ can be avoided. Choosing ϵ very small, around $\epsilon \approx 10^{-12}$, is large enough to avoid the singular case and it does not influence the error measurement $\epsilon_{SI,rel}^k$ significantly. The benefit of $\epsilon_{SI,rel}^k$ is that this error measure is invariant against scaling or adding an offset to a signal, which will be the case for example if the units of a signal are changed from meter to millimeter or from degree Celsius to Kelvin. Therefore it is sensible to determine the mean value of different scalar signals in (2.92). This error measure can be utilized for an online strategy deciding how to determine the input in optimal manner, more details are denoted in Section 2.4.2.

2.4 Implementation Aspects

The following section will deal with aspects about the implementation of the Model-based Pre-Step Stabilization Method, in focus are the workflow of the coupling method, a tool for an online determination how the input is computed, the learning phase of the system identification methods and the so-called switch phase.

The structure of a co-simulation utilizing the Model-based Pre-Step Stabilization Method for the coupling of N subsystems is depicted in Figure 2.4, for the case of approximated Interface Jacobians. The term SI denotes the system identification method, the RLS or the MOESP method, afterwards the quality assessement denoted

as QA is executed. Based on this information the Error-based Phase Check and the Learning Phase Check is performed. Additionally all N output vectors $\boldsymbol{y}_i, i = 1, \ldots, N$ are merged together to one global output vector $\boldsymbol{y}$, which is utilized in the main steps. There are three different options of the main steps as depicted by dotted lines in Figure 2.4, it should be emphasized, that based on the Learning Phase Check and the Error-based Phase Check, only one of the three options is carried out in a communication step, but this option can differ for any communication step. The three options are:
(1) the model-based coupling method consisting of the classical three main steps, the Error Differential Equation, the Model-based Extrapolation and the Input Optimization;
(2) the modified model-based coupling approach, which differs from the model-based coupling approach by skipping the Input Optimization;
(3) the signal-based coupling approach, typically ZOH or FOH coupling, there are no Interface Jacobians required.
By the execution of the main steps the global input vector $\boldsymbol{u}$ is determined. This vector is divided into the appropriate subsystem inputs $\boldsymbol{u}_i, i = 1, \ldots, N$. These are sent to the subsystems and there a step is performed and the next communication step starts from the beginning. It should be mentioned that in general the in- and outputs of the subsystem are vectors from different size.

2.4.1 Workflow of the Coupling Method

In the following the procedure or the workflow of the Model-based Pre-Step Stabilization Method is discussed, the details of the coupling method are described in Section 2.2. The focus will be on the order of the execution of the computation steps and on highlighting the dependencies between each steps. It should be mentioned that the following section is valid for a discrete-time and continuous time-subsystem representation. The workflow in Figure 2.5 shows the procedure of the Model-based Pre-Step Stabilization Method for one communication step. It starts after the simulation of all subsystems and ends with the determination of the inputs for every subsystem. As the coupling method is a model-based method, the workflow starts with the access of the required Interface Jacobians for every subsystem. The Interface Jacobians can be provided by the subsystem itself or they can be approximated, for practical examples typically an approximation is unavoidable. Therefore one can decide whether the RLS or the MOESP method should be utilized as system identification method, both will result in a discrete-time representation.
The Interface Jacobians of every subsystem are assembled to the global Interface Jacobians, representing the behavior of the whole co-simulation. It should be emphasized that due to these global matrices the mutual coupling effects and influence between the subsystems can be taken into account, this is a crucial part of the improved coupling performance of the introduced method. The global Interface Jacobians are required for the computation of the global main steps, the approximation of the monolithic outputs and the model-based extrapolation of these outputs over the next communication step.
The next step is to solve the global Error Differential Equation, to approximate the

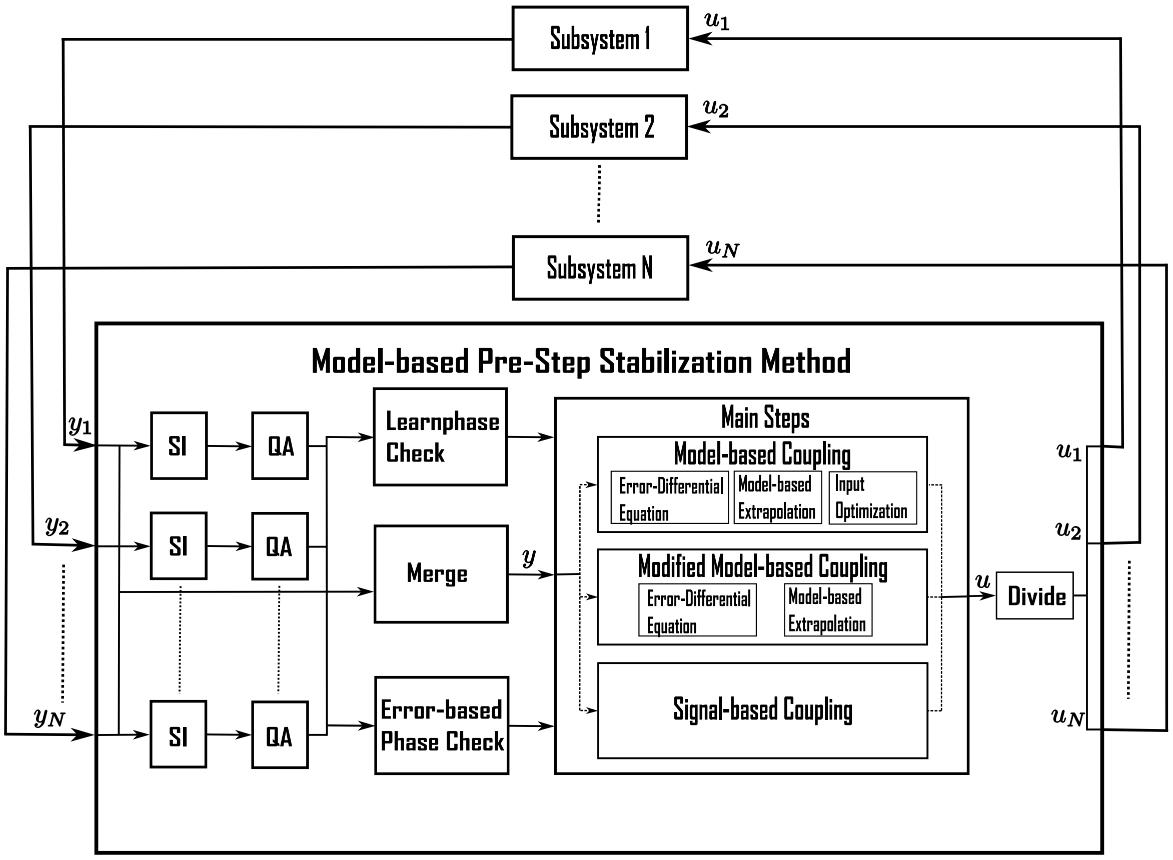


Figure 2.4: *Structure of a co-simulation with multiple subsystems coupled via the Model-based Pre-Step Stabilization Method*

monolithic output of every subsystem for the last communication step. Additional to the global Interface Jacobians, the extrapolation error, previously denoted as $\boldsymbol{\epsilon}$, of the last communication step is required for the computation of the monolithic output. The next step is to extrapolate the approximated monolithic output over the next communication step. This extrapolation is based on the global Interface Jacobians and therefore the mutual coupling between the different subsystems is considered. The monolithic outputs are the initial values for the model-based extrapolation. The last step is to determine the inputs for every subsystem, here it is important to mention that this computation is local, this means it can be computed for every subsystem independent from the others. Only the Interface Jacobians of each subsystem and not the global ones are required. After the optimization of the inputs for every subsystem, the inputs are sent to each subsystem and then all subsystems are able to simulate the next step, after that the worklflow starts from the beginning.

2.4.2 Error-based Phase Check

The quality assessment described in Section 2.3.3 is the basis of the Error-based Phase Check, where the decision is made how the input for the next communication step is

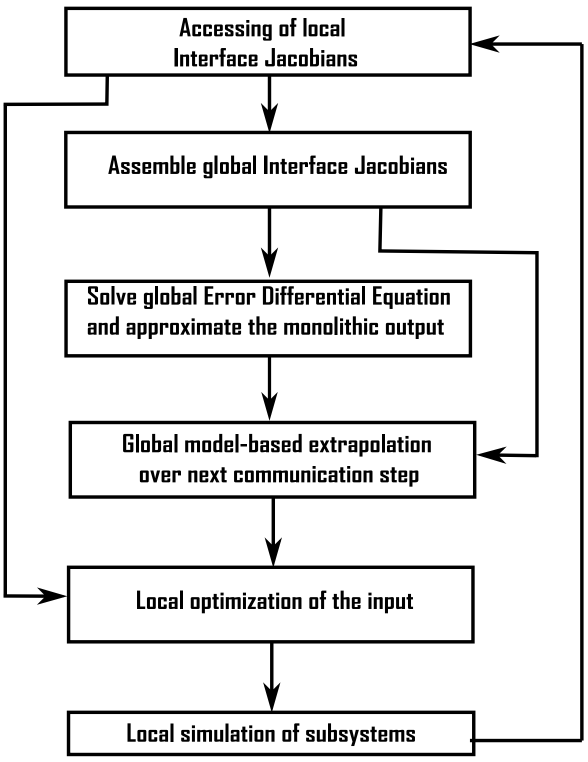

Accessing of local Interface Jacobians
Assemble global Interface Jacobians
Solve global Error Differential Equation and approximate the monolithic output
Global model-based extrapolation over next communication step
Local optimization of the input
Local simulation of subsystems

Figure 2.5: *Workflow of the Model-based Pre-Step Stabilization Method*

computed. As depicted in Figure 2.6, the relative errors $\epsilon^i_{SI,rel}$ of the system identification are computed for every subsystem $\boldsymbol{S}_i$ and then they are summarized in the global error measure $\epsilon_{SI,global}$, defined as

$$\epsilon_{SI,global} := \sqrt{\sum_i (\epsilon^i_{SI,rel})^2}, \tag{2.93}$$

which represents the Euclidean norm, of the vector containing all $\epsilon^i_{SI,rel}$. Based on $\epsilon_{SI,global}$ there are three different strategies for the determination of the input for the next communication step:
(1) $\epsilon_{SI,global} \geqslant th_{SBC}$: This is the case where the system identification failed, because the error $\epsilon_{SI,global}$ is to high to use the identified subsystem models, therefore a classical signal-based coupling method (SBC) is chosen for this communication step, i.e. ZOH- or FOH-extrapolation of the input. Due to the definition of $\epsilon_{SI,global}$ in (2.93) it is obvious that a high approximation error in just one subsystem can be the reason that $\epsilon_{SI,global} \geqslant th_{SBC}$ holds. This is necessary due to the mutual coupling and therefore possible high influence between the subsystems.
(2) $th_{SBC} > \epsilon_{SI,global} \geqslant th_{mod.}$: This represents that the system identification results are not as bad as in the previous case, but still not accurate enough to utilize the standard Model-based Pre-Step Stabilization Method. Therefore a modification of the coupling method is used, which should be less sensitive to approximation errors, according [23]. The difference to the classical coupling method is, that instead of the input optimization, the extrapolated monolithic output from Step 2, the model-based extrapolation, is directly utilized as the input for the next communication step. This

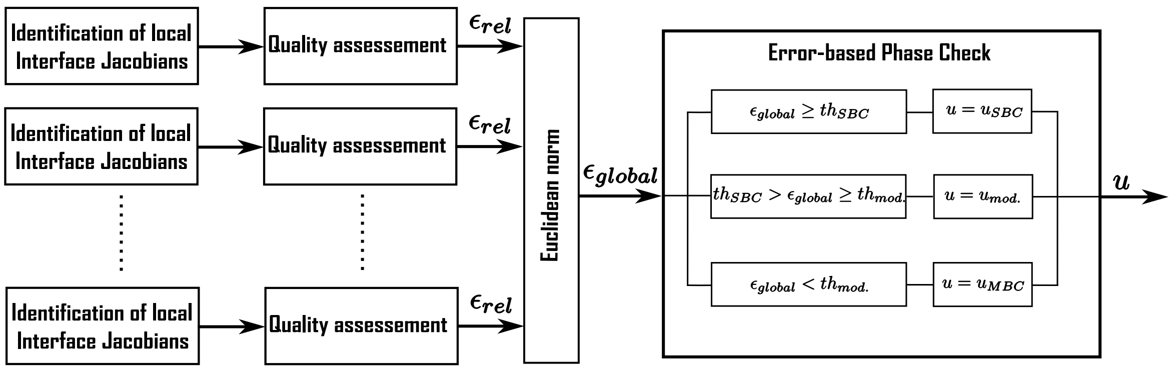

Identification of local Interface Jacobians
Quality assessement
ϵ_{rel}
Identification of local Interface Jacobians
Quality assessement
ϵ_{rel}
Identification of local Interface Jacobians
Quality assessement
ϵ_{rel}
Euclidean norm
ϵ_{global}
Error-based Phase Check
$\epsilon_{global} \geq th_{SBC}$
$u = u_{SBC}$
$th_{SBC} > \epsilon_{global} \geq th_{mod.}$
$u = u_{mod.}$
$\epsilon_{global} < th_{mod.}$
$u = u_{MBC}$
u

Figure 2.6: *Workflow of the SI-based Input Check*

means

$$\boldsymbol{u} = \boldsymbol{L}\hat{\boldsymbol{y}},$$

where $\boldsymbol{u}$ and $\hat{\boldsymbol{y}}$ denote the vector containing the appropriate quantities of all subsystems. This modified version holds for the continuous-time and discrete-time version of the coupling method.
(3) $\epsilon_{SI,global} < th_{mod.}$: If the approximation error for all subsystems is sufficiently small, the Model-based Pre-Step Stabilization Method is utilized for the coupling of every subsystem.
It is important to emphasize the necessity that all $\epsilon^{i}_{SI,rel}$ have to be sufficiently small to utilize the Model-based Pre-Step Stabilization Method for the classical and the modified version. Because even though just one subsystem might be badly approximated, this subsystem may have an influence to all others due to the global structure of Step 1 and Step 2 of the coupling method. How to choose the important thresholds th_{SBC} and $th_{mod.}$ will be further discussed and analyzed in Chapter 3, it is obvious that $th_{SBC} \geqslant th_{mod.}$ should hold. For the case of $th_{SBC} = th_{mod.}$ the modified version of the coupling method will never be utilized.

2.4.3 Learning Phase

A learning phase is required because the system identification methods, i.e. the RLS and the MOESP method, need some time to gather enough samples to learn the behavior of the system. In this time the signals are coupled in signal-based manner, because no information about the Interface Jacobians is available. The actual length of this phase depends highly on the example and the required accuracy of the approximated Interface Jacobians. The time should be chosen in such a way that the gathered data contain enough information about the subsystem. The rule of thumb for choosing the learning phase is, that it should be as small as possible but as long as required. Due to the fact that the Model-based Pre-Step Stabilization Method is mainly applied to systems where signal-based coupling generates insufficient results, the learning phase needs to be as small as possible, so that the simulation does not diverge or show odd results in that time.

For the RLS method it is possible to specify the initial parameters $\boldsymbol{p}$, i.e. the initial Interface Jacobians, and therefore it should be possible to keep the learning phase to a minimum. Such initial parameters can be determined by previous runs or from theoretical considerations about the subsystems. The influence of the length of the learning phase will be analyzed in Chapter 3.

2.4.4 Switch Phase

After the learning phase the coupling method is switched to its regular working mode. Due to the fact that this switch can stimulate some unwanted effects if it is instantaneously, i.e. from one communication step to the next, this switching is smoothed with a hyperbolicus tangent function, a similar approach can be found in [38]. The time how long the switching process should last is defined in the user parameter *timeSwitching*. During the switching process the inputs $\boldsymbol{u}_{SBC}$, computed by the signal-based coupling approach, and $\boldsymbol{u}_{MBC}$, computed by the Model-based Pre-Step Stabilization Method, are determined and they are merged together to $\boldsymbol{u}$, which are the inputs sent to the subsystems. The merged inputs $\boldsymbol{u}$ are determined as

$$\boldsymbol{u} = \text{fac}_{SBC}\boldsymbol{u}_{SBC} + \text{fac}_{MBC}\boldsymbol{u}_{MBC}.$$

Here the quantities fac_{SBC} and fac_{MBC} are determined over the hyperbolicus tangent approach as

$$\begin{aligned} \text{fac}_{MBC}(t) &= \frac{1}{2}\left[\tanh\left(-5 + \frac{10}{timeSwichting}(t - T_{StartSwitching})\right) + 1\right], \\ \text{fac}_{SBC} &= 1 - \text{fac}_{MBC}. \end{aligned}$$

In Figure 2.7 the progress of fac_{SBC} and fac_{MBC} is depicted for different values of timeSwitching, there it is obvious the greater timeSwitching is, the smoother the transition from the learning phase to the standard coupling phase is. For displaying an instantaneously switch one should choose timeSwitching as the communication step size. The influence and the effects of timeSwitching on the co-simulation results will be analyzed in Chapter 3.

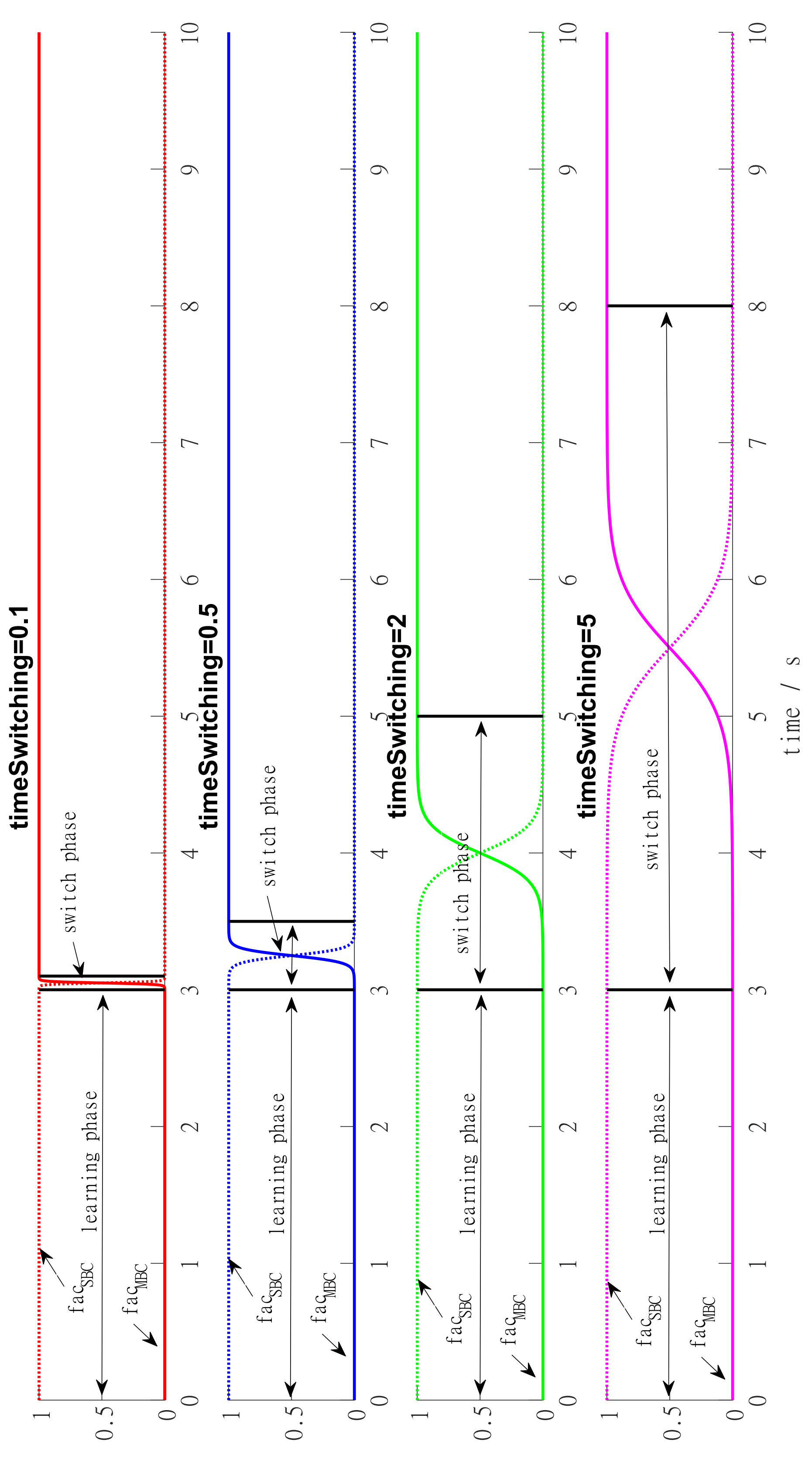


Figure 2.7: *The switching process for four different values of timeSwitching*

3

Analysis of the Method

The analysis of a method is required to give an insight in the behavior and the effects of the method. The analysis is indispensable to understand the method and establish trust in it. Especially in a multidisciplinary field with a lot of different influences like co-simulation, and a complex method like the Model-based Pre-Step Stabilization Method a sensitivity analysis is mandatory because there the question of how to parametrize the coupling method will be answered. The benefit of the coupling method in terms of stability will be shown by comparing it to other state of the art coupling methods. Therefore a stability analysis according the stiffness of the subsystems and communication step size will be performed and exponential and BIBO-stability will be verified, too. Additionally an accuracy analysis will be carried out to show the improved performance of the Model-based Pre-Step Stabilization Method.

The herein focused sensitivity, stability and accuracy analyses will be performed utilizing the dual mass oscillator example, which is a classical benchmark example of co-simulation, see e.g. [5, 6, 20, 22, 23]. The physical parameters, the stiffness and damping coefficients, are chosen appropriately so that the co-simulation is a stiff one, therefore the example is well designed for analyzing the Model-based Pre-Step Stabilization Method. The sensitivity analysis is a design of experiment(DoE) to examine the influence of various parameters of the coupling method to the co-simulation results, leading to guidelines for the parametrization of the coupling method for the case of approximated and exact Interface Jacobians. The stability analysis will show the improved stability of the coupling method according the damping and stiffness of the dual mass oscillator and the communication step size of the co-simulation. For the accuracy analysis of the coupling method the damping and stiffness of the dual mass oscillator is varied together with the communication step size of the co-simulation.

3.1 Dual Mass Oscillator

The dual mass oscillator is a linear and time-invariant benchmark example for co-simulation for force/displacement coupling, see for more details [29]. As it is depicted in Figure 3.1 it consists of two masses m_1 and m_2, which are fixed to a wall by a spring

and damper each, too. The coupling between the two masses is realized with a spring and a damper. The dual mass oscillator is separated into two subsystems S_1 and S_2, the subsystem S_1 consists of the mass m_1 and a spring, with the spring constant c_1, and a damper, with a damping factor of d_1, attached to the wall. The outputs of this subsystem are the position and the velocity of the mass. In the subsystem S_2 the mass m_2 is attached to the wall by a spring, with the spring constant c_2, and a damper, represented by the damping factor d_2. Also represented in S_2 is a spring, described by the spring constant c_k and a damper, with the damping factor d_k, which couples m_1 and m_2, the force between this two masses is the output of S_2. Both subsystems are stated in the following continuous-time state space representations:
Subsystem S_1:

$$\dot{\boldsymbol{x}}_1 = \boldsymbol{A}_1\boldsymbol{x}_1 + \boldsymbol{B}_1 u_1, \tag{3.1}$$
$$\boldsymbol{y}_1 = \boldsymbol{C}_1\boldsymbol{x}_1 \tag{3.2}$$

Subsystem S_2:

$$\dot{\boldsymbol{x}}_2 = \boldsymbol{A}_2\boldsymbol{x}_2 + \boldsymbol{B}_2\boldsymbol{u}_2, \tag{3.3}$$
$$y_2 = \boldsymbol{C}_2\boldsymbol{x}_2 + \boldsymbol{D}_2\boldsymbol{u}_2 \tag{3.4}$$

where

$$\boldsymbol{x}_1 = \begin{pmatrix} x_1 \\ \dot{x}_1 \end{pmatrix},\ \boldsymbol{A}_1 = \begin{pmatrix} 0 & 1 \\ -\frac{c_1}{m_1} & -\frac{d_1}{m_1} \end{pmatrix},\ \boldsymbol{B}_1 = \begin{pmatrix} 0 \\ \frac{1}{m_1} \end{pmatrix},\ \boldsymbol{C}_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$
$$\boldsymbol{x}_2 = \begin{pmatrix} x_2 \\ \dot{x}_2 \end{pmatrix},\ \boldsymbol{A}_2 = \begin{pmatrix} 0 & 1 \\ -\frac{c_2+c_k}{m_2} & -\frac{d_2+d_k}{m_2} \end{pmatrix},\ \boldsymbol{B}_2 = \begin{pmatrix} 0 & 0 \\ \frac{c_k}{m_2} & \frac{d_k}{m_2} \end{pmatrix},$$
$$\boldsymbol{C}_2 = \begin{pmatrix} c_k & d_k \end{pmatrix},\ \boldsymbol{D}_2 = \begin{pmatrix} -c_k & -d_k \end{pmatrix}.$$

The position and velocity of the mass m_1 respectively m_2 are described by the states x_1 and $\dot{x}_1$ respectively x_2 and $\dot{x}_2$. The chosen parameters are:

$$m_1 = 10 \text{ kg},\ m_2 = 10 \text{ kg},\ c_1 = 1e6 \text{ Nm}^{-1},$$
$$c_2 = 1e7 \text{ Nm}^{-1}, d_1 = 1 \text{ Nsm}^{-1},\ d_2 = 2 \text{ Nsm}^{-1}.$$

The initial conditions are set to:

$$x_1^0 = 0.1, \quad \dot{x}_1^0 = 0, \quad x_2^0 = 0, \quad \dot{x}_2^0 = 0. \tag{3.5}$$

The values for the the spring and damper realizing the coupling are set to

$$c_k = 1e5 \text{ Nm}^{-1},\ d_k = 1e3 \text{ Nsm}^{-1},$$

for the sensitivity analysis, for the stability analysis and for the accuracy analysis these parameters vary. If the communication step size is not varied for analysis purposes, it is fixed $\Delta T = 1\ ms$ and the subsystems are solved by the explicit Euler method with a constant step size of $\delta t = 0.01\ ms$. The scheduling of the co-simulation is parallel and therefore the inputs of both subsystems S_1 and S_2 have to be extrapolated.

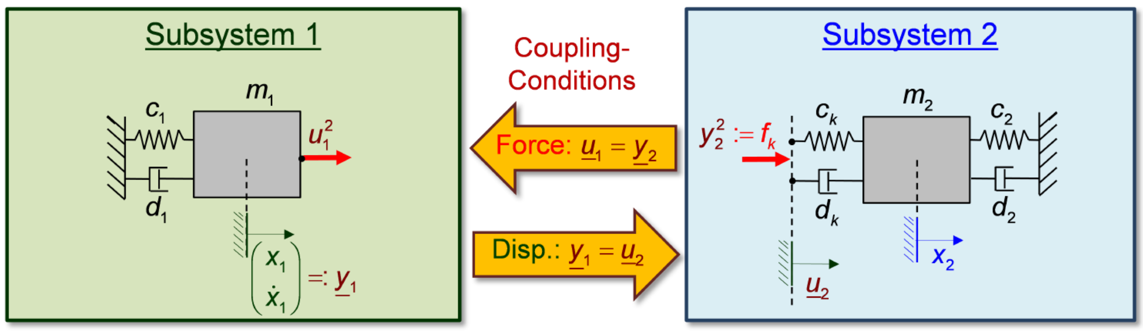

Subsystem 1
m_1
c_1
d_1
u_1^2
$\begin{pmatrix} x_1 \\ \dot{x}_1 \end{pmatrix} =: \underline{y}_1$
Coupling-Conditions
Force: $\underline{u}_1 = \underline{y}_2$
Disp.: $\underline{y}_1 = \underline{u}_2$
Subsystem 2
$y_2^2 := f_k$
c_k
m_2
c_2
d_k
d_2
x_2
$\underline{u}_2$

Figure 3.1: *Dual mass oscillator separated into two subsystems*

c	$c_{k_{max}}$	$c_{k_{min}}$	n_c
5e5 Nm^{-1}	2.5e5 Nm^{-1}	2.5e4 Nm^{-1}	1.5
d	$d_{k_{max}}$	$d_{k_{min}}$	n_d
1e3 Nsm^{-1}	3e3 Nsm^{-1}	5e2 Nsm^{-1}	1.5

Table 3.1: *Numerical values for the non-linear characterizations of the spring and damper coefficients*

3.1.1 Non-linear Dual Mass Oscillator

The analysis for the case of approximated Interface Jacobians is of minor interest for linear time-invariant subsystems, as the above described dual mass oscillator is. Therefore the dual mass oscillator is adapted by a non-linear spring and damper between the two masses m_1 and m_2. This means the constant spring c_k and damper d_k coefficients are replaced by time-variant quantities, represented by the characteristics in Figure 3.2 and the equations

$$c_k := \max\left(\min\left(c|x_2[1] - u_2[1]|^{n_c - 1}, c_{k_{max}}\right), c_{k_{min}}\right), \tag{3.6}$$
$$d_k := \max\left(\min\left(d|x_2[2] - u_2[2]|^{n_d - 1}, d_{k_{max}}\right), d_{k_{min}}\right), \tag{3.7}$$

the chosen values are stated in Table 3.1. The term $x_2[1] - u_2[1]$ describes the displacement of the coupling spring from their idle state, this means the difference of the positions of m_1 and m_2. Analog represents $x_2[2] - u_2[2]$ the difference between the velocities of the two masses, which is the basis for the time-variant damper coefficient d_k. In Table 3.1 the term $n_c = n_d = 1.5$ is stated which means that for a higher difference in the position respectively velocity of the two masses the spring respectively damper coefficient increases until their maximum, such spring and dampers are called progressive[1]. These non-linear adaptions to the dual mass oscillator makes the online adaption of the identified subsystems necessary and therefore this case is better suited to analyze the Model-based Pre-Step Stabilization utilizing approximated Interface Jacobians.

[1]Progressive springs and dampers are for exampled used in suspension springs.

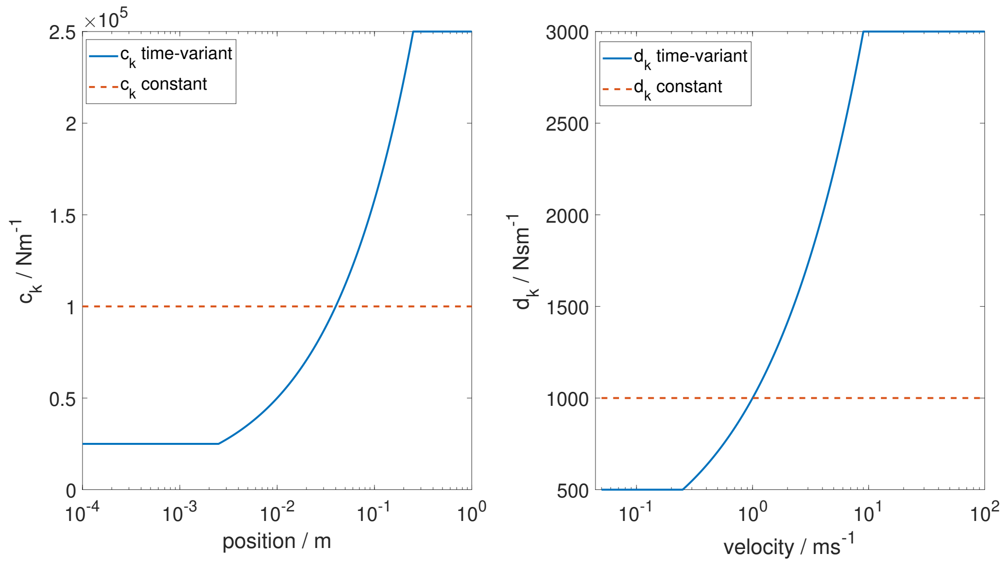


Figure 3.2: *Comparison of time-variant and constant spring and damper coefficients*

3.1.2 Extension with External Force

A dual mass oscillator without any external force is for the case of approximated Interface Jacobians from minor interest, because the input is not sufficiently exciting and the simulation time can only be chosen quite small. Therefore an external force is added to the dual mass oscillator leading to an extended subsystem S_1, where the external force is added in the following way

$$\dot{\boldsymbol{x}}_1 = \boldsymbol{A}_1\boldsymbol{x}_1 + \boldsymbol{B}_1 u_1 + \begin{pmatrix} 0 \\ \frac{1}{m_1} \end{pmatrix} F_{ex.}(t). \tag{3.8}$$

The external force $F_{ex.}$ is defined as

$$F_{ex.}(t) = \begin{cases} 5e5\sin(10t) & t = 0.05\,k \text{ for } k \in \mathbb{N} \\ 0 & \text{otherwise} \end{cases}$$

Due to the impulses of $F_{ex.}$ the input is persistently exciting and therefore the dual mass oscillator is better suited to analyze the coupling method for the case of approximated the Interface Jacobians.

3.2 Sensitivity Analysis

The understanding of how a system or a process works is crucial in many fields e.g. engineering, business or medicine. Especially of interest is the influence of parameters to the outputs of a system, this means how sensitive the system is according to parameter variations. The experimental approach is to study the changes in the output while changing the input parameters, this is the basic idea of a design of experiment,

for more details see [39].
It is quite obvious that if the adaption of a certain parameter results in a significant change in the output, this parameter is of high importance for the system. If otherwise, the outputs are nearly constant while varying some other parameter, this one is not significant for the system. To capture those terms more precisely, the so-called effects are introduced, the computation of those effects is the main part of the so-called analysis of variance. By changing one parameter and keeping all others steady the so-called main effect of this parameter can be determined. To investigate if a combination of certain parameters are of mayor or minor importance for the system, the so-called interaction effects are computed. To determine all possible interaction effects between the parameters a full factorial experiment has to be carried out. There all different parameter combinations are included, leading to a factorial number of experiments which have to be run. For example if the system consists of three analyzed parameters where the first parameter has three and the others two levels, the number of needed experiments is $3 \cdot 2^2 \cdot n = 12 \cdot n$. Here n stands for the number of repetitions for each parameter set. The repetitions are needed in case of uncontrollable influence to the system, e.g. measurement noise, natural deviation or stochastic aspects. Due to the fact that the number of experiments increases exponentially, the parameters and especially the number of levels for each parameter should be chosen carefully.
Based on the data from all simulation runs an optimal set of parameter should be determined. This will be done by ascending sorting according to the error measure of all performed runs, because an error measure of 0 would be optimal. This results in a list of indices, where the first element of the list contains the run with the lowest error and the last one the run with the highest error. The influence of the parameters to the error measure is then determined by plotting all parameters in the same order as in the list above. Therefore the parameters leading to the best results are plotted at the first position and the parameters leading to the highest error are plotted at the last position. Based on those sorted list of parameters the optimal set of parameter is determined by taking the mean value of all analyzed parameters, where the corresponding error is only 10% greater than the lowest error of all runs. This technique is taking into account, that there may be different combinations of nearly optimal parameters, therefore the mean value over a certain number of sufficiently good results is better than taking the parameters of the best run exclusively. This is especially important because the goal is to identify a general set of optimal parameters and therefore those parameters should be as robust as possible.
The structure of a design of experiment analysis contains the following steps:

1. planning
2. running
3. analyzing
4. concluding

The first step planning means, that the analyzed parameters and their levels have to be determined. There it should be kept in mind, that the number of required experiments increases exponentially, with the number of analyzed parameters and their levels. Additionally external or uncontrollable influences should be considered.

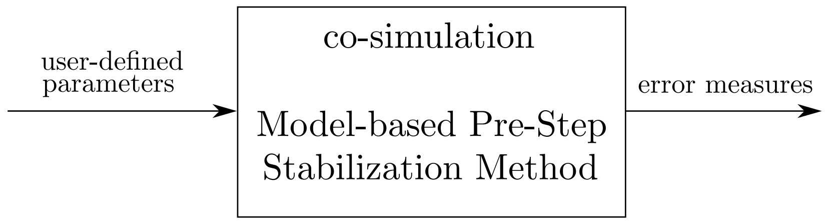


Figure 3.3: *Setup of the sensitivity analysis for the Model-based Pre-Step Stabilization Method*

The higher those influences are expected, or if they are unknown, the higher the number of repetitions should be chosen. Running the experiments is the most time consuming part of the analysis. Therefore one should make clear, that the results are stored well structured and organized, so that nothing can be mixed up or even lost. The third step is the analysis of the experiments, in the analysis of variance the effects are computed and it is determined how significant the parameters are. The ascending sorting analysis results in an optimal set of parameters. It should be pointed out that both analyses are based on the same data. The last step is to draw a conclusion based on the two analyses, which means answering the question of the significance of the parameters and which parameters are optimal.

The performed sensitivity analysis of the Model-based Pre-Step Stabilization Method will answer the question of how to parametrize the coupling method correctly and how significant the influence of the parameters are, the setup is depicted in Figure 3.3. Therefore a design of experiment analysis will be utilized for the discrete-time and continuous-time version of the coupling method. The continuous-time version will utilize exact Interface Jacobians and therefore, the focus is on analyzing the coupling method without any approximation error in the Interface Jacobians, the utilized example will be the linear time-invariant dual mass oscillator. Analyzing the influence of the approximation of the Interface Jacobians the discrete-time version of the method and the non-linear dual mass oscillator extended with an external force is chosen. Due to the fact that the analyzed system, the co-simulation utilizing the Model-based Pre-Step Stabilization Method, is a simulation, there are no real measurements included. Therefore it is completely free of any stochastic behavior and so there are no uncontrollable influences and therefore there are no repetitions required, i.e. $n = 1$.

3.2.1 Exact Interface Jacobians

For the case of exact Interface Jacobians the continuous-time version of the Model-based Pre-Step Stabilization Method is utilized, because if exact knowledge about the Interface Jacobians is available, it is typically given in continuous-time manner. The utilized error measure is defined as

$$\epsilon_{mono,global} := \sqrt{\sum_i (\epsilon^i_{mono,rel})^2}, \tag{3.9}$$

with

$$\epsilon^k_{mono,rel} := \frac{1}{n}\sum_{i=1}^{n} \frac{|\boldsymbol{y}^k[i] - \boldsymbol{y}^k_{mono}[i]|}{\max\limits_{j=1,\ldots,k}(\boldsymbol{y}^j_{mono}[i]) - \min\limits_{j=1,\ldots,k}(\boldsymbol{y}^j_{mono}[i]) + \epsilon}. \tag{3.10}$$

The first step of the design of the experiment is to choose the analyzed parameters and their levels. The analyzed parameters and their type are stated in Table 3.2 and will be discussed in the following.

If the boolean parameter useErrDiffEqu is true the Error Differential Equation is solved as the first step of the coupling method, all details to the Error Differential Equation can be found in Section 2.2.1. If the parameter useErrDiffEqu is set to false, the solving of the Error Differential Equation is skipped, i.e. $\boldsymbol{\delta}$ is set to $\mathbf{0}$, for the definition of $\boldsymbol{\delta}$ see (2.26).

By the parameter optionErrDiffEqu the solver of the Error Differential Equation can be chosen. The value *exact* stands for an exact solver, utilizing the matrix exponential and convolution, by this method no numerical discretization error is added. If the term *ex. euler* is chosen the classical explicit Euler method, see e.g [40], is utilized.

The number of discretization steps for solving the Error Differential Equation over one communication step can be adjusted by the parameter stepsErrDiffEqu. It is from integer type, because it stands for the number of intermediate steps for solving the ordinary differential equation in one communication step. This means if the communication step size $\Delta T = 10ms$ and stepsErrDiffEqu is set to 100, the step size of the solver of the Error Differential Equation is $\frac{10ms}{100} = 0.1ms$. The here discussed intermediate steps or values should not be mixed up with the intermediate values of the subsystems, there is no connection.

The parameters optionExtrapolation and stepsExtrapolation are analogously defined as optionErrDiffEqu and stepsErrDiffEqu, with the difference that optionExtrapolation and stepsExtrapolation define the solver and its steps for the model-based extrapolation, the second step of the Model-based Pre-Step Stabilization Method, for details see Section 2.2.1.

The third step of the coupling method, the input optimization, can be influenced by the parameter stepsDiscreteSystemMatrices. This parameter stands for the number of intermediate values for calculating the values Φ_B, Φ_C and Φ_D, see therefore (2.45) - (2.47). This means, the higher this parameter the more accurate the computation of these quantities will be.

For the integer parameters, namely stepsErrDiffEqu, stepsExtrapolation and stepsDiscreteSystemMatrices, the chosen levels are stated in Table 3.3. The levels are all chosen equally, because so it is easier to compare the results. All combinations of all six analyzed parameters will be investigated, this means this analysis of variance consists of a full factorial design of experiment. Therefore the number of runs can be easily computed via

$$\mathit{neededRuns} = 2 \cdot 2 \cdot 4 \cdot 2 \cdot 4 \cdot 4 = 512,$$

based on the number of levels of each analyzed parameter. This means that for generating the required data one has to perform 512 runs with different parameters. Due to the fact that all data is simulated there are no further unknown influences to the simulation, this means that there is no need for any repetitions, i.e. $n = 1$.

optionErrDiffEqu	stepsErrDiffEqu	optionExtrapolation
exact or *ex. euler*	integer	*exact* or *ex. euler*

stepsExtrapolation	stepsDiscreteSystemMatrices	useErrDiffEqu
integer	integer	boolean

Table 3.2: *Analyzed parameters of the design of experiment for the case of exact Interface Jacobians*

stepsErrDiffEqu	stepsExtrapolation	stepsDiscreteSystemMatrices
1	1	1
10	10	10
100	100	100
1000	1000	1000

Table 3.3: *Levels of the integer type analyzed parameters*

Analysis of Variance

The analysis of variance investigates how sensitive the Model-based Pre-Step Stabilization Method is against changes in the parameters. The following definitions and derivations are based on [39]. As an illustration of how to compute the needed quantities for the analysis of variance, assume the following example:
There are three analyzed parameters A, B and C with a levels for A, b levels for B and c levels for C. The simulated results are stored in a three dimensional array or a tensor, called y, there y_{ijk} denotes the result based on the $i-$th level of A, $j-$th level of B and $k-$th level of C. The term

$$y_{i\cdot\cdot} := \sum_{j=1}^{b}\sum_{k=1}^{c} y_{ijk}$$

denotes the total sum of all results, with the i-th level of the parameter A, analog the quantities $y_{\cdot j\cdot}, y_{\cdot\cdot k}$ are defined. Additionally one can define

$$y_{ij\cdot} := \sum_{k=1}^{c} y_{ijk},$$

which represents the sum of all results with the $i-$th level of A and the $j-$th level of B, analog are $y_{i\cdot k}$, $y_{\cdot jk}$ defined. The term

$$y_{\cdots} := \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c} y_{ijk}$$

stands for the sum of all results. Based on these quantities the required sum of squares can be computed. The impact of a single parameter A the so-called sum of squares for the main effect SS_A is defined as

$$SS_A := \frac{1}{bc}\sum_{i=1}^{a} y_{i\cdot\cdot}^2 - \frac{y_{\cdots}^2}{abc},$$

analog SS_B and SS_C can be defined. The two-parameter interaction sum of squares SS_{AB} is defined as

$$SS_{AB} = \frac{1}{c}\sum_{i=1}^{a}\sum_{j=1}^{b} y_{ij\cdot}^2 - \frac{y_{\dots}^2}{abc} - SS_A - SS_B,$$

analog the quantities SS_{AC} and SS_{BC} can be defined. The degrees of freedom DoF of a single parameter A or a combination of A and B can be computed as

$$DoF_A := a - 1$$

for a single parameter, or

$$DoF_{AB} := (a-1)\cdot(b-1)$$

for a combination of two parameters. Based on the DoF_A the mean square MS_A of SS_A can be determined by

$$MS_A := \frac{SS_A}{DoF_A}. \tag{3.11}$$

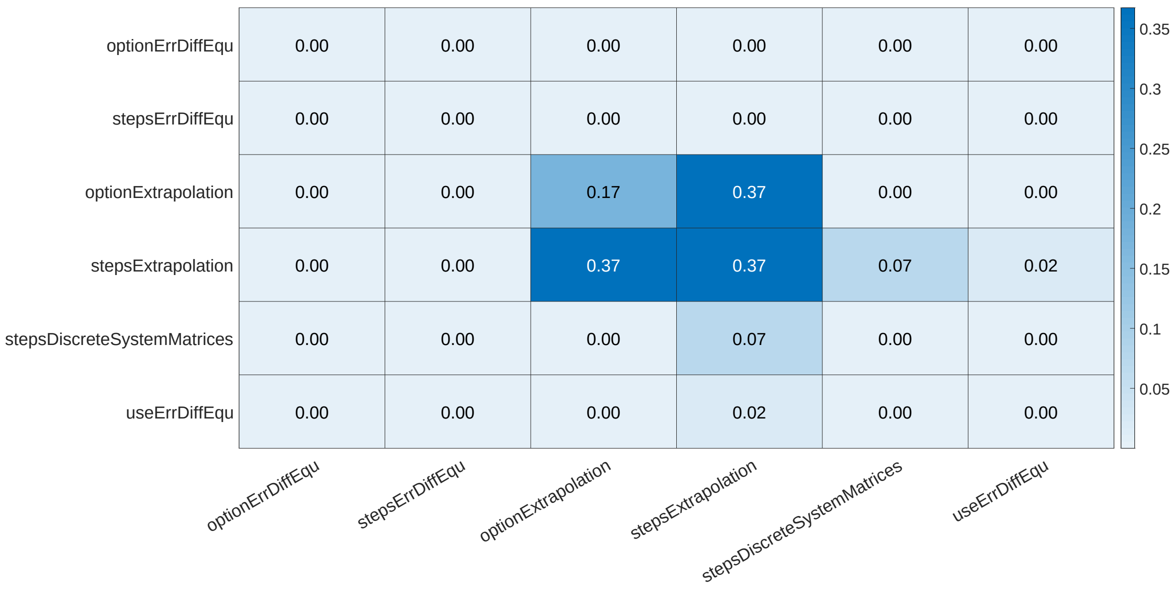
Analog the MS_{AB} for the two-factor interaction effects is defined as

$$MS_{AB} := \frac{SS_{AB}}{DoF_{AB}}. \tag{3.12}$$

This example illustrates how to compute the required mean square for the main and interaction effect. The following design of experiment will consist of more than three parameters, therefore the needed quantities have to be generalized. The results of the analysis of variance are stated in Table 3.4. Additionally these results are depicted in a heatmap in Figure 3.4, the diagonal entries of the heatmap represents the mean sum of squares for the main effects and the off-diagonal entries stand for the interaction effect of the parameters denoted at x- and y-axis. Due to the fact that the sum of squares for interaction effects are symmetric, i.e. $SS_{AB} = SS_{BA}$, the heatmap is symmetric too. At the heatmap and the corresponding table it is clearly visible that the parameters stepsExtrapolation and optionExtrapolation are the most significant ones. Also the interaction of those two parameters is highly significant. In Figure 3.5 the main effects of all six analyzed parameters are depicted, there and also in Table 3.4 and in Figure 3.4, it can be seen that the parameter optionErrDiffEqu and stepsErrDiffEqu are of nearly no significance. Both parameters are also not interacting with any other, as it is displayed in the off-diagonal entries of the heatmap and the interaction effect plots in Figure 3.6 and 3.7, as all lines are parallel or even coincident.
Of low significance are stepsDiscreteSystemMatrices and useErrDiffEqu. As depicted in Figure 3.10 and 3.11 there is an observable interaction between stepsDiscreteSystemMatrices, stepsExtrapolation and useErrDiffEqu. The main effect $MS_{stepsDiscreteSystemMatrices} = 0.07\%$ is rather small but the interaction with stepsExtrapolation is significant with $MS_{stepsDiscreteSystemMatrices-stepsExtrapolation} = 7.26\%$. Additionally the impact of stepsDiscreteSystemMatrices is shown in Figure 3.5, there

Parameters Main and Two Factor Interaction Effects	MS_X	Rate / %
optionErrDiffEqu	5.00e-12	0
stepsErrDiffEqu	7.56e-8	0
optionExtrapolation	6.26e2	17.32
stepsExtrapolation	1.27e3	36.72
stepsDiscreteSystemMatrices	2.86e1	0.07
useErrDiffEqu	1.03e1	0
optionErrDiffEqu – stepsErrDiffEqu	1.36e-12	0
optionErrDiffEqu – optionExtrapolation	4.32e-12	0
optionErrDiffEqu – stepsExtrapolation	3.00e-11	0
optionErrDiffEqu – stepsDiscreteSystemMatrices	2.32e-11	0
optionErrDiffEqu – useErrDiffEqu	0	0
stepsErrDiffEqu – optionExtrapolation	2.73e-9	0
stepsErrDiffEqu – stepsExtrapolation	3.55e-9	0
stepsErrDiffEqu – stepsDiscreteSystemMatrices	1.56e-8	0
stepsErrDiffEqu – useErrDiffEqu	7.56e-8	0
optionExtrapolation – stepsExtrapolation	1.27e3	36.72
optionExtrapolation – stepsDiscreteSystemMatrices	3.12e1	0.01
optionExtrapolation – useErrDiffEqu	1.04e1	0.26
stepsExtrapolation – stepsDiscreteSystemMatrices	6.64e2	7.26
stepsExtrapolation – useErrDiffEqu	3.96e1	1.87
stepsDiscreteSystemMatrices – useErrDiffEqu	3.09e1	0

Table 3.4: *Results of the Analysis of Variance for the case of exact Interface Jacobians*



	optionErrDiffEqu	stepsErrDiffEqu	optionExtrapolation	stepsExtrapolation	stepsDiscreteSystemMatrices	useErrDiffEqu
optionErrDiffEqu	0.00	0.00	0.00	0.00	0.00	0.00
stepsErrDiffEqu	0.00	0.00	0.00	0.00	0.00	0.00
optionExtrapolation	0.00	0.00	0.17	0.37	0.00	0.00
stepsExtrapolation	0.00	0.00	0.37	0.37	0.07	0.02
stepsDiscreteSystemMatrices	0.00	0.00	0.00	0.07	0.00	0.00
useErrDiffEqu	0.00	0.00	0.00	0.02	0.00	0.00

Figure 3.4: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for exact Interface Jacobians - Sum of Squares of Main and Interaction Effects*

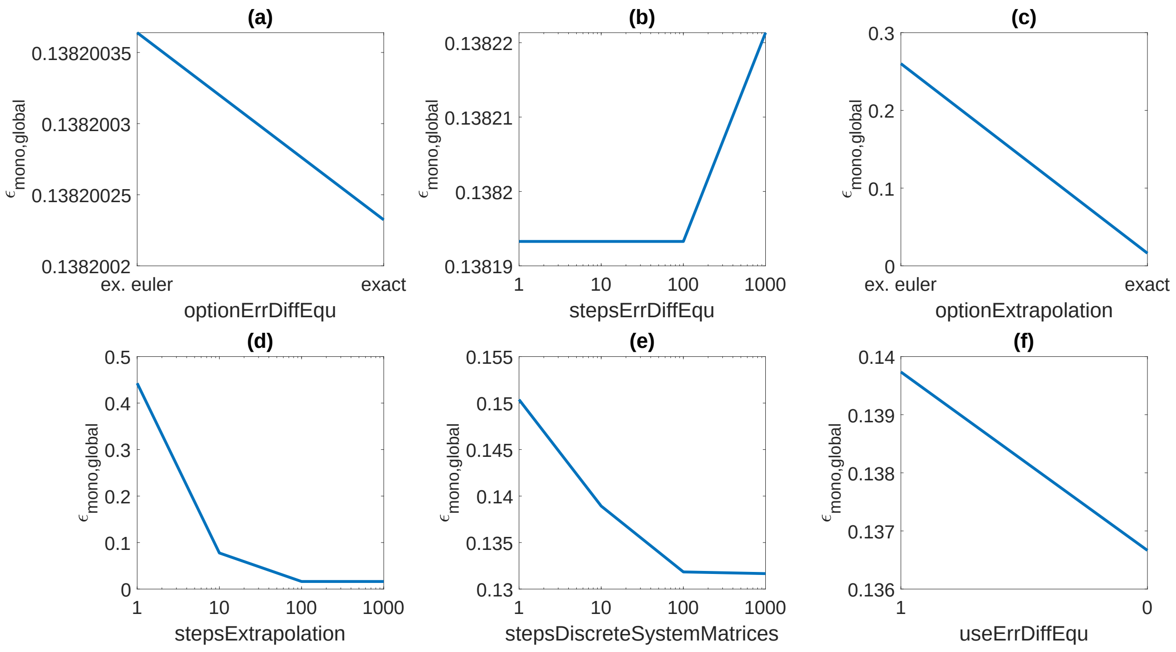


Figure 3.5: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for exact Interface Jacobians - Main Effects*

it is obvious that with a growing value for stepsDiscreteSystemMatrices the error $\epsilon_{mono,global}$ is decreasing, this influence is not as high as for the parameter stepsExtrapolation, but it can be observed.
This leads to the conclusion that from the three main steps of the analyzed coupling method the second step, the model-based extrapolation is the most significant. Especially of interest is that, utilizing the Error Differential Equation or not, i.e. setting useErrDiffEqu to *true* or *false*, does not, impact the results much, because the effect $MS_{useErrDiffEqu} = 0.0026\%$ is roughly greater than zero and also the corresponding interaction effects are all nearly zero. From this point of view it is reasonable, that also stepsErrDiffEqu and optionErrDiffEqu are of nearly no significance, because they only effect how accurate the Error Differential Equation is solved.
Therefore the focus in the following is, to have a closer look on the significant parameters stepsExtrapolation and the optionExtrapolation. In Figure 3.5 one can see, that for optionExtrapolation is set to *exact*, every run is sufficiently accurate. The same effect can be observed by setting stepsExtrapolation to 100 or 1000. The interaction between those two parameters is depicted in Figure 3.8 and 3.9. There it is obvious that with increasing stepsExtrapolation there is no difference between the results utilizing optionExtrapolation set to *ex. euler* or *exact*. This is in contrast to the results where stepsExtrapolation is set to 1 or 10, there is an obvious gap between the results based on the two different values of optionExtrapolation. This behaviour is reasonable because the lower the step size for the explicit Euler method is, the more accurate the solution is. Additionally one can see that for the case of optionExtrapolation is set to *exact* the number of stepsExtrapolation has no influence. This is explainable because regardless of the number of intermediate steps, an exact solver will always generates

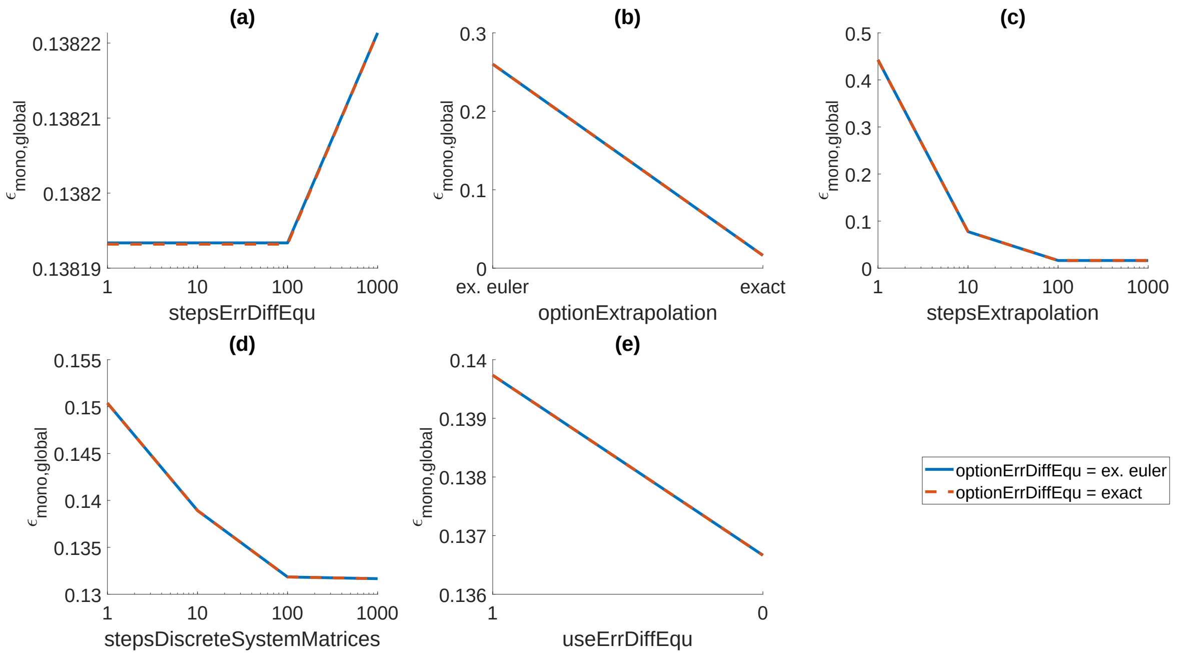


Figure 3.6: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for exact Interface Jacobians - Interaction Effect for optErrDiffEqu*

the same result. Therefore one can say, use optionExtrapolation as *exact* in combination with stepsExtrapolation is set to 1 or set optionExtrapolation to *ex. euler* and stepsExtrapolation to 1000. This means there is a strong interaction between those two parameters and they are also, independent from each other, significant for the coupling method.

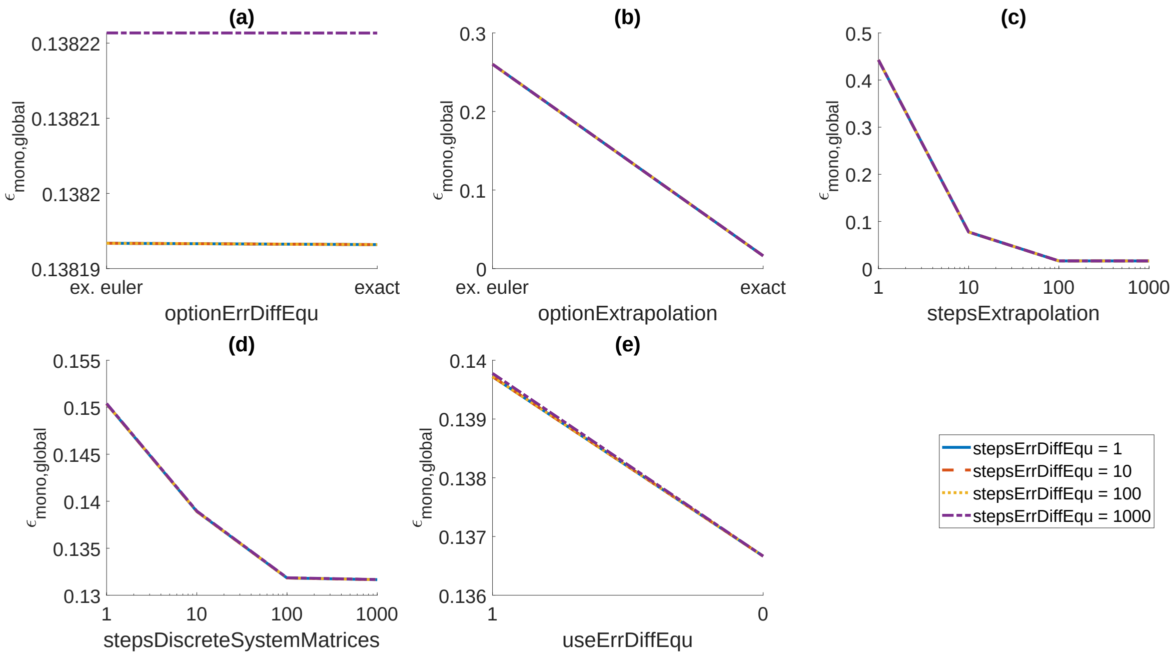


Figure 3.7: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for exact Interface Jacobians - Interaction Effect for stepsErrDiffEqu*

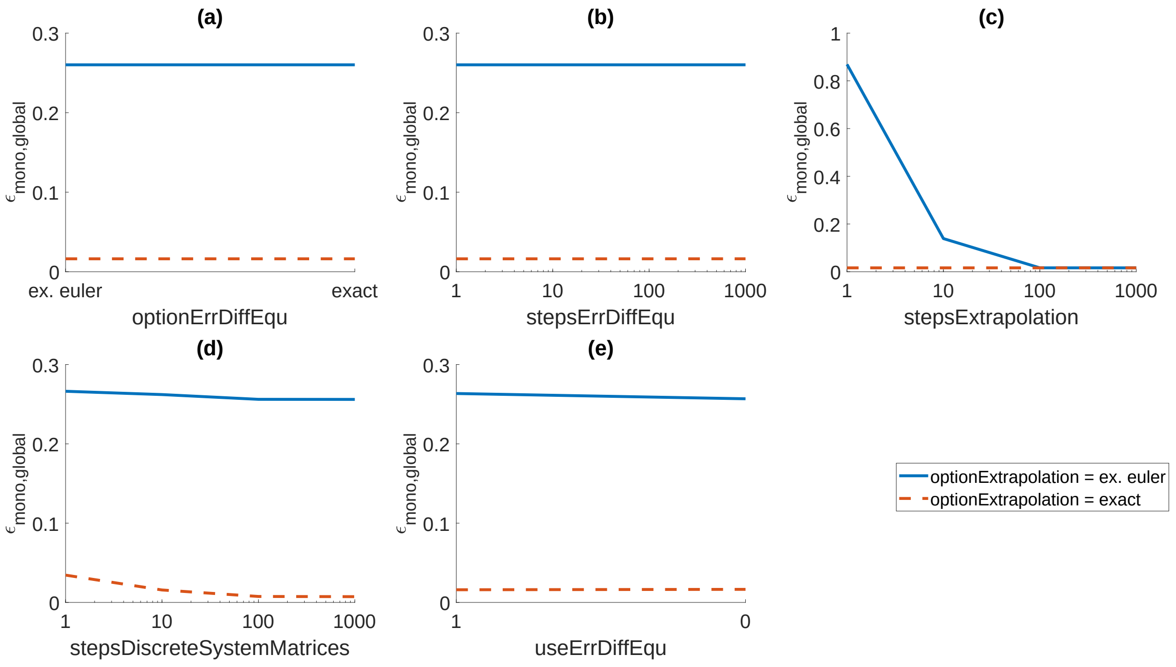


Figure 3.8: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for exact Interface Jacobians - Interaction Effect for optExtrapolation*

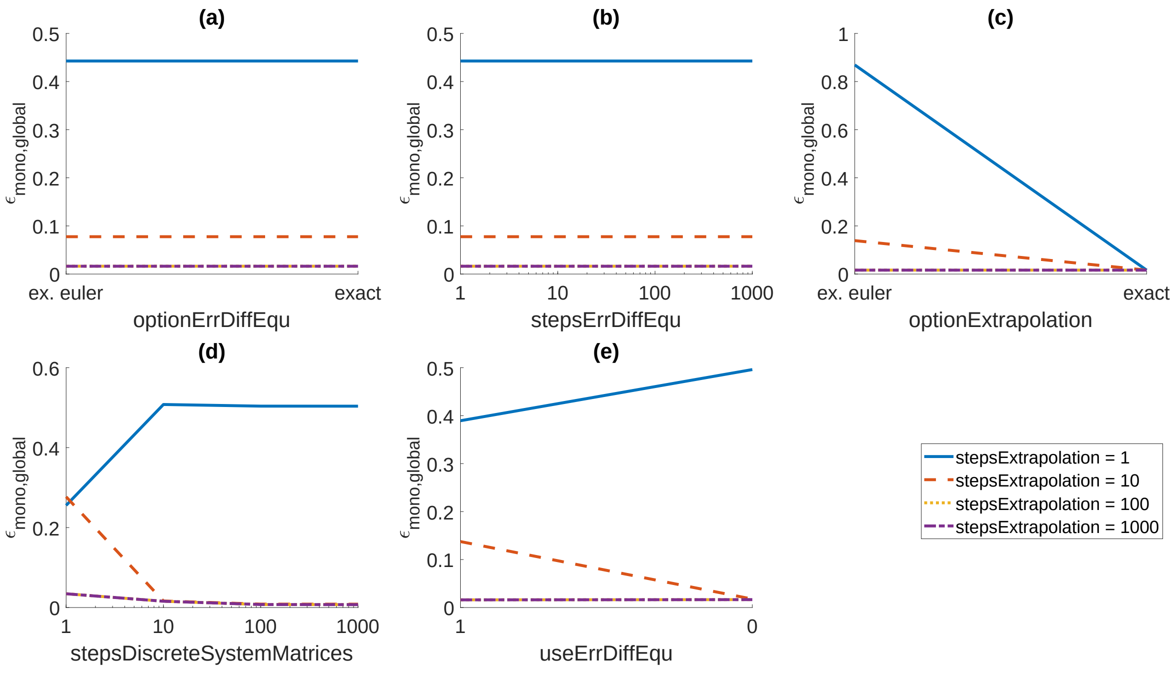


Figure 3.9: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for exact Interface Jacobians - Interaction Effect for stepsExtrapolation*

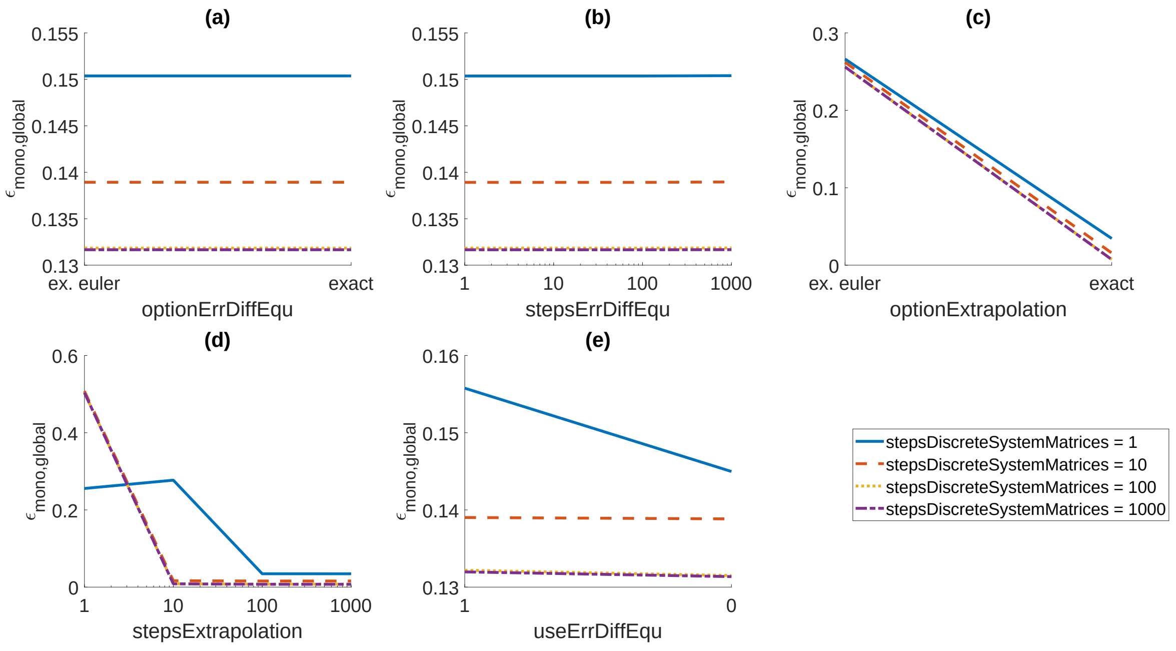


Figure 3.10: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for exact Interface Jacobians - Interaction Effect for stepsDiscreteSystemMatrices*

Ascending Sorting Analysis

The question of how to parametrize the Model-based Pre-Step Stabilization Method will be answered by the determination of an optimal set of parameters, via the second

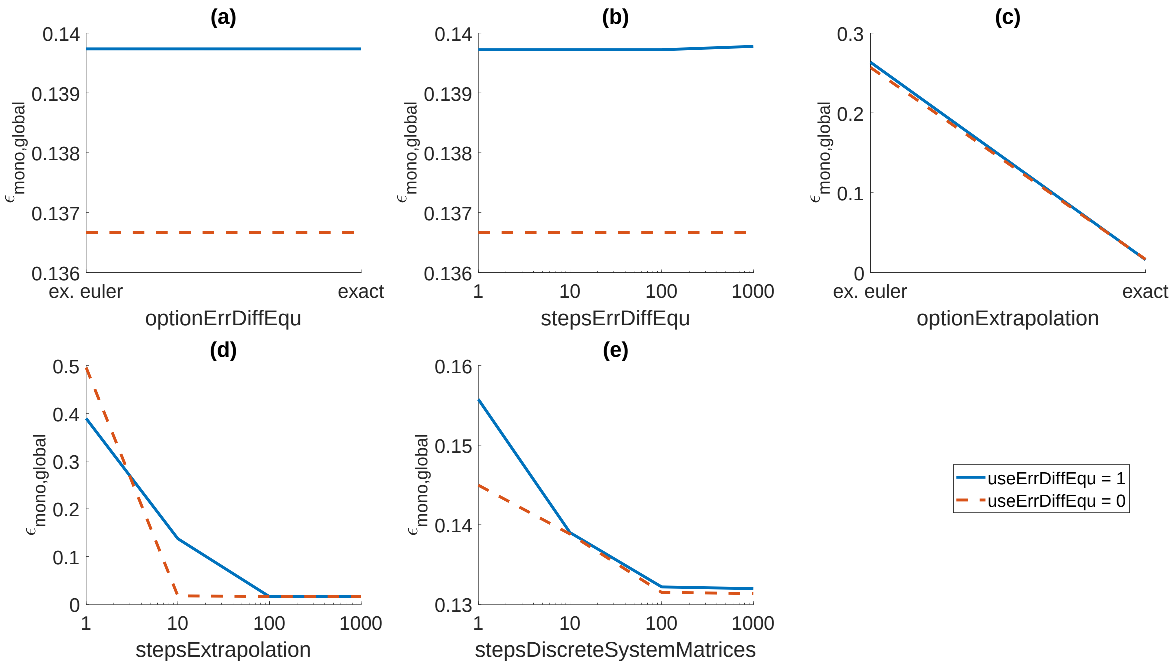


Figure 3.11: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for exact Interface Jacobians - Interaction Effect for boolUseErrDiff*

part of the design of experiment the so-called Ascending Sorting Analysis. Based on the sorted runs, according to the error measure $\epsilon_{mono,global}$, all six analyzed parameters will be rearranged in the same order. This means that in Figure 3.13 and 3.14 the parameters corresponding to a low error are depicted on the left side. If there is a specific pattern recognizable, especially for the parameters corresponding to the low errors, this means that a certain value or values of the parameter are from special interest. Due to the fact that there are many different runs with almost an optimal error, see therefore Figure 3.12, the optimal set of parameters is defined from all runs where the error is maximally 10 % higher than the best result, the corresponding parameters are called nearly optimal. The interpretation of the Figures 3.13 and 3.14 is easier with the help of the histogram, see therefore Figure 3.15 and 3.16. In these histograms only the results from the nearly optimal runs are depicted, as only these values are of interest for the computation of the optimal set of parameters. For all integer type parameters, the weighted mean over the data depicted in the histogram represents the optimal parameters. For the qualitative parameters the value is chosen which appears most in the histogram. The optimal set of parameters is stated in Table 3.5 and will be discussed in the following.
In Figure 3.13 one can see that for the parameters optionErrDiffEqu and stepsErrDiffEqu no pattern is recognizable, the values are arbitrary distributed and therefore no specific value is required for computing the best results. The same can be observed in the histogram in Figure 3.15, where every value appears with the same frequency for both parameters. The optimal parameter for optionErrDiffEqu is set to *exact* because it appears more often in the histogram. For stepsErrDiffEqu a weighted mean based

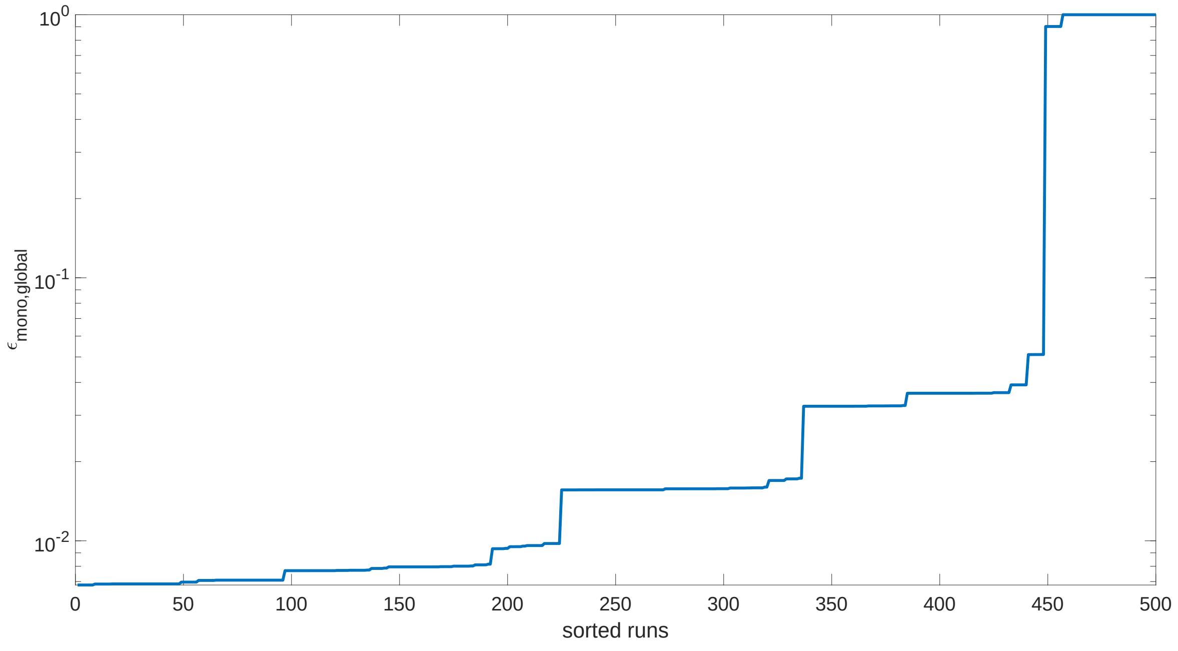


Figure 3.12: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for exact Interface Jacobians - sorted* $\epsilon_{mono,global}$

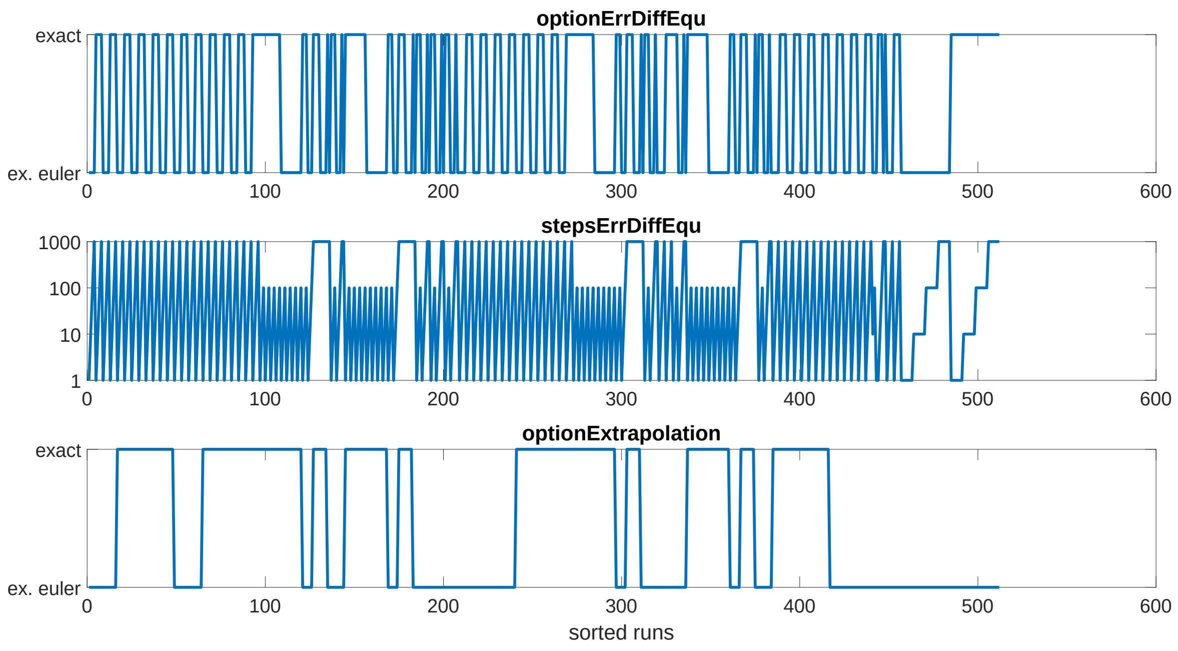


Figure 3.13: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for exact Interface Jacobians - sorted runs of analyzed parameter*

on the histogram is determined, leading to

$$\text{stepsErrDiffEqu}_{opt.} = (25 \cdot 1 + 24 \cdot 10 + 24 \cdot 100 + 24 \cdot 1000)\,\frac{1}{97} \approx 275.$$

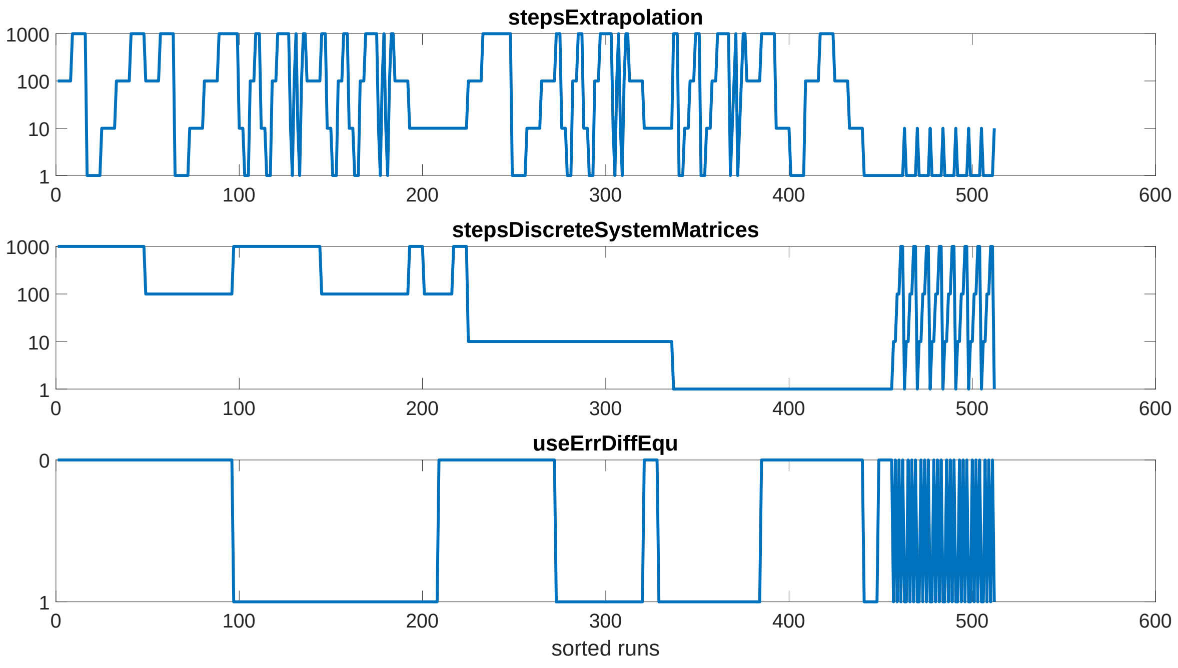


Figure 3.14: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for exact Interface Jacobians - sorted runs of analyzed parameter*

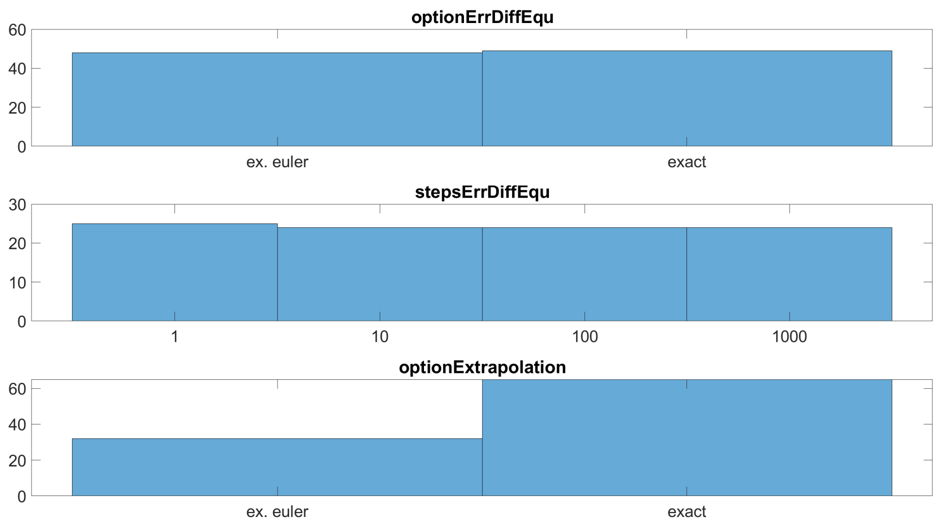


Figure 3.15: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for exact Interface Jacobians - histogram of nearly optimal parameters*

By comparing the parameter optionExtrapolation in Figure 3.13 and stepsExtrapolation in 3.14 a pattern can be clearly seen. For runs according low errors hold, if optionExtrapolation is set to *ex. euler* than stepsExtrapolation is 100 or 1000. For the case, that optionExtrapolation is *exact* the choice of stepsExtrapolation is arbi-

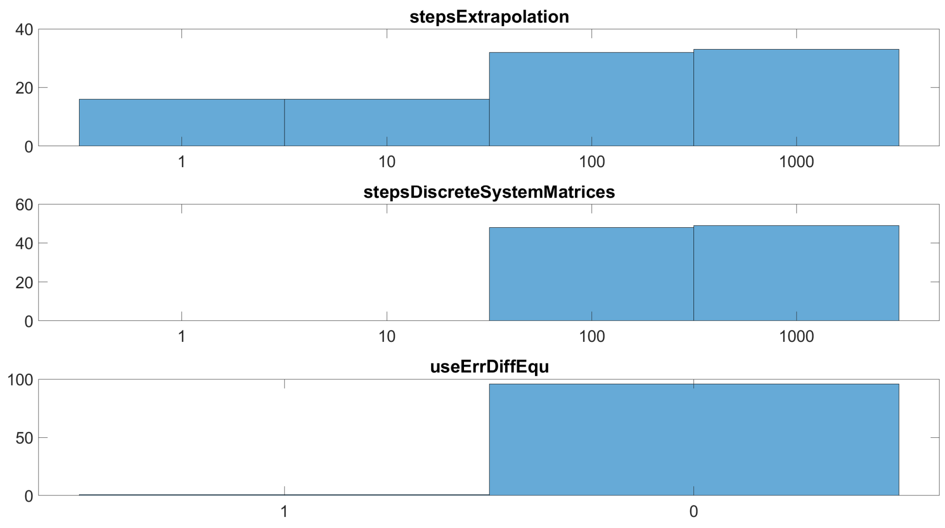


Figure 3.16: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for exact Interface Jacobians - histogram of nearly optimal parameters*

Parameter	Value
optionErrDiffEqu	*exact*
stepsErrDiffEqu	275
optionExtrapolation	*exact*
stepsExtrapolation	371
stepDiscreteSystemMatrices	543
useErrDiffEqu	*false*

Table 3.5: *Optimal set of parameters for the case of exact Interface Jacobians*

trary. On the right side of these figures one can see, that the choice of *ex. euler* as optionExtrapolation and stepsExtrapolation to 1 or 10, is resulting in high errors. This perfectly fits the observations from the analysis of variance. There it has been detected that optionExtrapolation and stepsExtrapolation have a strong interaction. In Figures 3.8 and 3.9 one can observe the phenomenon as described above. This leads to the conclusion that the best results can be observed by choosing optionExtrapolation as *exact*, the parameter stepsExtrapolation can be set to 1. The Ascending Sorting analysis can not handle interaction effects and therefore the optimal value of stepsExtrapolation is computed by the weighted mean based on the histogram in Figure 3.16, resulting in

$$\text{stepsExtrapolation}_{opt.} = (16 \cdot 1 + 16 \cdot 10 + 32 \cdot 100 + 33 \cdot 1000)\,\frac{1}{97} \approx 371.$$

This higher value of stepsExtrapolation will not effect the quality of the results in a negative way. The optimal value for optionExtrapolation is set to *exact* because it has a higher frequency than *ex. euler* in the histogram in Figure 3.15.

For stepsDiscreteSystemMatrices a clear pattern is recognizable, one should choose the parameter 1000 or 100. Choosing the values 1 or 10 is leading to bad results, as it is depicted in Figure 3.14. The optimal value is computed as a weighted mean, based on the histogram in Figure 3.16, leading to

$$\text{stepsDiscreteSystemMatrices}_{opt.} = (48 \cdot 100 + 59 \cdot 1000)\,\frac{1}{97} \approx 543.$$

For the parameter useErrDiffEqu the pattern and also the histogram is clear, the optimal value of useErrDiffEqu should be set to *false*.
The results of the co-simulation using the determined optimal parameters from Table 3.5 are depicted in Figure 3.17 and the corresponding coupling errors are depicted in Figure 3.18. The coupling error is defined as the difference of the monolithic and the co-simulation result. From the two figures it is obvious that the set of optimal parameters is a good choice because the co-simulated results are really close to the mono-simulated results, resulting in small coupling errors.

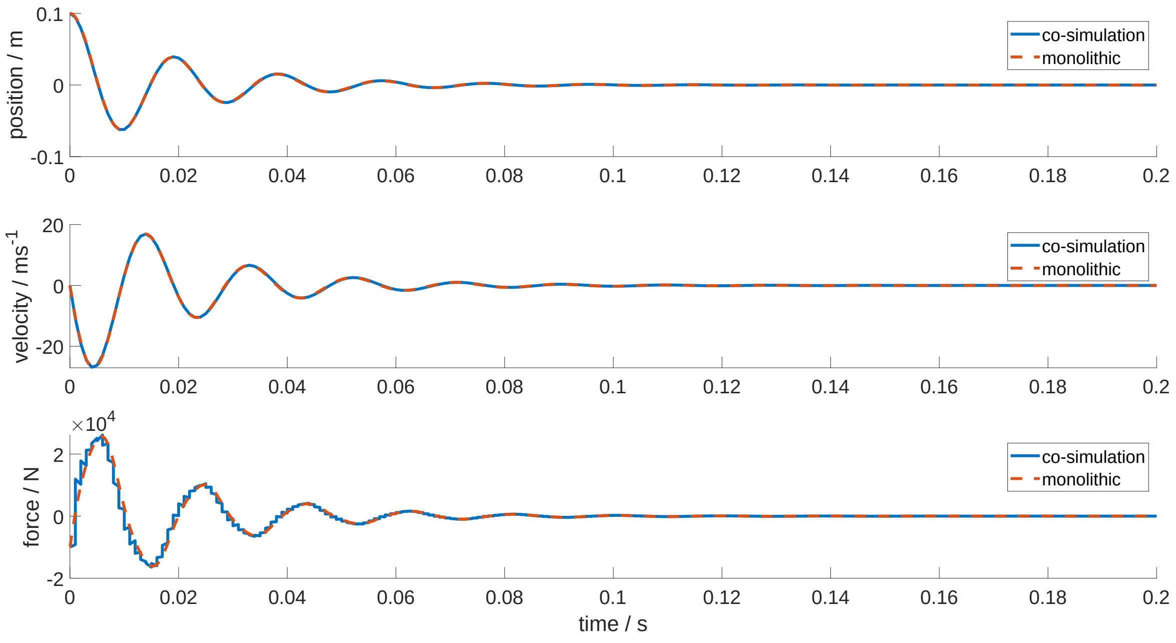


Figure 3.17: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for exact Interface Jacobians - position, velocity and force for optimal parameters*

Guideline for Parameters

This paragraph summarizes the results of the design of experiment for the case of exact Interface Jacobians, leading to guidelines for parametrizing the Model-based Pre-Step Stabilization Method. The analysis of variance has proofen that the major influence arises from two parameters, optionExtrapolation and stepsExtrapolation.

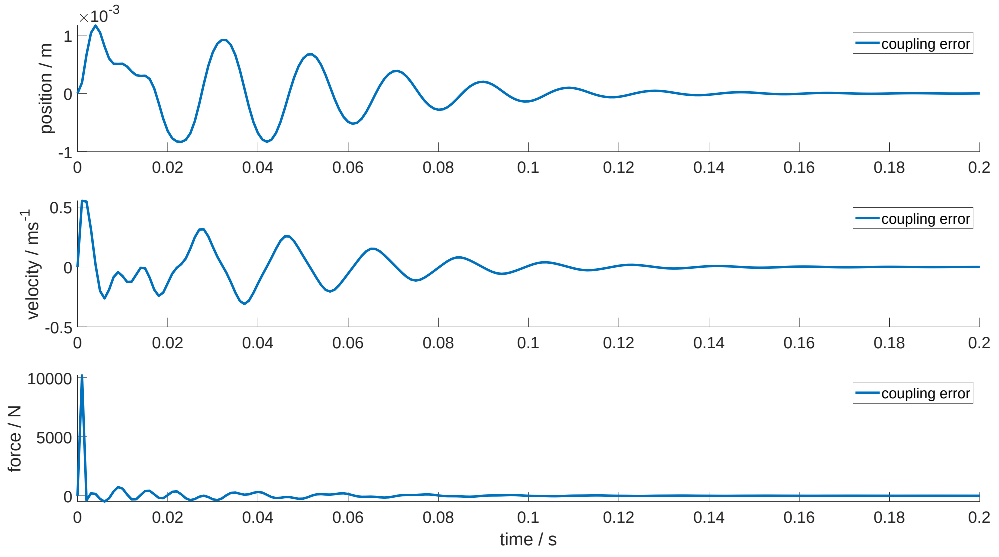


Figure 3.18: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for exact Interface Jacobians - coupling error of position, velocity and force for optimal parameters*

All other parameters are more or less not critical for the coupling method. Based on the Ascending Sorting Analysis an optimal set of parameters is determined, see Table 3.5 for the values of the six analyzed parameters. Guideline for parametrizing:

1. useErrDiffEqu: The analysis has shown that setting the parameter to *false* leads to better results. This means the Error Differential Equation is skipped. Therefore the choice of the parameters optionErrDiffEqu and stepsErrDiffEqu is of no influence. If otherwise, one wants to use the Error Differential Equation one should use optionErrDiffEqu set to *exact* and stepsErrDiffEqu can be set to 1.
2. stepDiscreteSystemMatrices: This parameter should be chosen as high as possible, but increasing this value leads to a higher computational effort. Therefore and due to the fact that between 100 and 1000 the error $\epsilon_{mono,global}$ is nearly the same, one should not choose stepDiscreteSystemMatrices greater than 1000, the determined optimal value of 543 is a good choice.
3. optionExtrapolation: If one is interested in the highest possible accuracy one should always choose *exact*, this guarantees the best results, but it will cost computational time. If the focus is drawn to a faster co-simulation one should choose *ex. euler* as parameter. In the following it will be pointed out, that there is a high connection of optionExtrapolation and stepsExtrapolation.
4. stepsExtrapolation: It can be clearly divided into two cases. First, if optionExtrapolation is set to *exact*, all values of stepsExtrapolation generates the same results and due to minimizing the computational effort one should choose stepsExtrapolation as 1. It should be really emphasized that, this is only valid, for *exact* as the value of optionExtrapolation. Second, if optionExtrapolation is

choosen as *ex. euler*, the parameter stepsExtrapolation should be chosen as high as possible, considering the aspect of increasing computational effort.

3.2.2 Approximated Interface Jacobians

For the case of approximated Interface Jacobians the influence of the user-defined parameters is analyzed in the following. The Interface Jacobians can be approximated with two different system identification methods a Recursive Least Squares Approach (RLS) or the Multivariable Output Error State Space Method (MOESP), both are derived and described in detail in Section 2.3. Due to the fact that those two approximation methods have different parameters the analysis is split into two parts, one utilizing the RLS, the other the MOESP as system identification method. The parameters of the design of experiment which are valid for both cases are stated in Table 3.6, these parameters will be explained in the following.
The parameter timeLearning defines the length of the learning phase, where the co-simulation is signal-based coupling, i.e. zero-order or first-order hold. This is unavoidable in the case of approximated Interface Jacobians because it needs a certain amount of time to gather a sufficient amount of data, so that the online system identification methods can work properly. Online system identification means, that only data in the present or the past of the simulation can be used, therefore a learning phase, to gather enough information is mandatory for utilizing online system identification methods. For more details about the learning phase see Section 2.4.3.
Between the learning phase and the standard coupling phase, there has to be a switching. To keep the disturbance of this switching as small as possible, the switching is performed smoothly. The duration of this switch phase is defined by the parameter timeSwitchphase, the higher it is, the smoother the interference of the switching will be. For more details see Section 2.4.4.
If the approximation errors of the Interface Jacobians are sufficiently high, the coupling method will not perform the standard model-based coupling schema. The error-based phase check, see Section 2.4.2, will determine in which phase the coupling method works, which means how the subsystems are coupled. The parameter thresholdMbc2sbcCoupling represents th_{SBC} and thresholdMbcNoInputOpt stands for $th_{mod.}$ in Section 2.4.2. It should be denoted that the condition

$$th_{mod.} \leqslant th_{SBC} \tag{3.13}$$

should be always fulfilled. This functionality can be seen as a safety inquiry, so that a bad approximation in a single step can not lead to critical consequences.
The parameter useErrDiffEqu is the same as in the case of exact Interface Jacobians, it describes whether the Error Differential Equation should be utilized or not, for details see Section 3.2.1. For the parameters of type real in Table 3.6 the levels have to be defined. As typical for a DoE one has to think of the number of required experiments when defining the number of levels, therefore one is always limited. The chosen levels are stated in Table 3.7. Here it should be emphasized that although the condition thresholdMbcNoInputOpt $\leqslant$ thresholdMbc2sbcCoupling holds, all full factorial combinations of the parameters will be computed. This is important for the computation

timeLearning	timeSwitchphase	thresholdMbc-2sbcCoupling	thresholdMbc-NoInputOpt	useErrDiffEqu
real	real	real	real	boolean

Table 3.6: *Analyzed parameters of the DoE for the case of approximated Interface Jacobians for RLS and MOESP as system identification*

timeLearning	timeSwitchphase	thresholdMbc-2sbcCoupling	thresholdMbc-NoInputOpt
0.1	0.002	0.01	0.01
0.2	0.01	0.05	0.05
0.5	0.05	0.1	0.1
		0.5	0.5
		1	1
		2	2
		5	5

Table 3.7: *Levels of the integer and real type analyzed parameters for RLS and MOESP as system identification*

of the sum of squares for the main and interaction effects.
The case of utilizing the RLS as system identification method will be discussed first. The analyzed parameters and their levels are stated in Table 3.8. The parameter λ is called the forgetting factor, it describes the weighting of the samples, for more details see [17, 18]. The latest sample has the highest weight and the others are weighted in descending order. One can roughly say that around $\frac{1}{1-\lambda}$ data samples are of significant impact for the system identification. Therefore there is a big gap between choosing $\lambda = 0.99$ or $\lambda = 0.999$, for the value $\lambda = 1$ there is no forgetting and all samples are weighted the same.
The parameters M and N determines the order of the identified difference equation (2.86). N stands for the number of considered values of the output itself and M defines the number of considered values of the corresponding input, see also (2.88). It should be denoted that for every input there will be the same number of considered samples. The levels with 1 and 2 each are chosen because the Dual Mass Oscillator, see Section 3.1, has only two states in each subsystem. Therefore an approximation with three or more states would be senseless.
By utilizing the RLS as system identification method, there are eight analyzed parameters in total, based on the different number of levels, the number of required simulations can be computed as

$$neededRuns = \underbrace{3 \cdot 3 \cdot 7 \cdot 7 \cdot 2}_{\text{general parameter}} \cdot \underbrace{5 \cdot 2 \cdot 2}_{\text{RLS parameter}} = 17640.$$

The MOESP method requires two parameters, the modelOrder and the windowSize, the chosen levels are stated in Table 3.9. The parameter modelOrder stands for the number of states of the underlying state space model, for more details of how the MOESP method approximates the Interface Jacobians see Section 2.3.1. It should be

λ	M	N
real	integer	integer
0.95	1	1
0.98	2	2
0.99		
0.999		
1		

Table 3.8: *Analyzed parameters and their levels of the DoE for the case of approximated Interface Jacobians for RLS as system identification*

modelOrder	windowSize
integer	integer
1	50
2	200
	500
	1000

Table 3.9: *Analyzed parameters and their levels of the DoE for the case of approximated Interface Jacobians for MOESP as system identification*

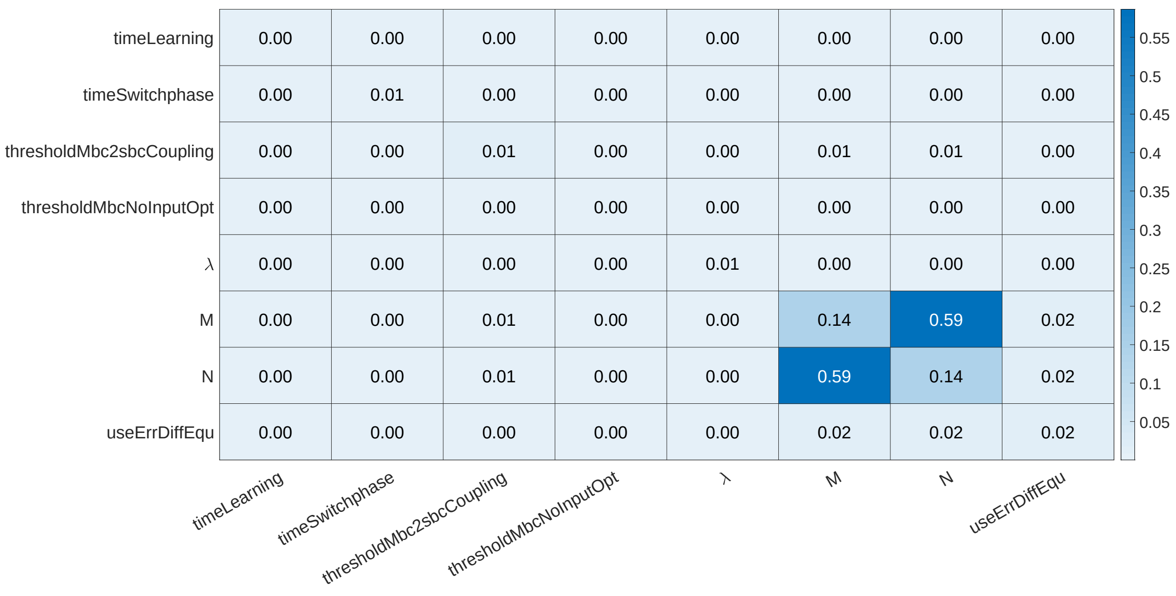
denoted that choosing modelOrder greater than two would not make any sense for that example, because the number of states for both subsystems is two. For both subsystems the same number of states is chosen, due to the fact that the subsystems are from similar structure.
In contrast to the RLS the MOESP requires a moving window of considered samples. The parameter windowSize describes the number of samples in this moving window. The larger the windowSize is, the more data is taken into account for the approximation of the Interface Jacobians. The drawback is that all data in this window are weighted the same and therefore the reaction to changes in the system dynamics is slow. Due to the fact that the sampling rate, i.e. the communication step size, is 1 ms, this means that the windowSize levels of $50, 200, 500, 1000$ stand for moving windows with a length of $0.05, 0.2, 0.5$ and 1 second. Due to the fact that it is an online system identification, the window is filled until the correct number of samples are stored and then it is always shifted, keeping its size constant, as a classical moving window. This means that for the value of 1000 always the whole history of the simulation is taken into account, because the simulation ends at 1 second. The number of required runs is based on the number of levels of all seven analyzed parameters and can be computed as

$$neededRuns = \underbrace{3 \cdot 3 \cdot 7 \cdot 7 \cdot 2}_{\text{general parameter}} \cdot \underbrace{2 \cdot 4}_{\text{MOESP parameter}} = 7056.$$

Analysis of Variance

The analysis of variance investigates, how much influence the analyzed parameters of the Model-based Pre-Step Stabilization Method, according the overall co-simulation result have. This leads to the conclusion which parameters are important and should therefore be chosen with care and for which the choice is not crucial. As mentioned before, this design of experiment will be split up into two cases, one for utilizing the RLS as system identification method and for the other one, the MOESP method is the system identification tool. Therefore the following analysis of variance will be structured into these two parts.

RLS as System Identification First the analysis of variance for the RLS based coupling method will be discussed. The computed mean sum of squares for the main MS_X and interaction effects MS_{XY} are stated in Table 3.10. The definitions of MS_X and MS_{XY} and how to compute them are stated in Section 3.2.1. Additionally the main and interaction effects are visualized as a heatmap in Figure 3.19, the diagonal entries of the heatmap represent the mean sum of squares for the main effects and the off-diagonal entries the mean sum of squares for the interaction effects, according the parameter denoted at the x- and y-axis. There it is clearly visible that the interaction effect of N and M has the most influence, followed by main effect of M and N. Therefore they will be discussed in further detail.



	timeLearning	timeSwitchphase	thresholdMbc2sbcCoupling	thresholdMbcNoInputOpt	λ	M	N	useErrDiffEqu
timeLearning	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
timeSwitchphase	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
thresholdMbc2sbcCoupling	0.00	0.00	0.01	0.00	0.00	0.01	0.01	0.00
thresholdMbcNoInputOpt	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
λ	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00
M	0.00	0.00	0.01	0.00	0.00	0.14	0.59	0.02
N	0.00	0.00	0.01	0.00	0.00	0.59	0.14	0.02
useErrDiffEqu	0.00	0.00	0.00	0.00	0.00	0.02	0.02	0.02

Figure 3.19: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by RLS - Sum of Squares of Main and Interaction Effects*

In Figure 3.20 (a) one can see that the parameter timeLearning is of minor significance, additionally in the interaction plot in Figure 3.21 it is obvious that there is no interaction with any other parameter of significant importance. The same phenomenon can

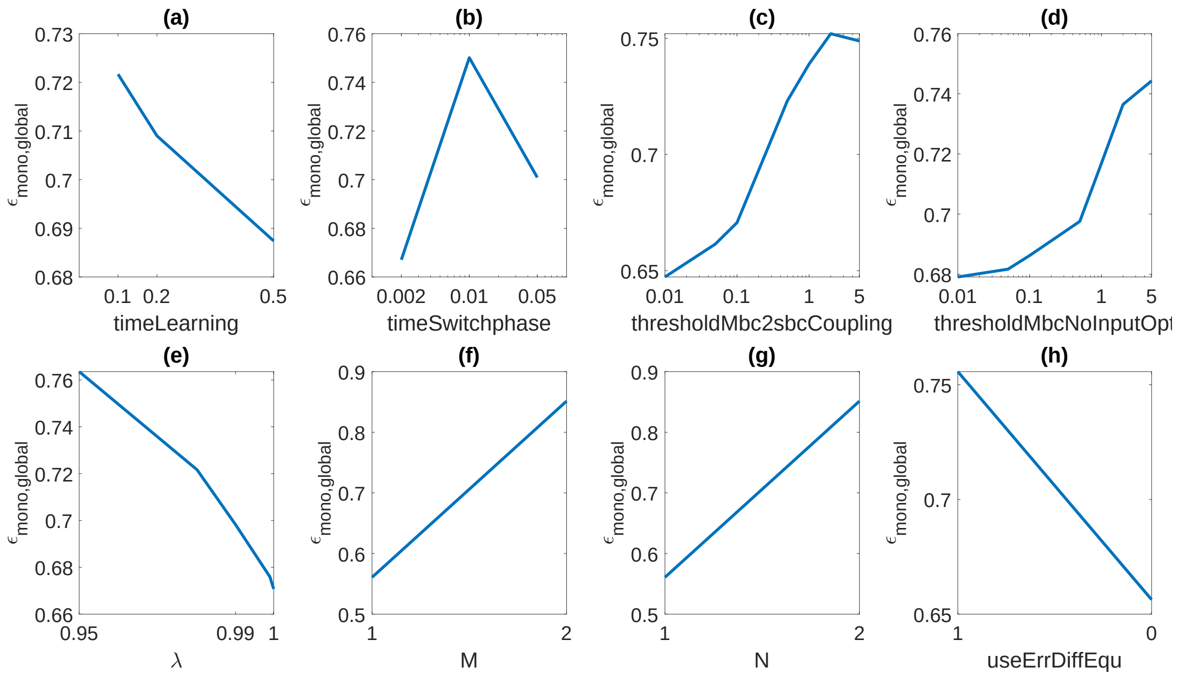


Figure 3.20: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by RLS - Main Effects*

be observed for the parameter timeSwitchphase in Figure 3.20 (b) and Figure 3.22. Both statements are supported by the fact that all main and interaction effects according timeLearning and timeSwitchphase are from rate 0, see therefore Table 3.10. The main effect plot of thresholdMbc2sbcCoupling and thresholdMbcNoInputOpt in Figure 3.20 (c) and (d) shows a slight tendency for choosing small values, but the heatmap shows that the effects are of nearly zero significance for the coupling method. The interaction plots for both parameters look similar, there is no interaction with other parameters, except for the case of the interaction of the parameters with themselves. This arises from the condition thresholdMbcNoInputOpt $\leqslant$ thresholdMbc2sbcCoupling.
The choice of λ is of minor importance to the performance of the Model-based Pre-Step Stabilization Method, because the main effect and also the interaction effects visualized in the heatmap in Figure 3.19 or stated in Table 3.10 are nearly zero. This fact is supported by the interaction plots in Figure 3.25, as λ shows no significant interaction with any other parameter.
Whether the Error Differential Equation is utilized or not is of minor significance for the analyzed coupling method because the main and interaction effects in Table 3.10 are nearly zero. Due to the main effect plot in Figure 3.20 (h) and the interaction plots in Figure 3.28 the choice of setting useErrDiffEqu to *false* seems legit, because the error is less than by utilizing the Error Differential Equation.
As mentioned before, most significant for the coupling method is the choice of the parameters M and N. The first thing one can observe is that the main effect plots in Figure 3.20 (f) and (g) look identical, and both show better results by choosing the

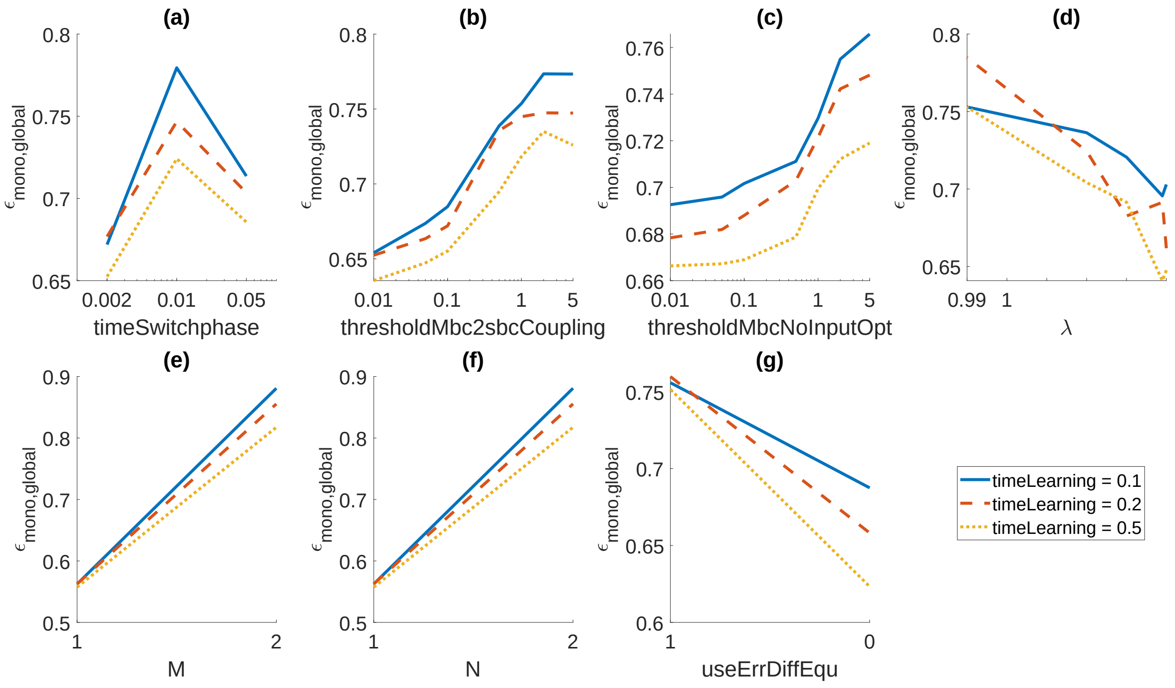


Figure 3.21: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by RLS - Interaction Effects of timeLearning*

value of 1. The interaction plots in Figure 3.26 and 3.27 show that with all other parameters there is nearly no interaction, but the interaction of M and N is highly significant. This is indicated in subplot (f) by the crossing of the lines, because this means that if M is set to a value of 1 the best choice for N is also 1, but if M is set to 2 also N should be chosen as 2. In other words the parameters M and N should always be chosen the same, because choosing M and N with different values is leading to high errors. This is the reason for the high mean square of sums for the interaction of M and N. The high main effect value can be explained by the main effect subplots (f) and (g) because choosing M and N to a value of 1 is significantly reducing the error. Comparing the scaling of the y-axis of all main effect plots in Figure 3.20 leads also to the conclusion that the choice of M and N has the most influence for the analyzed coupling method.

Parameters Main and Two Factor Interaction Effects	MS_X	Normalized / %
timeLearning	3	0.1
timeSwitchphase	20	0.8
thresholdMbc2sbcCoupling	30	1.2
thresholdMbcNoInputOpt	11	9.4
λ	20	0.8
M	372	14.3
N	372	14.3
useErrDiffEqu	44	1.7
timeLearning – timeSwitchphase	1	0
timeLearning – thresholdMbc2sbcCoupling	1	0
timeLearning – thresholdMbcNoInputOpt	0	0
timeLearning – λ	3	0.1
timeLearning – M	2	0
timeLearning – N	2	0
timeLearning – useErrDiffEqu	3	0.1
timeSwitchphase – thresholdMbc2sbcCoupling	7	0.3
timeSwitchphase – thresholdMbcNoInputOpt	0	0
timeSwitchphase – λ	4	0.1
timeSwitchphase – M	10	0.4
timeSwitchphase – N	10	0.4
timeSwitchphase – useErrDiffEqu	7	0.3
thresholdMbc2sbcCoupling – thresholdMbcNoInputOpt	10	0.4
thresholdMbc2sbcCoupling – λ	6	0.2
thresholdMbc2sbcCoupling – M	18	0.7
thresholdMbc2sbcCoupling – N	18	0.7
thresholdMbc2sbcCoupling – useErrDiffEqu	5	0.2
thresholdMbcNoInputOpt – λ	2	0
thresholdMbcNoInputOpt – M	0	0
thresholdMbcNoInputOpt – N	0	0
thresholdMbcNoInputOpt – useErrDiffEqu	1	0
λ – M	2	0
λ – N	2	0
λ – useErrDiffEqu	3	0.1
M – N	1524	58.7
M – useErrDiffEqu	41	1.6
N – useErrDiffEqu	41	1.6

Table 3.10: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by RLS - Sum of Squares of Main and Interaction Effects*

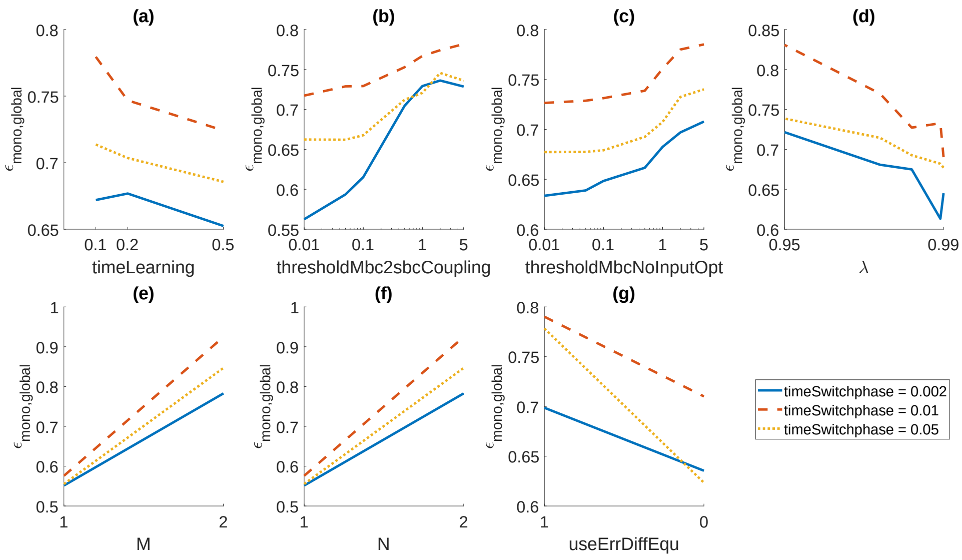


Figure 3.22: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by RLS - Interaction Effects of timeSwitchphase*

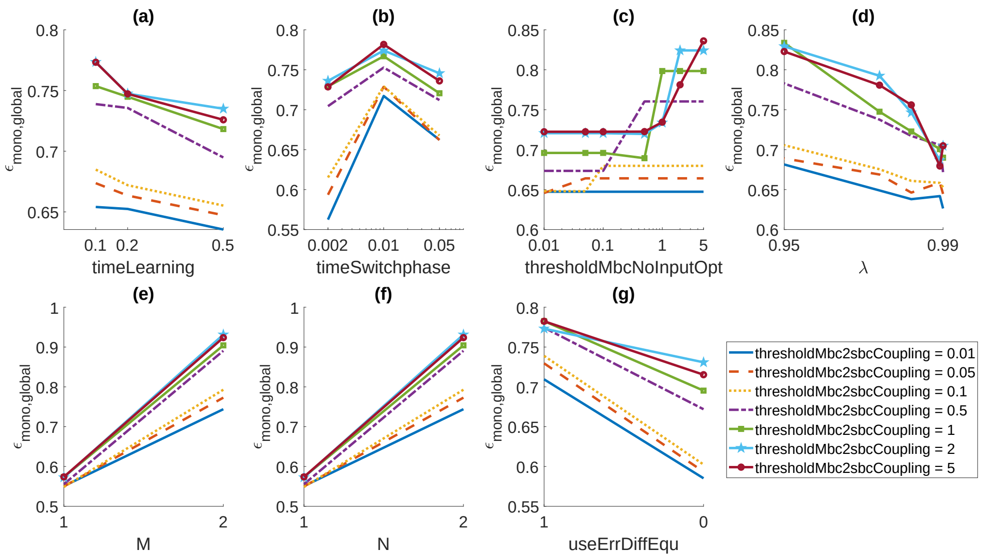


Figure 3.23: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by RLS - Interaction Effects of thresholdMbc2SbcCoupling*

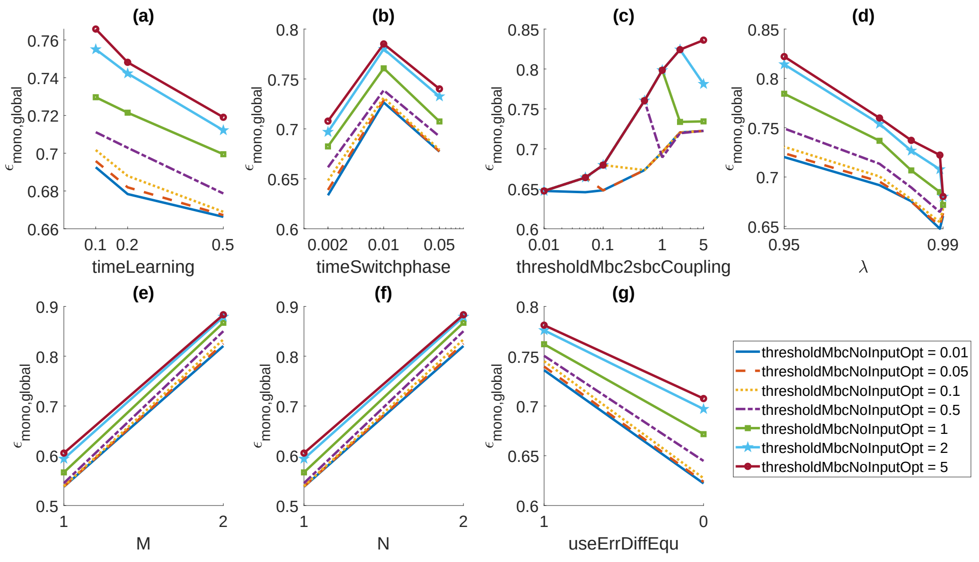


Figure 3.24: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by RLS - Interaction Effects of thresholdMbc2NoInputOpt*

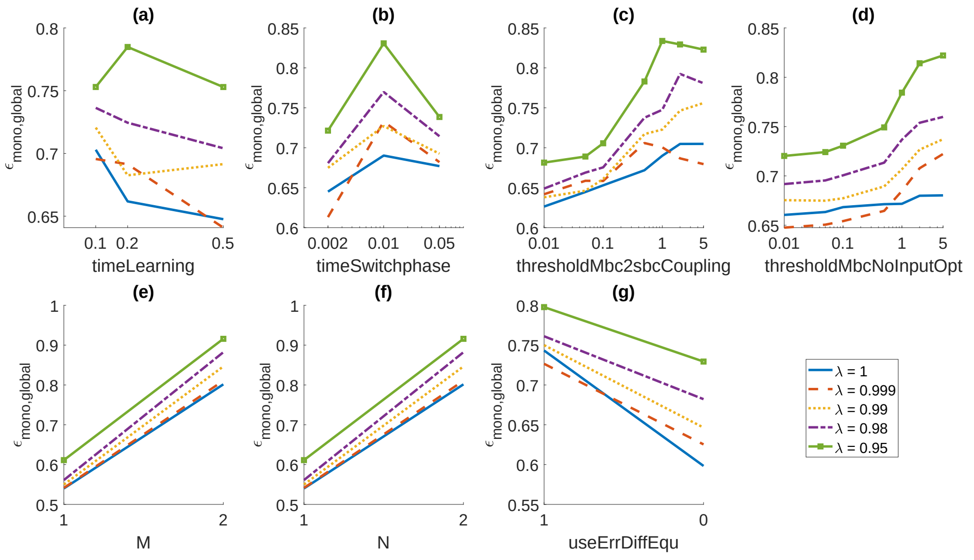


Figure 3.25: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by RLS - Interaction Effects of λ*

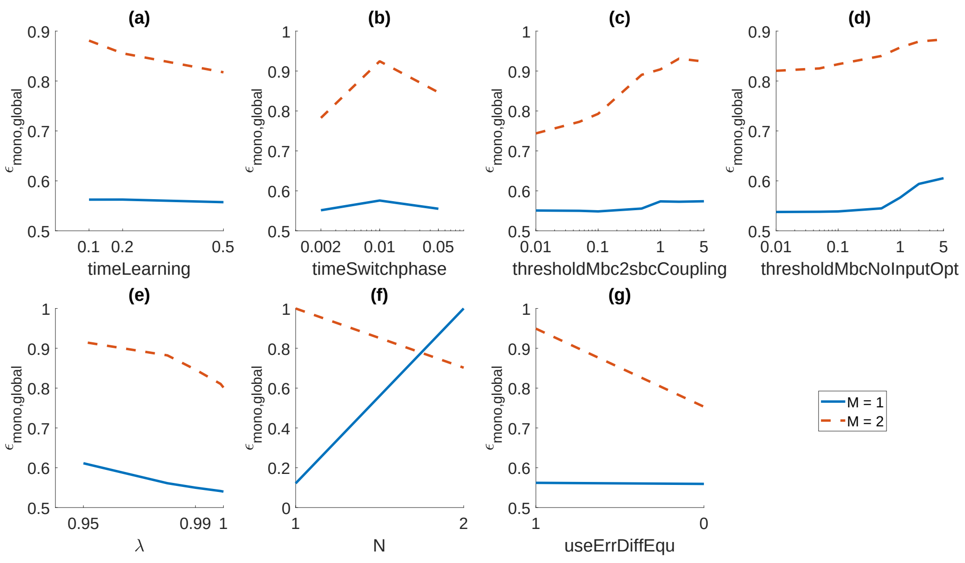


Figure 3.26: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by RLS - Interaction Effects of M*

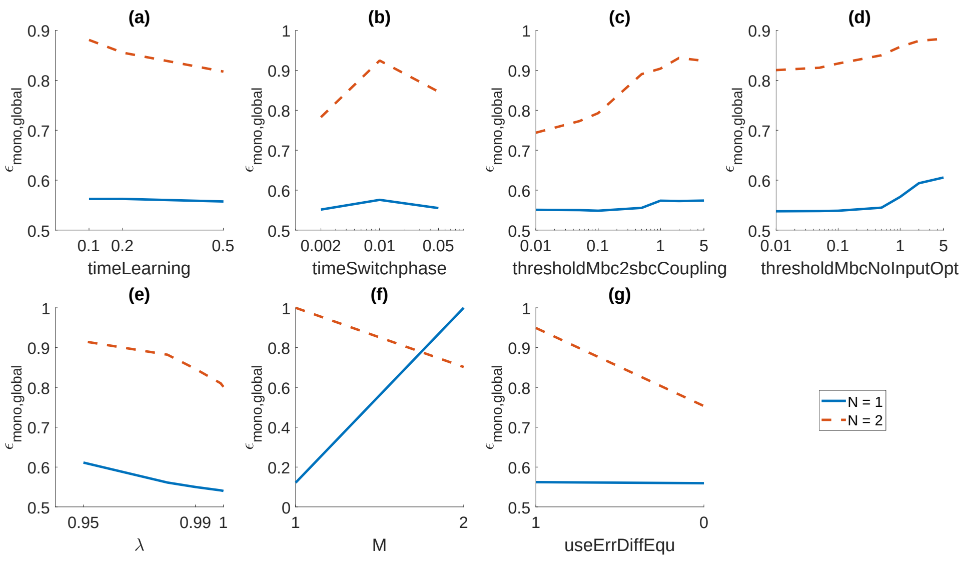


Figure 3.27: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by RLS - Interaction Effects of N*

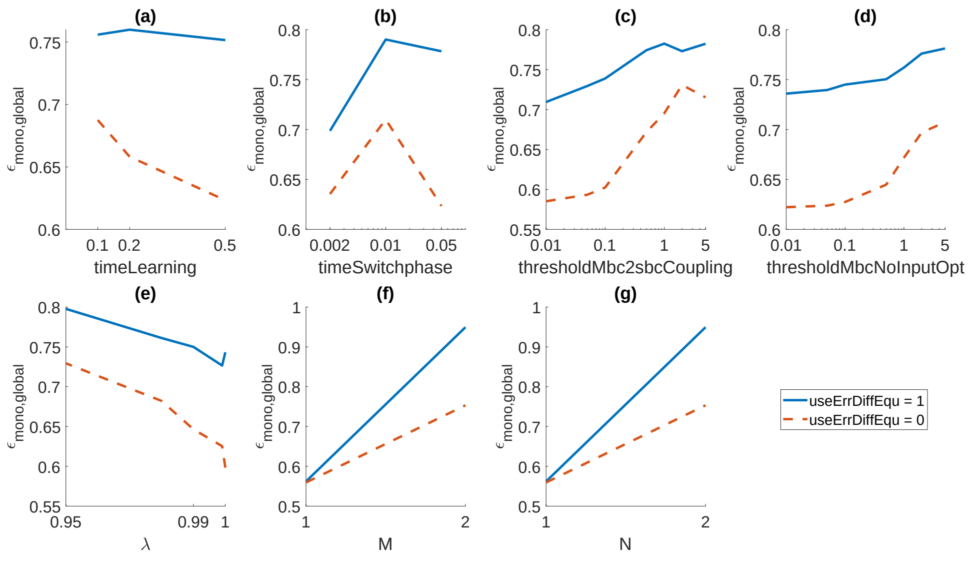
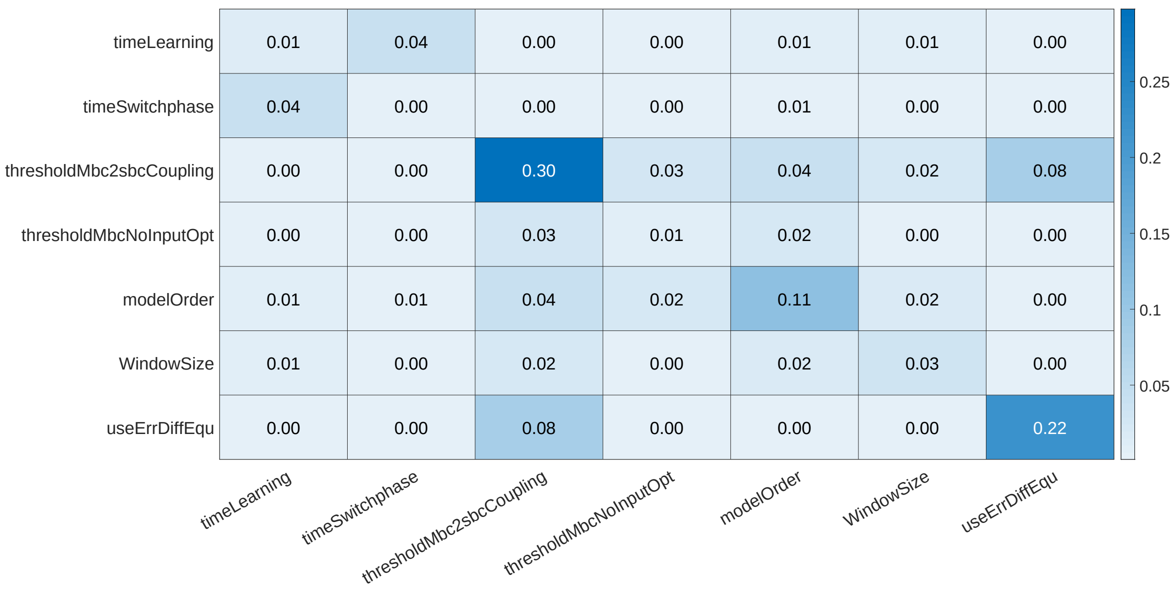


Figure 3.28: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by RLS - Interaction Effects of useErrDiffEqu*

MOESP as System Identification In the following the analysis of variance for the case of approximated Interface Jacobians by the MOESP method is stated. The main and two-factor interaction effects are stated in Table 3.11 and visualized in a heatmap in Figure 3.29. There one sees, that the three parameters thresholdMbc2sbcCoupling, modelOrder and useErrDiffEqu have a high significance for the coupling method. Therefore those three parameters will be discussed in further detail.
The parameter timeLearning is of minor importance, as its main effect is 0.01, only the interaction with the parameter timeSwitchphase with a normalized mean square of sums of 0.04 is observable. This interaction effect is also depicted in the interaction plot in Figure 3.31 (a), as there is a crossing of the lines. With all other parameters there is no detectable interaction from timeLearning.
Except for the interaction mentioned above, there is no remarkable effect, neither an interaction nor a main effect, for the parameter timeSwitchphase, as it is depicted in Figure 3.32 (b) - (f). This means that this parameter is of minor importance for the coupling method.
The parameter thresholdMbcNoInputOpt has a negligible main effect, as it is depicted in Figure 3.30. Additionally, as it is depicted in Figures 3.34 (c) and (d), the only two observable interactions are with the parameters thresholdMbc2sbcCoupling and modelOrder. In all other subplots no interaction is recognizable.
The size of the moving window of the MOESP method is described by the parameter windowSize. The heatmap in Figure 3.29 shows clearly a small but observable interaction with modelOrder and thresholdMbc2sbcCoupling. In the interaction plots in Figure 3.36, it is clear to see that, always the interaction with windowSize set to 50 differs from the other three values. The same behaviour can be observed at the main effect plot in Figure 3.30 (e). This means that there is a difference if windowSize is set to 50, but for the values $200, 500$ and 1000 the difference is rather small.
The highest influence has the main effect of thresholdMbc2sbcCoupling, with a normalized mean square of sums of 0.30, the same can be observed in Figure 3.30 (c), comparing the y-axis of the different main effects it is clear that thresholdMbc2sbcCoupling is of major influence for the coupling method. The interaction is compared to its main effect rather low, only with the parameters thresholdMbc2sbcCoupling, modelOrder and useErrDiffEqu small dependencies are recognizable.
The choice of the modelOrder is of importance for the coupling method, as a normalized mean square of 0.11 denotes. The interaction with the other six parameters is rather small and of negligible importance, as it is depicted in Figure 3.35.
Whether the Error Differential Equation is utilized or not is of high importance, as the value of 0.22 for the normalized mean square shows. This fact is supported by the main effect plot in Figure 3.30 (g). Only the interaction with the parameter thresholdMbc2sbcCoupling is recognizable, see therefore heatmap in Figure 3.29, all other interactions have a normalized effect of nearly 0.

Parameters Main and Two Factor Interaction Effects	Absolute	Normalized / %
timeLearning	5.9	1.5
timeSwitchphase	2.0	0.5
thresholdMbc2sbcCoupling	120.8	29.8
thresholdMbcNoInputOpt	3.8	0.9
modelOrder	46.3	11.4
windowSize	12.4	3.1
useErrDiffEqu	90.1	22.2
timeLearning – timeSwitchphase	17.1	4.2
timeLearning – thresholdMbc2sbcCoupling	1.3	0.3
timeLearning – thresholdMbcNoInputOpt	0.6	0.1
timeLearning – modelOrder	3.4	0.8
timeLearning – windowSize	2.5	0.6
timeLearning – useErrDiffEqu	1.6	0.4
timeSwitchphase – thresholdMbc2sbcCoupling	1.5	0.4
timeSwitchphase – thresholdMbcNoInputOpt	0.4	0.1
timeSwitchphase – modelOrder	2.2	0.5
timeSwitchphase – windowSize	0.5	0.1
timeSwitchphase – useErrDiffEqu	1.7	0.4
thresholdMbc2sbcCoupling – thresholdMbcNoInputOpt	11.2	2.8
thresholdMbc2sbcCoupling – modelOrder	15.7	3.9
thresholdMbc2sbcCoupling – windowSize	9.2	2.3
thresholdMbc2sbcCoupling – useErrDiffEqu	34.2	8.4
thresholdMbcNoInputOpt – modelOrder	9.6	2.4
thresholdMbcNoInputOpt – windowSize	1.0	0.2
thresholdMbcNoInputOpt – useErrDiffEqu	0.9	0.2
modelOrder – windowSize	7.9	2
modelOrder – useErrDiffEqu	0.6	0.1
windowSize – useErrDiffEqu	1.3	0.3

Table 3.11: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by MOESP - Sum of Squares of Main and Interaction Effects*



	timeLearning	timeSwitchphase	thresholdMbc2sbcCoupling	thresholdMbcNoInputOpt	modelOrder	WindowSize	useErrDiffEqu
timeLearning	0.01	0.04	0.00	0.00	0.01	0.01	0.00
timeSwitchphase	0.04	0.00	0.00	0.00	0.01	0.00	0.00
thresholdMbc2sbcCoupling	0.00	0.00	0.30	0.03	0.04	0.02	0.08
thresholdMbcNoInputOpt	0.00	0.00	0.03	0.01	0.02	0.00	0.00
modelOrder	0.01	0.01	0.04	0.02	0.11	0.02	0.00
WindowSize	0.01	0.00	0.02	0.00	0.02	0.03	0.00
useErrDiffEqu	0.00	0.00	0.08	0.00	0.00	0.00	0.22

Figure 3.29: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by MOESP - Sum of Squares of Main and Interaction Effects*

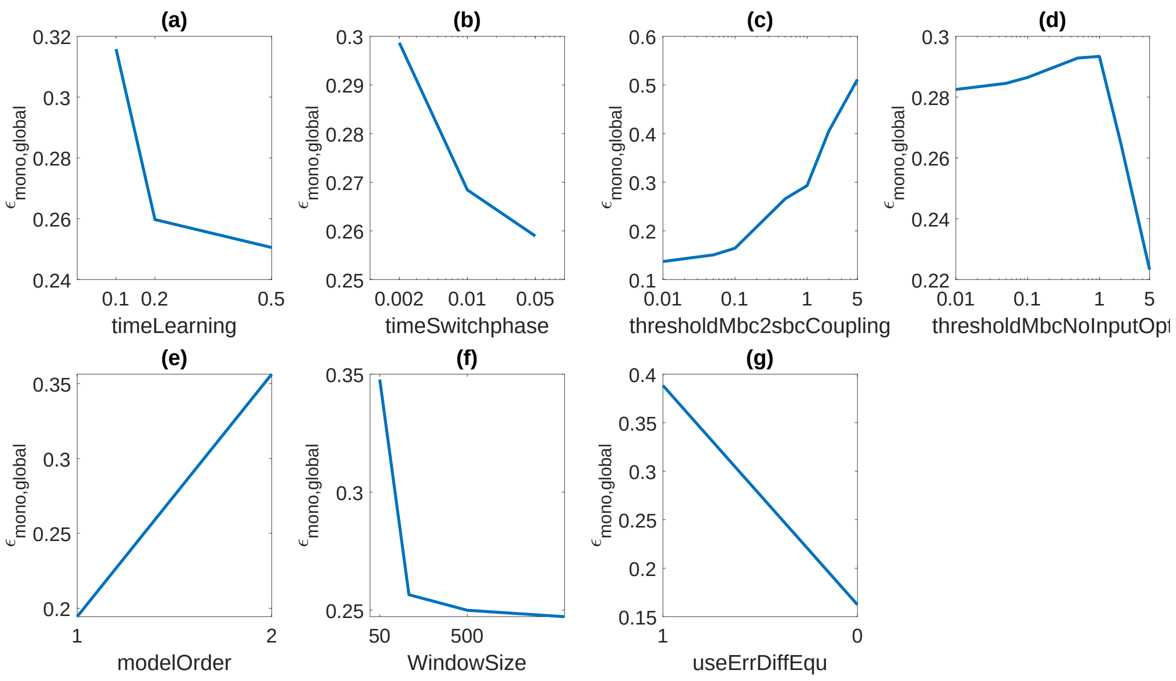


Figure 3.30: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by MOESP - Main Effects*

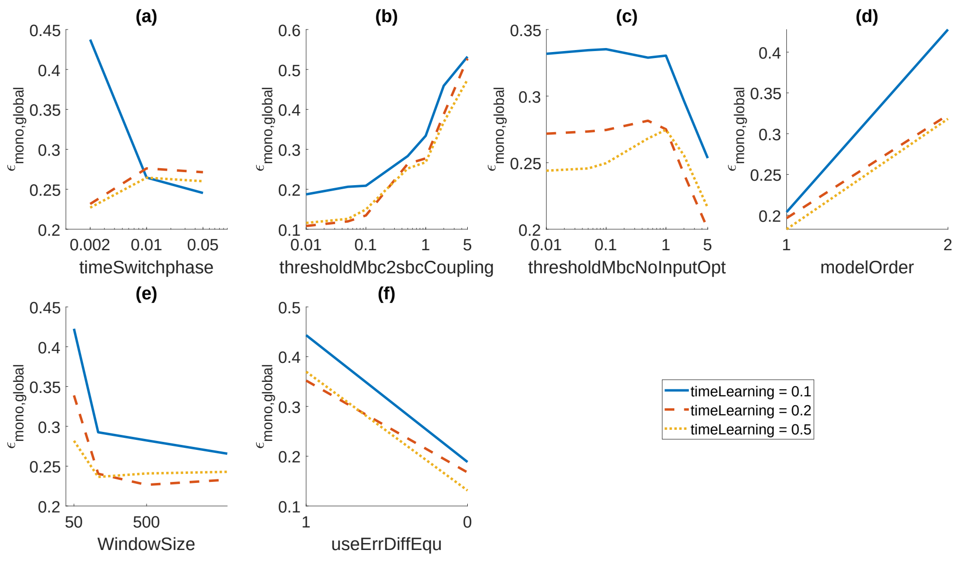


Figure 3.31: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by MOESP - Interaction Effects of timeLearning*

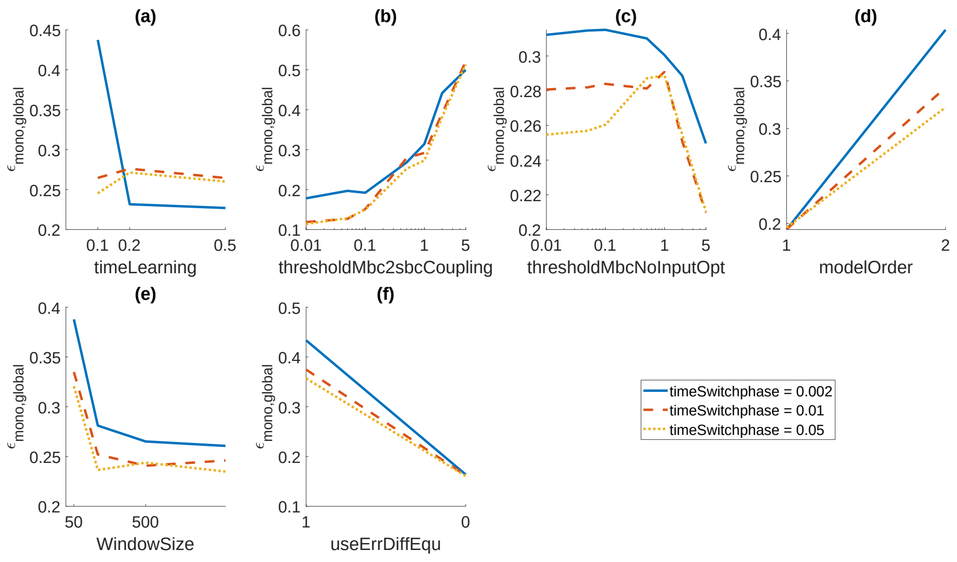


Figure 3.32: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by MOESP - Interaction Effects of timeSwitchphase*

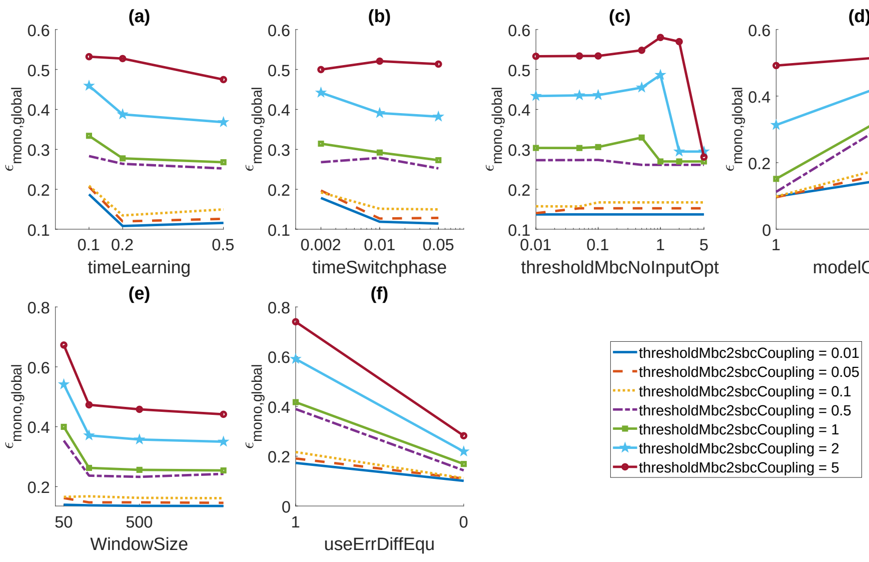


Figure 3.33: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by MOESP - Interaction Effects of thresholdMbc2SbcCoupling*

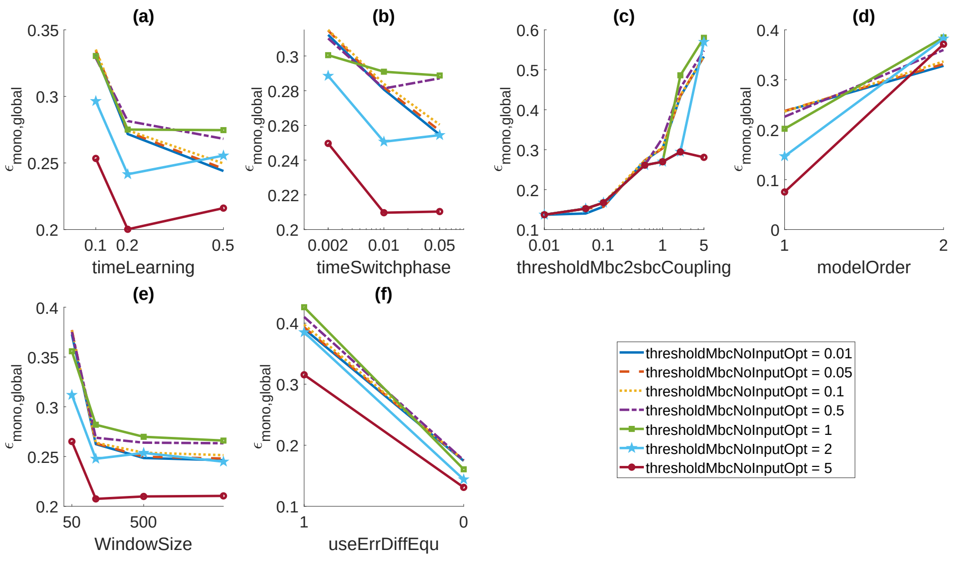


Figure 3.34: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by MOESP - Interaction Effects of thresholdMbc2NoInputOpt*

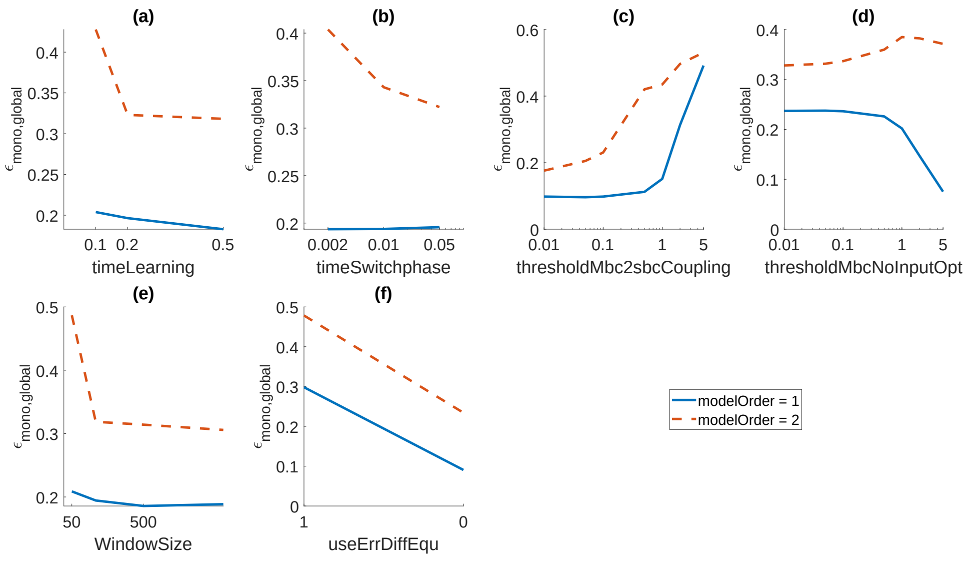


Figure 3.35: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by MOESP - Interaction Effects of modelOrder*

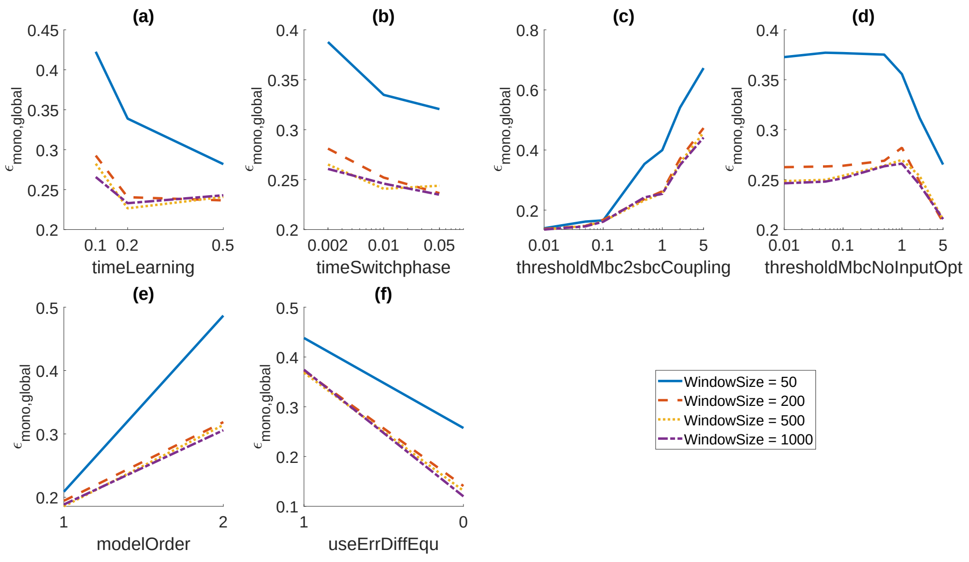


Figure 3.36: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by MOESP - Interaction Effects of windowSize*

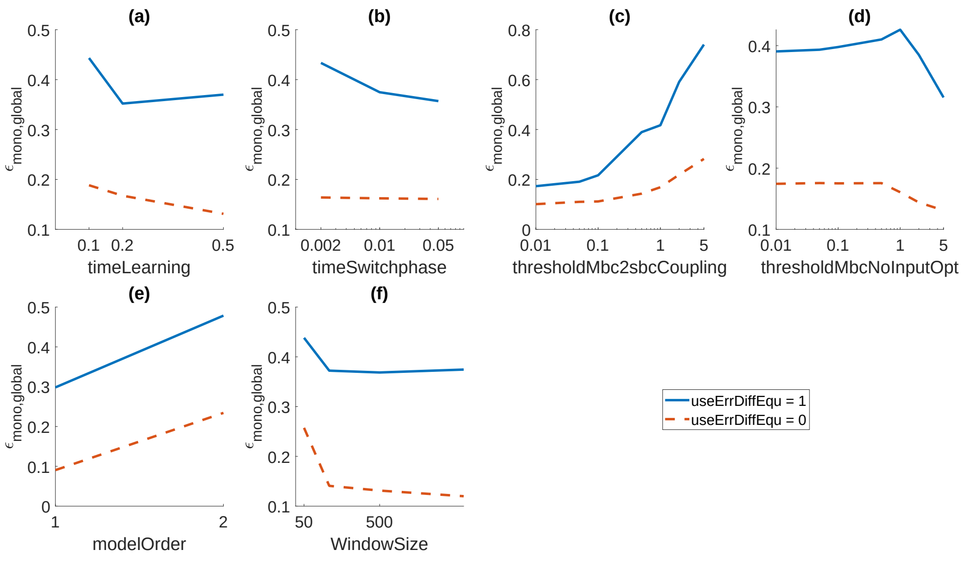


Figure 3.37: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by MOESP - Interaction Effects of useErrDiffEqu*

Ascending Sorting Analysis

The Ascending Sorting Analysis is analogous to the case of utilizing exact Interface Jacobians, see therefore Section 3.2.1. The goal of this analysis is to determine an optimal set of parameters for the Model-based Pre-Step Stabilization Method, in the case of utilizing approximated Interface Jacobians. The simulation runs are sorted in ascending order according to the error measure $\epsilon_{mono,global}$, as it is depicted in Figure 3.38 or 3.45. As it is the case for the analysis of variance in the previous section, this analysis is split up into the same two parts. The first part will deal with the analysis of the coupling method utilizing the RLS method as system identification method, for approximating the required Interface Jacobians. In the second part, the MOESP method is used instead of the RLS.

RLS as System Identification Beginning with the RLS as system identification method, it should be mentioned that the maximum error is set to $\epsilon_{mono,global} = 1$, as it can be seen by the saturation of $\epsilon_{mono,global}$ starting at approximately run 5800, in Figure 3.38. In Figures 3.39 and 3.40 all eight analyzed parameters are rearranged according to the sorted runs depicted in Figure 3.38. It should be denoted that only the first 500 runs are depicted because the focus is drawn to the simulation results with a low error measure. For the computation of the optimal parameters the average over all parameter values, according simulation runs where $\epsilon_{mono,global}$ is only 10% higher than its minimum, are taken into account, this means that only the first 179 simulation runs are considered.
For the parameter timeLearning there is no obvious pattern recognizable in Figure 3.39, but in the first histogram in Figure 3.41 one can observe that the value 0.2 has the highest frequency. But the trend to a specific value for this parameter is not completely clear. The optimal value for timeLearning is determined with

$$\text{timeLearning}_{opt.} = \big(50 \cdot 0.1 + 99 \cdot 0.2 + 30 \cdot 0.5\big)\frac{1}{179} \approx 0.22.$$

The second subplot in Figure 3.39 and 3.41 refers to the parameter timeSwitchphase. There it is recognizable, that the parameter should be chosen small, the value 0.05 is of minor frequency. This fact is especially evident from the histogram, based on this data the optimal value is determined as

$$\text{timeSwitchphase}_{opt.} = \big(101 \cdot 0.002 + 78 \cdot 0.01\big)\frac{1}{179} \approx 0.005.$$

For the parameter thresholdMbc2sbcCoupling the pattern is extremely obvious, because only the two highest values 2 and 5 appear in the first 400 runs. It should be mentioned, that there are 7 possible values of thresholdMbc2sbcCoupling, as it is stated in Table 3.7. Also in the histogram the obvious pattern is easily observable. The optimal value is computed as

$$\text{thresholdMbc2sbcCoupling}_{opt.} = \big(78 \cdot 2 + 92 \cdot 5\big)\frac{1}{179} \approx 3.54.$$

At a first glance on the fourth subplot in Figure 3.39 one can observe, that the parameter thresholdMbcNoInputOpt should be chosen rather small than big, because the values 2 and 5 are not displayed in the first 500 runs. In the corresponding histogram in Figure 3.41 it is clear to see, that small values are preferred. The optimal value can be computed as

$$\begin{aligned}\text{thresholdMbcNoInputOpt}_{opt.} = \big(44 \cdot 0.01 &+ 44 \cdot 0.05 + 43 \cdot 0.1 + 37 \cdot 0.5 + \dots \\ &+ 12 \cdot 1\big)\frac{1}{179} \approx 0.21.\end{aligned}$$

For the parameter λ a specific pattern is recognizable, the values 0.98 and 0.95 should be avoided. Based on the histogram, a more detailed view for the optimal choice of λ is possible. There the highest frequency is at the value of 0.999, but the difference to 1 is rather small. Based on that data the optimal choice for this parameter is determined as

$$\lambda_{opt.} = \big(60 \cdot 1 + 72 \cdot 0.999 + 33 \cdot 0.99 + 14 \cdot 0.98\big)\frac{1}{179} \approx 0.996.$$

The parameters M and N have identical results and it is quite obvious that the value of 1 is preferred. There are just some rare cases where M and N are set to 2. Remarkable is that there is not a single case where M and N are chosen differently for the first 500 simulation runs, as one can see by comparing the second and third subplot in Figure 3.40. The optimal choice for these parameters can be clearly set to

$$\begin{aligned}\text{M}_{opt.} =& 1, \\ \text{N}_{opt.} =& 1.\end{aligned}$$

Whether the Error Differential Equation should be utilized or not, is decided by setting useErrDiffEqu to *true* or *false*. From the fourth subplot of Figure 3.40 a clear pattern is not identifiable. In the histogram the situation is more obvious, the value *true* appears approximatly twice as much as the value *false*. This means that useErrDiffEqu should be set *true*, but it should be emphasized that this choice is not clear, it stays uncertain. An overview of the eight analyzed optimal parameters is stated in Table 3.12. Figure 3.43 shows the position, velocity and the coupling force of the dual mass oscillator for the optimal parameter settings and in Figure 3.44 the corresponding coupling error for all three signals are depicted. As the co-simulated results fit the monolithic results quite well the optimal set of parameters seems well chosen. Additionally one can observe that after the learning phase, the oscillations of the coupling error is damped clearly, which shows the effect of the Model-based Pre-Step Stabilization Method.

Parameter	Value
timeLearning	0.22
timeSwitchphase	0.005
thresholdMbc2sbcCoupling	3.54
thresholdMbcNoInputOpt	0.21
λ	0.996
M	1
N	1
useErrDiffEqu	*true*

Table 3.12: *Optimal set of parameters for the case of approximated Interface Jacobians*

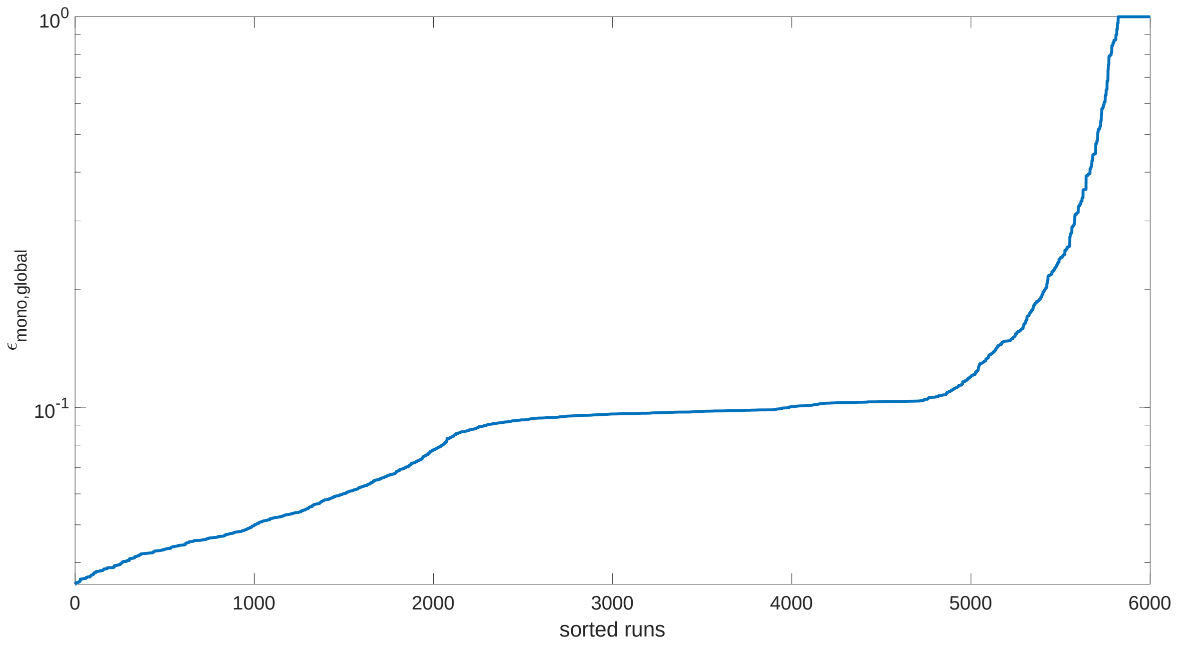


Figure 3.38: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by RLS - sorted $\epsilon_{mono,global}$*

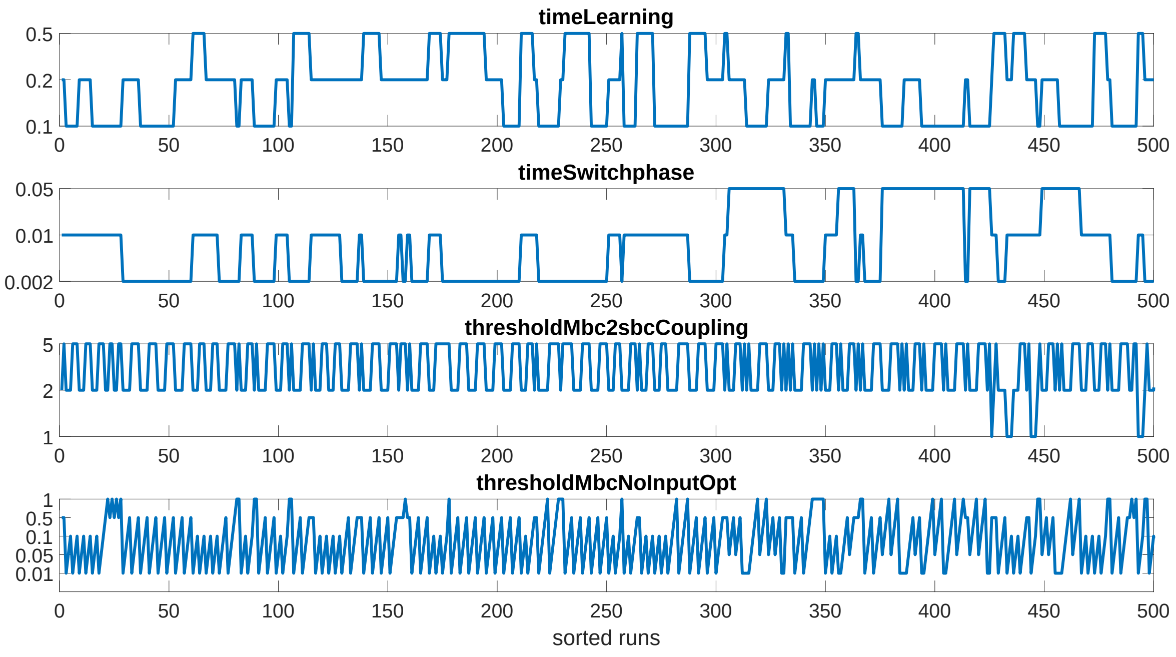


Figure 3.39: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by RLS - sorted runs of analyzed parameters*

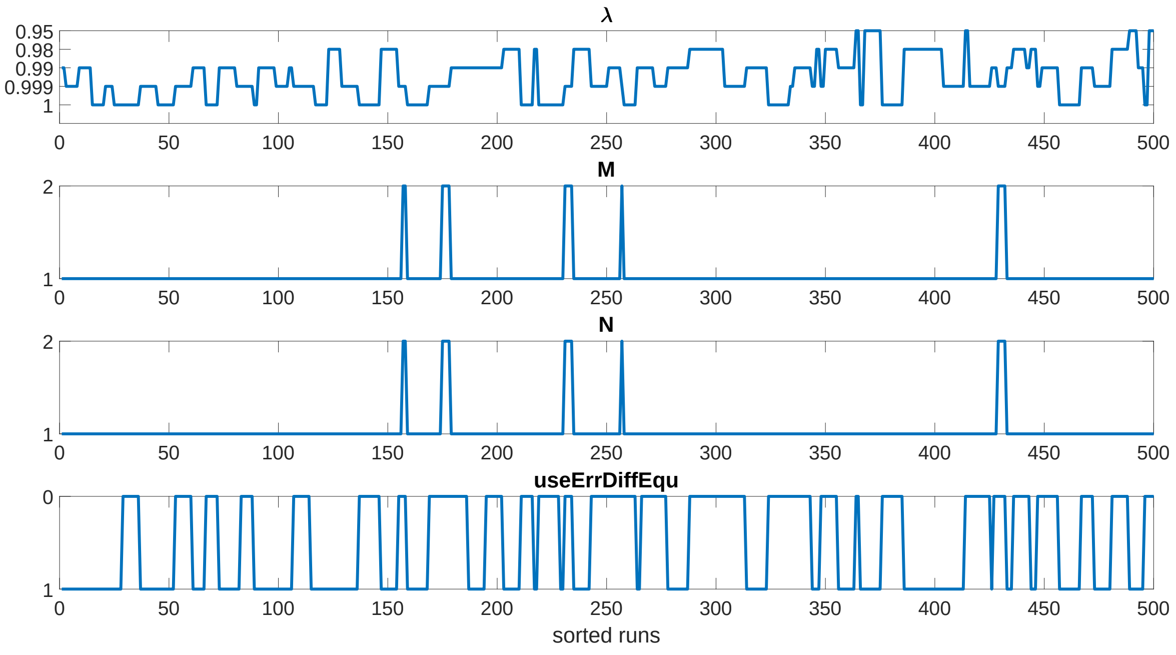


Figure 3.40: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by RLS - sorted runs of analyzed parameters*

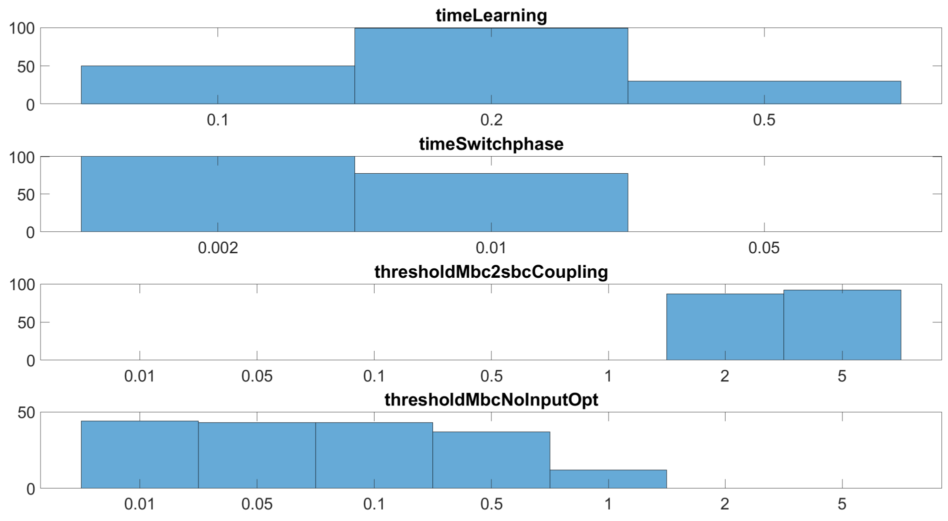


Figure 3.41: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by RLS - histogram of nearly optimal parameters*

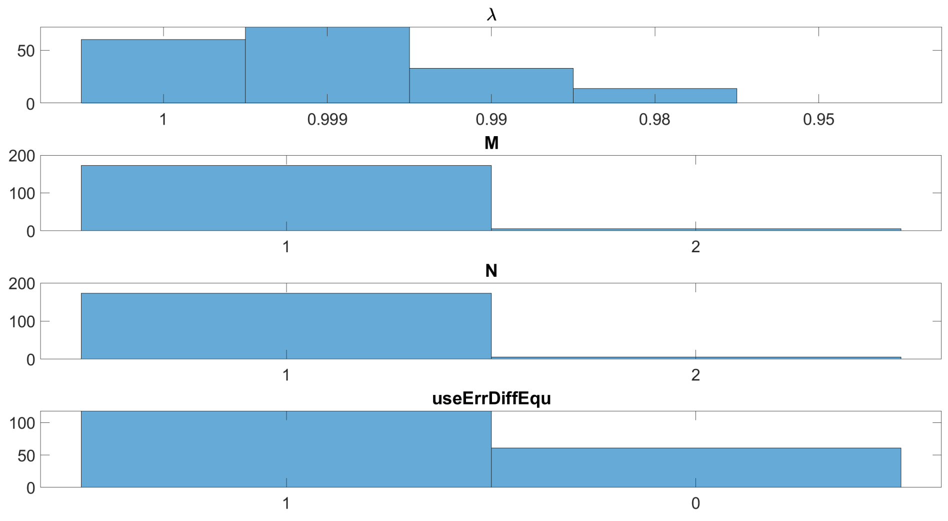


Figure 3.42: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by RLS - histogram of nearly optimal parameters*

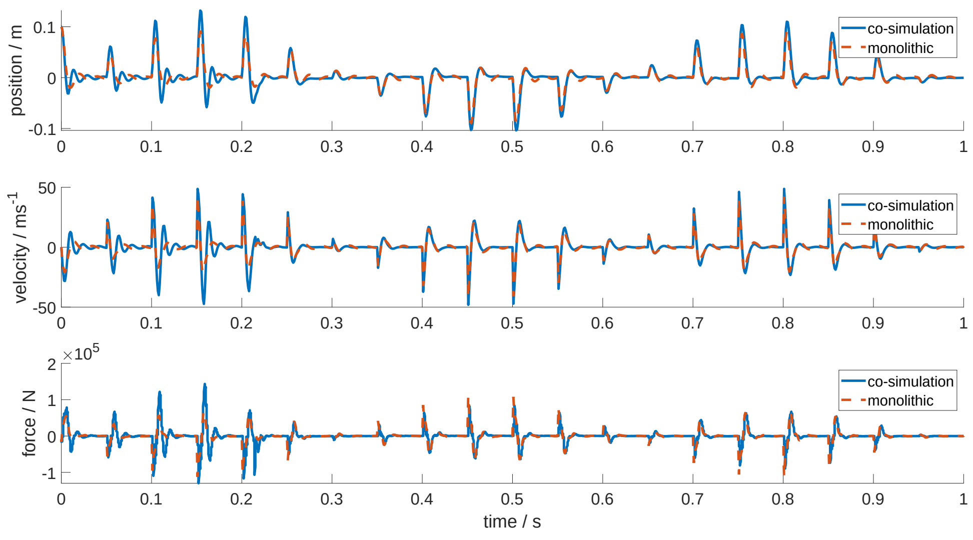


Figure 3.43: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by RLS - position, velocity and force co-simulated utilizing optimal parameters*

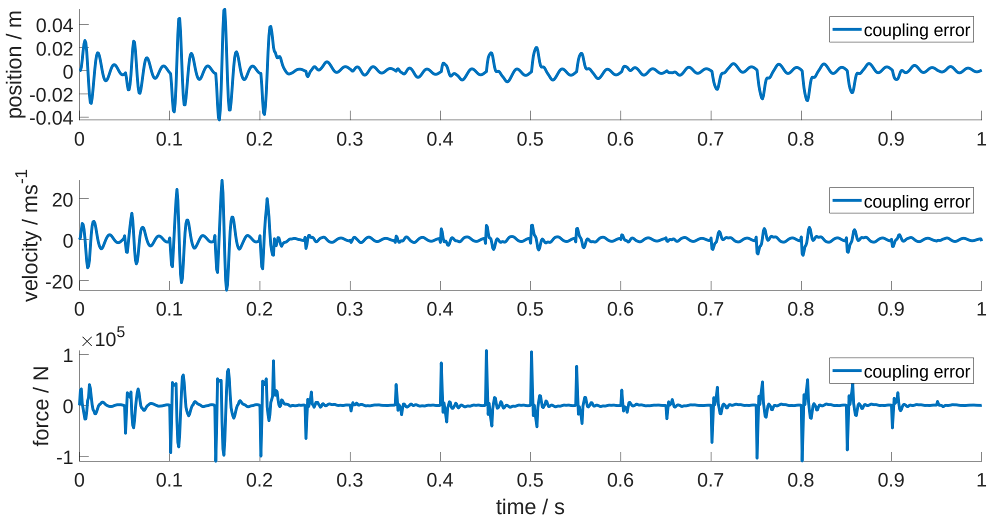


Figure 3.44: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by RLS - coupling error of position, velocity and force co-simulated utilizing optimal parameters*

MOESP as System Identification The following section deals with the Ascending Sorting Analysis for the case of utilizing Interface Jacobians approximated by the MOESP method. In Figure 3.45 the simulation runs are depicted in ascending sorted order according to the error measure $\epsilon_{mono,global}$, see (3.9) for its definition. From the sorted parameter plots in Figures 3.46 and 3.47 the optimal set of parameters can be determined by analyzing the patterns in it. Helpful for determining this set of parameters are the histograms in Figures 3.48 and 3.49. The optimal set of parameters will be determined as the average of the values depicted in those histograms. Due to the fact, that between the low errors the gap is wider than in case of utilizing the RLS, the condition of considering only parameters which correspond to errors which are maximal 10% greater than the lowest, would lead to only three entries in the histogram. Therefore the condition is adapted so that a minimum of 50 simulation runs are considered. The set of optimal parameters is stated in Table 3.13 and will be discussed in the following.
Starting with the parameter timeLearning one sees in the first subplot of Figure 3.46 that a clear pattern is hard to determine, therefore the histogram in Figure 3.46 shows that the value 0.2 appears mostly. The optimal value of timeLearning is determined by computing the mean of the data depicted in the histogram, resulting in

$$\text{timeLearning}_{opt.} = (5 \cdot 0.1 + 28 \cdot 0.2 + 17 \cdot 0.5)\,\frac{1}{50} \approx 0.29.$$

For the parameter timeSwitchphase it is obvious, that small values appear with a higher frequency, as one can observe in the second histogram in Figure 3.46. This means a fast changing between learn and model-based coupling phase is better for the performance of the coupling method. The optimal value is computed as

$$\text{timeSwitchphase}_{opt.} = (37 \cdot 0.002 + 13 \cdot 0.01)\,\frac{1}{50} \approx 0.004.$$

The third subplot in Figure 3.46 shows a really clear pattern, only the values 2 and 5 appear. This fact is also shown in the histogram according to the parameter thresholdMbc2sbcCoupling. The small values $0.01, \ldots, 1$ are not suited for generating results with a low error, as there is not a single case where thresholdMbc2sbcCoupling is less than 2 in the best 430 simulation runs, as it is depicted in the third subplot of Figure 3.46. The optimal value for the parameter is computed based on the data displayed in the histogram, this results in

$$\text{thresholdMbc2sbcCoupling}_{opt.} = (13 \cdot 2 + 37 \cdot 5 +)\,\frac{1}{50} \approx 4.22.$$

For the parameter thresholdMbcNoInputOpt no clear pattern is recognizable in the fourth subplot of Figure 3.46, the values look arbitrarily distributed. This fact is supported by the fourth histogram in Figure 3.48, every parameter appears at least 6 times, only values 2 and 5 have a higher frequency. The optimal value is computed as the mean of this data, leading to

$$\begin{aligned}\text{thresholdMbcNoInputOpt}_{opt.} = \big(&6 \cdot 0.01 + 6 \cdot 0.05 + 6 \cdot 0.1 + 6 \cdot 0.5 + \ldots \\ &6 \cdot 1 + 8 \cdot 2 + 12 \cdot 5 + \big)\frac{1}{50} \approx 1.72.\end{aligned}$$

Parameter	Value
timeLearning	0.29
timeSwitchphase	0.004
thresholdMbc2sbcCoupling	4.22
thresholdMbcNoInputOpt	1.72
modelOrder	1
windowSize	778
useErrDiffEqu	*false*

Table 3.13: *Optimal set of parameters for the case of Interface Jacobians approximated by MOESP*

How to choose the parameter modelOrder optimally is an easy question, as the pattern in the first subplot of Figure 3.47 is really clear, for every simulation run of the best 500 the model order has been set 1. Therefore the optimal value can be set to $\text{modelOrder}_{opt.} = 1$. For the parameter windowSize the pattern shows that high values are improving the results. From the second histogram in Figure 3.49 it is clear that high values have a positive influence on the analyzed coupling method, the optimal values is computed as the following

$$\text{windowSize}_{opt.} = (2 \cdot 200 + 19 \cdot 500 + 29 \cdot 1000)\,\frac{1}{50} \approx 778.$$

The question whether the Error Differential Equation should be utilized or not is hard to answer from the pattern in third subplot of Figure 3.47, because the best results are reached by utilizing it, but at the majority of the top 50 simulation runs the parameter useErrDiffEqu is set to *false*, as it is depicted in the histogram in Figure 3.49. Therefore the optimal value for useErrDiffEqu is set to *false*. Utilizing the optimal parameters stated in Table 3.13 leads to the simulation results and the corresponding coupling error depicted in Figure 3.50 and 3.51. There it is obvious to see that after the learning phase, the coupling error decreases clearly, as the oscillations in Figure 3.51 are damped.

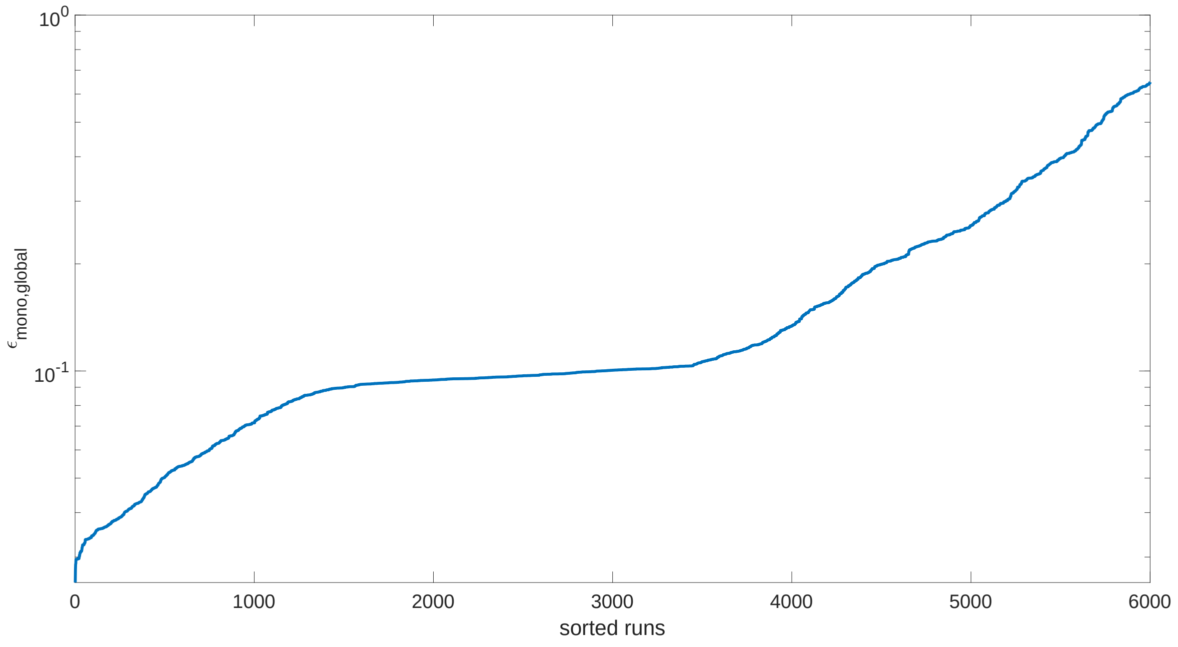


Figure 3.45: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by MOESP - sorted $\epsilon_{mono,global}$*

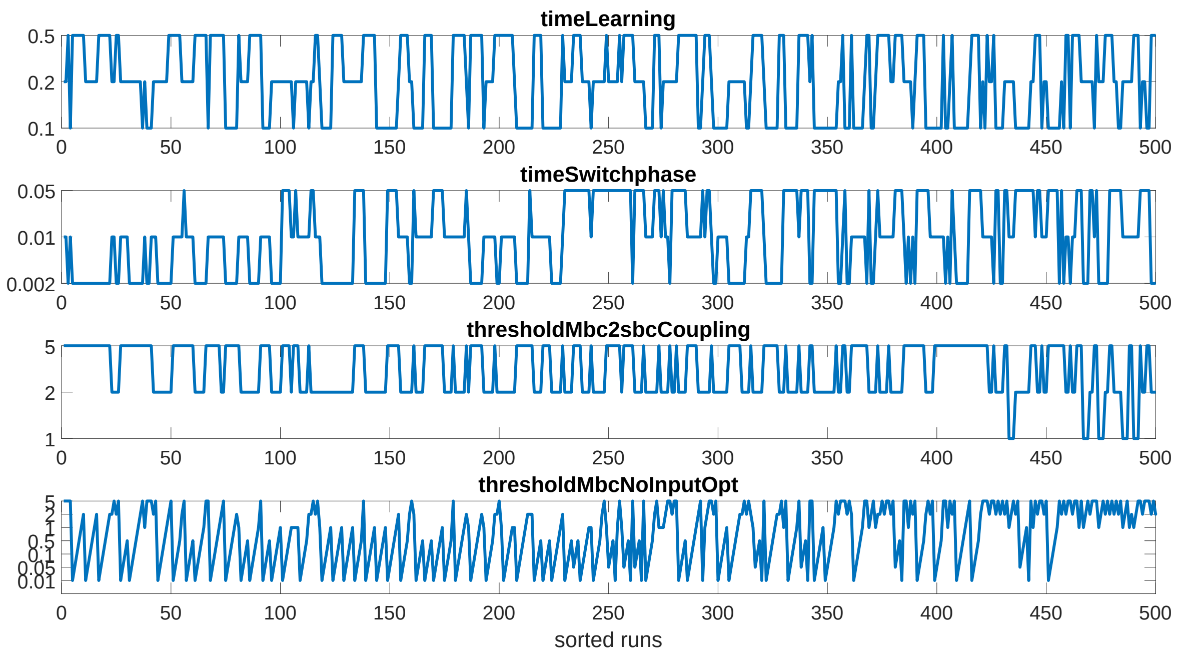


Figure 3.46: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by MOESP - sorted runs of analyzed parameters*

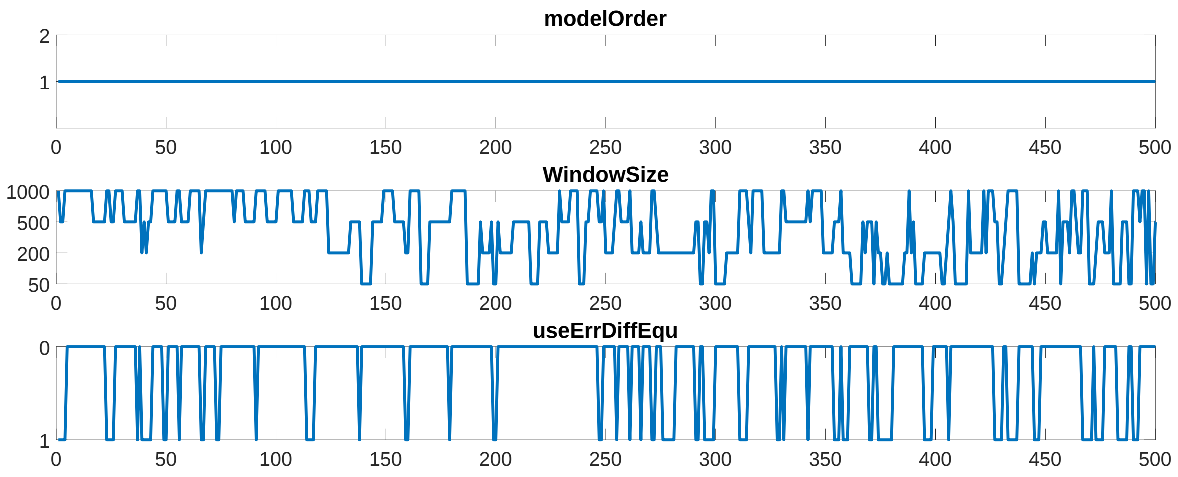


Figure 3.47: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by MOESP - sorted runs of analyzed parameters*

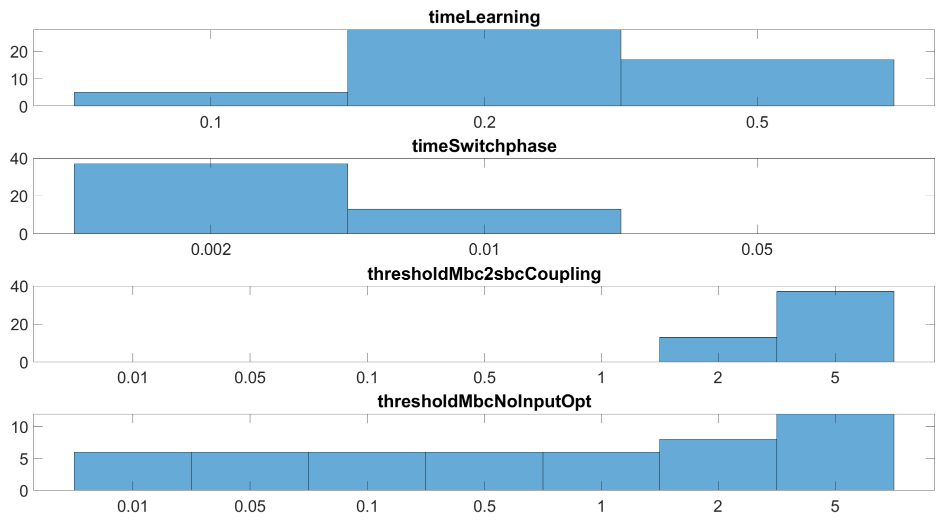


Figure 3.48: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by MOESP - histogram of nearly optimal parameters*

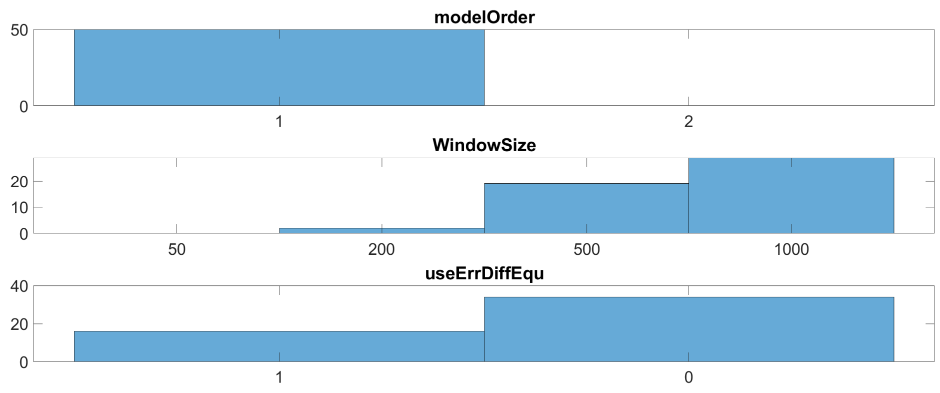


Figure 3.49: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by MOESP - histogram of nearly optimal parameters*

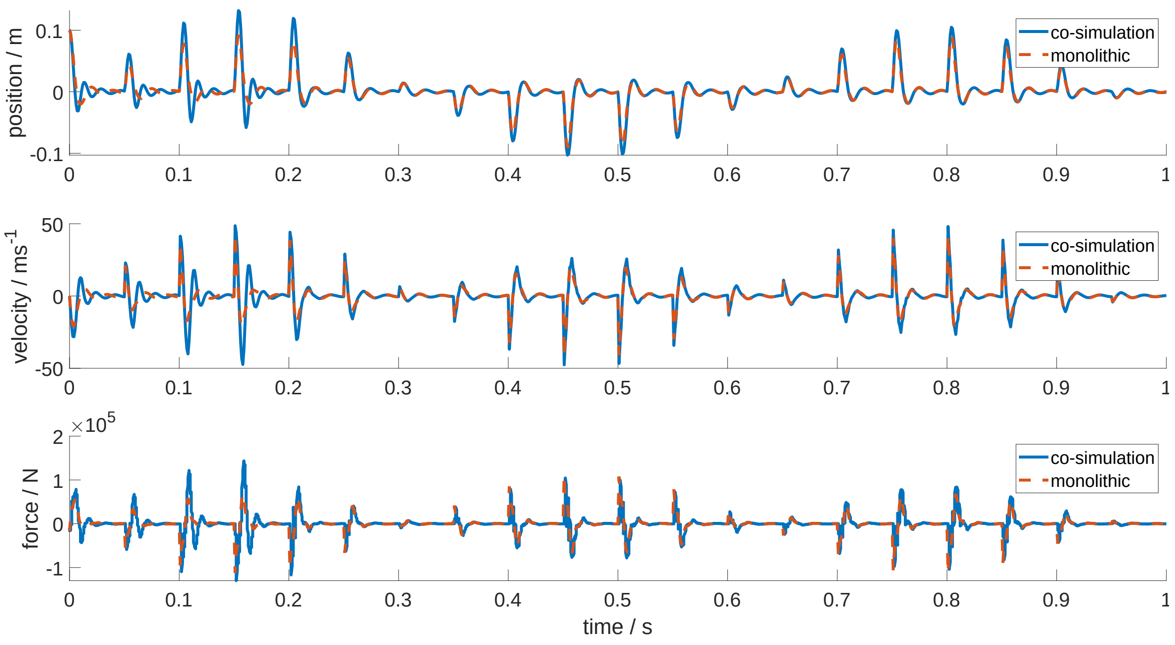


Figure 3.50: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by MOESP - position, velocity and force co-simulated utilizing optimal parameters*

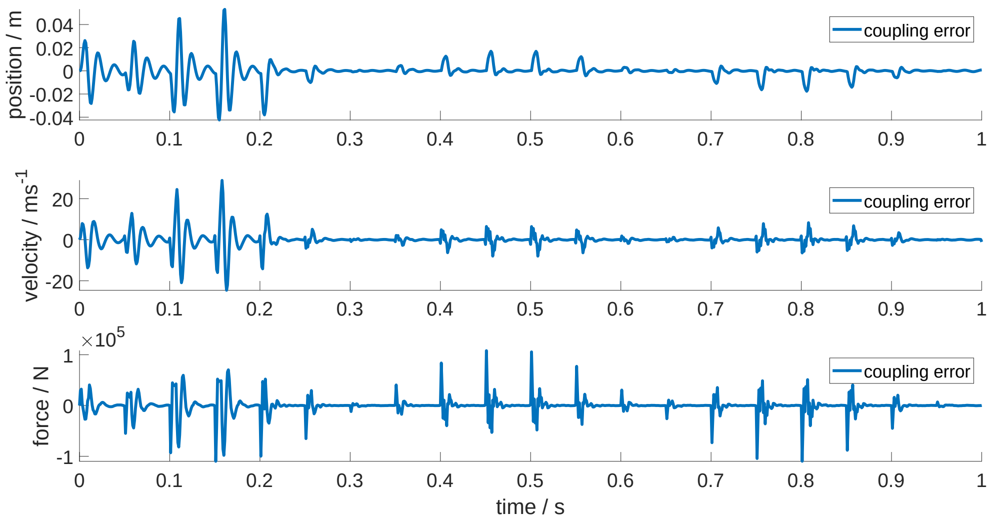


Figure 3.51: *Sensitivity analysis for the Model-based Pre-Step Stabilization Method for Interface Jacobians approximated by MOESP - coupling error of position, velocity and force co-simulated utilizing optimal parameters*

Guideline for Parameters

This guideline for choosing the parameters of the Model-based Pre-Step Stabilization Method, for the case of utilizing approximated Interface Jacobians, will summarize the design of experiment and compare the results of utilizing the RLS and the MOESP method as system identification method.
The parameters timeLearning, timeSwitchphase, thresholdMbc2sbcCoupling, thresholdMbcNoInputOpt, useErrDiffEqu are parameters which are always required, no matter whether the RLS or MOESP is utilized. Therefore these so-called general parameters can be easily compared in the following. The optimal values for timeLearning are in the case of utilizing the RLS $22ms$ and for utilizing the MOESP $29ms$, which is satisfying as those results are quite similar. Additionally the depicted significance, i.e. the main and interaction effects, in Figures 3.19 and 3.29 are in both cases not of high importance. Therefore the choice of timeLearning is independent from the selected system identification method. It should be emphasized, that for a specific example the choice of timeLearning will always depend on the size of the communication step size and the amount of data stored in the signals, because the input signals have to be sufficiently exciting, see for details Section 2.3.3.
The time for switching from the learn phase to the model-based coupling phase is determined by the parameter timeSwitchphase, no matter which system identification method is chosen the value is around $5ms$. Also the influence on the accuracy of the analyzed coupling method is in both cases negligible, i.e. the question of how to choose timeLearning is of minor importance. The value of timeSwitchphase should always be chosen by considering the communication step size, for the analyzed example the optimal value of $5ms$ is determined at a communication step size of $1ms$, which means the switching takes 5 communication steps.
The optimal value of the parameter thresholdMbc2sbcCoupling is nearly independent from the chosen system identification method, as selecting the RLS the optimal value is 3.54 and by choosing the MOESP method it is 4.22. Leading to the conclusion that a high value is the best option for accurate results of the Model-based Pre-Step Stabilization Method.
For the parameter thresholdMbcNoInputOpt it is not that clear, because the optimal value by utilizing the RLS is computed as 0.21 and the selection of the MOESP method is leading to an optimal value of 1.72. The difference in those optimal values is not dramatic, because for both cases the significance of the accuracy of the results is of minor importance, as the main and interaction effects depicted in Figure 3.19, for the RLS, and Figure 3.29, for the MOESP, are really low. Nevertheless it seems that the input optimization is less sensitive against approximation errors in the Interface Jacobians, if the MOESP method is utilized.
For the parameter useErrDiffEqu the optimal value determined by utilizing the RLS is true and for the MOESP it is false, therefore a closer look on the results leading to those values is necessary. Both optimal values are not clearly determined, for both cases the majority for *true* respectively *false* was only around 66%. Therefore there is not a high confidence in the determined optimal values. The heatmap depicted in Figure 3.19 shows that useErrDiffEqu is not significant for the performance of the

coupling method if the RLS is utilized. This is in contrast to the fact that the significance of useErrDiffEqu is stated as 22% if the MOESP is used, as it is stated in Figure 3.29 or Table 3.11. This leads to the conclusion that overall setting useErrDiffEqu to *false* may be the better choice but it can not be guaranteed and may depend on the example.
The parameter λ and windowSize can be compared because both parameters affect the number of considered samples for the online system identification. The RLS is a recursive method and the parameter λ weighted the samples descending, a rule of thumb is that approximately $\frac{1}{1-\lambda}$ samples are of importance for the RLS. For the MOESP method windowSize denotes the samples of the moving window. Comparing the optimal values for λ with 0.996, i.e. ≈ 400 considered samples, and 778 for modelOrder, leads to the fact that in both cases a high number of considered samples are beneficial. It should be pointed out that for the RLS method the significance of λ is negligible, as depicted in Figure 3.19 or Table 3.10. But for the MOESP method the choice of windowSize has slight influence, as one can see in Figure 3.29 or Table 3.11.
The number of states for identifying the required Interface Jacobians is determined by the parameter modelOrder if the MOESP method is utilized and if the RLS is used the parameters M and N imply this information. The parameter modelOrder and M, N can not be compared directly but the determined optimal values show similar results. The best values are for modelOrder 1 and also M and N should be set to 1 for generating the best results. Also the analysis of variance for these parameters shows strong similarities, because the choice of the values are of high significance for the coupling method. For the case of utilizing the RLS the interaction effect of M and N has a very high influence with 59%, also the main effect of M and N is high with 14%.
Comparing the analyzed results of utilizing the RLS and the MOESP for the approximation of the Interface Jacobians leads to the conclusion that the RLS based coupling method is easier to parametrize, because only the choice of M and N is highly significant. For the MOESP method more parameters influence the quality of the results and therefore the parametrization is more complicated. A second aspect is that the learning phase is slightly smaller if the RLS is utilized. Comparing the sorted error measures $\epsilon_{mono,global}$ in Figures 3.38 and 3.45 leads to the conclusion that the lowest $\epsilon_{mono,global}$ is reached by utilizing the MOESP method, but the average of the lowest 1000 or 2000 simulation runs is better if the RLS is utilized. Therefore and all in all, the RLS is the preferred system identification method because it is easier to parametrize and more robust, but the MOESP method is still a fine alternative.

3.3 Stability Analysis

The stability of a method is a crucial property and therefore it should be analyzed in detail. First of all, a clarification of the term stability is necessary, because different stability concepts exist and it should be made clear what kind of stability is investigated in this section. For co-simulation the so-called zero-stability is a fundamental property, because it means that with a convergence of the communication step size $\Delta T \rightarrow 0$ the co-simulation results converges too, see e.g. [1, 29]. Zero-stability can be guaranteed

for the dual mass oscillator because there is no algebraic loop, which can easily be seen because the direct feed-through of S_1 is zero. Zero-stability is a property of the subsystems and the modeling of the subsystems and therefore zero-stability can not be influenced by the coupling method. This means zero-stability is a basic assumption of a co-simulation in general.
The term numerical stability deals with the influence of errors arising from a finite communication step size, which includes for example the coupling error resulting from the unavoidable extrapolation of the subsystem inputs. This kind of stability is highly dependent on the coupling method and is therefore of interest. It should be mentioned, that numerical stability is always connected to a test problem. As the stability depends on the chosen test problem, it is sensible to utilize a widely known benchmark example. For this work the linear dual mass oscillator has been chosen, for details see Section 3.1. The stability analysis will analyze the case of exact Interface Jacobians only, because the approximation of Interface Jacobians always requires a learning phase during which the signal-based coupling would lead to instabilities, which would falsify the analysis. Therefore the focus is put on analysing the stability of the method without the influence of the approximation of the Interface Jacobians.
The stability analysis for the case of approximated Interface Jacobians is carried out regarding exponential and bounded-input bounded-output (BIBO) stability, for details see [25, 41]. The terms exponential and BIBO stability are treated simultaneously as they can be computed based on the same indicator, the so-called Bohl exponent. For free and time invariant systems Lyapunov stability is sufficient, but for time variant systems additional stability concepts exist. Uniform stability means that the transition matrix has to be bounded by an independent constant, this is in contrast to Lyapunov stability where this constant may depend on the initial time. Additionally the term asymptotic stability exist, this means that any perturbed motion tends to the original motion as the time tends to infinity, which means the limit of the transition matrix is the zero matrix as the time tends to infinity. Combining asymptotic and uniform stability, results in uniform asymptotic stable systems and it can be shown that these systems are equivalent to exponential stable systems. Therefore the concept of stability is for time variant systems more complicated as for time invariant ones and exponential stability is the most restrictive stability measure. Regarding disturbances, exponential stability is the most robust stability measure for free systems. For forced systems, i.e. the systems depend on an input, the stability is typically analyzed in the sense of BIBO-stability. This means that the system reacts on a bounded input sequence always with a bounded output sequence.

3.3.1 Numerical Stability

The numerical stability analysis deals with the case of exact Interface Jacobians and therefore the linear dual mass oscillator from Section 3.1 is chosen. As proposed in the previous section the RLS is chosen as system identification method. The parameters of the coupling method are based on the sensitivity analysis from the previous section and are stated in Table 3.5. For comparing the stability analysis of the Model-based Pre-Step Stabilization Method the classical zero-order-hold, the Nearly Energy

Preserving Coupling Element (NEPCE) and the Model-based Corrector (MBCorr) approach have been chosen. The NEPCE method, for details see [6, 7], is a signal based coupling method which compensates the coupling error in delayed manner. As the Model-based Pre-Step Stabilization Method, the MBCorr approach, for details see [5], utilizes information of the subsystems and therefore the MBCorr is an ideal candidate for comparison, as the assumptions and the utilized information is the same as for the Model-based Pre-Step Stabilization Method.
In the analysis the communication step size ΔT is varied from $0.5ms$ to $10ms$, additionally the stiffness c_k and the damping coefficient d_k of the spring-damper, representing the coupling between the two masses, is varied too. Based on every triple $(\Delta T, c_k, d_k)$ a co-simulation is performed, leading to three signals as results, each. These simulation results are the basis for the determination whether the co-simulation is stable or instable.
The triple $(\Delta T, c_k, d_k)$ represents an instable co-simulation, if the amplitude of one of the three results in the last 20% of the simulation time is greater than the amplitude in the first 20% of the simulation time. It should be pointed out that it is sufficient that only one of the three results has to fulfill the condition so that the whole co-simulation and respectively the triple $(\Delta T, c_k, d_k)$ is classified as instable. The stability regions are determined by the communication step size ΔT and the quantity $\frac{c_k}{d_k}$. If $\frac{c_k}{d_k} \to \infty$ this means, that the influence of the stiffness is much higher than of the damping, this can be approximately seen as a rigid connection and for this case the co-simulation is sensitive to extrapolation errors. Therefore an upper boundary according $\frac{c_k}{d_k}$ will exist. Combining (3.4) with the coupling condition $u_1 = y_2$ and $\boldsymbol{u}_2 = \boldsymbol{y}_1$ leads to

$$y_2 = c_k\left(x_2 - x_1\right) + d_k\left(\dot{x}_2 - \dot{x}_1\right),$$

where x_1 and x_2 denote the position of the two masses. Based on this equation one can see that if d_k is much greater than c_k the influence of the position of the masses is negligible for the coupling force y_2. Additionally the influence of x_1 for the subsystem S_2 is of minor importance, as one can see by the definition of $\boldsymbol{B}_2$ in (3.4). Therefore the coupling of x_1 is negligible but y_2 still influences x_1 and therefore instable behaviour according x_1 can be accumulated in S_1. This may lead to inaccurate results or in the worst case this can lead to an instable behaviour of the co-simulation. The choice of c_k and d_k, so that $c_k\left(x_2 - x_1\right)$ is approximately the same as $d_k\left(\dot{x}_2 - \dot{x}_1\right)$ represents the case without stability issues. Due to the fact that $\dot{x}_2 - \dot{x}_1$ is roughly 100 times larger than $x_2 - x_1$ the ratio $\frac{c_k}{d_k} \approx 100$ represents the unproblematic choice of c_k and d_k. Summarizing this means that for $\frac{c_k}{d_k} \to 0$ and $\frac{c_k}{d_k} \to \infty$ problems with the stability of the co-simulation are expected.
The stability regions of all considered coupling methods are depicted in Figure 3.52, for a quantitative comparison the surface area of all four coupling methods are computed as follows. On the x-axis the length unit l_x is logarithmically defined, i.e. between $\frac{c_k}{d_k} = 10^{-2}$ and $\frac{c_k}{d_k} = 10^{3}$ are $5l_x$. On the y-axis the length unit is linear defined, i.e. the difference of $\Delta = 8ms$ and $\Delta = 5ms$ is $3ms$. For zero-order-hold coupling the surface area is determined as $A_{zoh} := 7.5ms\, l_x$, for the NEPCE approach the corresponding stability region has a surface area of $A_{NEPCE} := 10.5ms\, l_x$. Utilizing the MBCorr approach is resulting in a stability region of size $A_{MBCorr} := 19.5ms\, l_x$. The

Model-based Pre-Step Stabilization Method generates the largest stability region, with a surface area of $A_{pre-step} := 62ms\, l_x$. Comparing this stability region to the other three is resulting in the fact, that the utilization of the presented coupling method is increasing the stability region by a factor of $\frac{A_{pre-step}}{A_{zoh}} = \frac{62}{7.5} \approx 8.2$ compared to zero-order-hold coupling. Comparing it to the NEPCE approach is leading to an increased stability region of $\frac{A_{pre-step}}{A_{NEPCE}} = \frac{62}{10.5} \approx 5.9$ and comparing the MBCorr approach with the Model-based Pre-Step Stabilization Method is leading to an increased stability region by a factor of $\frac{A_{pre-step}}{A_{MBCorr}} = \frac{62}{19.5} \approx 3.2$. The enlarged stability region of the presented coupling method can be explained by the fact, that it is the only method which compensates the coupling error in pre step manner.
Additionally one can observe, that there is a strong connection of the communication step size ΔT and the ratio $\frac{c_k}{d_k}$, as the shape of the stability regions for MBCorr, NEPCE and zero-order-hold coupling indicates. For a decreasing communication step size ΔT all stability regions are getting wider, this can be explained as for $\Delta T \to 0$ all combinations of c_k and d_k would lead to a stable co-simulation, independent of the chosen coupling method, as the chosen test problem is zero-stable. Therefore only the upper part of the stability region is of interest. The Model-based Corrector (MBCorr) approach enlarges the stability region compared to the signal-based coupling methods NEPCE and zero-order-hold. This can be explained by the fact that a model-based approach makes use of additional information about the subsystems, which leads to enlarged regions of stability. As NEPCE coupling compensates the extrapolation error, its stability region is enlarged compared to the classical zero-order-hold coupling. These nested stability regions are understandable as with an increase in the complexity of the coupling method the stability should increase, as additional information is exploited.
In Figure 3.53 the stability region utilizing the zero-order-hold coupling approach is depicted. There it can be seen that for $\frac{c_k}{d_k} < 0.2s^{-1}$ and $\frac{c_k}{d_k} > 1000s^{-1}$ all simulations are instable. Additionally one can observe, that by increasing ΔT the stability region narrows. For $\Delta T > 3ms$ no stable combination of c_k and d_k exists. The inverted V-shaped stability region shows clearly the strong connection of ΔT and $\frac{c_k}{d_k}$. The small stable region in the upper right corner arises from alising effects and can therefore be ignored, only connected stability regions are of importance. The NEPCE approach leads to the stability region depicted in Figure 3.54. As before, the inverted V-shape of the stability region is obvious, for $\Delta T > 4ms$ no stable combination of c_k and d_k exists. In Figure 3.55 the stability region of the Model-based Corrector approach is depicted, as before, the inverted V-shape of the stability region is evident. Additionally, one can observe that the upper boundary of the communication step size ΔT is $5ms$. The stability region of the Model-based Pre-Step Stabilization Method is depicted in Figure 3.56, there it is obvious that its shape is completely different from all others, especially in the region where $\frac{c_k}{d_k} < 0.1s^{-1}$ holds. The increased stability for small ratios of $\frac{c_k}{d_k}$ arises from the pre-step manner of the coupling method, because due to the utilization of the Interface Jacobians the instable behaviour triggered through x_1 can be compensated. Additionally, it can be observed that the stability region is nearly independent from the communication step size, as the stability region is of rectangular shape. According to the few stable points in the top, center part of Figure 3.56, a weak

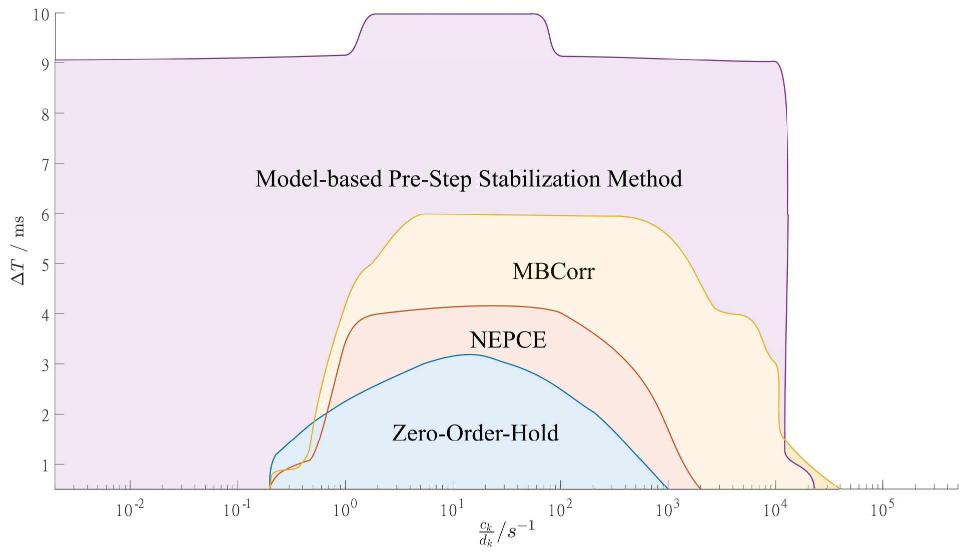


Figure 3.52: *Comparison of the stability regions for different co-simulation techniques*

dependency between ΔT and $\frac{c_k}{d_k}$ exists. As at the previous stability regions, the ratio $\frac{c_k}{d_k} \approx 10s^{-1}$ is more likely to lead to stable co-simulations. The ratio $\frac{c_k}{d_k}$ has an upper boundary for stable results as, none of the coupling methods achieve stable results for $\frac{c_k}{d_k} > 2 \cdot 10^4 s^{-1}$, at least for relevant communication step sizes around $1ms$.
This analysis has proven that with utilization of the Model-based Pre-Step Stabilization Method the numerical stability is improved. This means the co-simulation is stable for a wider range of c_k and d_k, it is only bounded by $\frac{c_k}{d_k} \approx 10^4 s^{-1}$. Especially remarkable is that in contrast to the other coupling methods, the stability region for the Model-based Pre-Step Stabilization Method is nearly independent from the communication step size ΔT, as one can see in Figure 3.52.

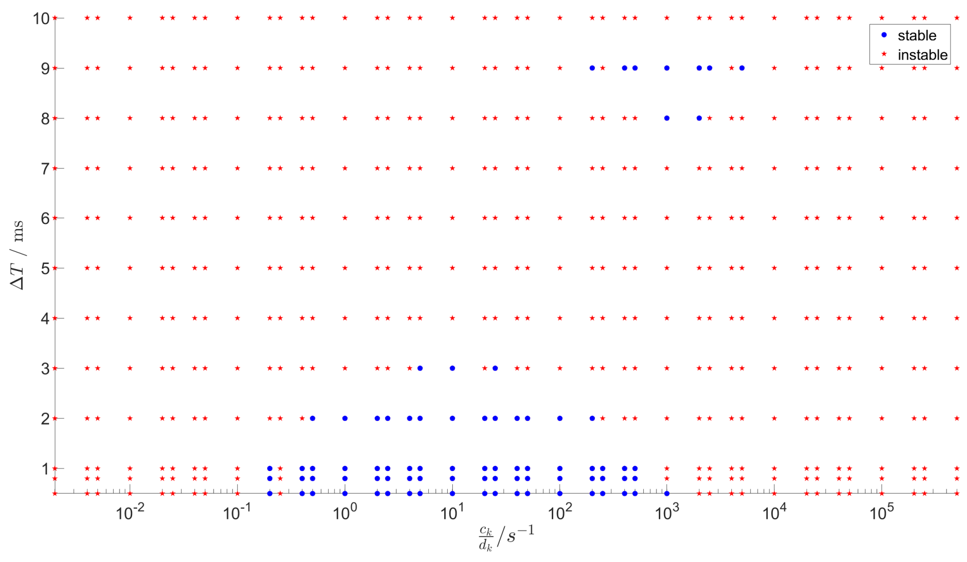


Figure 3.53: *Stability region for zero-order-hold coupling*

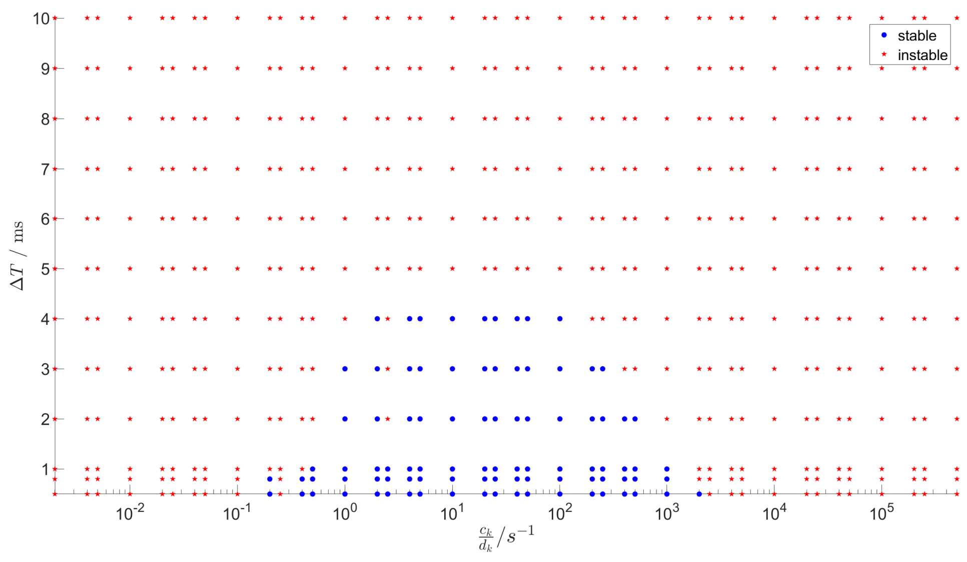


Figure 3.54: *Stability region for the NEPCE method*

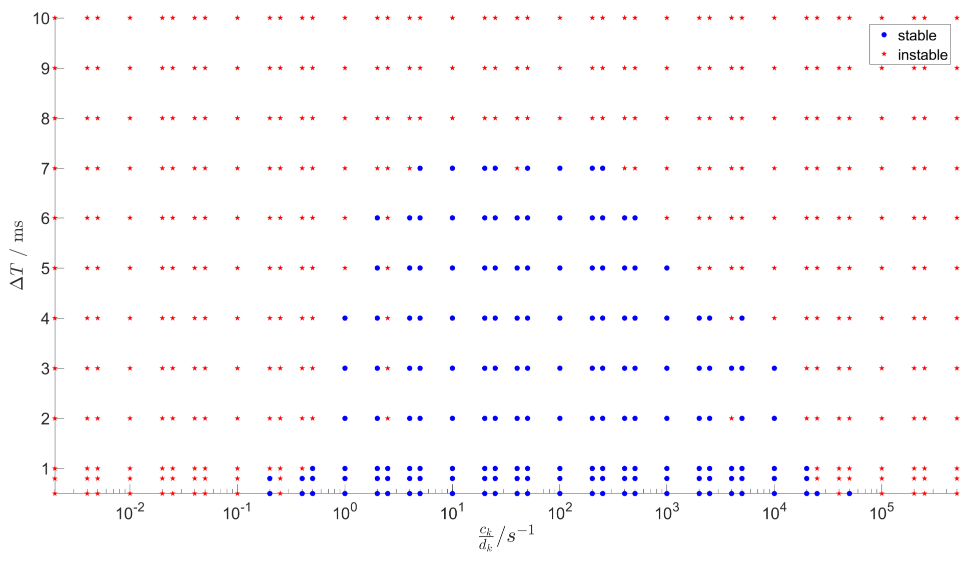


Figure 3.55: *Stability region for the Model-based Corrector Approach*

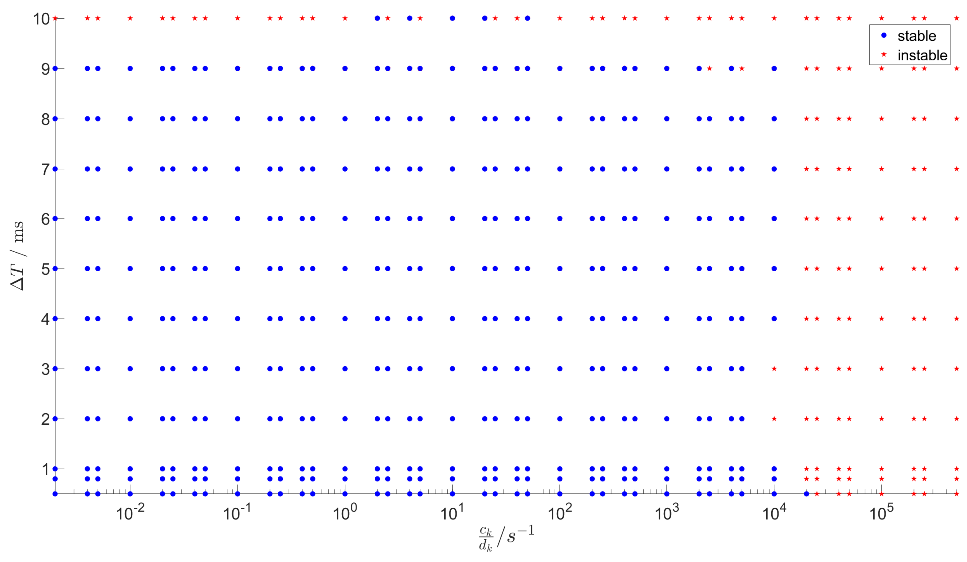


Figure 3.56: *Stability region for the Model-based Pre-Step Stabilization Method*

3.3.2 Exponential and BIBO-Stability

In contrast to the numerical stability analysis in the previous section, the focus of this section is to analyze the Model-based Pre-Step Stabilization Method in terms of exponential stability and BIBO-stability. Due to the fact that both stability concepts are suitable for time variant systems, the case of approximated Interface Jacobians can be analyzed. Therefore the nonlinear dual mass oscillator from Section 3.1.1 is chosen as test problem. This means, that the time variant character of the analyzed system is caused by the nonlinear character of the subsystem itself and additionally by the approximation of the Interface Jacobians, which leads to time varying parameters. As proposed in the previous section, the RLS has been chosen as system identification method. The parametrization of the coupling method is based on the sensitivity analysis of the previous section, the utilized parameters are stated in Table 3.12. All relevant facts and definitions for evaluating exponential and BIBO-stability will be stated in the following and are based on [25].
The discrete-time state space representation without direct feed-through is

$$\begin{aligned} \boldsymbol{x}^{k+1} &= \boldsymbol{\Phi}_{\boldsymbol{A}}^{k}\boldsymbol{x}^{k} + \boldsymbol{\Phi}_{\boldsymbol{B}}^{k}\boldsymbol{u}^{k+1}, \\ \boldsymbol{y}^{k+1} &= \boldsymbol{C}^{k}\boldsymbol{x}^{k+1}. \end{aligned} \tag{3.14}$$

An autonomous, free or unforced system can be written as

$$\boldsymbol{x}^{k+1} = \boldsymbol{\Phi}_{\boldsymbol{A}}^{k}\boldsymbol{x}^{k}. \tag{3.15}$$

Definition 2 (see [41]). *A linear system* (3.14) *is called* exponentially stable*, if there exist reals* $\alpha \geqslant 1$ *and* $\beta \in [0,1)$ *such that*

$$\|\boldsymbol{\Phi}^{k,k_0}\| \leqslant \alpha\beta^{k-k_0} \quad \textit{for all } k_0 \leqslant k. \tag{3.16}$$

The term

$$\boldsymbol{\Phi}^{k,k} := \boldsymbol{I}, \qquad \boldsymbol{\Phi}^{k,k_0} := \boldsymbol{\Phi}_{\boldsymbol{A}}^{k-1}\boldsymbol{\Phi}_{\boldsymbol{A}}^{k-2}\dots\boldsymbol{\Phi}_{\boldsymbol{A}}^{k_0} \text{ for all } k > k_0,$$

describes the transition matrix of (3.15). BIBO-stability means that any bounded input generates a bounded output, more precisely stated in the following definition.

Definition 3 (see [41]). *A linear time-variant system* (3.14) *is called* BIBO-stable*, if from* $sup_{k\in\mathbb{N}}|\boldsymbol{u}_k| < \infty$ *follows* $sup_{k\in\mathbb{N}}|\boldsymbol{y}_k| < \infty$.

The following theorem connects BIBO-stability and exponential stability.

Theorem 3.3.1 (see [41]). *A linear system* (3.14) *is BIBO-stable if* $\boldsymbol{\phi}_{\boldsymbol{A}}^{k}, \boldsymbol{\phi}_{\boldsymbol{B}}^{k}$ *and* $\mathbf{C}_k$ *are bounded in* k *and the free system* (3.15) *is exponential stable.*

As it is stated in (3.16), exponential stability is defined for one system and therefore, the first step is to combine the two subsystems S_1 and S_2 of the non-linear dual mass oscillator and the coupling element *CoupEle*, depicted in Figure 3.57, to one system $S_1 + S_2 + CoupEle$. The introduced subsystem *CoupEle* represents the Model-based

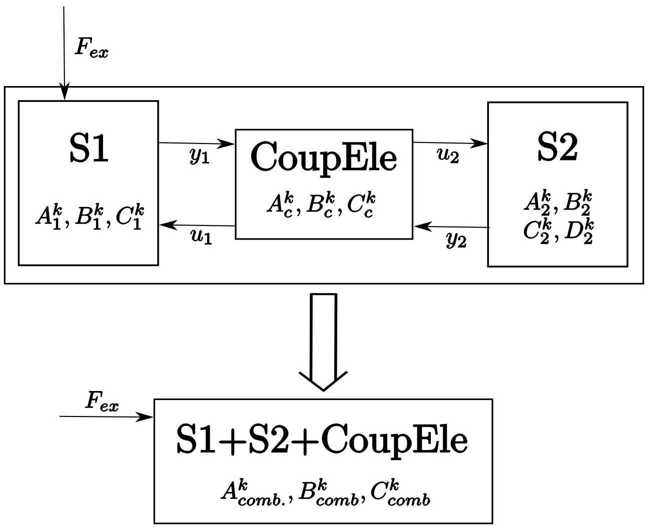


Figure 3.57: *Rearranging of the co-simulation of two subsystems S_1, S_2 coupled via the Coupling Element CoupEle into one system $S1 + S2 + CoupEle$*

Pre-Step Stabilization Method and its state space representation will be derived in the following.

Due to the fact that the exponential stability in [41] is defined for state space representation only, S_1, S_2 and *CoupEle* will be stated in discrete-time time-variant state space representations:

$$S1:$$

$$\boldsymbol{x}_1^{k+1} = \boldsymbol{\Phi}_{\boldsymbol{A}_1}^k \boldsymbol{x}_1^k + \boldsymbol{\Phi}_{\boldsymbol{B}_1}^k \boldsymbol{u}_1^{k+1} + \boldsymbol{\Phi}_{\boldsymbol{B}_{ex.}}^k F_{ex.}^{k+1}, \tag{3.17}$$

$$\boldsymbol{y}_1^{k+1} = \boldsymbol{C}_1^k \boldsymbol{x}_1^{k+1}, \tag{3.18}$$

$$S2:$$

$$\boldsymbol{x}_2^{k+1} = \boldsymbol{\Phi}_{\boldsymbol{A}_2}^k \boldsymbol{x}_2^k + \boldsymbol{\Phi}_{\boldsymbol{B}_2}^k \boldsymbol{u}_2^{k+1}, \tag{3.19}$$

$$\boldsymbol{y}_2^{k+1} = \boldsymbol{C}_2^k \boldsymbol{x}_2^{k+1} + \boldsymbol{D}_2^k \boldsymbol{u}_2^{k+1}, \tag{3.20}$$

$$CoupEle:$$

$$\boldsymbol{x}_c^{k+1} = \boldsymbol{\Phi}_{\boldsymbol{A}_c}^k \boldsymbol{x}_c^k + \boldsymbol{\Phi}_{\boldsymbol{B}_c}^k \boldsymbol{u}_c^{k+1}, \tag{3.21}$$

$$\boldsymbol{y}_c^{k+1} = \boldsymbol{C}_c^k \boldsymbol{x}_c^{k+1}. \tag{3.22}$$

The subsystems S_1 and S_2 represent the non-linear dual mass oscillator and are defined in Section 3.1.1. In the following the representation of *CoupEle* will be derived. The discrete-time method in Section 2.2.2 is the basis for this derivation.
The first step is the approximation of the monolithic output by the computation of $\boldsymbol{\delta}^k$ in

$$\boldsymbol{\delta}^k = (\boldsymbol{I} - \tilde{\boldsymbol{C}}_d^k)^{-1} \left[\tilde{\boldsymbol{A}}_d^k \boldsymbol{\delta}^{k-1} + \tilde{\boldsymbol{B}}_d^k \boldsymbol{\epsilon}^{k-1} + \tilde{\boldsymbol{D}}_d^k \boldsymbol{\epsilon}^k \right],$$

where the coupling error is defined as $\boldsymbol{\epsilon}^k = \boldsymbol{L}\boldsymbol{y}^k - \boldsymbol{u}^k$, inserting this in the equation above results in

$$\boldsymbol{\delta}^k = (\boldsymbol{I} - \tilde{\boldsymbol{C}}_d^k)^{-1} \left[\tilde{\boldsymbol{A}}_d^k \boldsymbol{\delta}^{k-1} + \tilde{\boldsymbol{B}}_d^k \left(\boldsymbol{L}\boldsymbol{y}^{k-1} - \boldsymbol{u}^{k-1} \right) + \tilde{\boldsymbol{D}}_d^k \left(\boldsymbol{L}\boldsymbol{y}^k - \boldsymbol{u}^k \right) \right]. \tag{3.23}$$

The model-based extrapolation of the monolithic output $\boldsymbol{y}^k + \boldsymbol{\delta}^k$ can be computed as

$$\hat{\boldsymbol{y}}^{k+1} = (\boldsymbol{I} - \hat{\boldsymbol{B}}^k)^{-1}\hat{\boldsymbol{A}}^k \left(\boldsymbol{y}^k + \boldsymbol{\delta}^k\right). \tag{3.24}$$

From inserting (2.71) in (2.70) the input optimization for two subsystems can be written as

$$\begin{aligned}\underbrace{\begin{pmatrix}\boldsymbol{u}_1^{k+1}\\ \boldsymbol{u}_2^{k+1}\end{pmatrix}}_{=:\boldsymbol{u}^{k+1}} = \hat{\boldsymbol{y}}^{k+1} - \frac{1}{2}\underbrace{\begin{pmatrix}\left(\frac{\partial S_{d_1}}{\partial \boldsymbol{u}^{k+1}}\right)^T\left(\frac{\partial S_{d_1}}{\partial \boldsymbol{u}^{k+1}}\left(\frac{\partial S_{d_1}}{\partial \boldsymbol{u}^{k+1}}\right)^T\right)^{-1} & \boldsymbol{0}\\ \boldsymbol{0} & \left(\frac{\partial S_{d_2}}{\partial \boldsymbol{u}^{k+1}}\right)^T\left(\frac{\partial S_{d_2}}{\partial \boldsymbol{u}^{k+1}}\left(\frac{\partial S_{d_2}}{\partial \boldsymbol{u}^{k+1}}\right)^T\right)^{-1}\end{pmatrix}}_{=:\boldsymbol{Mat}_1}\cdot\ldots\\ \cdot\left[\underbrace{\begin{pmatrix}\frac{\partial S_{d_1}}{\partial \boldsymbol{u}^{k+1}} - 2 & \boldsymbol{0}\\ \boldsymbol{0} & \frac{\partial S_{d_2}}{\partial \boldsymbol{u}^{k+1}} - 2\end{pmatrix}}_{=:\boldsymbol{Mat}_2}\hat{\boldsymbol{y}}^{k+1} + 2\underbrace{\begin{pmatrix}\frac{\partial S_{d_1}}{\partial \boldsymbol{y}^{k}} & \boldsymbol{0}\\ \boldsymbol{0} & \frac{\partial S_{d_2}}{\partial \boldsymbol{y}^{k}}\end{pmatrix}}_{=:\boldsymbol{Mat}_3}\boldsymbol{y}^k + 2\underbrace{\begin{pmatrix}\frac{\partial S_{d_1}}{\partial \boldsymbol{u}^{k}} & \boldsymbol{0}\\ \boldsymbol{0} & \frac{\partial S_{d_2}}{\partial \boldsymbol{u}^{k}}\end{pmatrix}}_{=:\boldsymbol{Mat}_4}\boldsymbol{u}^k\right].\end{aligned} \tag{3.25}$$

Inserting (3.23) in (3.24) results in

$$\begin{aligned}\hat{\boldsymbol{y}}^{k+1} &= (\boldsymbol{I} - \hat{\boldsymbol{B}}^k)^{-1}\hat{\boldsymbol{A}}^k\Big(\boldsymbol{y}^k + (\boldsymbol{I} - \tilde{\boldsymbol{C}}_d^k)^{-1}\cdot\ldots\\ &\cdot\Big[\tilde{\boldsymbol{A}}_d^k\boldsymbol{\delta}^{k-1} + \tilde{\boldsymbol{B}}_d^k\left(\boldsymbol{L}\boldsymbol{y}^{k-1} - \boldsymbol{u}^{k-1}\right) + \tilde{\boldsymbol{D}}_d^k\left(\boldsymbol{L}\boldsymbol{y}^{k} - \boldsymbol{u}^{k}\right)\Big]\Big).\end{aligned} \tag{3.26}$$

Combining the input optimization in (3.25) with the model-based extrapolation in (3.26) leads to

$$\begin{aligned}\boldsymbol{u}^{k+1} = &\underbrace{\left(\boldsymbol{I} - \frac{1}{2}\boldsymbol{Mat}_1\boldsymbol{Mat}_2\right)(\boldsymbol{I} - \hat{\boldsymbol{B}}^k)^{-1}\hat{\boldsymbol{A}}^k}_{=:\boldsymbol{Mat}_5}\Big(\boldsymbol{y}^k + (\boldsymbol{I} - \tilde{\boldsymbol{C}}_d^k)^{-1}\cdot\ldots\\ &\cdot\Big[\tilde{\boldsymbol{A}}_d^k\boldsymbol{\delta}^{k-1} + \tilde{\boldsymbol{B}}_d^k\left(\boldsymbol{L}\boldsymbol{y}^{k-1} - \boldsymbol{u}^{k-1}\right) + \tilde{\boldsymbol{D}}_d^k\left(\boldsymbol{L}\boldsymbol{y}^{k} - \boldsymbol{u}^{k}\right)\Big]\Big) + \ldots\\ &-\frac{1}{2}\boldsymbol{Mat}_1\left[\boldsymbol{Mat}_3\boldsymbol{y}^k + \boldsymbol{Mat}_4\boldsymbol{u}^k\right].\end{aligned}$$

After rearranging the equation reads as

$$\begin{aligned}
\boldsymbol{u}^{k+1} = &\underbrace{-\boldsymbol{Mat}_5(\boldsymbol{I}-\tilde{\boldsymbol{C}}_d^k)^{-1}\tilde{\boldsymbol{B}}_d^k}_{=:\boldsymbol{Mat}_{\boldsymbol{u}^{k-1}\to\boldsymbol{u}^{k+1}}}\boldsymbol{u}^{k-1} + \dots \\
&\underbrace{\left(-\boldsymbol{Mat}_5(\boldsymbol{I}-\tilde{\boldsymbol{C}}_d^k)^{-1}\tilde{\boldsymbol{D}}_d^k - \frac{1}{2}\boldsymbol{Mat}_1\boldsymbol{Mat}_4\right)}_{=:\boldsymbol{Mat}_{\boldsymbol{u}^{k}\to\boldsymbol{u}^{k+1}}}\boldsymbol{u}^{k} + \dots \\
&+\underbrace{\boldsymbol{Mat}_5(\boldsymbol{I}-\tilde{\boldsymbol{C}}_d^k)^{-1}\tilde{\boldsymbol{A}}_d^k}_{=:\boldsymbol{Mat}_{\boldsymbol{\delta}^{k-1}\to\boldsymbol{u}^{k+1}}}\boldsymbol{\delta}^{k-1} + \dots \\
&+\underbrace{\boldsymbol{Mat}_5(\boldsymbol{I}-\tilde{\boldsymbol{C}}_d^k)^{-1}\tilde{\boldsymbol{B}}_d^k\boldsymbol{L}}_{=:\boldsymbol{Mat}_{\boldsymbol{y}^{k-1}\to\boldsymbol{u}^{k+1}}}\boldsymbol{y}^{k-1} + \dots \\
&+\underbrace{\left(\boldsymbol{Mat}_5\left(\boldsymbol{I}+(\boldsymbol{I}-\tilde{\boldsymbol{C}}_d^k)^{-1}\tilde{\boldsymbol{D}}_d^k\boldsymbol{L}\right) - \frac{1}{2}\boldsymbol{Mat}_1\boldsymbol{Mat}_3\right)}_{=:\boldsymbol{Mat}_{\boldsymbol{y}^{k}\to\boldsymbol{u}^{k+1}}}\boldsymbol{y}^{k}.
\end{aligned} \tag{3.27}$$

CoupEle can be described as the state space system

$$\begin{pmatrix}\boldsymbol{u}^k\\ \boldsymbol{u}^{k+1}\\ \boldsymbol{\delta}^k\end{pmatrix} = \boldsymbol{\Phi}_{\boldsymbol{A}_c}^k\begin{pmatrix}\boldsymbol{u}^{k-1}\\ \boldsymbol{u}^{k}\\ \boldsymbol{\delta}^{k-1}\end{pmatrix} + \boldsymbol{\Phi}_{\boldsymbol{B}_c}^k\begin{pmatrix}\boldsymbol{y}^{k-1}\\ \boldsymbol{y}^{k}\end{pmatrix}, \tag{3.28}$$

with

$$\begin{pmatrix}\boldsymbol{u}^{k-1}\\ \boldsymbol{u}^{k}\\ \boldsymbol{\delta}^{k-1}\end{pmatrix}$$

as states and

$$\begin{pmatrix}\boldsymbol{y}^{k-1}\\ \boldsymbol{y}^{k}\end{pmatrix}$$

as inputs. The state matrix

$$\boldsymbol{\Phi}_{\boldsymbol{A}_c}^k := \begin{pmatrix}\boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0}\\ \boldsymbol{Mat}_{\boldsymbol{u}^{k-1}\to\boldsymbol{u}^{k+1}} & \boldsymbol{Mat}_{\boldsymbol{u}^{k}\to\boldsymbol{u}^{k+1}} & \boldsymbol{Mat}_{\boldsymbol{\delta}^{k-1}\to\boldsymbol{u}^{k+1}}\\ -(\boldsymbol{I}-\tilde{\boldsymbol{C}}_d^k)^{-1}\tilde{\boldsymbol{B}}_d^k & -(\boldsymbol{I}-\tilde{\boldsymbol{C}}_d^k)^{-1}\tilde{\boldsymbol{D}}_d^k & (\boldsymbol{I}-\tilde{\boldsymbol{C}}_d^k)^{-1}\tilde{\boldsymbol{A}}_d^k\end{pmatrix}$$

and the input matrix

$$\boldsymbol{\Phi}_{\boldsymbol{B}_c}^k := \begin{pmatrix}\boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{Mat}_{\boldsymbol{y}^{k-1}\to\boldsymbol{u}^{k+1}} & \boldsymbol{Mat}_{\boldsymbol{y}^{k}\to\boldsymbol{u}^{k+1}}\\ (\boldsymbol{I}-\tilde{\boldsymbol{C}}_d^k)^{-1}\tilde{\boldsymbol{B}}_d^k\boldsymbol{L} & (\boldsymbol{I}-\tilde{\boldsymbol{C}}_d^k)^{-1}\tilde{\boldsymbol{D}}_d^k\boldsymbol{L}\end{pmatrix}$$

can be derived from (3.23) and (3.27). The output equation from *CoupEle* reads as

$$\boldsymbol{u}^{k+1} = \underbrace{\begin{pmatrix} \mathbf{0} & \boldsymbol{I} & \mathbf{0} \end{pmatrix}}_{=:\boldsymbol{C}_c^k} \begin{pmatrix} \boldsymbol{u}^k \\ \boldsymbol{u}^{k+1} \\ \boldsymbol{\delta}^k \end{pmatrix}. \tag{3.29}$$

The subsystems S_1, S_2 and *CoupEle* will be combined into one system, as it is shown in Figure 3.57. Therefore the subsystems S_1 and S_2 are combined and rearranged, leading to

$$\begin{aligned} \boldsymbol{y}^{k+1} = &\begin{pmatrix} \boldsymbol{C}_1^k \boldsymbol{\Phi}_{\boldsymbol{A}_1}^k & \mathbf{0} \\ \mathbf{0} & \boldsymbol{C}_2^k \boldsymbol{\Phi}_{\boldsymbol{A}_2}^k \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_1^k \\ \boldsymbol{x}_2^k \end{pmatrix} + \dots \\ &+ \underbrace{\begin{pmatrix} \boldsymbol{C}_1^k \boldsymbol{\Phi}_{\boldsymbol{B}_1}^k & \mathbf{0} \\ \mathbf{0} & \boldsymbol{C}_2^k \boldsymbol{\Phi}_{\boldsymbol{B}_2}^k + \boldsymbol{D}_2^k \end{pmatrix}}_{=:\boldsymbol{Mat}_{CB}^k} \boldsymbol{u}^{k+1} + \begin{pmatrix} \boldsymbol{\Phi}_{\boldsymbol{B_{ex.}}}^k \\ \mathbf{0} \end{pmatrix} F_{ex.}^{k+1}. \end{aligned}$$

The output of *CoupEle*, the quantity $\boldsymbol{u}^{k+1}$, is the input for S_1 and S_2 and therefore (3.28) and (3.29) is inserted in the equation above, resulting in

$$\begin{aligned} \boldsymbol{y}^{k+1} = &\begin{pmatrix} \boldsymbol{C}_1^k \boldsymbol{\Phi}_{\boldsymbol{A}_1}^k & \mathbf{0} \\ \mathbf{0} & \boldsymbol{C}_2^k \boldsymbol{\Phi}_{\boldsymbol{A}_2}^k \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_1^k \\ \boldsymbol{x}_2^k \end{pmatrix} + \boldsymbol{Mat}_{CB}^k \boldsymbol{C}_c^k \boldsymbol{\Phi}_{\boldsymbol{A}_c}^k \begin{pmatrix} \boldsymbol{u}^{k-1} \\ \boldsymbol{u}^k \\ \boldsymbol{\delta}^{k-1} \end{pmatrix} + \dots \\ &+ \boldsymbol{Mat}_{CB}^k \boldsymbol{C}_c^k \boldsymbol{\Phi}_{\boldsymbol{B}_c}^k \begin{pmatrix} \boldsymbol{y}^{k-1} \\ \boldsymbol{y}^k \end{pmatrix} + \begin{pmatrix} \boldsymbol{\Phi}_{\boldsymbol{B_{ex.}}}^k \\ \mathbf{0} \end{pmatrix} F_{ex.}^{k+1}. \end{aligned}$$

The states of the combined system $S1 + S2 + CoupEle$ can be written as

$$\begin{pmatrix} \boldsymbol{u}^{k-1} \\ \boldsymbol{u}^k \\ \boldsymbol{\delta}^{k-1} \\ \boldsymbol{y}^{k-1} \\ \boldsymbol{y}^k \\ \boldsymbol{x}_1^k \\ \boldsymbol{x}_2^k \end{pmatrix}.$$

Therefore the system can be stated as

$$\begin{pmatrix} \boldsymbol{u}^k \\ \boldsymbol{u}^{k+1} \\ \boldsymbol{\delta}^k \\ \boldsymbol{y}^k \\ \boldsymbol{y}^{k+1} \\ \boldsymbol{x}_1^{k+1} \\ \boldsymbol{x}_2^{k+1} \end{pmatrix} = \boldsymbol{\phi}_{\boldsymbol{A_{comb}}}^k \begin{pmatrix} \boldsymbol{u}^{k-1} \\ \boldsymbol{u}^k \\ \boldsymbol{\delta}^{k-1} \\ \boldsymbol{y}^{k-1} \\ \boldsymbol{y}^k \\ \boldsymbol{x}_1^k \\ \boldsymbol{x}_2^k \end{pmatrix} + \underbrace{\begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \boldsymbol{\Phi}_{\boldsymbol{B_{ex.}}}^k \\ \mathbf{0} \end{pmatrix}}_{=:\phi_{\boldsymbol{B_{comb}}}^k} F_{ex.}^{k+1}, \tag{3.30}$$

$len_{rate}[\,]\ \backslash t_{sim}\ [sec]$	1.0	2.0	3.0	4.0	5.0
0.25	0.9990	0.9846	0.9843	0.9838	0.9820
0.5	0.9870	0.9830	0.9800	0.9811	0.9789
0.75	0.9811	0.982	0.9808	0.9800	0.9800
1	0.9825	0.9801	0.9789	0.9790	0.9792

Table 3.14: *Computed Bohl exponents for the co-simulation of the nonlinear dual mass oscillator coupled with the Model-based Pre-Step Stabilization Method*

with

$$\boldsymbol{\phi}^k_{\boldsymbol{A_{comb}}} := \begin{pmatrix} & & & & & \mathbf{0} & \mathbf{0} \\ & \boldsymbol{\Phi}^k_{\boldsymbol{A}_c} & & & \boldsymbol{\Phi}^k_{\boldsymbol{B}_c} & \mathbf{0} & \mathbf{0} \\ & & & & & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{I} & \mathbf{0} & \mathbf{0} \\ & \boldsymbol{Mat}^k_{CB}\boldsymbol{C}^k_c\boldsymbol{\phi}^k_{\boldsymbol{A}_c} & & & \boldsymbol{Mat}^k_{CB}\boldsymbol{C}^k_c\boldsymbol{\phi}^k_{\boldsymbol{B}_c} & \boldsymbol{C}^k_1\boldsymbol{\phi}^k_{\boldsymbol{A}_1} & \boldsymbol{C}^k_2\boldsymbol{\phi}^k_{\boldsymbol{A}_2} \\ \mathbf{0} & \begin{pmatrix} \boldsymbol{\phi}^k_{\boldsymbol{B}_1} \\ \mathbf{0} \end{pmatrix} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{\phi}^k_{\boldsymbol{A}_1} & \mathbf{0} \\ \mathbf{0} & \begin{pmatrix} \mathbf{0} \\ \boldsymbol{\phi}^k_{\boldsymbol{B}_2} \end{pmatrix} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{\phi}^k_{\boldsymbol{A}_2} \end{pmatrix}.$$

The output equation of $S1 + S2 + CoupEle$ reads as

$$\boldsymbol{y}^{k+1} = \underbrace{\begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{I} & \mathbf{0} & \mathbf{0} \end{pmatrix}}_{\boldsymbol{C}^k_{\boldsymbol{comb}}} \begin{pmatrix} \boldsymbol{u}^k \\ \boldsymbol{u}^{k+1} \\ \boldsymbol{\delta}^k \\ \boldsymbol{y}^k \\ \boldsymbol{y}^{k+1} \\ \boldsymbol{x}_1^{k+1} \\ \boldsymbol{x}_2^{k+1} \end{pmatrix}.$$

The system in (3.30) will be analyzed according to exponential stability. To evaluate whether a system is exponential stable or not the so-called Bohl exponent

$$\overline{\beta}(\boldsymbol{\Phi}^k_{\boldsymbol{A}}) := \limsup_{i\to\infty} \sup_{k\geqslant 0} \left|\left|\boldsymbol{\Phi}^{k+i,k}\right|\right|^{\frac{1}{i}}$$

can be utilized, because the condition

$$\overline{\beta}(\boldsymbol{\Phi}^k_{\boldsymbol{A}}) < 1$$

is sufficient and necessary that a system (3.14) is exponential stable, see therefore [42]. The Bohl exponent $\overline{\beta}(\boldsymbol{\phi}^k_{\boldsymbol{A_{comb}}})$ is determined with a numerical robust method, which is stated in [25]. The computed Bohl exponents are stated in Table 3.14, the simulation time t_{sim} and the interval length rate len_{rate} are varied. The transient time is set to 0.5 seconds, the interval length len is then determined via $len = len_{rate}(t_{sim} - 0.5)$. From the fact that $\overline{\beta}(\boldsymbol{\phi}^k_{\boldsymbol{A_{comb}}}) \approx 0.98$ the system (3.30) is exponential stable. Due to Theorem 3.3.1 and the fact that (3.30) is exponential stable and that the matrices $\boldsymbol{\phi}^k_{\boldsymbol{A_{comb}}}, \boldsymbol{\phi}^k_{\boldsymbol{B_{comb}}}, \boldsymbol{\phi}^k_{\boldsymbol{C_{comb}}}$ are bounded for all k, it follows that $S1 + S2 + CoupEle$ is a BIBO-stable system.

3.4 Accuracy Analysis

Although the main focus of the Model-based Pre-Step Stabilization Method is to improve the performance of a co-simulation in terms of stability, the accuracy of the co-simulation should not be ignored. Therefore the accuracy of the invented coupling method will be compared to the classical zero-order-hold, the Nearly Energy Preserving Coupling Element and the Model-based Corrector approach in the following. Due to the fact, that it is of interest how the accuracy of the co-simulation depends on the stiffness c_k and damping d_k of the spring, representing the coupling between the two masses, and the communication step size ΔT, the chosen example is the linear dual mass oscillator from Section 3.1. Therefore the continuous version of the Model-based Pre-Step Stabilization Method, with exact Interface Jacobians is utilized, the parameters of the coupling method are the optimized ones, stated in Table 3.5. The measure for the accuracy is the error $\epsilon_{mono,global}$ defined in (3.9), this means all three coupled signals, the position x_1 and velocity $\dot{x}_1$ of the mass in subsystem 1 and the coupling force F between the two masses are considered with equal weight.
In Figure 3.58 the accuracy of zero-order-hold coupling dependent on the communication step size ΔT and the rate $\frac{c_k}{d_k}$ is depicted. The quantity $\frac{c_k}{d_k}$ has been chosen for the same reasons as described in Section 3.2.2. It should be pointed out, that the color of the points in the figure represents $\epsilon_{mono,global}$, the maximum is set to 1 because $\epsilon_{mono,global}$ is a relative error and therefore a differentiation between errors over 100% is negligible. One can see in Figure 3.58 that zero-order-hold coupling generates results with a good accuracy for $\frac{c_k}{d_k} \in [0.5s^{-1}, 1000s^{-1}]$ for $\Delta T \leqslant 1ms$. For $\Delta T = 2ms$ or $\Delta T = 3ms$ the range of $\frac{c_k}{d_k}$ narrows symmetrically around $\frac{c_k}{d_k} = 10$. For a communication step size larger than $3ms$ no sufficiently accurate result is possible. As an exception the results with $\Delta T = 10ms$ and $\frac{c_k}{d_k} \in [50s^{-1}, 2000s^{-1}]$ act but these results can be ignored, because they origin from aliasing effects and therefore those results are of no interest. The accuracy plot of the NEPCE method in Figure 3.59 looks similar to the one of the zero-order-hold coupling, the only difference is that also for $\Delta T = 4ms$ and $\Delta T = 5ms$ sufficiently accurate results exist. This means that due to the utilization of the NEPCE method the usage of a higher communication step size is possible. As supposed, the usage of model-based coupling methods leads to an increased accuracy, as it is depicted for the Model-based Corrector approach in Figure 3.60. There it is obvious that the area of sufficiently accurate results is enlarged, there accurate results exists up to $\Delta T = 8ms$. As it has been detected in the stability analysis in Section 3.3.1, there is an interaction between the communication step size ΔT and $\frac{c_k}{d_k}$, as the inverted V-shape of the region of accurate results in Figures 3.58 - 3.60 indicates. The region of accurate results for the Model-based Pre-Step Stabilization Method is depicted in Figure 3.61, there it is obvious that the area is larger than the ones before. This means, that the analyzed coupling method has the effect, that for a wider range of $\frac{c_k}{d_k}$, a large communication step size can be chosen and the results will still be accurate. Therefore the Model-based Pre-Step Stabilization Method is not as sensitive for the choice of the communication step size ΔT as all other mentioned coupling methods. Additionally, the interaction of ΔT and $\frac{c_k}{d_k}$ is less than compared with the other coupling methods, as the region of accurate results is approximately

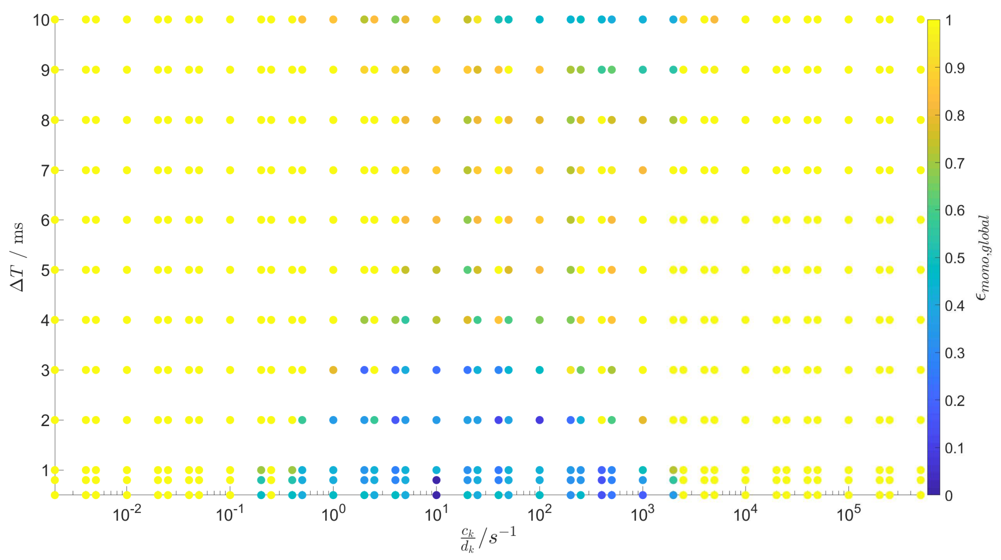


Figure 3.58: *Accuracy of zero-order-hold coupling depending on ΔT and $\frac{c_k}{d_k}$*

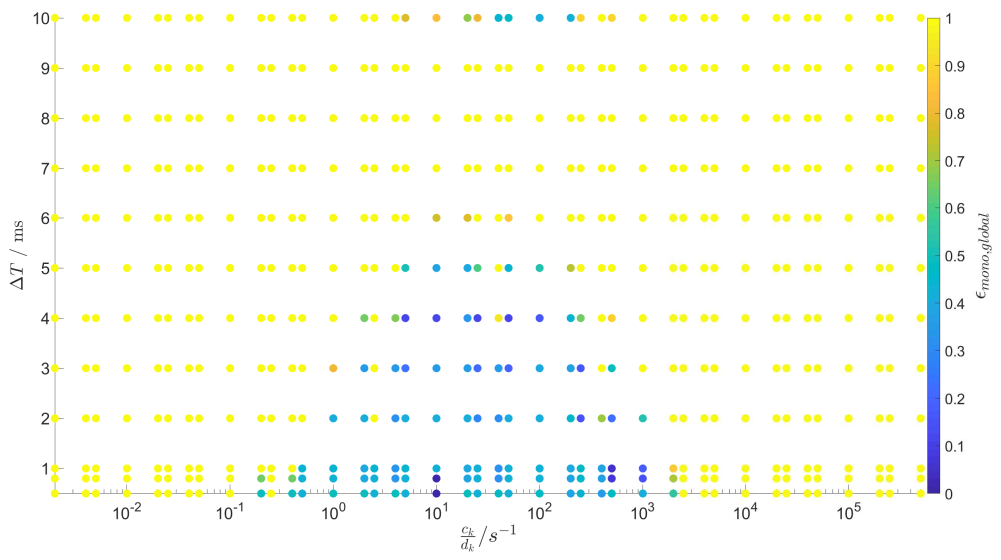


Figure 3.59: *Accuracy of NEPCE coupling depending on ΔT and $\frac{c_k}{d_k}$*

rectangular.
The progression of $\epsilon_{mono,global}$ for fixed values of $c_k = 2e5Nm^{-1}$ and $d_k = 200Nsm^{-1}$ and a varying communication step size are depicted in Figure 3.62, for all four mentioned coupling methods. There again it should be denoted that the maximum error is set to 1. So one can observe that the zero-order-hold coupling (ZOH) and the NEPCE coupling are inaccurate for $\Delta T \geqslant 2ms$. The declining for ZOH coupling for $\Delta T > 7ms$ origins from aliasing effects and can therefore be ignored. The model-based

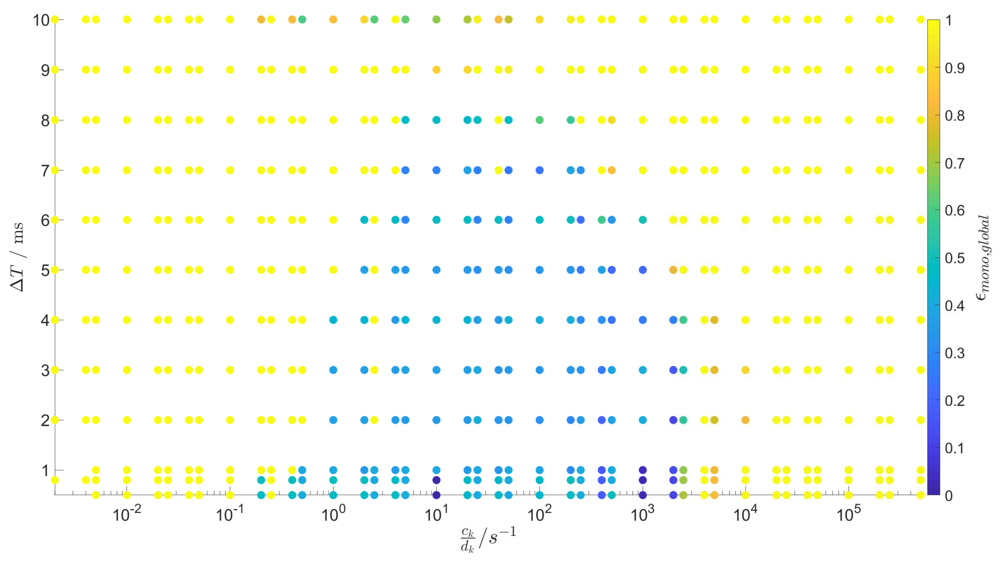


Figure 3.60: *Accuracy of the Model-based Corrector approach depending on* ΔT *and* $\frac{c_k}{d_k}$

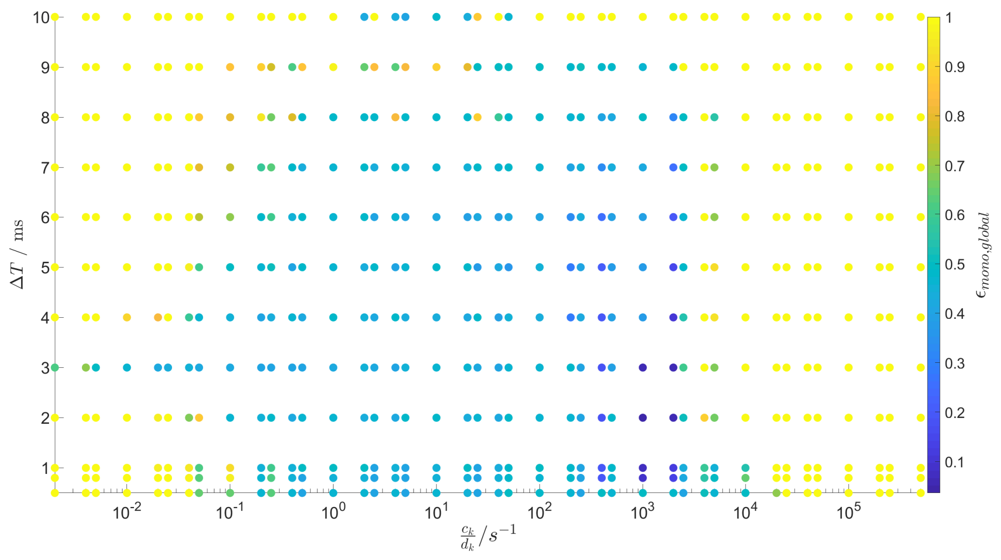


Figure 3.61: *Accuracy of the Model-based Pre-Step Stabilization Method depending on* ΔT *and* $\frac{c_k}{d_k}$

coupling methods show an obvious increase by the border of accurate results for the communication step size. For the Model-based Corrector approach accurate results are possible for $\Delta T = 5ms$ and for the Model-based Pre-Step Stabilization Method even up to $9ms$. Additionally it should be denoted that for all $\Delta T > 2ms$ the error $\epsilon_{mono,global}$ of the Pre-Step method is obviously smaller than the error of the MBCorr

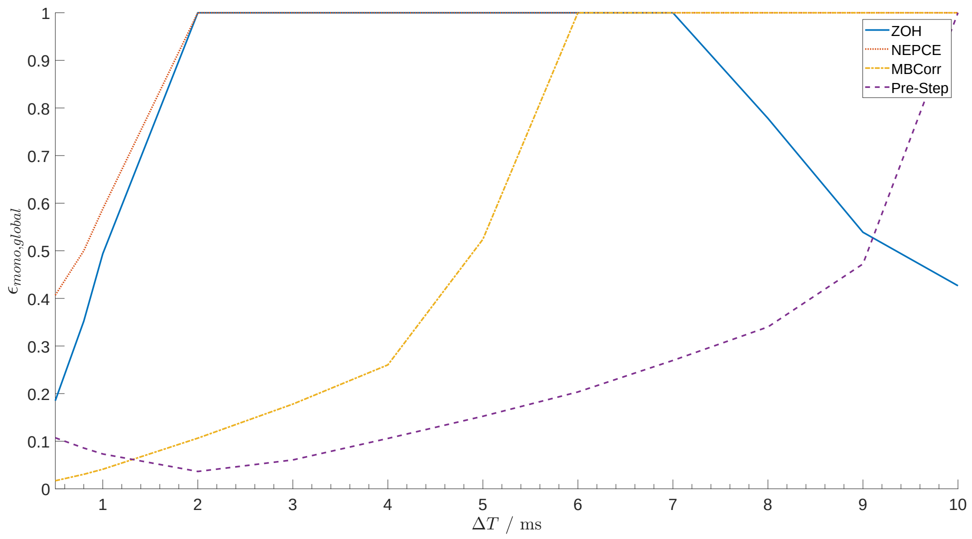


Figure 3.62: *Comparison of $\epsilon_{mono,global}$ of different co-simulation coupling methods at varying communication step size ΔT for $c_k = 2e5$ and $d_k = 200$*

method. Communication step sizes smaller than $0.5ms$ are of no interest for the analyzed method, due to two reasons. First, the invented co-simulation coupling method is a stabilization method and is therefore designed to handle large communication step sizes. Second, the classical zero-order-hold coupling and other signal based coupling methods generate for small communication step sizes sufficiently accurate results and therefore there is no need for improvements. Summarizing this means, that the improvements by utilizing the Model-based Pre-Step Stabilization Method are depicted in Figure 3.62, in the area between the MBCorr result and the Pre-Step result from $[2ms, 10ms]$.
For $\Delta T = 1ms$ the position x_1, the velocity $\dot{x}_1$ and the coupling force F signal are depicted in Figure 3.63, there all four different coupling methods and the monolithic results, acting as reference solution, are shown. There one can see that both signal-based coupling methods, the ZOH and the NEPCE method are instable and therefore the results are of no use. Both model-based methods show similar results, both are equally and satisfyingly accurate. The results of MBCorr and the Pre-Step method by enlarging the communication step size to $\Delta T = 2ms$ are depicted in 3.64. There it is obvious that both results are quite accurate but especially in the force signal there is a clear difference between the two results. The Pre-Step result precedes and the MBCorr result follows the monolithic result. For a closer look on this phenomenon a zoom in Figure 3.65 is given. There one can see the difference between the two model-based approaches clearly. In the communication point at $t = 0.02s$ the Pre-Step Method computes an approximation of the monolithic result and then optimizes the input in such a way that at $t = 0.02s + \Delta T = 0.022s$ the co-simulated result coincides with the approximated monolithic result. Due to the direct feed-through term in the second subsystem of the dual mass oscillator, see (3.4), this optimized input effects instan-

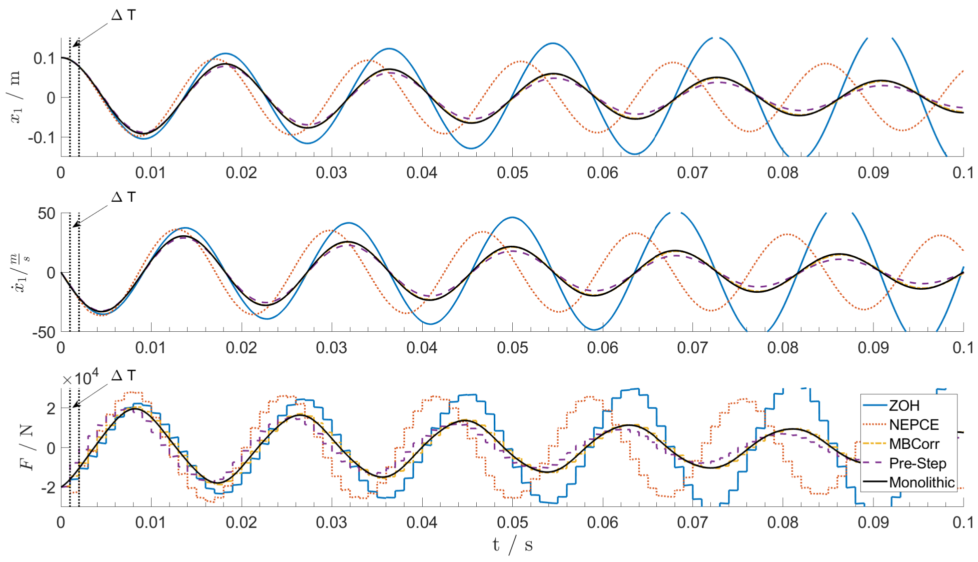


Figure 3.63: *Comparison of position x_1, velocity $\dot{x}_1$ and coupling force F of different co-simulation coupling methods at $\Delta T = 1ms$ for $c_k = 2e5$ and $d_k = 200$*

taneously the subsystem resulting in the discontinuous gap in co-simulation result at $t = 0.02s$, therefore the Pre-Step method precedes the monolithic result. The Model-based Corrector approach tries to preserve in the co-simulated result the same energy as in the monolithic result. This means that the integral over $[0.02s, 0.022s]$ from the monolithic and co-simulated result should coincide, i.e. the two shaded regions in Figure 3.65 should be from equal size. This phenomenon is a natural property of these coupling methods and does not origin from a special choice of the example or of the parameters. Figure 3.66 depicts the result of the two model-based coupling methods with a communication step size $\Delta T = 6ms$. There it is obvious that the MBCorr result is instable and is therefore of no use. The result from the Model-based Pre-Step Stabilization Method is still stable and from sufficient accuracy, taken into account that $\Delta T = 6ms$ is a large choice for this example.

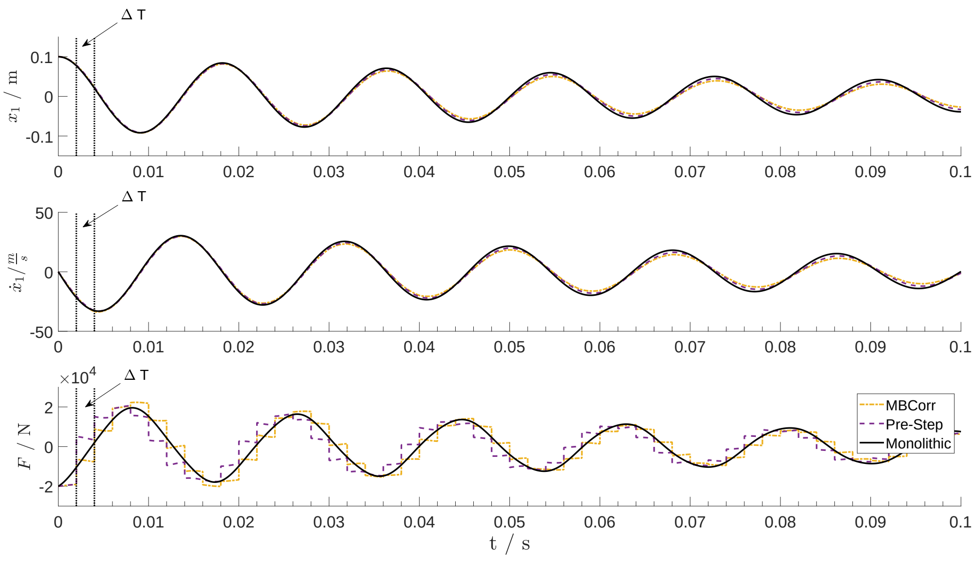


Figure 3.64: *Comparison of position x_1, velocity $\dot{x}_1$ and coupling force F of the Model-based Corrector and the Model-based Pre-Step Stabilization Method at $\Delta T = 2ms$ for $c_k = 2e5$ and $d_k = 200$*

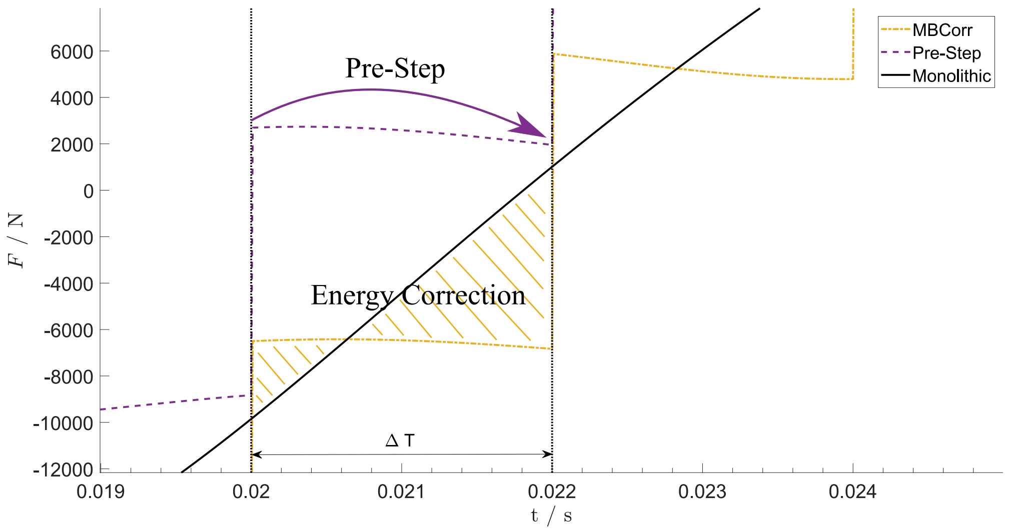


Figure 3.65: *Comparison of the Model-based Pre-Step Stabilization method and the Model-based Corrector approach*

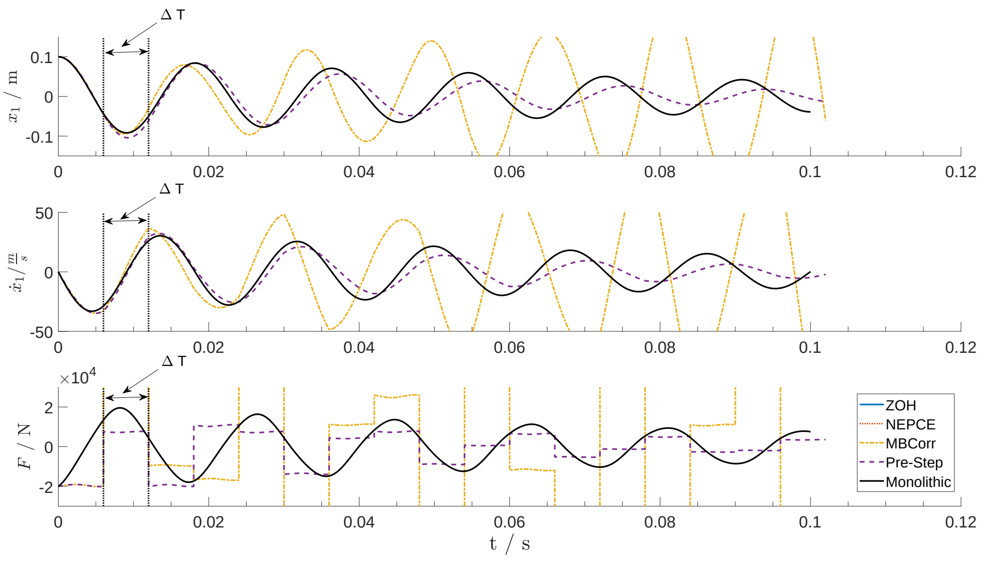


Figure 3.66: *Comparison of position x_1, velocity $\dot{x}_1$ and coupling force F of the Model-based Corrector and the Model-based Pre-Step Stabilization Method at $\Delta T = 6ms$ for $c_k = 2e5$ and $d_k = 200$*

4 Evaluation

The evaluation of a new method is the crucial part to demonstrate the improved performance. There the advantages and benefits of the introduced method should be made clear, obviously and reasonably.

This chapter deals with the evaluation of the Model-based Pre-Step Stabilization Method, it is compared to state of the art signal- and model-based coupling methods. As evaluation example the co-simulation of a helicopter and its controller has been chosen. In the following, the setup of the co-simulation will be stated and explained in detail, including a description of both subsystems, the helicopter and the controller. Additional information and further details about the helicopter and its controller can be found in [43, 44, 45, 46]. The results of the co-simulation will be analyzed in detail, with a focus on the stability and accuracy for large communication step sizes.

4.1 Helicopter-Control Co-Simulation

In this section the co-simulation of a helicopter and its controller will be treated. The helicopter and the controller model will be stated and discussed in the following, both models origin from [44] and additional details about the helicopter and its controller can be found in [43, 45, 46]. The aircraft model represents a Westland Lynx with around 4500 kg gross weight, which is used for multiple purposes in military. The helicopter consists of a twin engine and a four blade semi-rigid main rotor. The controller is from type H_∞ and the pilot is simulated by stating the reference commandos which are required for the controller. The utilized design is called full-authority-controller, which means that the controller has the full control over all blade angels. More details about the helicopter and the controller will be stated in the following subsections.

In the following the various reasons for choosing this example for the evaluation of the Model-based Pre-Step Stabilization Method will be discussed:
(1) Although the helicopter and its controller represent a complex system the exact state space representations are accessible. Therefore the influence of approximated and exact Interface Jacobians can be analyzed and studied in detail.
(2) The co-simulation of a helicopter is challenging because the natural, physical

dynamics of a helicopter are strongly interconnected, i.e. the states have strong mutual couplings, and a helicopter is in general an instable system. Therefore the unavoidable extrapolation error of a co-simulation can be critical in terms of accuracy and stability.
(3) This co-simulation is stiff, because the stiffness of the monolithic approach is determined by

$$\frac{\max|\lambda(\boldsymbol{A_{mono}})|}{\min|\lambda(\boldsymbol{A_{mono}})|} \approx 3.3e5,$$

where $|\lambda(\boldsymbol{A_{mono}})|$ denotes the absolute value of the eigenvalues of the monolithic state matrix $\boldsymbol{A_{mono}}$. Due to the fact that a co-simulation can be seen as an integration method, like the well-known Explicit Euler method, the expression stiff co-simulation origins from the theory of numerical solvers for ordinary differential equations. For classical integration methods the stability depends on the sampling rate or the step size of the integration and the so-called stiffness of the example, the higher the stiffness the lower the sampling rate has to be. Typical as example the well-known Dahlquist Equation $\dot{x} = \lambda x$ is chosen and for this case the stiffness is represented by the parameter λ. For general examples the stiffness is determined by the quotient of the highest absolute eigenvalue divided by the smallest one $\frac{\lambda_{max}}{\lambda_{min}}$. Due to this quotient, it is possible to evaluate the stiffness of a co-simulation, based on the eigenvalues of the monolithic state matrix. This determined stiffness of the co-simulation represents an upper boundary of the sampling rate, in the case of co-simulation the sampling rate is the communication step size ΔT. From an other point of view, a stiff system consists of fast and slow dynamics and to keep the system stable the sampling rate has to be chosen according to the fast dynamic.
Due to these reasons, this co-simulation example is well chosen for evaluating the introduced coupling method. In the upper part of Figure 4.1 the standard control circuit is shown, which represents the helicopter and its control. The pilot is represented by the specification of the commandos $\boldsymbol{r}$, therefore this term is called pilot commandos. The lower part in this figure shows the co-simulation, the controller is stated as subsystem S_2 and the helicopter is represented by the subsystem S_1. The block between the two subsystems represents the co-simulation coupling, this means every subsystem has their own solver and is only communicating with each other at discrete time points, the communication time points. The communication between the subsystems is exclusive by exchanging its output signals $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$. The co-simulation coupling will be executed in parallel and therefore both subsystem inputs $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ have to be extrapolated. The data exchange is for both subsystems at the same communication time points, the time between two time points is denoted as the communication step size ΔT and is constant for the whole simulation.
Due to the fact that the influence of the approximation of the required Interface Jacobians will be investigated, the continuous-time version, for the utilization of exact Interface Jacobians, and the discrete-time version, for the use of approximated Interface Jacobians, of the Model-based Pre-Step Stabilization Method will be applied. All needed settings and parameters for both versions of the coupling method are stated and derived in Section 3.2, and can be found in Table 3.12 for the discrete-time version

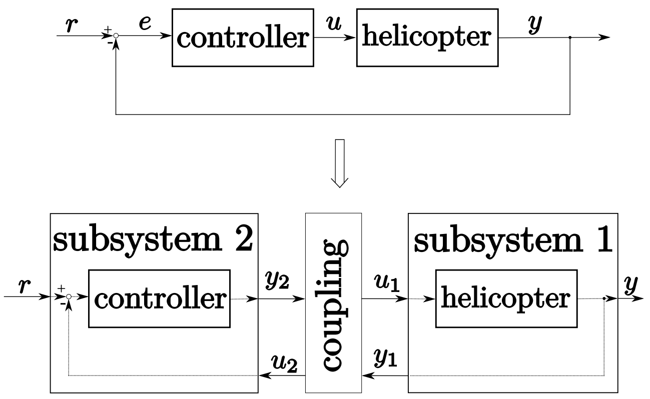


Figure 4.1: *Control circuit of the helicopter transformed into a co-simulation of two subsystems*

and in Table 3.5 for the continuous-time version. Due to the fact that the timeLearning and the timeSwitchphase are parameters which depend on the chosen communication step size ΔT, they have to be adjusted to fit the different communication step size in the helicopter control co-simulation. In the sensitivity analysis $\Delta T = 1ms$ and for the helicopter control co-simulation the communication step size is around $\Delta T = 100ms$ and therefore the chosen value of timeLearning with $12s$ is of the same character as $0.22s$ for the dual mass oscillator, the same holds for the timeSwitchphase with an chosen value of $1s$ for the following example. The simulation end time is for all runs set to $150s$.

4.1.1 Helicopter Model

The modeling of a helicopter is a challenging task as the dynamics of the rotors and aerodynamics are highly coupled and sensitive. As described and discussed in [43], the modeling can be differentiated into three levels with a growing complexity of the model. The level 1 modeling consists of rigid blades and a linear 2-D model of aerodynamics. By increasing the complexity for a level 2 modeled helicopter, a number of blade elastic are considered, e.g. the second order rotor flapping and coning modes, are included. The aerodynamics are then described as nonlinear 3-D equations and local effects of the blades are considered. In a level 3 modeled helicopter the rotor is modeled in detail, with all elastic modes and the aerodynamics are considering turbulences and therefore a full wake analysis is carried out. The more complex the helicopter is, the closer the simulated results are to the real flight behaviour. Depending on the task one is interested in, the level of modeling should be chosen. For example if one is interested in the design of the rotor, only a level 3 modeled helicopter will provide helpful data. If one is interested to design a controller or conduct a parametric study of the flight quality for example, a level 1 helicopter model will be sufficient. The helicopter model used in this co-simulation is based on the non-linear Rationalised Helicopter Model from [43], which is level 2 modeled. As it is shown in [43] experimental data has proven that the Rationalised Helicopter Model is of sufficient accuracy. The model itself is instable due to strong cross couplings, which origin from the strong coupling of the natural dynamics of a helicopter. As the focus lies on the evaluation of the Model-based Pre-Step Stabilization Method, a linearized state space model of the Rationalised Helicopter Model is sufficiently, as shown in [44, 46]. The linearized model is a reasonable choice, as the focus of the analysis will lie on the comparison of the co-simulated and the monolithic results and not to real flight data. In the following the linearized state space model of the helicopter will be stated and discussed. The subsystem 1 inherits the linear continuous-time time-invariant state space model

$$\dot{\boldsymbol{x}}_1 = \boldsymbol{A}_1\boldsymbol{x}_1 + \boldsymbol{B}_1\boldsymbol{u}_1, \tag{4.1}$$

$$\boldsymbol{y}_1 = \boldsymbol{C}_1\boldsymbol{x}_1. \tag{4.2}$$

with

$$\boldsymbol{x}_1 := \begin{pmatrix} \theta \\ \phi \\ p \\ q \\ \xi \\ v_x \\ v_y \\ v_z \end{pmatrix} \tag{4.3}$$

where θ denotes the pitch and ϕ the roll attitude. The states p and q stand for the roll and pitch rate and ξ describes the yaw rate, with v_x, v_y, v_z the velocity of the helicopter in forward, lateral and vertical direction is denoted. Figure 4.2 describes the orientation of the helicopter, it should be mentioned that the positive z-direction is pointing towards the ground. The numerical values of the matrices $\boldsymbol{A}_1, \boldsymbol{B}_1$ and $\boldsymbol{C}_1$ can be found in [47]. Computing the eigenvalues $\lambda_{\boldsymbol{A}_1}$ of $\boldsymbol{A}_1$ is resulting in $\max(\mathrm{Re}(\lambda_{\boldsymbol{A}_1})) = 0.2342$, which means that the helicopter is an instable system. The output of the helicopter $\boldsymbol{y}_1$ consists of

$$\boldsymbol{y}_1 := \begin{pmatrix} \dot{H} \\ \theta \\ \phi \\ \dot{\psi} \\ p \\ q \end{pmatrix}, \tag{4.4}$$

where $\dot{H}$ stands for the heave velocity and $\dot{\psi}$ for the heading rate. The outputs of the helicopter $\boldsymbol{y}_1$ are divided into controller outputs $\boldsymbol{y_{ctrl}} := \left(\dot{H}, \theta, \phi, \dot{\psi}\right)^T$ and rate signals $\boldsymbol{y_{rate}} := (p, q)^T$. The input of the helicopter $\boldsymbol{u}_1$ are the four blade angels

- main rotor collective,
- longitudinal collective,
- lateral cyclic,
- tail rotor collective,

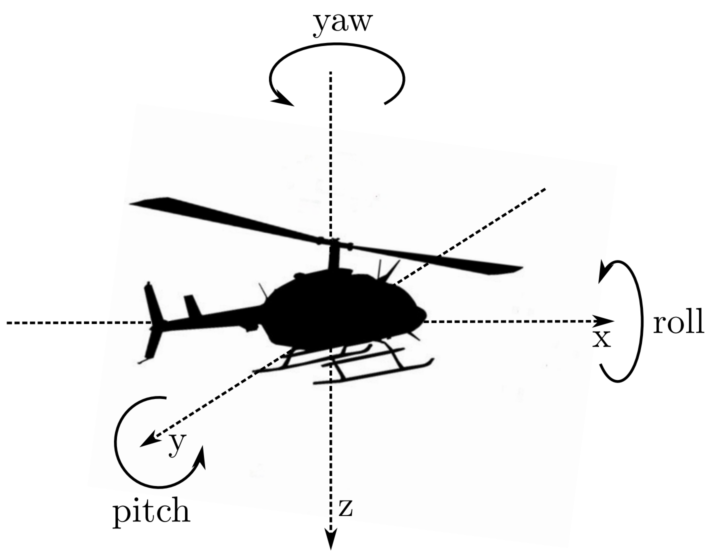


Figure 4.2: *Illustration of the three movement directions of the helicopter*

who are determined by the controller, which will be discussed in detail in the next section. In the following the effect of the four blade angels to the motion of the helicopter will be discussed. The main rotor collective changes all the blade angels at the same time and therefore is mainly controlling the lift of the helicopter, represented mainly by the heave velocity $\dot{H}$ of the helicopter. The longitudinal collective changes the angels of the blade in such a way that the helicopter moves longitudinally, therefore the helicopter is tilting which is mainly described by the pitch θ of the helicopter. The lateral cyclic is responsible for the movement of the helicopter in lateral direction, therefore the angels of the blade are changed in such a way that mainly the roll is influenced. The tail rotor collective is used to compensate the torque generated by the main rotor and stops the helicopter from spinning around, it can also be utilized to move the helicopter in lateral direction. It should be emphasized that the motion of a helicopter is not as decoupled as it has been described, the dynamics are actually highly coupled.
As one part of the evaluation will be to investigate the influence of utilizing a system identification for approximating the required Interface Jacobians, the exact Interface Jacobians are needed too. Based on the numerical values of $\boldsymbol{A}_1, \boldsymbol{B}_1$ and $\boldsymbol{C}_1$ from [47] the exact Interface Jacobians $\frac{\partial S_1}{\partial \boldsymbol{y}}$, $\frac{\partial S_1}{\partial \boldsymbol{u}}$ and $\frac{\partial S_1}{\partial \dot{\boldsymbol{u}}}$ can be computed by applying the

transformation rules (2.12) and (2.13), resulting in

$$\frac{\partial S_1}{\partial \boldsymbol{y}} = \begin{pmatrix} -2.91e-01 & 1.37e-04 & 1.70e-05 & 4.45e-02 & -1.51e-01 & 1.69e-01 \\ 0.00e+00 & 0.00e+00 & 0.00e+00 & 5.34e-02 & 0.00e+00 & 1.00e+00 \\ 0.00e+00 & 0.00e+00 & 0.00e+00 & 5.95e-02 & 1.00e+00 & 1.16e-06 \\ -1.38e-03 & 0.00e+00 & 0.00e+00 & -7.35e-01 & -2.06e+00 & -3.91e-01 \\ -5.79e-03 & 0.00e+00 & 0.00e+00 & -6.36e-02 & -1.16e+01 & -2.55e+00 \\ 3.16e-03 & 0.00e+00 & 0.00e+00 & -3.29e-16 & 4.39e-01 & -2.00e+00 \end{pmatrix} \tag{4.5}$$

$$\frac{\partial S_1}{\partial \boldsymbol{u}} = \begin{pmatrix} 4.82e+00 & -3.11e-02 & -3.00e-02 & 1.25e-02 \\ 0.00e+00 & 0.00e+00 & 0.00e+00 & 0.00e+00 \\ 0.00e+00 & 0.00e+00 & 0.00e+00 & 0.00e+00 \\ 3.06e-01 & -1.05e-02 & -4.97e-01 & -2.07e-01 \\ 1.24e-01 & 8.28e-02 & -2.75e+00 & -1.79e-02 \\ -3.64e-02 & 4.75e-01 & 1.43e-02 & 0.00e+00 \end{pmatrix}. \tag{4.6}$$

Due to the fact, that the direct feed-through term of helicopter is zero it is clear that $\frac{\partial S_1}{\partial \dot{\boldsymbol{u}}} = \boldsymbol{0}$ holds, see therefore (2.14). The subsystem is solved with an numerical solver, the explicit Euler method with a step size of $2ms$, for details of the chosen numerical solver, see for example [40]. The initial conditions are $\boldsymbol{x}(\boldsymbol{0}) = \boldsymbol{0}$, which represents that the helicopter is standing on the ground and both rotors are standing still.

4.1.2 Controller Model

Control theory is a huge field of research with a long history and mostly the aim of the controller is to damp disturbances from the outside, or trying to follow a given reference track, as accurate as possible. Due to the assumptions and requirements different control strategies can be applied, e.g. adaptive control, optimal control, stochastic control or robust control. Robust control is the part which will be in the focus of this section, as the controller, representing the subsystem 2 of the co-simulation, is from type $\mathcal{H}_\infty$. This controller has been chosen, as it is suited for tracking a given reference signal. As the focus of this chapter is the evaluation of the Model-based Pre-Step Stabilization method and not the design of the controller, the details of $\mathcal{H}_\infty$ controller in general and for this example can be found e.g. in [44, 45]. Nevertheless the most important facts about $\mathcal{H}_\infty$ control theory will be stated in this section.
As the designed controller is a so-called full authority controller, the pilot is only defining the reference quantities $\boldsymbol{r}$, further called pilot commands. As it is shown in Figure 4.1 the input of the controller is

$$\boldsymbol{e} := \begin{pmatrix} \boldsymbol{r} - \boldsymbol{y}_{ctrl} \\ \boldsymbol{0} - \boldsymbol{y}_{rate} \end{pmatrix}.$$

The differentiation of the outputs of the helicopter into $\boldsymbol{y}_{ctrl}$ and $\boldsymbol{y}_{rate}$ origins from the fact that only for the heave velocity $\dot{H}$, the pitch attitude p, the roll attitude r and the heading rate $\dot{\psi}$ a reference signal, the pilot commands, are defined. For the roll and pitch rate the reference signal is always constantly zero.
In the following, a brief summary of $\mathcal{H}_\infty$ control will be stated, based on [44], additional information can be found for example in [45, 48, 49]. The theory of $\mathcal{H}_\infty$ control can

be assigned to the broad topic of robust control, there the aim is to design controllers which compensate the influence of disturbances sufficiently or improve the accuracy of a tracking problem. In Figure 4.3 the basic structure of a $\mathcal{H}_\infty$ controller is shown, there $\boldsymbol{u}$ denotes the control signals, $\boldsymbol{v}$ the measured signals, $\boldsymbol{w}$ the exogenous input, typically a disturbance or command signals and $\boldsymbol{z}$ stands for the error signals, which should be minimized by the controller. The aim of $\mathcal{H}_\infty$ control is to find a controller K which minimizes

$$||\boldsymbol{F}_l(P,K)||_\infty,$$

where $\boldsymbol{F}_l(P,K)$ denotes the closed-loop transfer function from $\boldsymbol{w}$ to $\boldsymbol{z}$, i.e. $\boldsymbol{z} = F_l(P,K)\boldsymbol{w}$. The $||\cdot||_\infty$ can be interpreted in the frequency domain as

$$||\boldsymbol{F}_l(P,K)||_\infty = \max_\omega \overline{\sigma}\left(\boldsymbol{F}_l(P,K)(j\omega)\right)$$

where ω stands for the frequency, j denotes the imaginary unit and $\overline{\sigma}(\boldsymbol{A})$ stands for the maximum singular value of the matrix $\boldsymbol{A}$. In the time domain holds

$$||\boldsymbol{F}_l(P,K)||_\infty = \max_{\boldsymbol{w}(t)\neq 0} \frac{||\boldsymbol{z}(t)||_2}{||\boldsymbol{w}(t)||_2}$$

with $||\boldsymbol{z}(t)||_2 := \sqrt{\int_0^\infty \sum_i (z_i(t))dt}$. Roughly speaking one can say that minimizing $||\boldsymbol{F}_l(P,K)||_\infty$ means that the reaction of P to an exogenous input $\boldsymbol{w}$ should be minimized. A version of $\mathcal{H}_\infty$ control is the so-called mixed-sensitivity approach. The difference to the classical one is that instead of one, two transfer functions are considered, namely $\boldsymbol{S}$, the transfer function from $\boldsymbol{w}$ to the output, and $\boldsymbol{KS}$, the transfer function from $\boldsymbol{w}$ to $\boldsymbol{u}$. In Figure 4.4 the structure of the $\boldsymbol{S}/\boldsymbol{KS}$ mixed-sensitivity $\mathcal{H}_\infty$ control loop is depicted, there it is visible that there are two error signals $\boldsymbol{z}_1 = \boldsymbol{W}_1(\boldsymbol{r} - \boldsymbol{y})$ and $\boldsymbol{z}_2 = \boldsymbol{W}_2\boldsymbol{u}$. The quantity $\boldsymbol{r}$ denotes reference signals which should be tracked. Therefore the optimization problem can be formulated as finding a controller K which minimizes

$$\left\| \begin{pmatrix} \boldsymbol{W}_1\boldsymbol{S} \\ \boldsymbol{W}_2\boldsymbol{KS} \end{pmatrix} \right\|_\infty .$$

For the utilized controller of this co-simulation, the $\boldsymbol{S}/\boldsymbol{KS}$ mixed-sensitivity approach has been augmented by a scaling $\boldsymbol{W}_3$ of the reference signals $\boldsymbol{r}$, so that all signals can be equally well tracked. For details of how the weights $\boldsymbol{W}_1, \boldsymbol{W}_2$ and $\boldsymbol{W}_3$ are chosen, see [44, 45], all needed information for computing the specific controller can be found in [47].

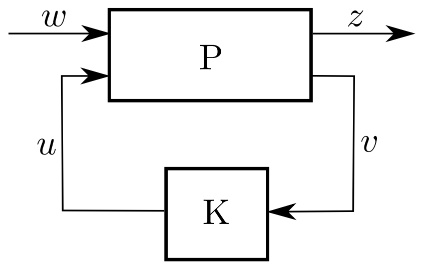


Figure 4.3: *General control design, from [44]*

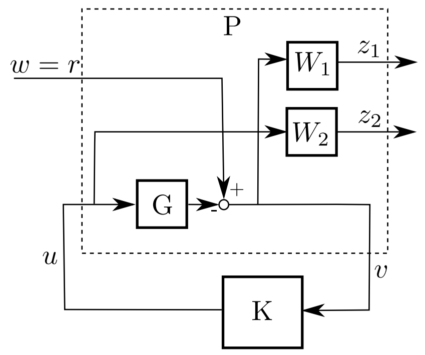


Figure 4.4: *S/KS mixed-sensitivity minimization in standard form (tracking), from [44]*

The utilized controller for the helicopter control co-simulation is described as the following linear, continuous-time, time-invariant state space system

$$\dot{\boldsymbol{x}}_2 = \boldsymbol{A}_2\boldsymbol{x}_2 + \boldsymbol{B}_2\boldsymbol{u}_2, \tag{4.7}$$

$$\boldsymbol{y}_2 = \boldsymbol{C}_2\boldsymbol{x}_2. \tag{4.8}$$

The number of states is 20 and the input $\boldsymbol{u}_2$ is the difference of the outputs of the helicopter and the reference signals, this means

$$\boldsymbol{u}_2 := \begin{pmatrix} \dot{H}_{ref} - \dot{H} \\ \theta_{ref} - \theta \\ \phi_{ref} - \phi \\ \dot{\psi}_{ref} - \dot{\psi} \\ -p \\ -q \end{pmatrix}.$$

The outputs of the controller $\boldsymbol{y}_2$ are the same four signals, which are the inputs of the helicopter, namely

- main rotor collective,
- longitudinal collective,
- lateral cyclic,
- tail rotor collective.

The physical connection and description of the in- and outputs of the controller is stated in the previous section. Computing the eigenvalues $\lambda_{\boldsymbol{A}_2}$ of $\boldsymbol{A}_2$ one clearly sees that the subsystem of the controller is stiff because $\frac{\max(\mathrm{abs}(\Re(\lambda_{\boldsymbol{A}_2})))}{\min(\mathrm{abs}(\Re(\lambda_{\boldsymbol{A}_2})))} \approx 3.4e5$. This makes it even more challenging to use such a controller as a subsystem for co-simulation. As the main part of the evaluation of the invented coupling method is to analyze the effect of approximated Interface Jacobians, the exact Interface Jacobians $\frac{\partial S_2}{\partial \boldsymbol{y}}$, $\frac{\partial S_2}{\partial \boldsymbol{u}}$ and $\frac{\partial S_2}{\partial \dot{\boldsymbol{u}}}$ are required. By applying (2.12) and (2.13) to $\boldsymbol{A_2}, \boldsymbol{B_2}$ and $\boldsymbol{C_2}$, the exact Interface

Jacobians can be determined as

$$\frac{\partial S_2}{\partial \boldsymbol{y}} = \begin{pmatrix} -2.25e+01 & -3.69e+01 & -3.05e+00 & 4.07e+01 \\ -4.99e+01 & -2.13e+02 & -1.78e+01 & 2.37e+02 \\ -4.83e+00 & -2.06e+01 & -2.78e-01 & 2.55e+01 \\ 5.41e+01 & 2.32e+02 & 2.19e+01 & -2.67e+02 \end{pmatrix} \tag{4.9}$$

$$\frac{\partial S_2}{\partial \boldsymbol{u}} = \begin{pmatrix} 6.34e+01 & 1.90e+02 & -4.93e+01 & 1.87e+02 & -2.68e+01 & 1.25e+01 \\ 2.84e+02 & 1.12e+03 & -2.86e+02 & 1.08e+03 & -1.55e+02 & 7.33e+01 \\ 2.77e+01 & 1.06e+02 & -4.58e+01 & 1.16e+02 & -1.60e+01 & 5.33e+00 \\ -3.04e+02 & -1.19e+03 & 3.32e+02 & -1.25e+03 & 1.69e+02 & -7.66e+01 \end{pmatrix}. \tag{4.10}$$

As there is no direct feed-through term in (4.8) the $\frac{\partial S_2}{\partial u}$ is a zero matrix with 4 columns and 6 rows. As the subsystem representing the helicopter, also the controller, is solved by the explicit Euler method with a step size of $2ms$, for details of the explicit Euler method see [40]. The initial conditions are set to zero for all states. Neither the helicopter nor the controller model consist of a direct feed-through which means that the co-simulation is zero-stable.

Before running the co-simulation the pilot commands $\boldsymbol{r}$ have to be defined. The four choosable signals of the pilot are the heave velocity $\dot{H}_{ref}$, the pitch attitude θ_{ref}, the roll attitude ϕ_{ref} and the heading rate $\dot{\psi}_{ref}$, the remaining two outputs of the helicopter the pitch rate and the roll rate are always specified as zero reference signals. As this selection influences the whole simulation, five different versions are tested and analyzed. The versions differ in aspects of amplitude, simultaneous combination of the four signals and discontinuous gaps. The first version is depicted in Figure 4.5, there it is obvious that no simultaneous action in the pilot commands exists, this means

$$\text{If } \boldsymbol{r}[i](t) \neq 0 \Rightarrow \boldsymbol{r}[j](t) = 0 \quad \forall j \neq i \text{ for every } t.$$

In other words this means, that if one of the four pilot commands is unequal zero, all others are zero for this moment. The idea behind this choice is that the dynamics of the whole simulation should be easier to handle for the co-simulation, as there is no simultaneous excitation origin from the pilot commands. Additionally this version is free of discontinuous gaps which should also be less challenging for the co-simulation. In the second version of the pilot commands, only the heave velocity and the heading rate are unequal zero, as depicted in Figure 4.6. The heave velocity consists of different steps according height and duration and is therefore highly discontinuous. Simultaneously the heading rate is varying between different parts of sinusoidal and steps functions. Therefore this version may be more challenging, as there is more simultaneous action and discontinuous gaps. The main difference of the third version, depicted in Figure 4.7, is that the pitch attitude is unequal zero. The discontinuous steps of the heave velocity are more regular and also the heading rate does not change its dynamics as often and fast as in version two. The fourth version of pilot commands is depicted in Figure 4.8, there again only the heave velocity and the heading rate are unequal zero. But the difference to the version two is that there are no discontinuous gaps in

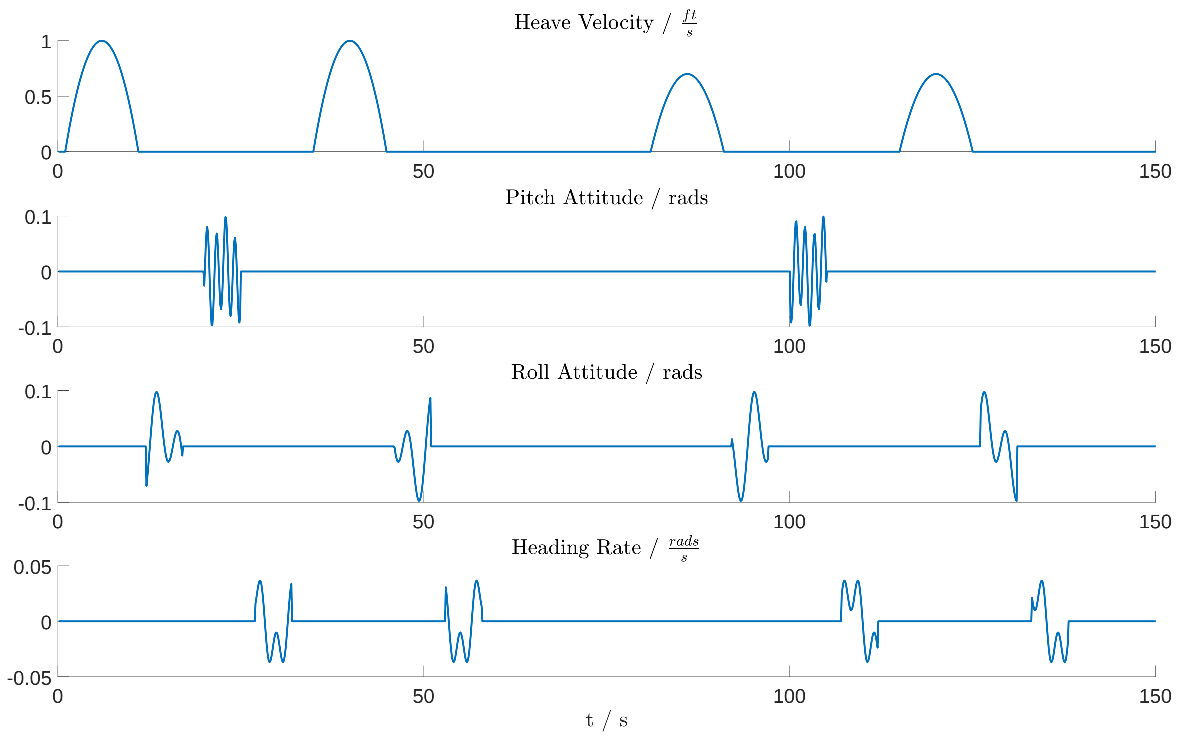


Figure 4.5: *Pilot commands version 1*

the heave velocity and also the heading rate consists of only one sinusoidal part with only one pause. Therefore different results between version two and four should be mainly based on the discontinuities of the pilot commands. In Figure 4.9 all four signals are varying nearly all the time, this means there is a lot of action simultaneously. Especially noticeable is that the heading rate consists of only discountinuous steps. Therefore the fifth version should be the most challenging one for the co-simulation.

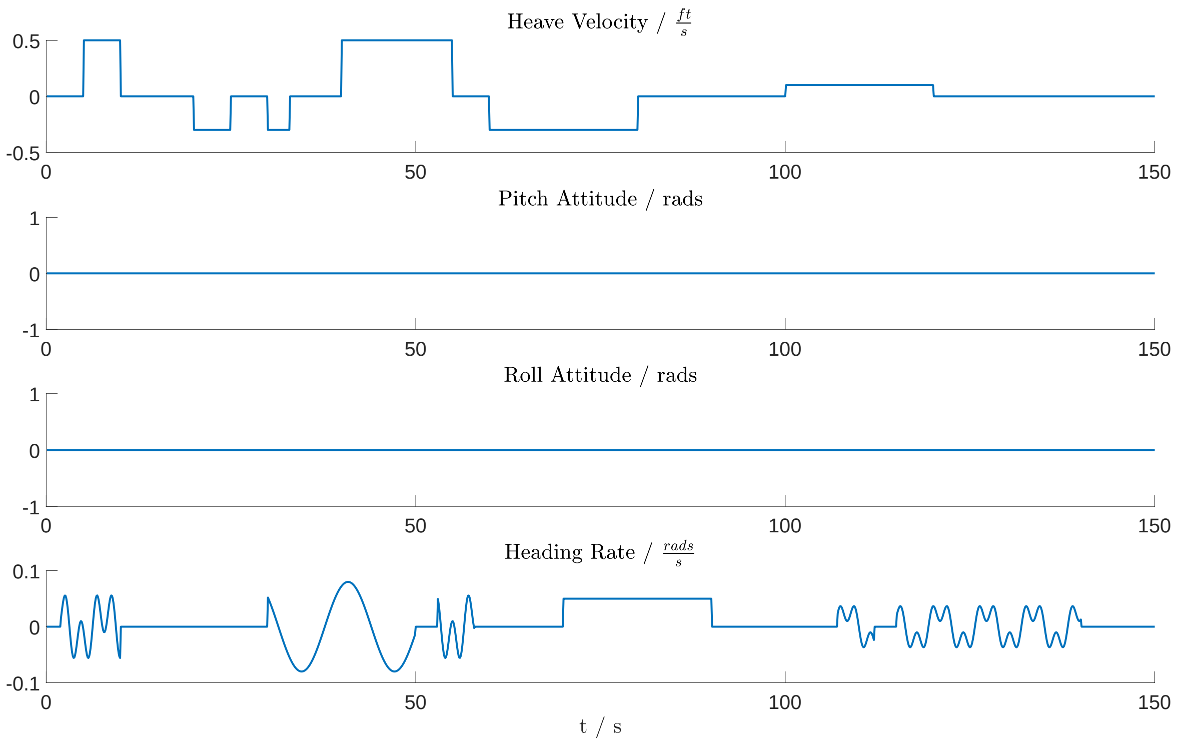


Figure 4.6: *Pilot commands version 2*

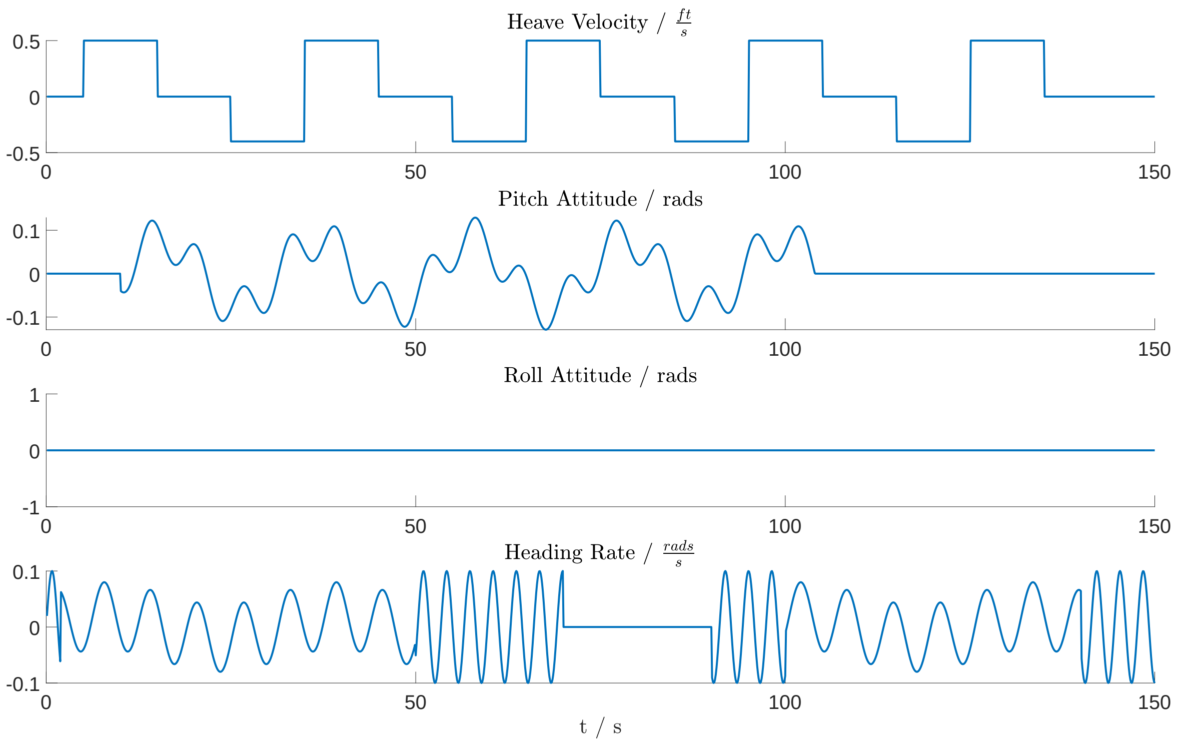


Figure 4.7: *Pilot commands version 3*

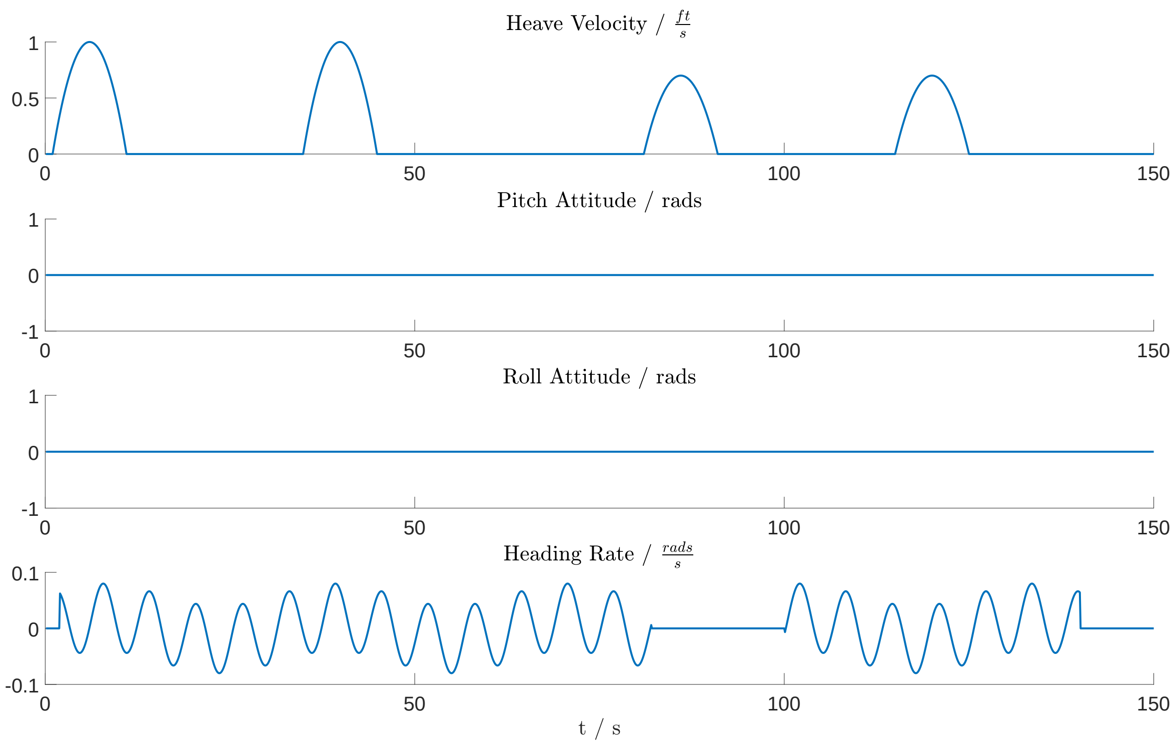


Figure 4.8: *Pilot commands version 4*

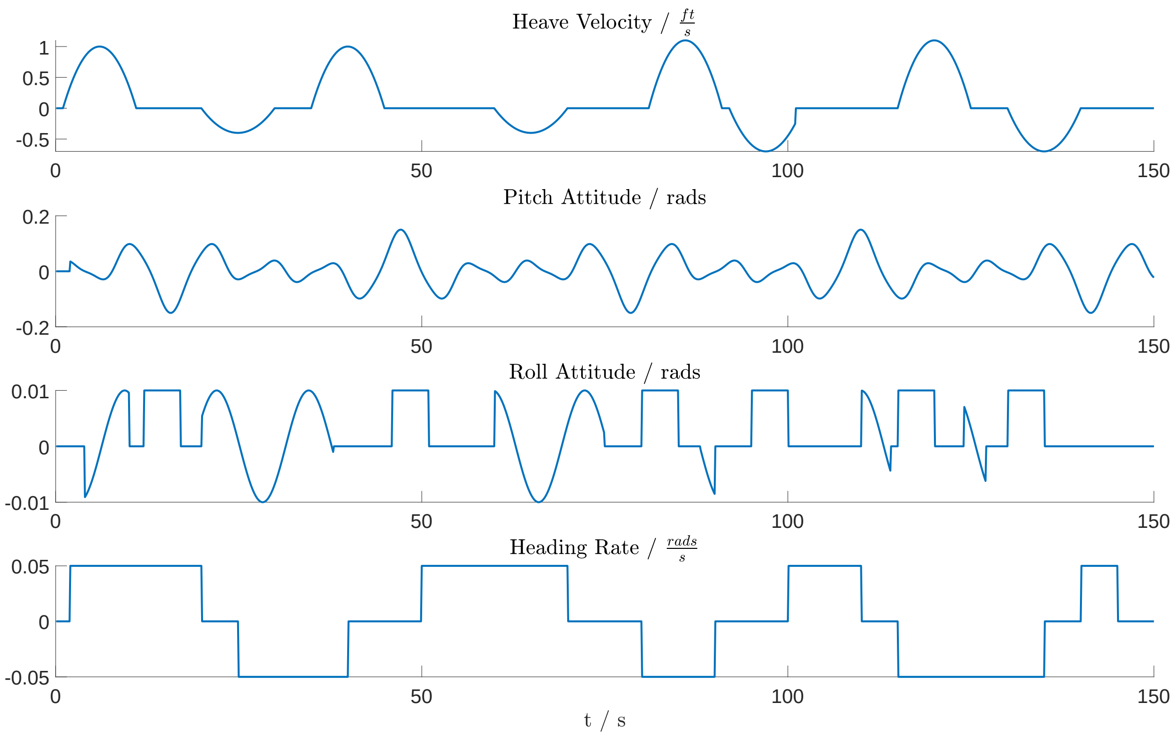


Figure 4.9: *Pilot commands version 5*

4.1.3 Results

First of all the effect of approximation errors of the required Interface Jacobians will be investigated, this is possible because the exact Interface Jacobians of both subsystems are accessible, see therefore (4.5), (4.6), (4.9) and (4.10). In Figure 4.10 the heave velocity, the heading rate and also the z-velocity of the helicopter are depicted utilizing the Model-based Pre-Step Stabilization Method with approximated and exact Interface Jacobians. The surprising fact is that the utilization of exact Interface Jacobians is leading immediately to instable results, as just a few seconds after the learning phase, where in both cases the coupling is ZOH, the result starts to oscillate. The reason for this paradox behavior is deep-set in the definition and computation of the Interface Jacobians and will be discussed in the following. Both subsystems are stated as classical continuous-time state space model, represented by the matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ and $\boldsymbol{D}$, therefore the transformation rule (2.12)

$$\frac{\partial S_1}{\partial \boldsymbol{y}_1} = \boldsymbol{C}_1(t)\boldsymbol{A}_1(t)\boldsymbol{C}_1^{-1}(t),$$

is required to compute the Interface Jacobian subsystem representation. There the problematic inversion of $\boldsymbol{C}_1$ is clearly visible. The subsystem 1, representing the helicopter, consists of 8 states and 6 outputs and therefore, as discussed in Section 2.1.1, the inversion of $\boldsymbol{C}_1$ will be computed utilizing the pseudo inverse. This is the reason for the instable behaviour because some dynamics are cancelled out by inverting the matrix $\boldsymbol{C}_1$ and therefore the Interface Jacobian $\frac{\partial S_1}{\partial \boldsymbol{y}_1}$ does not represent the real behaviour of subsystem 1 sufficiently. Also in subsystem 2, representing the controller, the number of states with 20 is higher than the number of outputs with 4 and therefore the required inversion of $\boldsymbol{C}_2$ for computing $\frac{\partial S_2}{\partial \boldsymbol{y}_2}$ cancels even more dynamics and information about the subsystem. For this example the exact Interface Jacobians are not usable because the number of states is higher than the number of outputs and the states are highly coupled, due to the natural dynamics of a helicopter and its control. Figure 4.10 shows that the effect of this subsystem representation based on the exact Interface Jacobians is huge, the co-simulation gets nearly immediately instable.

Here the question arises how can approximated Interface Jacobians be a better subsystem description as the exact ones? The reason for this, is the choice that the number of states is the same as the number of outputs, for the case of using the RLS as system identification this means setting $N = M = 1$. On the one hand, this is motivated by the fact that the inversion of $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ is then unproblematic, because the matrices are square and the outputs are linear independent by assumption. On the other hand also the sensitivity analysis in Section 3.2.2 shows, that the best choice is to choose $N = M = 1$. This means the complex subsystem of the helicopter and its controller are approximated by a time-variant less complex Interface Jacobian description. The required accuracy of the description can be reached by the time-varying characteristics of the system identification methods. This effect is of interest because the situation of the two subsystems is typical and therefore this will also be valid for other examples. Summarizing this means, that the utilization of exact Interface Jacobians is not a guarantee for a better performance of the co-simulation, the helicopter control

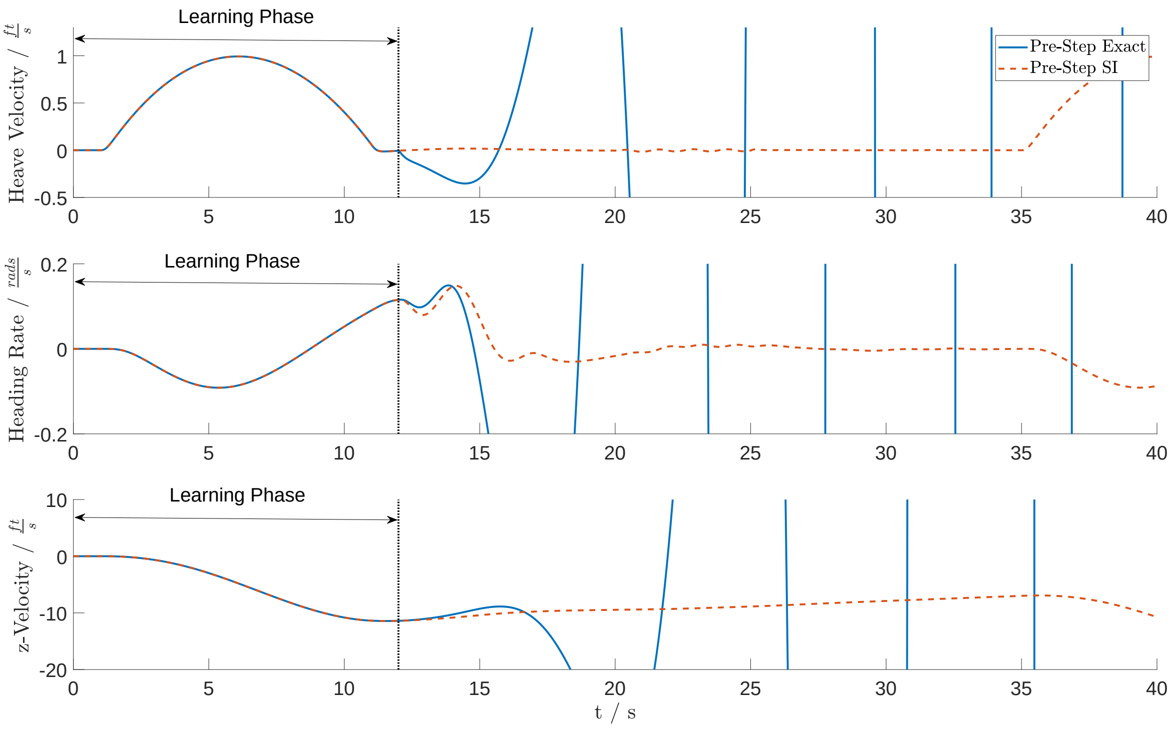


Figure 4.10: *Helicopter outputs for exact and approximated Interface Jacobians*

co-simulation shows that it can destabilize the whole simulation. Especially for the case that the number of states is much higher and the states are highly coupled in the subsystems, the utilization of the approximated Interface Jacobians seems more favorable. This fact is further reinforced by considering, that typically the access to the exact Interface Jacobians requires a high effort and these may be really challenging. All in all this example shows, that approximated Interface Jacobians may lead to much better results than exact ones.

For the evaluation of the results of the co-simulation, the classical error measure $\epsilon_{mono,global}$ defined in (3.9) and the stability measure $\sigma_{mono,global}$ are utilized. The stability measure $\sigma_{mono,global}$ is defined as

$$\sigma_{mono,global} := \frac{1}{6}\sum_{i=1}^{6}\sqrt{evar(\boldsymbol{y}_1[i] - \boldsymbol{y}_{1,mono}[i])} + \frac{1}{4}\sum_{i=1}^{4}\sqrt{evar(\boldsymbol{y}_2[i] - \boldsymbol{y}_{2,mono}[i])}, \tag{4.11}$$

where $\boldsymbol{y}_{1,mono}[i]$ stands for the i-th scalar signal of the monolithic output of the helicopter and $\boldsymbol{y}_{2,mono}[i]$ represents the i-th scalar signal of the monolithic output of the controller. The function *evar* computes a special variance of a signal. The difference to the classical variance is that, a time varying mean value of the signal will be taken into account, for more details see [50]. It should be mentioned that, if the mean value of the signal is constant the classical variance coincides with the computed variance of *evar*. The reason why the *evar* function is utilized and not the classical variance

is that in the classical variance a stationary error, i.e. $\boldsymbol{y}_1[i] - \boldsymbol{y}_{1,mono}[i] = const. \neq 0$ for a certain time, will affect the variance highly and by utilizing *evar* this stationary errors will be cancelled and only oscillations of $\boldsymbol{y}_1[i] - \boldsymbol{y}_{1,mono}[i]$ will be considered. The effect of stationary errors will be captured by the error measure $\epsilon_{mono,global}$ and therefore the utilization of *evar* is required to separate the error measure and the stability measure from the influence of stationary errors. Therefore one can say that $\epsilon_{mono,global}$ describes the accuracy and $\sigma_{mono,global}$ denotes the stability of the results, for sure these two concepts are connected and intertwined but a useful separation is possible.
Based on the error measure $\epsilon_{mono,global}$ and the stability measure $\sigma_{mono,global}$ the Model-based Pre-Step Stabilization Method (Pre-Step SI) is compared to zero-order-hold (ZOH) coupling, the Nearly Energy Preserving Coupling Element (NEPCE) approach, and the Model-based Corrector approach with exact (MBCorr exact) and approximated (MBCorr SI) subsystem information. As the utilization of exact Interface Jacobians for the Model-based Pre-Step Stabilization Method leads to instable results, this coupling method is ignored in the further chapter. In Figure 4.11 the results of the different coupling methods, utilizing version 1 of the pilot commands, are depicted depending on the communication step size ΔT, which varies between $[0.01s, 0.3s]$. The maximum value of $\epsilon_{mono,global}$ and $\sigma_{mono,global}$ is set to 1, because results with such a high value of $\epsilon_{mono,global}$ or $\sigma_{mono,global}$ are useless results. In the figure it is obvious that for $\Delta T \rightarrow 0s$, $\epsilon_{mono,global}$ and $\sigma_{mono,global}$ are both decreasing for all coupling methods except for the NEPCE coupling, which acts differently than the others. Until $\Delta T = 0.1s$ the results of ZOH, MBCorr exact and Pre-Step SI nearly coincide, only the MBCorr SI results are worse for $\Delta T \geqslant 0.08s$. Until $\Delta T = 0.14s$ the Pre-Step SI method is just slightly better than ZOH but for $\Delta T > 0.14s$ this changes drastically as ZOH coupling gets instable, as $\sigma_{mono,global}$ increases highly. For $\Delta T \in [0.16s, 0.2s]$ the Model-based Pre-Step Stabilization Method is the only coupling method which produces stable results. Utilizing version 2 of the pilot commands is leading to the results depicted in Figure 4.12, there the results are similar to the ones recently discussed. The only noticable difference is, that the MBCorr SI result is stable up to $\Delta T = 0.12s$. Comparing the Figures 4.11 - 4.15 it is obvious that all five diagrams look very similar, this means that the choice of the pilot commands is not critical for the co-simulation of the helicopter and its control. The error and oscillations measures vary just in a small range, the tendency is for all five tested versions of the pilot commands the same. Therefore in the following chapter only the first version of the pilot commands will be utilized.

As the first version of the pilot commands is utilized, the highest communication step size where all compared coupling methods generate useful results is $\Delta T = 0.08s$ and therefore all outputs of the helicopter and its control will be further discussed based on a communication step size of $\Delta T = 0.08s$. The six simulation results of the helicopter outputs are depicted in Figures 4.16 and 4.17, there the five co-simulation coupling methods are compared to the monolithic results. At the heave velocity it is obvious to see, that the accuracy of all coupling methods, except for NEPCE, is sufficient. Especially of interest is that the MBCorr SI result has two peaks at $t \approx 12s$ and

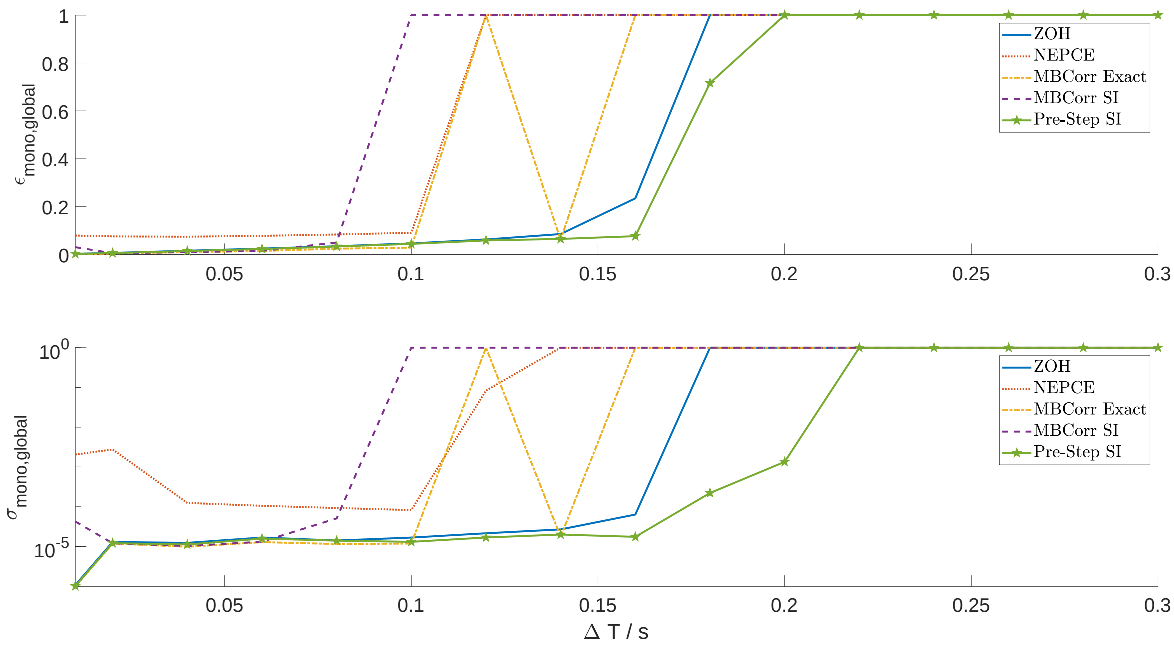


Figure 4.11: *Error measure $\epsilon_{mono,global}$ and stability measure $\sigma_{mono,global}$ dependent on ΔT for pilot commands version 1*

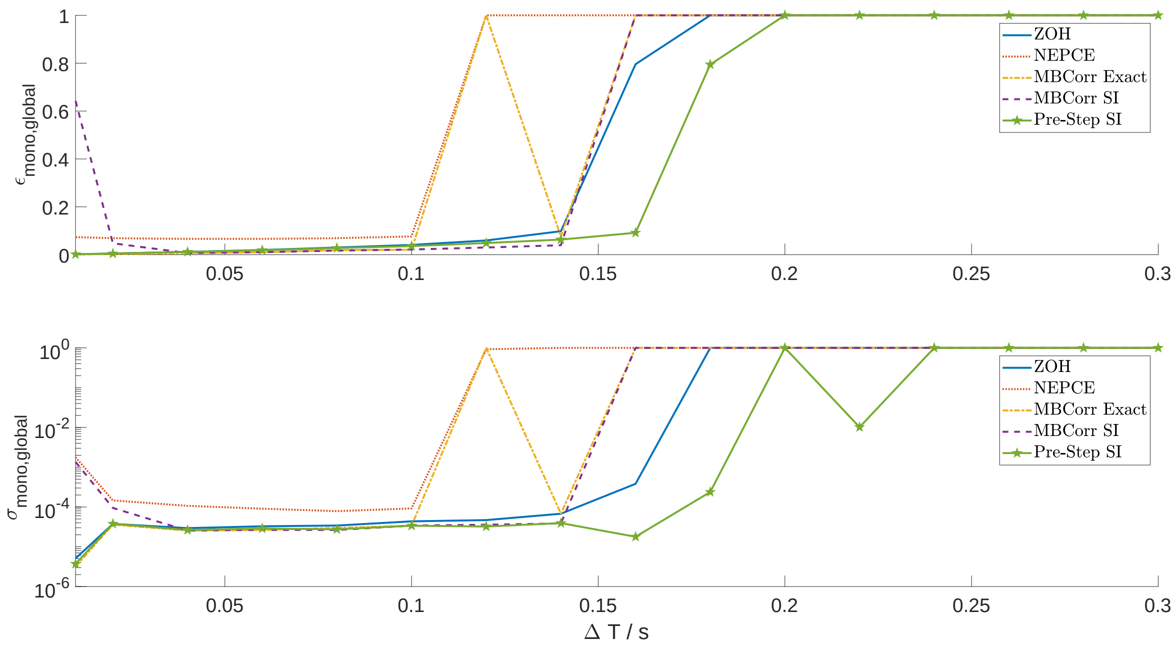


Figure 4.12: *Error measure $\epsilon_{mono,global}$ and stability measure $\sigma_{mono,global}$ dependent on ΔT for pilot commands version 2*

$t \approx 80s$, the first one coincides with the end of the learning phase. Comparing this to the other five outputs of the helicopter, one sees in every signal a peak at $t \approx 12s$ and $t \approx 80s$. Especially in the pitch attitude for $t \in [80s, 95s]$ there is an obvious deviation

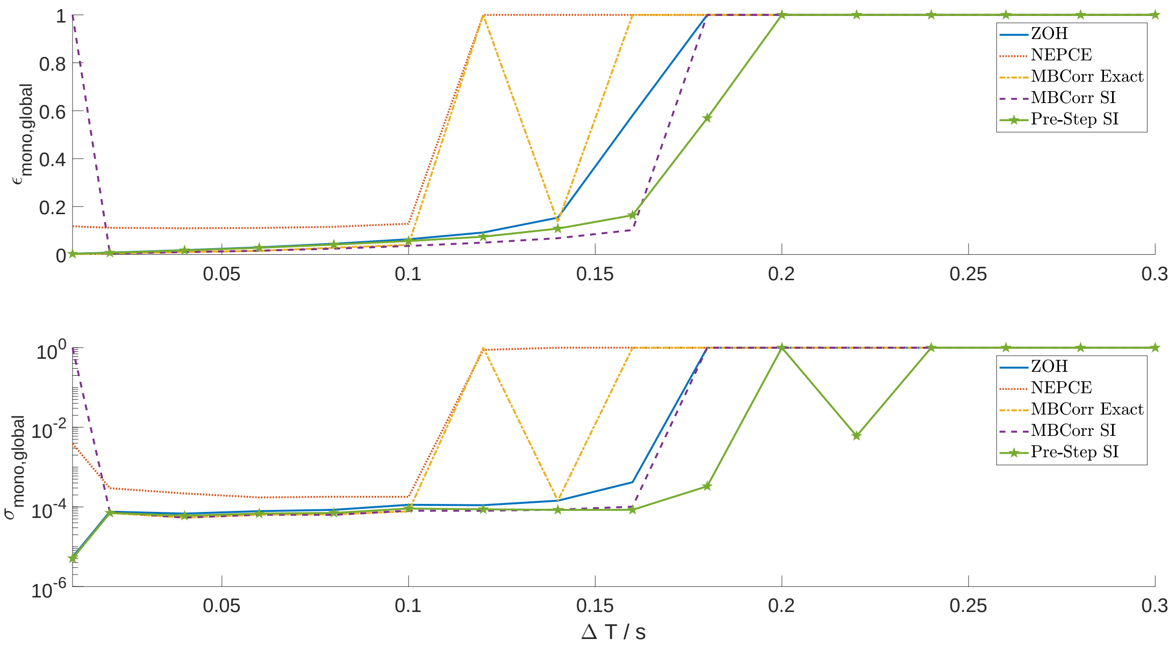


Figure 4.13: *Error measure $\epsilon_{mono,global}$ and stability measure $\sigma_{mono,global}$ dependent on ΔT for pilot commands version 3*

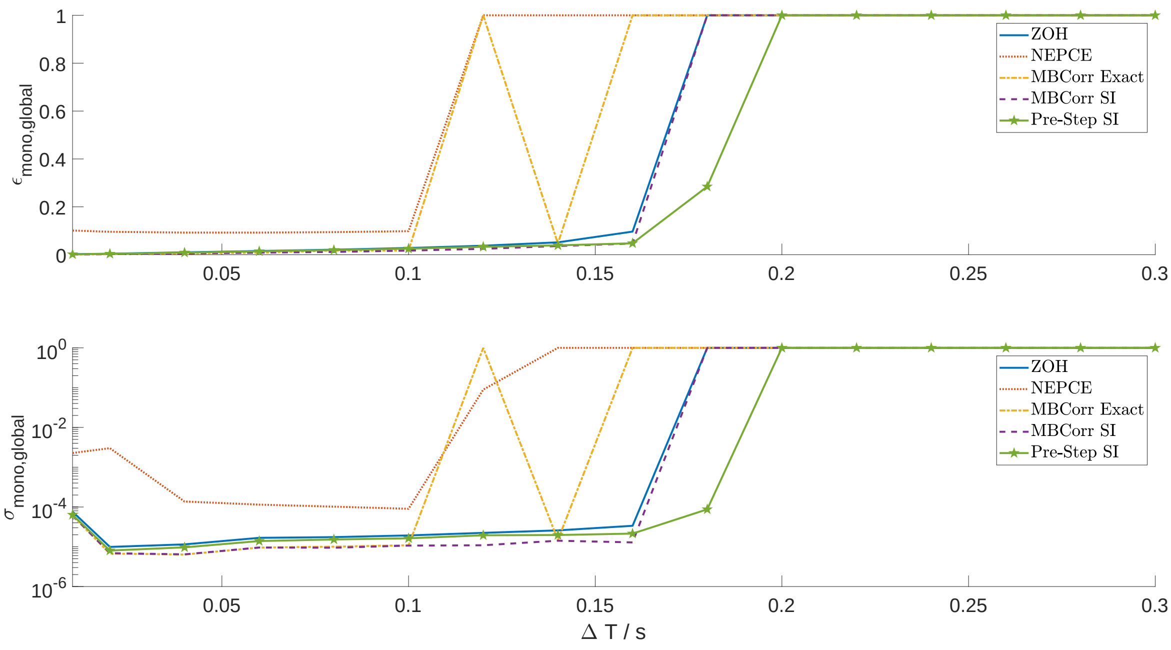


Figure 4.14: *Error measure $\epsilon_{mono,global}$ and stability measure $\sigma_{mono,global}$ dependent on ΔT for pilot commands version 4*

from the monolithic result. The NEPCE method shows nearly permanent deviations from the monolithic result. Summarizing the six helicopter outputs, depicted in Figures 4.16 and 4.17, shows that the ZOH, the MBCorr exact and the Pre-Step SI

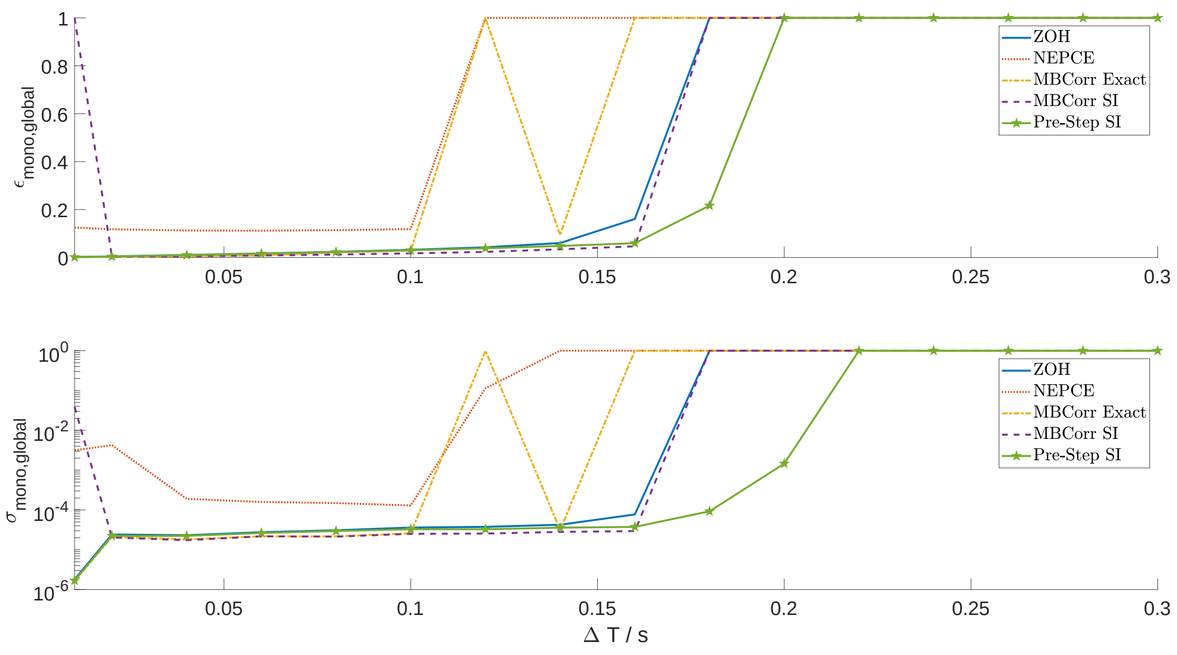


Figure 4.15: *Error measure $\epsilon_{mono,global}$ and stability measure $\sigma_{mono,global}$ dependent on ΔT for pilot commands version 5*

coincide with the monolithic results, which means the performance is for this case satisfying. It is also obvious that the NEPCE approach has drawbacks in terms of accuracy and the MBCorr SI approach has two major deviations, one immediately after the learning phase and the other after a pause of around 20 seconds. Even though the velocity of the helicopter is not an output of the subsystem, one can analyze the results for the purpose of evaluation of the utilized coupling methods, see for the x-, y- and z-velocity of the helicopter Figure 4.18. As mentioned in the other helicopter outputs the NEPCE method is inaccurate and the MBCorr SI result has some small peaks in the z-velocity at $t \approx 12s$ and $t \approx 80s$. Additional to those small peaks, there is a nearly permanent deviation from the monolithic result in the velocity in x-direction. Comparing all those nine outputs of the helicopter, the heave velocity, the pitch and roll attitude, the heading rate, the pitch and roll rate, and the velocity in all three cartesian directions, the same effects can be more or less investigated in every signal, this reflects the strong interconnection and coupling in the subsystem of the helicopter itself.
The four outputs of the controller are depicted in Figures 4.19 and 4.20. Comparing the main rotor collective signal with the heave velocity, one sees that there is a strong connection. Additionally the two observed peaks of MBCorr SI are specially remarkable, as they are $\approx$ 5 times higher than the amplitude of the monolithic signal. The two peaks in all MBCorr SI results, outputs of the helicopter and the controller, are correlating strongly, as they all appear at $\approx 12s$ and $\approx 80s$. This correlation is a further evidence that the two subsystems are strongly interconnected. The oscillations of the NEPCE approach are significant in the time intervals $[1s, 11s], [35s, 45s], [81s, 91s]$ and $[115s, 125s]$, they correlate exactly with the parts

where the heave velocity is changing. This leads to the conclusion that the change in the heave velocity is leading to the oscillations and deviations of all NEPCE results. For the ZOH, the MBCorr exact and the Pre-Step SI results all four outputs of the controller are of satisfying accuracy and free of any unwanted oscillations.
One of the keys of the improved performance of the Model-based Pre-Step Stabilization Method is, that complex subsystems are approximated with less complex surrogate models, with the approximated Interface Jacobians, which are time variant. The time variant subsystem description is achieved by utilizing the RLS and therefore the identified parameters can vary over time and adapt themselves appropriately. As $N = M = 1$ is chosen, the difference equation of a scalar output signal, see (2.87), e.g. the heave velocity or the main rotor collective, can be written as

$$y^k = a_1^k y^{k-1} + b_{11}^k u_1^{k-1} + \ldots + b_{m1}^k u_m^{k-1}, \tag{4.12}$$

where $m = 4$ for the subsystem 1 and $m = 6$ for the subsystem 2, as this is the number of inputs of the subsystems. The superscript index ^{k} denotes the time variant aspect of the parameters $a_1^k, b_{11}^k, \ldots, b_{m1}^k$. In the Figures 4.21 - 4.24 the progression of all identified parameters is depicted over the full simulation time, including the learning phase from $0s$ until $12s$. In the learning phase the RLS is identifying the parameter, but the coupling is not influenced by the parameters, as the coupling is exclusively carried out by the ZOH approach in the learning phase. The high oscillations at the beginning of the simulation, roughly around the first 5 seconds, origin from the fact, that for all parameters the initial values are set to zero. It should be mentioned that in all following parameter plots the parameter a_1 is denoted to left y-axis and all other parameters to the right y-axis, this is required as they differ significantly in their range. Considering the above mentioned four figures there are two main observations. First, all parameters are really time varying, except for a_1 of the heave velocity which seems to be nearly constant, it is only varying in the range of $[0.96, 0.97]$. This means that all parameters are really adapting themselves over time and therefore the choice of a time-variant Interface Jacobian subsystem description is necessary and mandatory. The lack of complexity in the surrogate models, compared to the real subsystems, is compensated by the automatic adaption of the parameters. The second observation is that, none of the parameters is zero, or even near zero, this means that every input of a subsystem is influencing every output of the subsystem, this reflects the strong coupling in the subsystems. As depicted in Figure 4.21 the heave velocity is mainly influenced by the parameter b_{11} which denotes the influence of the main rotor collective signal from the controller, this fits perfectly with the modeling of the subsystems as discussed in the previous section. It seems that the main rotor collective has the most influence on all outputs of the helicopter as the parameter b_{11} has the biggest range, as depicted in Figures 4.21 and 4.22. But the influence does not depend on the value of the identified parameters exclusively, the range of the input signal is of the same importance. In Figures 4.19 and 4.20 the range of the main rotor collective is $\approx 0.1 rads$, for the longitudinal cyclic $\approx 1.5 rads$, for the lateral cyclic $\approx 1 rads$ and for the tail rotor collective $\approx 1 rads$. Comparing the ranges to the identified parameters of the pitch attitude one sees that the main rotor collective and the longitudinal collective, represented through parameter b_{21}, has the

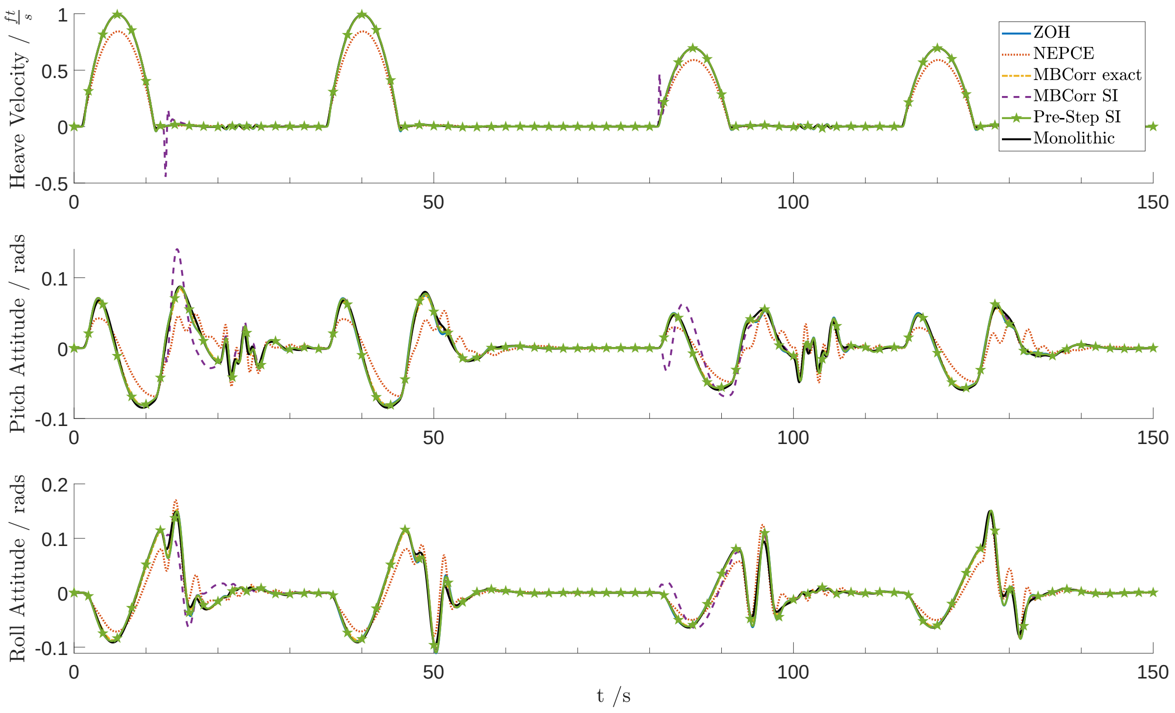


Figure 4.16: *Helicopter outputs for $\Delta T = 0.08s$: heave velocity, pitch attitude and roll attitude*

most influence. For the roll attitude the influence of the main rotor collective is not as strong as before and therefore the lateral cyclic, b_{31}, and the tail rotor collective, b_{41} are of significance. These observations fit the physical description of the blade angels of the helicopter. For the other signals the influence of the four inputs to the outputs are in the same range. But in general the progression of the identified parameters shows clearly the fact that every move of the controller is influencing the whole behaviour of the helicopter, which is again an evidence of the strongly coupled dynamics of this co-simulation.

Figures 4.23 and 4.24 show the progression of the identified parameters of the four outputs of the controller, the main rotor collective, the longitudinal cyclic, the lateral cyclic and the tail rotor collective. In the beginning of the simulation there are higher oscillations than at the helicopter outputs, but after the learning phase the progression of the parameter is smooth and free of oscillations. For analyzing the influence of the inputs of the controller to its outputs the range of the inputs is required, from Figure 4.16 follows that the range of the heave velocity is $\approx 1\frac{ft}{s}$, for the pitch attitude it is $\approx 0.2rads$ and for the roll attitude the range can be determined as $\approx 0.2rads$. The range of the heading rate is $\approx 0.1\frac{rads}{s}$, the range of the roll rate is $\approx 0.4\frac{rads}{s}$ and for the pitch rate the range can be stated as $\approx 0.3\frac{rads}{s}$, which can be derived from Figure 4.17. Combining the progression of the input parameters b_{i1} for the main rotor collective, see Figure 4.23, one can observe that the heave velocity, the roll and pitch attitude are of importance and that the heading rate and especially the roll and pitch rate are significant for the main rotor collective. For the longitudinal cyclic,

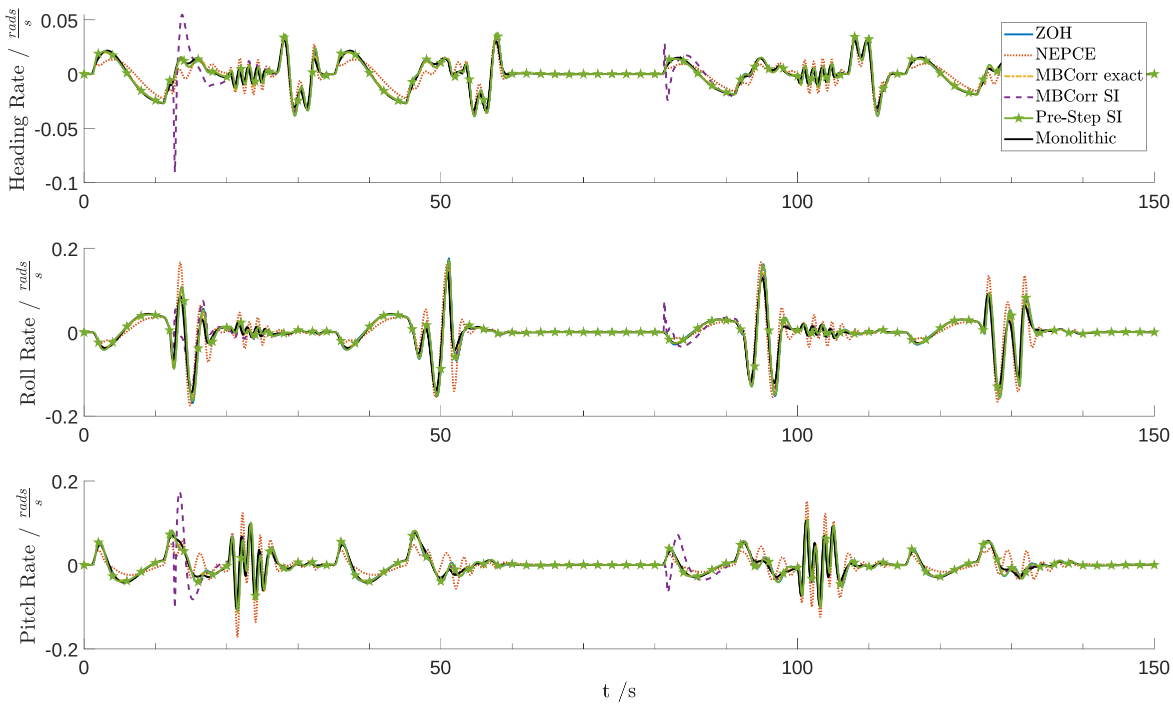


Figure 4.17: *Helicopter outputs for* $\Delta T = 0.08s$*: heading rate, roll rate and pitch rate*

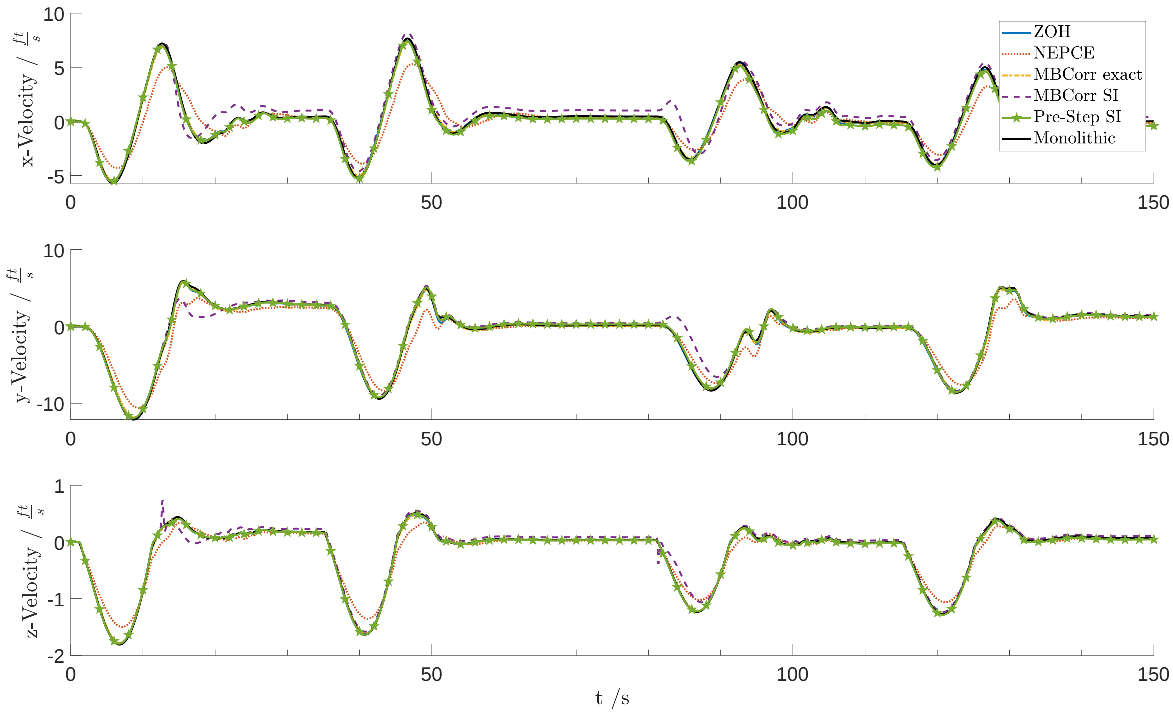


Figure 4.18: *X-,y- and z-velocity of the helicopter for* $\Delta T = 0.08s$

only the influence of the heave velocity is not significant, all other five inputs are of

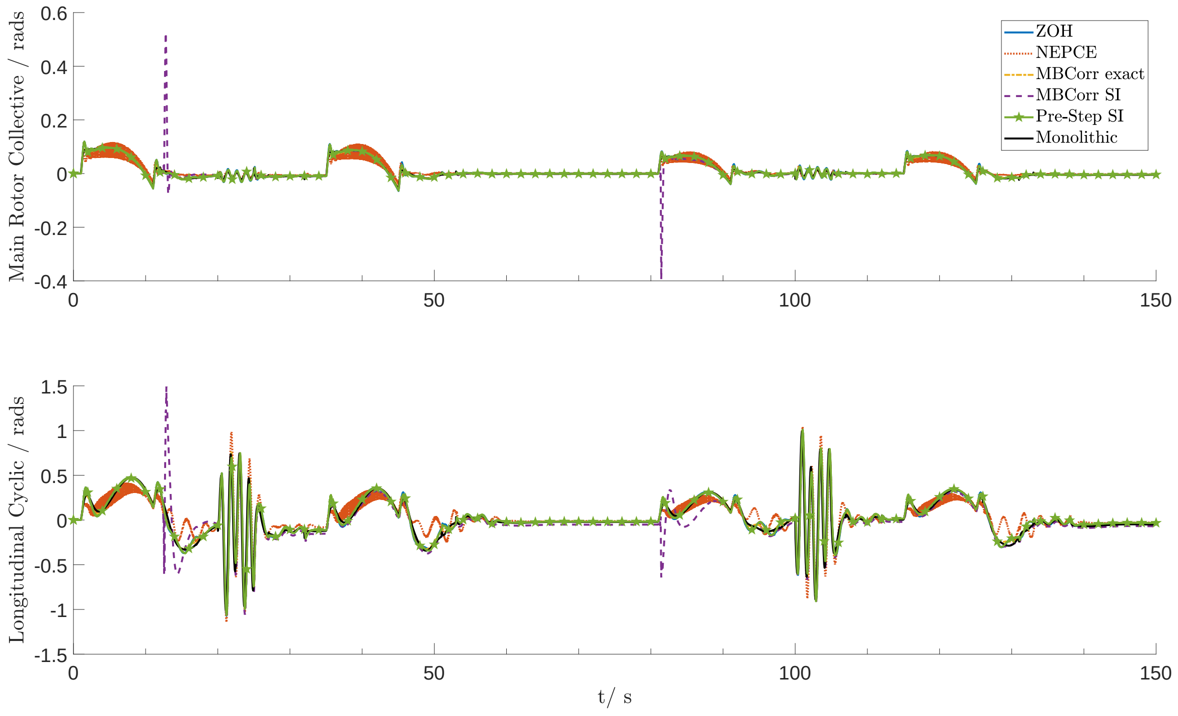


Figure 4.19: *Controller outputs for $\Delta T = 0.08s$: main rotor collective and longitudinal cyclic*

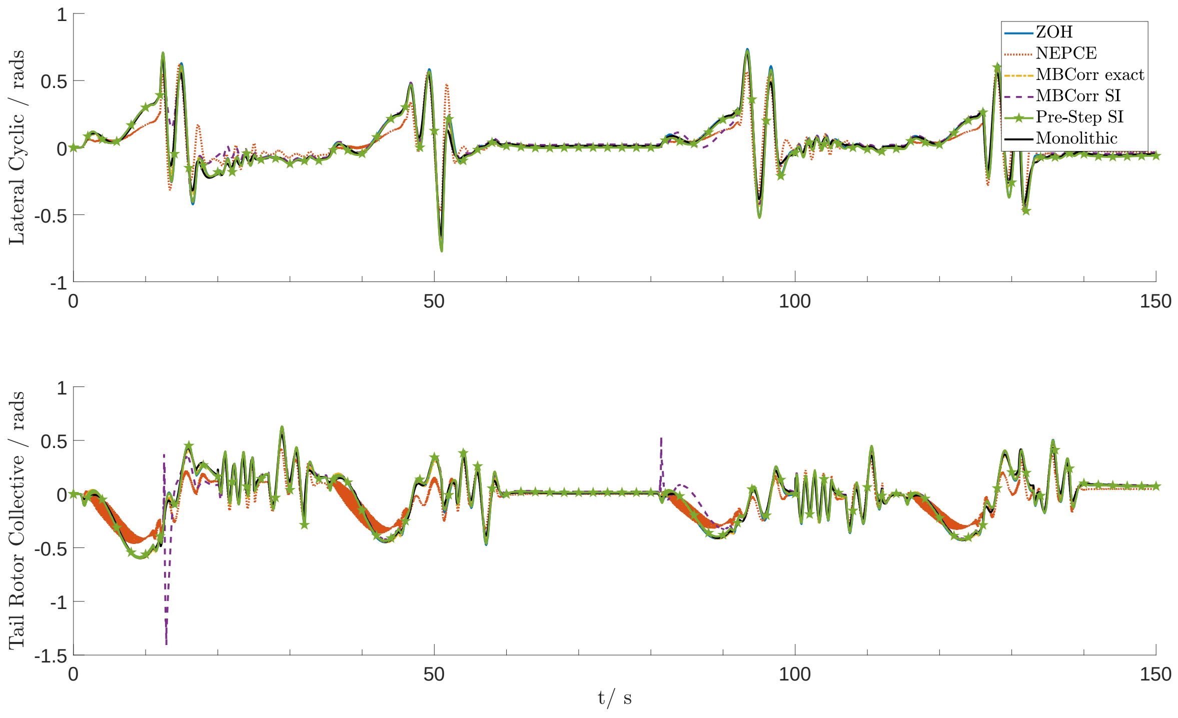


Figure 4.20: *Controller outputs for $\Delta T = 0.08s$: lateral cyclic and tail rotor collective*

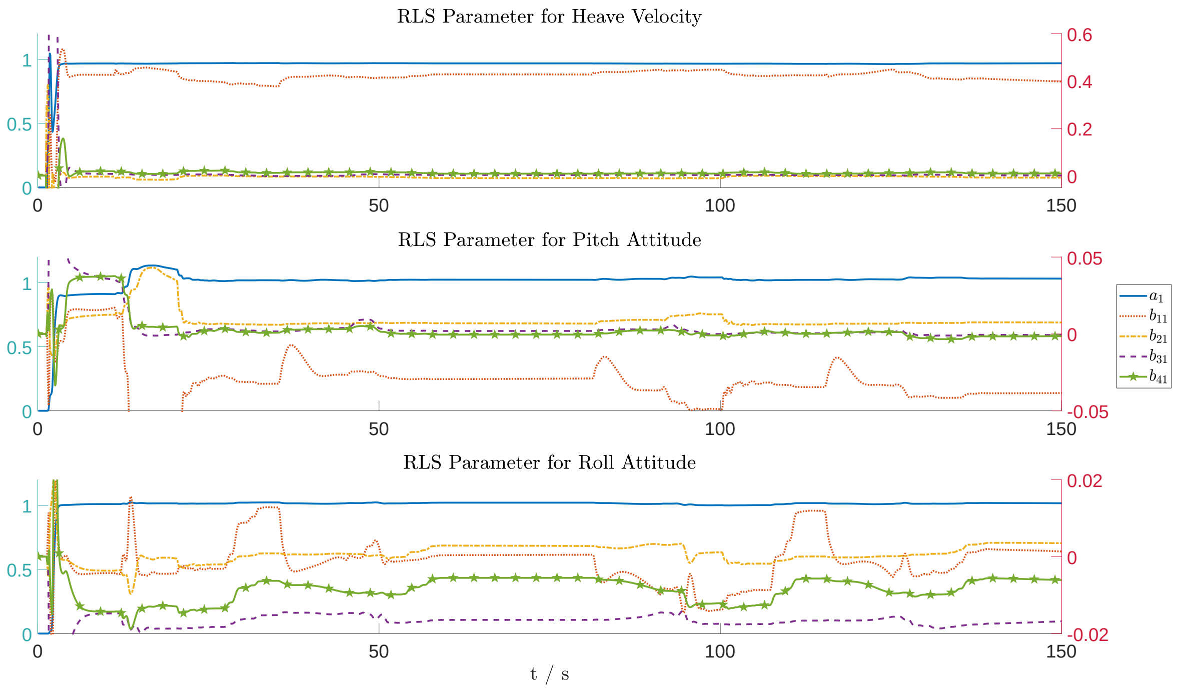


Figure 4.21: *Identified parameters for helicopter outputs: heave velocity, pitch attitude and roll attitude*

higher importance, especially the pitch rate is significant. Based on the progression of the identified parameters of the lateral cyclic, depicted in Figure 4.24, the significant influence of the roll rate and the pitch attitude is obvious. The heading rate and the heave velocity are of importance for the lateral cyclic. For the tail rotor collective signal, the most important inputs are the heading rate and the pitch rate. The other four inputs are of less or no significant influence.
As the results are mainly based on the approximated Interface Jacobians, the quality of this approximated subsystem description is of high importance. Therefore in the upper part of Figure 4.25 the measure $\epsilon_{SI,global}$ of the error of the system identification method is stated, for its definition see (2.93). There it is obvious that for the whole simulation the measure is under 0.15, which is an accurate result as $\epsilon_{SI,global}$ is an relative measure, especially if one is considering the fact that only the highest peaks are in the range of $[0.1, 0.15]$. For the main part of the simulation $\epsilon_{SI,global}$ is around 0.02, which can be seen as an approximation error of around 2%. At the end of the learning phase, denoted by the black doted line at $t = 12s$, the highest peaks are reached, but this is clear as the coupling method is switched from ZOH to the pre-step coupling. To damp the influence of these high peaks the switch phase has been introduced, for details see Section 2.4.4. The lower part of Figure 4.25 shows in which coupling phase the Model-based Pre-Step Stabilization method is running during the simulation. Starting with the learning phase for $12s$, followed by the switch phase for $2s$. From the rest of the simulation the coupling method stays in the pre-step phase, because

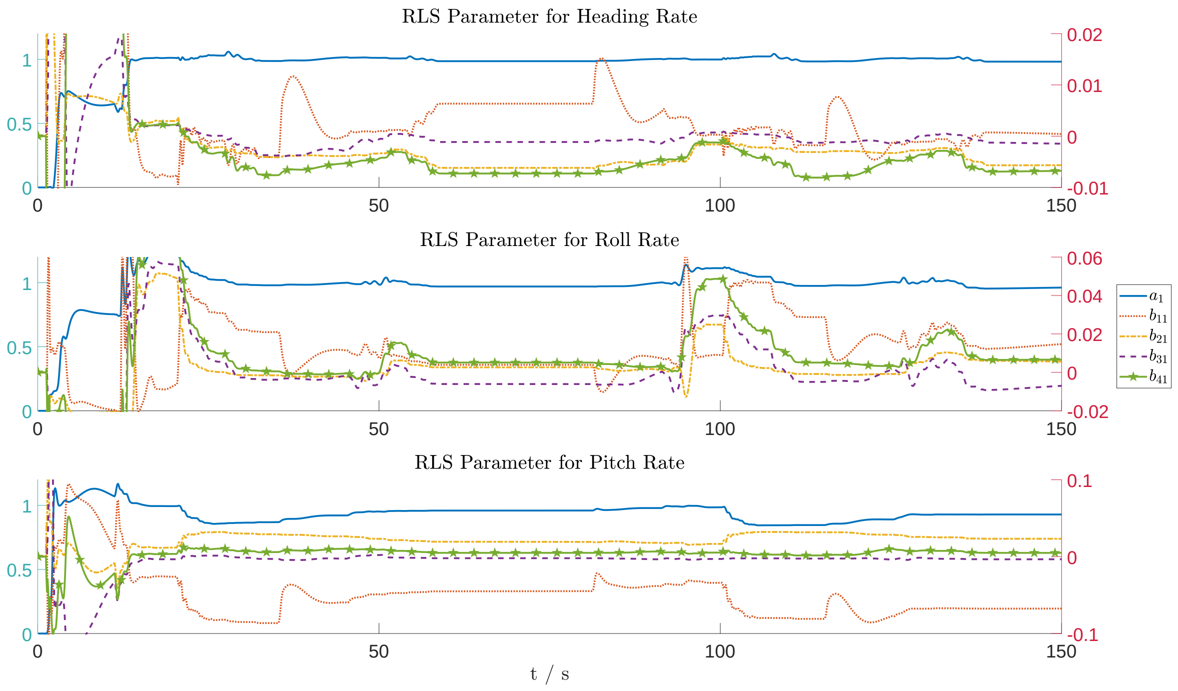


Figure 4.22: *Identified parameters for helicopter outputs: heading rate, roll rate and pitch rate*

the $\epsilon_{SI,global}$ never reaches the boundaries of $thresholdMbc2sbcCoupling = 3.54$ or $thresholdMbcNoInputOpt = 0.21$, which would be necessary to switch in the SBC only phase or the modified pre-step phase. This switching between the phases is only required if the approximation is too weak, therefore no switching from the pre-step phase is the preferred case. For more details of this error-based phase check see Section 2.4.2.

As the stability of the co-simulation is the main focus of the Model-based Pre-Step Stabilization the question arises why the coupling gets instable at $\approx \Delta T = 0.2s$, as it is depicted in Figure 4.11. The answer is based on the learning phase, where the first $12s$ of the co-simulation the coupling is ZOH, because if already in the learning phase the results are oscillating, the system identification method identified completely wrong Interface Jacobians and therefore the whole Model-based Pre-Step Stabilization will fail. This problem will be treated by a shorter learning phase of $2s$ and improved initial parameters. The shorter learning phase will decrease the influence of the instable behaviour of the ZOH in the learning phase drastically and as the learning phase is too short to completely identify the model without any previous knowledge, well-chosen initial parameters are required. The initial parameters are saved from a previous co-simulation with $\Delta T = 0.1s$ at around $t \approx 8s$ because at that time the parameters are nearly steady. It is important to take initial parameters from the beginning of the simulation as they vary over the time significantly and therefore

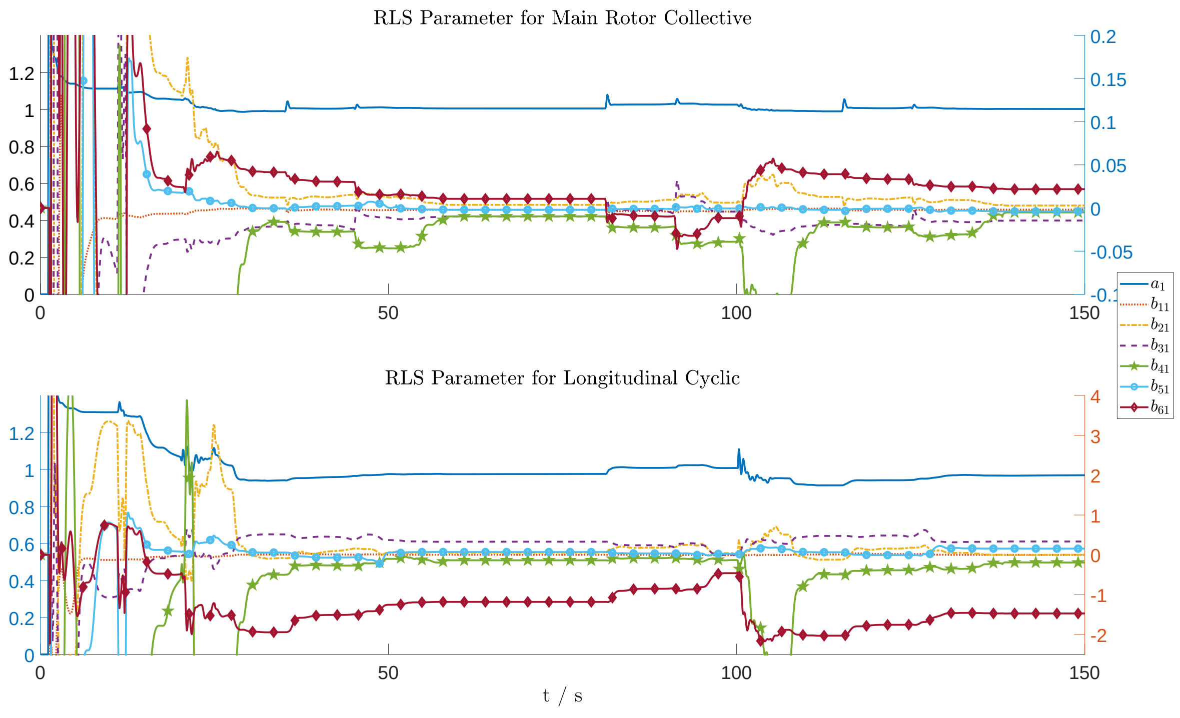


Figure 4.23: *Identified parameters for controller outputs: main rotor collective and longitudinal cyclic*

the parameters are saved at the first moment when all parameters where nearly steady. Due to the fact that these parameters are the basis for a discrete-time subsystem description they have to be recalculated according to the utilized communication step size.

This recalculation will be derived by transforming a difference equation to the double sampling rate, this means the parameters of the difference equation have to change accordingly to the sampling rate. The concept of this derivation will be generalized so that a difference equation can be recalculated to any sampling rate which is greater than the original one. Starting with inserting the difference equation from (4.12) in

$$y^{k+1} = a_1^{k+1} y^k + b_{11}^{k+1} u_1^k + \ldots + b_{m1}^{k+1} u_m^k,$$

leads to

$$y^{k+1} = a_1^{k+1} \left(a_1^k y^{k-1} + b_{11}^k u_1^{k-1} + \ldots + b_{m1}^k u_m^{k-1} \right) + b_{11}^{k+1} u_1^k + \ldots + b_{m1}^{k+1} u_m^k.$$

Rearranging it results in

$$y^{k+1} = a_1^{k+1} a_1^k y^{k-1} + a_1^{k+1} b_{11}^k u_1^{k-1} + b_{11}^{k+1} u_1^k + \ldots + a_1^{k+1} b_{m1}^k u_m^{k-1} + b_{1m1}^{k+1} u_m^k.$$

One should assume that the sampling rate is doubled, i.e. the values $u_1^k, \ldots, u_m^k$ are not available, only the values at time points $k-1$ and $k+1$ are accessible, therefore

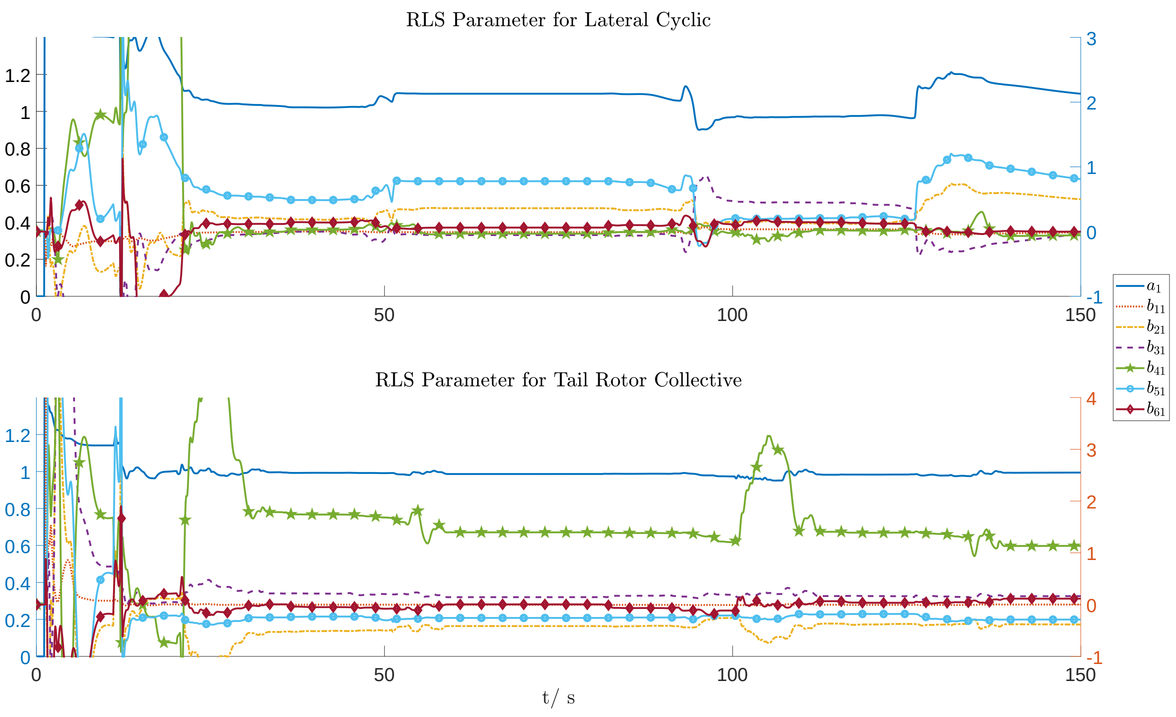


Figure 4.24: *Identified parameters for controller outputs: lateral cyclic and tail rotor collective*

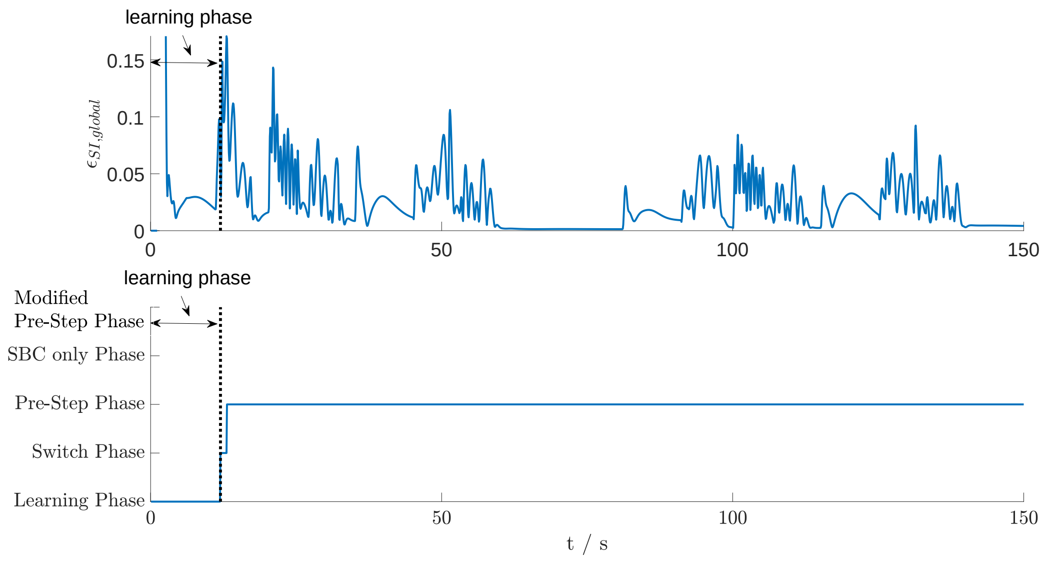


Figure 4.25: *Quality of the identified Interface Jacobians and the progression of the Error-based Phase Check*

the assumption holds that

$$u_k = u_{k-1}.$$

As the parameters are taken from a steady phase, we can assume that

$$\begin{aligned} a_1^{k+1} &= a_1^k =: a_1, \\ b_{11}^{k+1} &= b_{11}^k =: b_{11}, \\ &\vdots \\ b_{m1}^{k+1} &= b_{m1}^k =: b_{m1}. \end{aligned}$$

Considering the assumptions is leading to

$$y^{k+1} = (a_1)^2 y^{k-1} + (a_1 b_{11} + b_{11})\, u_1^{k-1} + \ldots + (a_1 b_{m1} + b_{m1})\, u_m^{k-1}.$$

Comparing the above equation with the difference equation of the double sampling rate

$$y^{k+1} = \tilde{a}_1 y^{k-1} + \tilde{b}_{11} u_1^{k-1} + \ldots + \tilde{b}_{m1} u_m^{k-1},$$

leads to

$$\begin{aligned} \tilde{a}_1 &= (a_1)^2, \\ \tilde{b}_{11} &= a_1 b_{11} + b_{11}, \\ &\vdots \\ \tilde{b}_{m1} &= a_1 b_{m1} + b_{m1}, \end{aligned}$$

which are the transformations rules to recalculate identified parameters to the double sampling rate. Analog, it can be shown that for any integer multiple of the original sampling rate H, i.e. the enlarged sampling rate $\tilde{H} = kH$, for $k \in \mathbb{N}$, the required parameters can be calculated as

$$\tilde{a}_1 := (a_1)^{\frac{\tilde{H}}{H}}, \tag{4.13}$$

$$\tilde{b}_{j1} := b_{j1} \sum_{i=0}^{\frac{\tilde{H}}{H}} (a_1)^i, \tag{4.14}$$

for $j = 1, \ldots, m$.

In the following a generalization of this transformation to a general sampling rate $\tilde{H}$, which has to be greater than the original one H, will be stated. The quantities

$$\begin{aligned} \Delta &= \left\lfloor \frac{\tilde{H}}{H} \right\rfloor, \\ \delta &= \frac{\tilde{H}}{H} - \Delta, \end{aligned}$$

describe the integer part and the decimal part of the ratio $\frac{\tilde{H}}{H}$. The transformed parameters are determined via

$$\begin{aligned} \tilde{a}_1 &= \tilde{a}_{low,1}(1-\delta) + \tilde{a}_{up,1}\delta, \\ \tilde{b}_{11} &= \tilde{b}_{low,11}(1-\delta) + \tilde{b}_{up,11}\delta, \\ &\vdots \\ \tilde{b}_{m1} &= \tilde{b}_{low,m1}(1-\delta) + \tilde{b}_{up,m1}\delta, \end{aligned}$$

with

$$\begin{aligned}
\tilde{a}_{low,1} &:= (a_1)^{\Delta}, \\
\tilde{a}_{up,1} &:= (a_1)^{\Delta+1}, \\
\tilde{b}_{low,11} &:= b_{11} \sum_{i=0}^{\Delta} (a_1)^i, \\
\tilde{b}_{up,11} &:= b_{11} \sum_{i=0}^{\Delta+1} (a_1)^i, \\
&\vdots \\
\tilde{b}_{low,m1} &:= b_{m1} \sum_{i=0}^{\Delta} (a_1)^i, \\
\tilde{b}_{up,m1} &:= b_{m1} \sum_{i=0}^{\Delta+1} (a_1)^i.
\end{aligned}$$

This means the parameters are always transformed exactly for the integer case and if the ratio $\frac{\tilde{H}}{H}$ is of non-integer type, i.e. $\delta \neq 0$, the parameters are linear interpolated. The benefit of using the linear interpolation is that, there are no problems if one of the parameter is negative, because therefore the transformed parameter might get an imaginary part. The loss of accuracy through this linear interpolation can be neglected for two reasons. First, the transformed parameters are only used as initial parameters and will therefore adapt themselves anyway. Second, the parameters utilized for the transforming originate from a previous run and they are saved from a moment, where nearly all signals are steady, therefore the error of the saving of the parameters predominate the errors introduced to the linear interpolation for case of $\delta > 0$.
The effect of the shortening of the learning phase, from $12s$ to $2s$, and replacing the zero initial parameters with the recalculated ones described above, is depicted in Figure 4.26. There the major improvement in terms of accuracy $\epsilon_{mono,global}$ and stability $\sigma_{mono,global}$ is clearly visible for $\Delta T > 0.18s$. Due to the loaded initial parameters it is possible to generate stable results where also the accuracy is sufficient. This means that the Model-based Pre-Step Stabilization Method is able to generate a stable co-simulation, with a sufficient accuracy, even for the case of large communication steps. The problem of an instable learning phase, due to ZOH coupling, can be overcome by shortening the learning phase dramatically and setting recalculated initial parameters from an previous stable run. For the demonstration that the results are really improved, the six helicopter output signals are depicted in Figures 4.27 and 4.28. In all six subplots the same effects can be seen, for the Pre-Step SI result with the $12s$ learning phase and no initial parameters, there are strong oscillations from the beginning until $t \approx 25s$. These oscillations origin from the instable behavior of the ZOH coupling in the learning phase. After these oscillations the result stabilizes, but it is obviously less accurate than the results obtained by setting the recalculated initial parameters and shortening the learning phase to $2s$. The difference in terms of accuracy also at the end of the simulation is impressive, the difference between the monolithic and the Pre-Step SI loaded Initial Parameter result is nearly not visible. Having a

closer look at the result regarding the velocity in Figure 4.29, the improved accuracy of the results by loading the initial parameters is even more obvious. Especially for the case of the velocity in z-direction the monolithic result is nearly coinciding with the pre-step results utilizing the loaded initial parameters. The same effects of high oscillations at the classical pre-step results and an obvious improvement in terms of accuracy can be seen at the four outputs of the controller in Figures 4.30 and 4.31. Especially of interest is that the classical pre-step result is always superimposed by high frequent oscillations, also at the end of the simulation. The pre-step results, with utilizing the recalculated initial parameters, are free of these oscillations and the gap to the monolithic result is significantly smaller. As the only difference of the two pre-step results are the identified Interface Jacobians, represented by the parameters a_1 and $b_{11}, \ldots, b_{m1}$, the progression of the identified parameters is of special interest and is therefore depicted in Figure 4.32 for both cases, for the output signal heave velocity of the helicopter. As one can see, the parameters with initial values unequal zero are free of any oscillations and also the values at the end of the simulation are significantly different. This means that the wrong choice of initial conditions and a too long learning phase have significant influence for the whole simulation, not only at the beginning of the simulation. The loaded initial parameters seem like a good choice, as there is no major adaption of the parameter over the time, small and smooth adaptions are always necessary as the less complex Interface Jacobians are approximating more complex subsystems by a time-variant approach. At a first glance, the results of the error measure of the system identification $\epsilon_{SI,global}$ depicted in the upper part of Figure 4.33 seem paradox, as the pre-step results are way better than the ones with the loaded initial parameters. But the reason is easy to see in (2.92), as the scaling of the error measure is done by the amplitude of the past samples of the signal and for the case of no loaded initial parameters there are high oscillations, where the peak values are up 100 times higher than for the results with loaded initial parameters. This leads to the conclusion that $\epsilon_{SI,global}$ is smaller although the actual approximation is worse than the other. This means the measure $\epsilon_{SI,global}$ is not suited for signals with high oscillations. Comparing the progression of $\epsilon_{SI,global}$ in Figure 4.33 for the case of loaded initial parameter with the one of the classical pre-step results in Figure 4.25 one can easily see that the amplitude and also the oscillations are very similar, which is an additional hint that the version with the loaded initial parameters works really well. As at the beginning of the simulation, in the first $\approx 20s$, there are some high peaks of $\epsilon_{SI,global}$ for the case of loaded initial parameters, the modified pre-step phase is activated for a few short periods. As explained in detail in Section 2.4.2 the modified pre-step phase is a special version of the Model-based Pre-Step Stabilization method which is less sensitive against approximation errors in the Interface Jacobians. The need for this phase switching is to damp the influence of these approximation errors before they can lead to weak or even instable results, one can see this switching as a security function, for preventing the coupling getting instable. The phase switching works well as after $\approx 20s$ there is no further need for any switching.

Figure 4.26 shows that the results, if the loaded initial parameters are utilized, are stable up to $\Delta = 0.3s$. As demonstration that also the accuracy, especially after $\approx 20s$, is sufficient, the six outputs of the helicopter are depicted in Figures 4.34 and

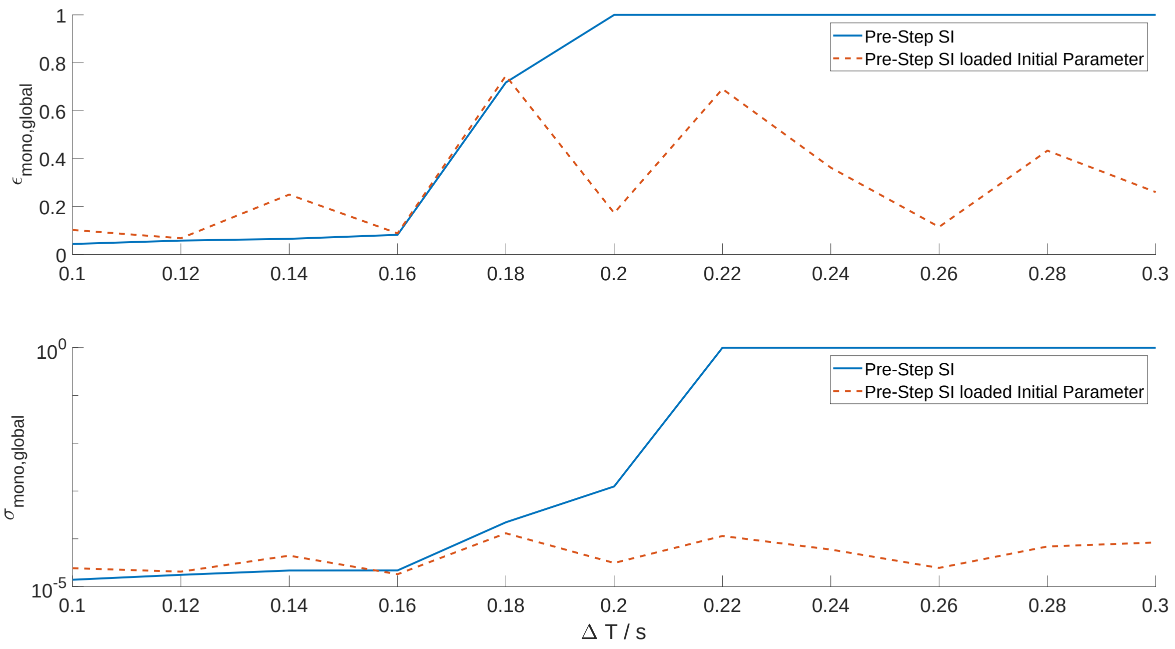


Figure 4.26: *Error measure $\epsilon_{mono,global}$ and stability measure $\sigma_{mono,global}$ dependent on ΔT for pilot commands version 1 for the case of loaded initial parameters*

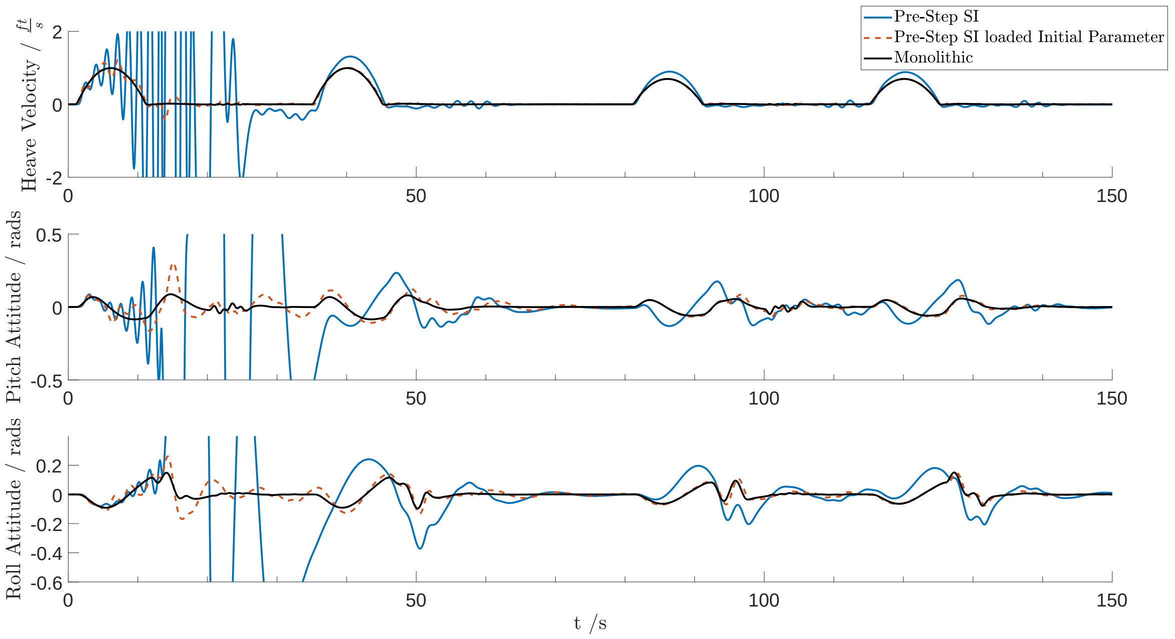


Figure 4.27: *Helicopter outputs for $\Delta T = 0.2s$ for the case of loaded initial parameters: heave velocity, pitch attitude and roll attitude*

4.35. It should be emphasized, that there are no high frequent oscillations in the results and that the heave velocity nearly coincides with the monolithic result and also for the other five signals the accuracy after a short period is sufficient. The velocity

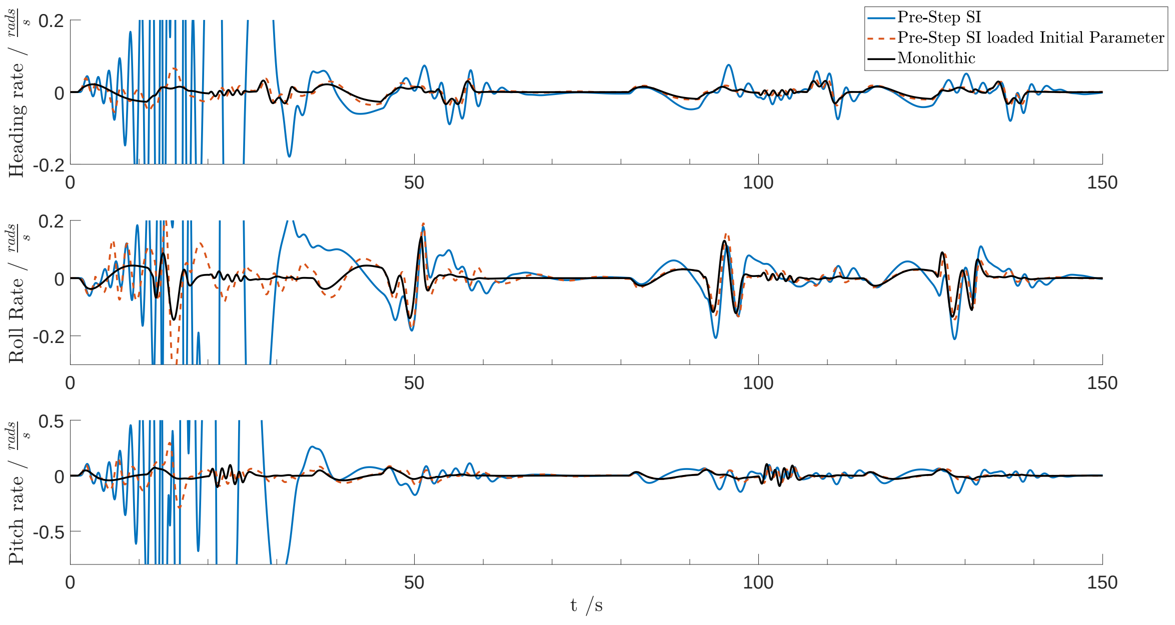


Figure 4.28: *Helicopter outputs for $\Delta T = 0.2s$ for the case of loaded initial parameters: heading rate, roll rate and pitch rate*

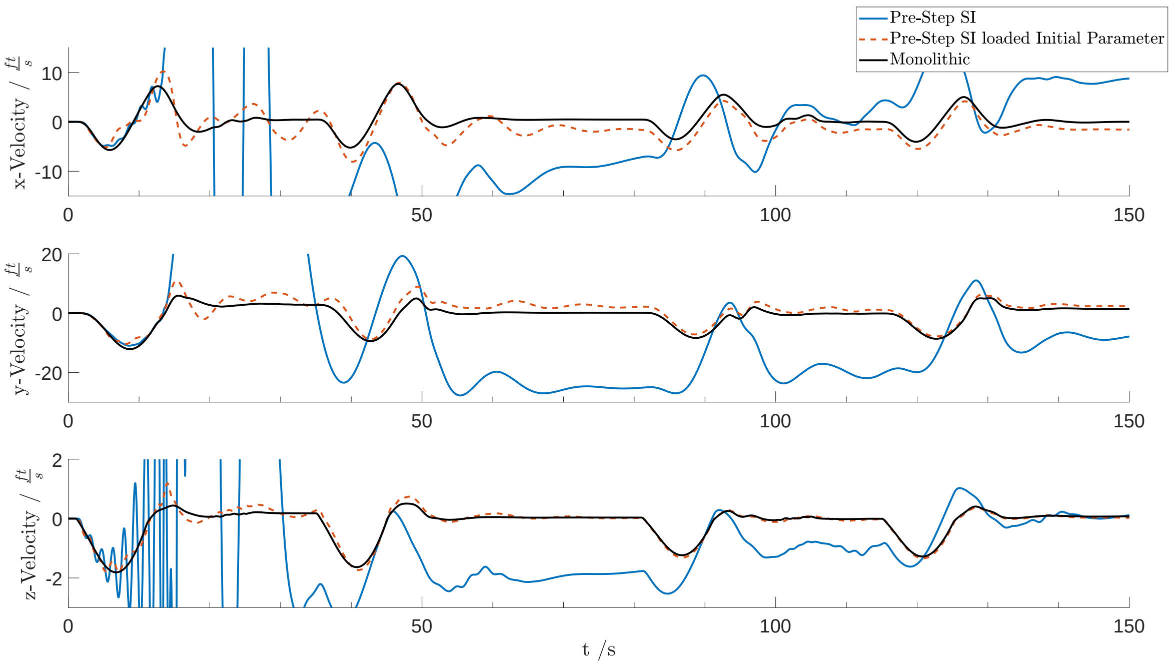


Figure 4.29: *x-, y- and z-velocity of the helicopter for $\Delta T = 0.2s$ for the case of loaded initial parameters*

of the helicopter is depicted in Figure 4.36, there it is remarkable how accurate the co-simulated z-velocity fits to the monolithic result. The outputs of the controller are depicted in Figures 4.37 and 4.38, there one sees that, there are high oscillations for

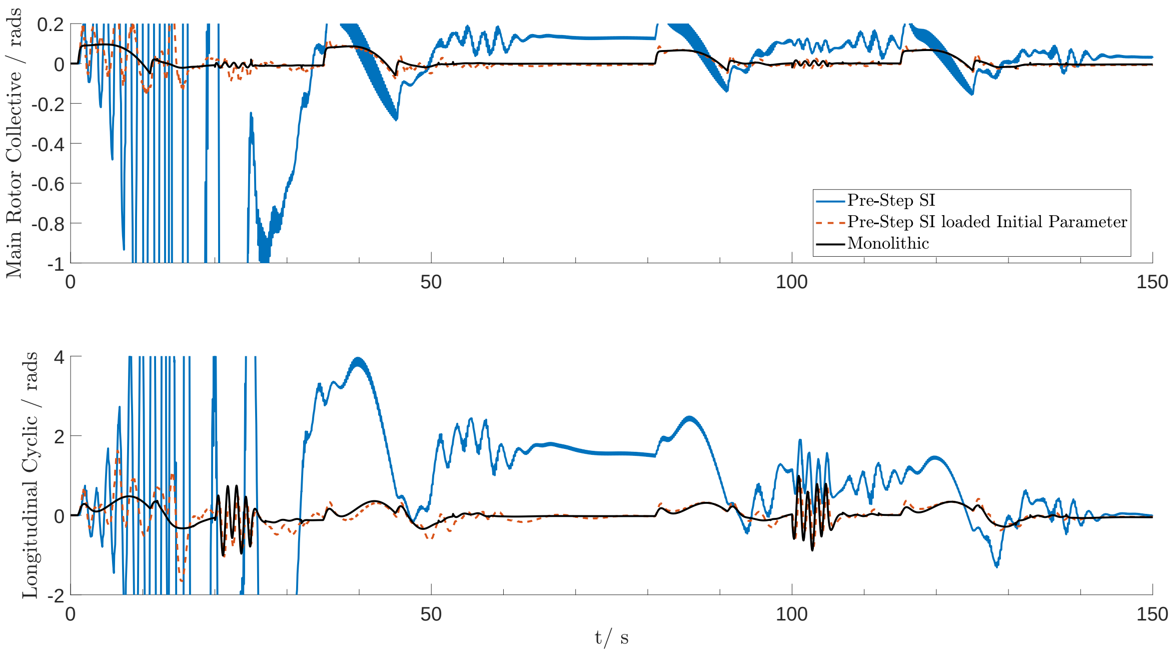


Figure 4.30: *Controller outputs for $\Delta T = 0.2s$ for the case of loaded initial parameters: main rotor collective and longitudinal cyclic*

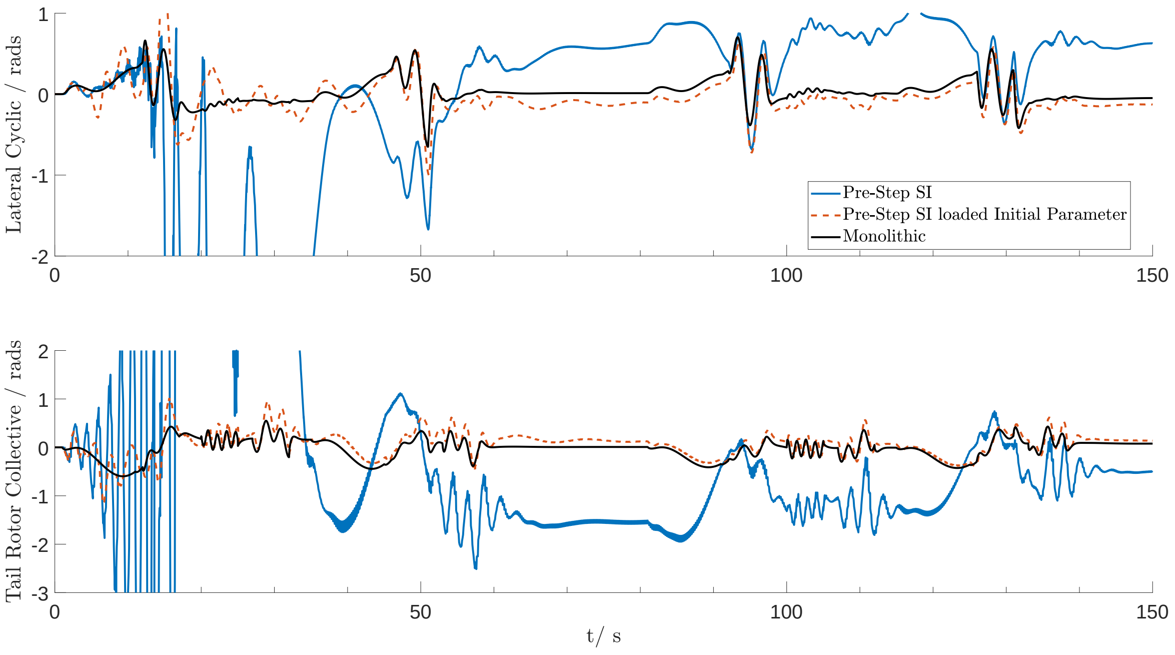


Figure 4.31: *Controller outputs for $\Delta T = 0.2s$ for the case of loaded initial parameters: lateral cyclic and tail rotor collective*

the first 20 seconds but afterwards the results are free of oscillations and the accuracy is satisfying. Especially if one is considering the fact that a constant gap to the monolithic results is not as harmful as oscillations. Summarizing this means, that the

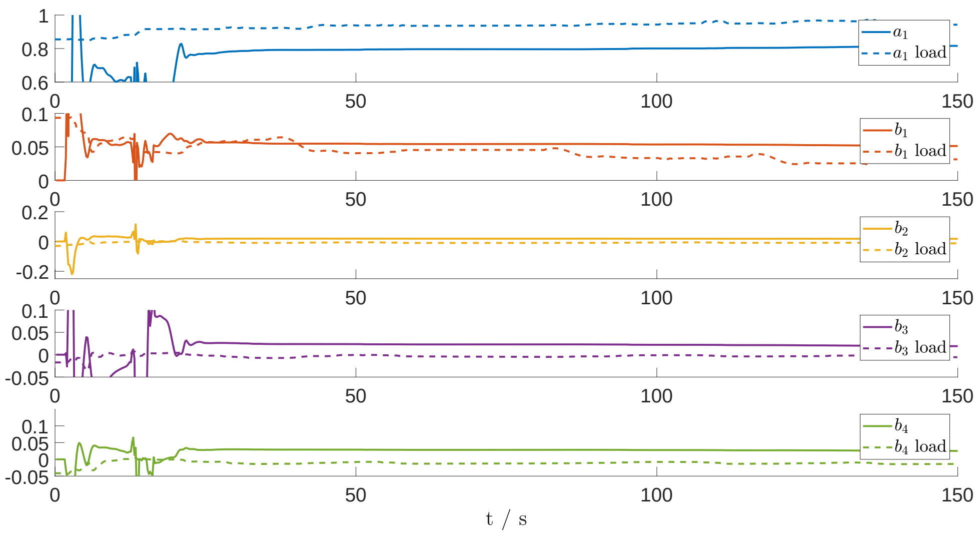


Figure 4.32: *Comparison of the identified parameters for the case with and without loaded initial parameters, for the helicopter output heave velocity*

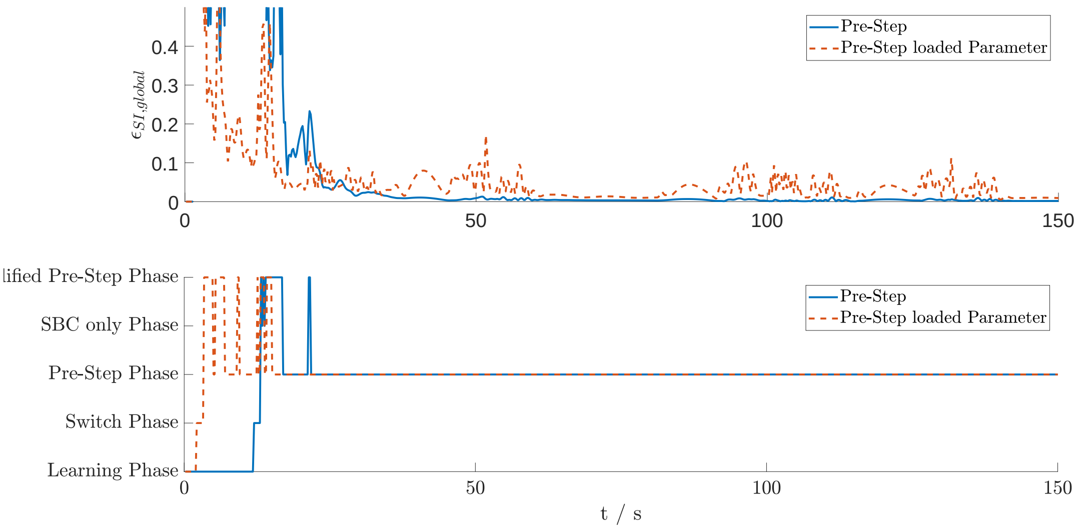


Figure 4.33: *Quality of the identified Interface Jacobians and the progression of the Error-based Phase Check for the case of loaded initial parameters*

option to load initial parameters and so keeping the learning phase to a minimum, is improving the performance of the Model-based Pre-Step Stabilization in terms of stability and accuracy for large communication step sizes.

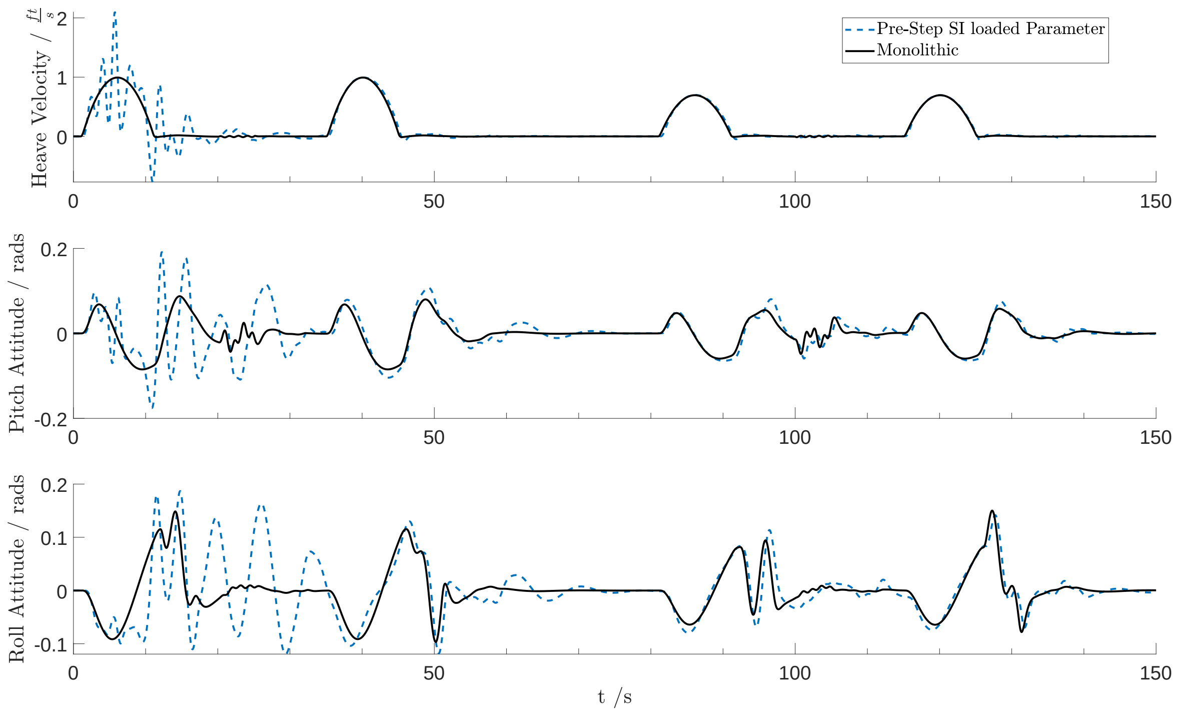


Figure 4.34: *Helicopter outputs for $\Delta T = 0.3s$ for the case of loaded initial parameters: heave velocity, pitch attitude and roll attitude*

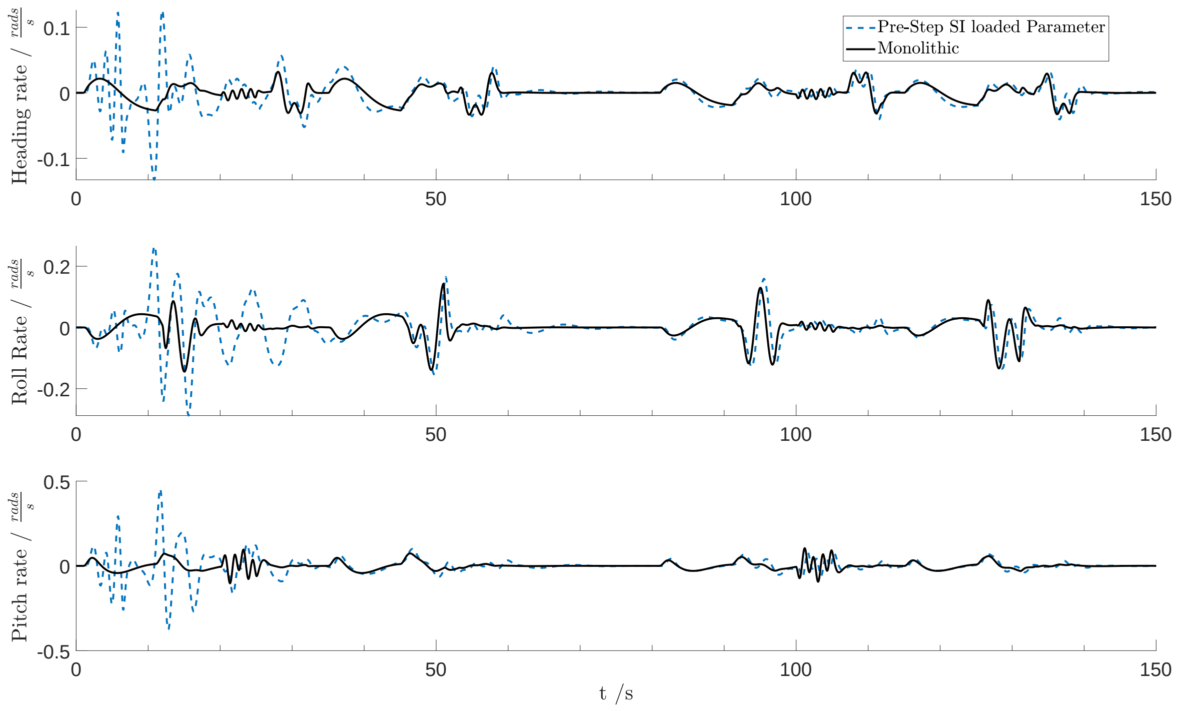


Figure 4.35: *Helicopter outputs for $\Delta T = 0.3s$ for the case of loaded initial parameters: heading rate, roll rate and pitch rate*

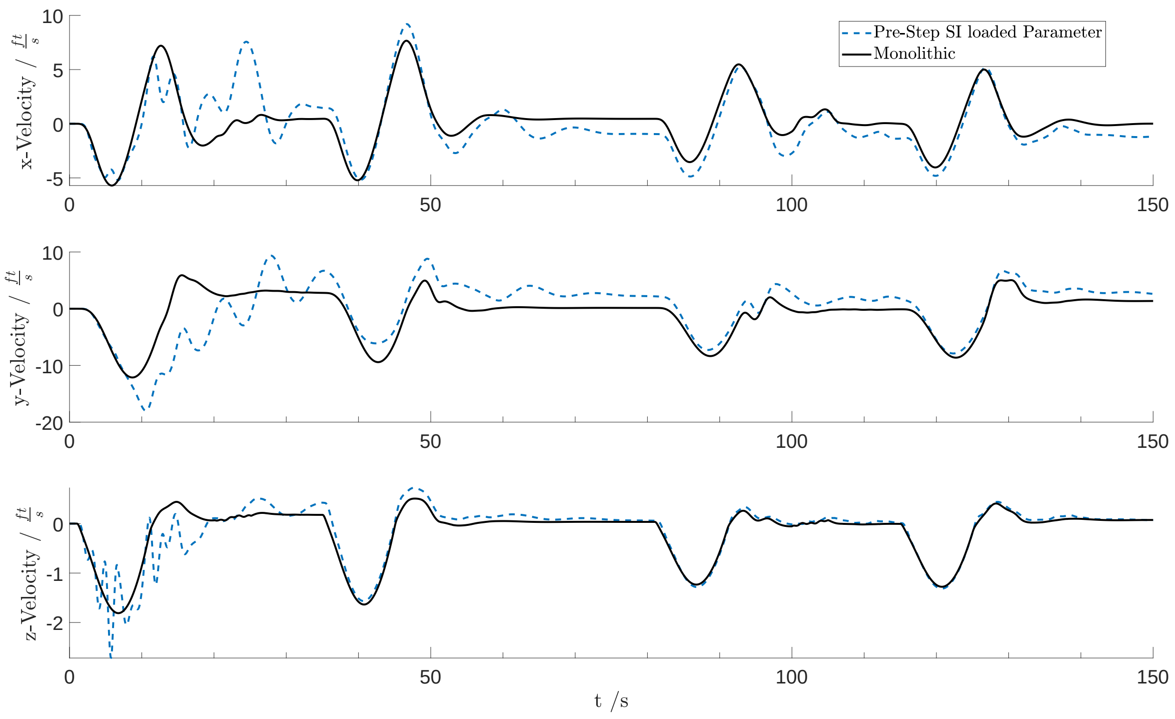


Figure 4.36: *X-, y- and z-velocity of the helicopter for $\Delta T = 0.3s$ for the case of loaded initial parameters*

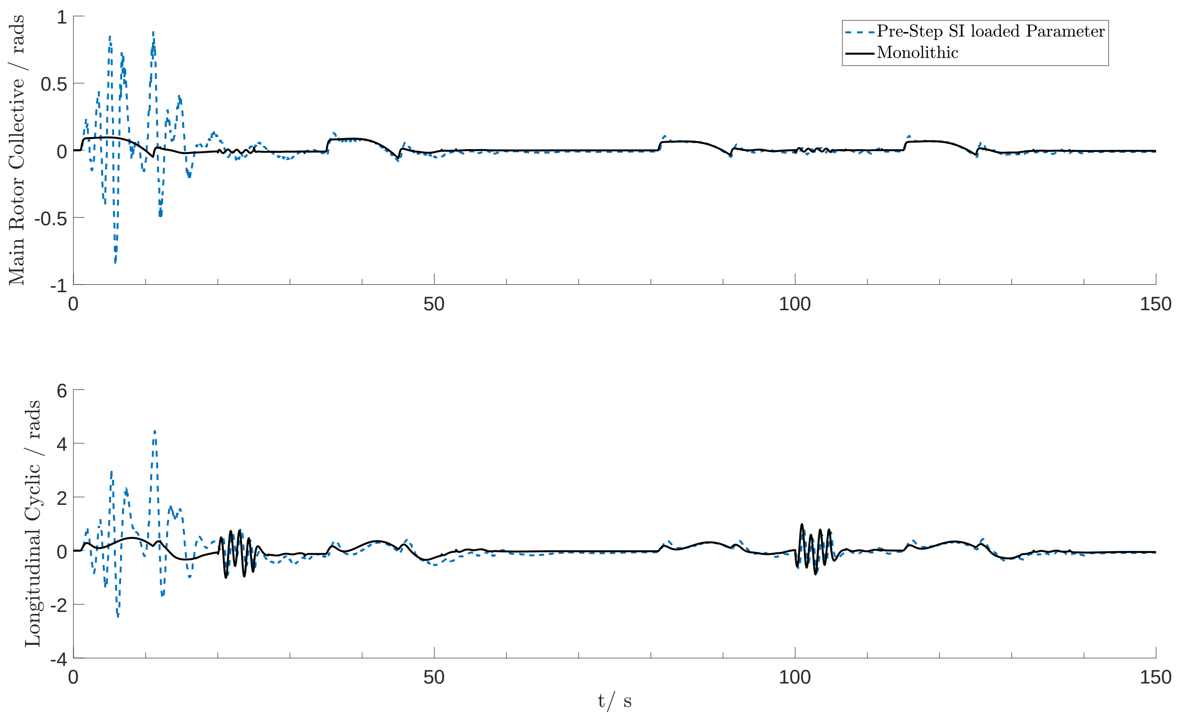


Figure 4.37: *Controller outputs for $\Delta T = 0.3s$ for the case of loaded initial parameters: main rotor collective and longitudinal cyclic*

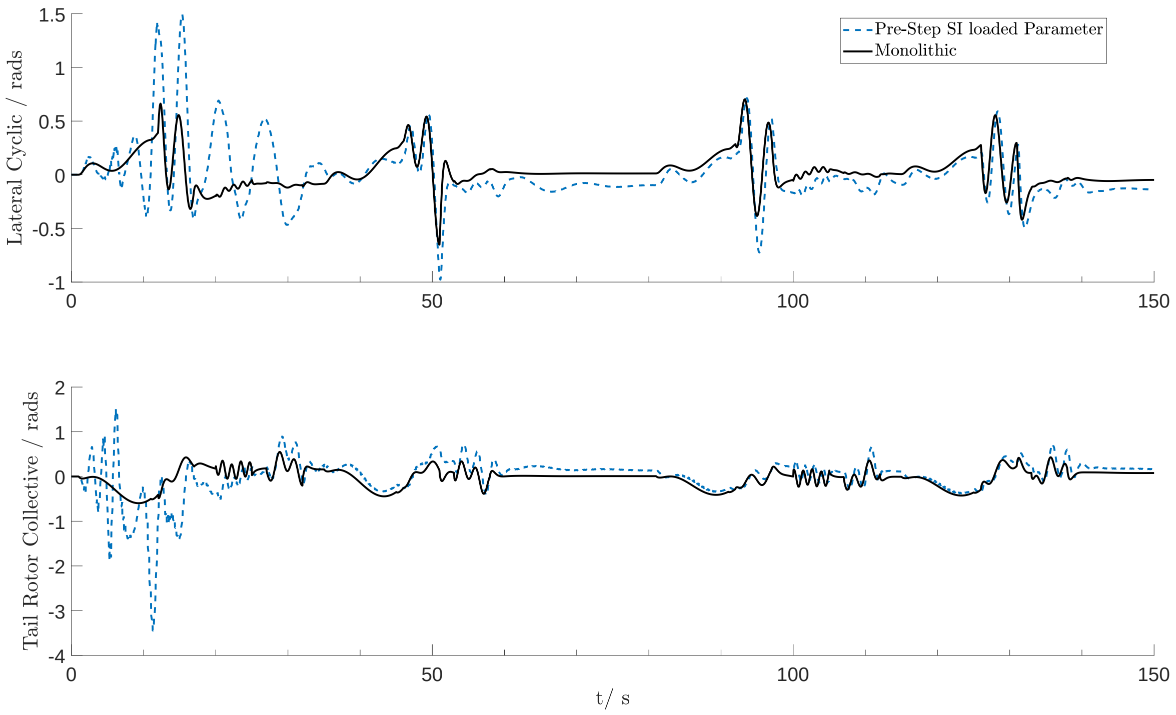


Figure 4.38: *Controller outputs for $\Delta T = 0.3s$ for the case of loaded initial parameters: lateral cyclic and tail rotor collective*

4.2 Summary and Conclusion of the Helicopter-Control Co-Simulation

As evaluation of the Model-based Pre-Step Stabilization Method, the coupling method has been compared to the classical ZOH coupling, NEPCE method and the Model-based Corrector approach with exact (MBCorr exact) and approximated (MBCorr SI) subsystem information, at the co-simulation of a helicopter and its control. The two major outcomes of this evaluation will be stated and discussed in the following.
First, the full potential of the Model-based Pre-Step Stabilization Method can be seen, if the learning phase is drastically shortened, as otherwise the simulation already gets instable in the learning phase for larger communication step-sizes. For such a short learning phase, well chosen initial parameters are required. These parameters can be accessed from a previous run with a smaller communication step size, where all results are stable and accurate. Figure 4.26 shows the satisfying improvements in terms of accuracy and stability by utilizing the recalculated initial parameters.
Second, the fact that utilizing the Model-based Pre-Step Stabilization Method with approximated instead of exact Interface Jacobians is leading to much better results is surprising and sounds paradox. The fact that the Interface Jacobian subsystem description is derived by a transformation rule where the inversion of the output matrix $\boldsymbol{C}$ is necessary, is the origin for the instable behaviour for the case of exact subsystem information. Because the number of states in the subsystems is higher than the outputs and therefore the utilization of the pseudo inverse, for inverting $\boldsymbol{C}$, is canceling

out too much significant dynamics. This effect can be investigated at the helicopter control co-simulation as the states of the subsystems are highly coupled. The key to overcome the problem of inverting $\boldsymbol{C}$ is to choose the number of states, for the approximated Interface Jacobians, equally to the number of outputs. Therefore the complex subsystems, with a high number of states, are approximated by less complex surrogate models which are time-varying, this means the loss of complexity is compensated by a self-adaption of the approximated Interface Jacobians.

5

Summary and Conclusion

The improved stability and accuracy of the Model-based Pre-Step Stabilization Method will lead to a more trustworthy co-simulation. Therefore this coupling method will facilitate the way to include co-simulation into the virtual validation processes of the future.

5.1 Model-based Pre-Step Stabilization Method

The goal of the Model-based Pre-Step Stabilization Method is to stabilize the co-simulation of multiple subsystems by exploiting subsystem information, the so-called Interface Jacobians. The method itself consists of three main steps:

1. Approximating the monolithic output, by solving the Error Differential Equation, which is based on the extrapolation error of the last communication step.
2. Extrapolating the determined monolithic output one communication step into the future, under the consideration of cross couplings between the subsystems.
3. Optimizing the input for every subsystem in such a way, that the co-simulated output of the subsystem fits the extrapolated monolithic one.

As the subsystem description can be in continuous- or discrete-time manner, two appropriate versions of the Model-based Pre-Step Stabilization Method exists, which are stated in Section 2.2 with all further details to the three main steps. These three main steps are based on the Interface Jacobians and therefore the question how to access these mandatory quantities is of high significance. As most subsystems can be described as state space systems, the connection to the Interface Jacobian subsystem description is derived and stated in Section 2.1. On the one hand the Interface Jacobians can be provided by the subsystem itself and on the other hand the Interface Jacobians can be approximated by the invented coupling method, based on the in- and the outputs of the subsystems. For this purpose, two system identification methods have been presented and analyzed, the Multivariable Output Error State Space (MOESP) approach and the Recursive Least Square (RLS) method, all required details are stated in Section 2.3. Due to the fact, that the quality of the approximated Interface Jacobians is important for the coupling method, the quality measure is determined in every communication step. Based on this measure the error based check decides in every communication step, whether the classical Model-based Pre-Step Stabilization Method or the modified version, where step three is skipped, or whether only zero-order-hold coupling is performed. For more details, see Section 2.4, which deals with the implementation aspects. Further discussed in this section, are the workflow of the coupling method and the learning and switch phase. The learning phase is required for the system identification methods to gather enough data, so that the calculated Interface Jacobians are useful. The switch phase is applied right after the end of the learning phase, because the switching from the signal-based coupling in the learning phase, to the pre-step coupling, might lead to deviations or oscillations which should be smoothed and damped by this switching approach.

5.2 Analysis

The sensitivity analysis of the invented coupling method analyzed the case of exact and approximated Interface Jacobians. As typically exact Interface Jacobians, provided by the subsystems themselves, are stated in continuous-time manner, this case is focusing on the continuous-time version of the Model-based Pre-Step Stabilization Method. The analysis of variance has shown, that the second step of the coupling method, the model-based extrapolation, is the most significant one. Due to the so-called Ascending Sorting Analysis, an optimal set of parameters for the coupling method has been determined, see therefore Table 3.5.
By analyzing the case of approximated Interface Jacobians, the discrete-time version of the Model-based Pre-Step Stabilization Method has been in focus. This sensitivity analysis has shown, that the RLS and the MOESP approach are both useable system identification methods. The difference is, that the parametrization of the RLS is easier and the average results are better, i.e. it is more robust, and therefore the RLS is the preferred choice of system identification method for approximating the Interface Jacobians. The Ascending Sorting Analysis determined an optimal set of parameters for utilizing the RLS and the MOESP method, see therefore Tables 3.12 and 3.13.

As the main goal of the Model-based Pre-Step Stabilization Method is to increase the co-simulation performance in terms of stability, a stability analysis has been carried out. Therefore the dual mass oscillator co-simulation has been chosen as the test problem. The co-simulation is investigated with regard to three different types of stability: numerical stability, exponential stability and bounded-input bounded-output (BIBO) stability. Additionally it should be stated, that the co-simulation is zero-stable, as there are no algebraic loops.
Numerical stability deals with the influence of errors originating from a finite communication step size, e.g. the coupling error. For analyzing this kind of stability, the stiffness and damping coefficients of the spring damper, connecting the two masses, and the communication step size are varied. Therefore the linear dual mass oscillator with a time-invariant subsystem description is chosen and therefore the case of exact Interface Jacobians is the appropriate one to investigate. The Model-based Pre-Step Stabilization Method shows a clearly enlarged stability region compared to the other state of the art co-simulation coupling techniques, see therefore Figure 3.52. This

proves that the invented coupling method is stabilizing the co-simulation, which is especially important for the use of large communication steps.
The terms exponential and BIBO stability can both be investigated for time-variant subsystem descriptions and therefore the non-linear dual mass oscillator, extended with an external force has been chosen as test problem. As the subsystem description is time-variant, the choice of the approximated Interface Jacobians is appropriate, i.e. the discrete-time version of the invented coupling method will be utilized. Exponential and BIBO stability can be simultaneously determined by evaluating the so-called Bohl exponent. As the computed Bohl exponent is less than 1 and the time-variant matrices are bounded in every step it follows that the co-simulation utilizing the Model-based Pre-Step Stabilization Method is exponential and BIBO stable.

Although the stability is the main focus of this thesis, the accuracy of the coupling method should not be neglected. The influence of the stiffness and damper coefficients and the communication step size, according the accuracy of the co-simulation has been investigated, as reference the monolithic results were utilized. The region of accurate results is quite similar to the stability region, which means the invented coupling method is not only improving the stability, also the accuracy of the co-simulation is enhanced. For certain stiffness and damper coefficients, the effect of the communication step size according the accuracy of the results is investigated in Figure 3.62. There it is clearly visible, that by utilizing the Model-based Pre-Step Stabilization Method the choice of a larger communication step size is possible without any loss in the quality of the results.

5.3 Evaluation

In Chapter 4 the Model-based Pre-Step Stabilization Method is evaluated at the co-simulation of a helicopter and its control. This co-simulation has been chosen mainly for the following two reasons. First, the exact Interface Jacobians are accessible and therefore the effect of approximating the Interface Jacobians can be investigated. Second, the helicopter represents an instable subsystem and the controller a stiff one and therefore the co-simulation is challenging. Additionally the states of both subsystems are highly coupled, as the dynamics of a helicopter are in nature highly coupled.
The influence of the five different, pre-defined versions of the pilot commands is not significant for the co-simulation, this makes the evaluation less complicated, as only one version of the pilot commands is utilized for the further evaluation. Comparing the cases of exact and approximated Interface Jacobians results in the paradox, that the approximated Interface Jacobians lead to much better results than the exact ones, the results utilized by the exact Interface Jacobians are even instable. This leads to the conclusion, that it is not possible to guarantee that utilizing exact Interface Jacobians is leading to better results. As described in detail in Section 4.1.3, the reason for this paradox effect is, that the subsystems have more states than outputs and additionally the states in the subsystems are highly coupled. Taken into account that accessing exact Interface Jacobians is hard to achieve, as nearly no simulation software is supporting such a feature right now, the preferred way is to approximate the Interface

Jacobians.
First, it should be mentioned, that utilizing the Model-based Pre-Step Stabilization Method is leading to improved results according stability and accuracy, especially the unwanted oscillations in the results are damped. As the RLS is utilized as system identification method the influence of the choice of the initial values has been investigated. With using zero initial values, the learning phase has to be sufficiently large to get stable results. There the problem arises, that if the co-simulation starts to oscillate already in the learning phase, the approximated Interface Jacobians are not sufficiently accurate, so the coupling by the Model-based Pre-Step Stabilization Method will oscillate and gets instable too. By setting appropriate initial values, loaded from a previous stable run with a lower communication step size, the learning phase can be shrinked to a minimum. As now the instable behaviour in the learning phase can be avoided, the region of stable and accurate results is clearly enlarged. This means that one can set the communication step size larger, without a significant loss in the quality of the results in terms of stability and accuracy, which represents the main goal of the coupling method. Therefore the evaluation stated that with utilizing the Model-based Pre-Step Stabilization Method the co-simulation performance in terms of stability and accuracy is clearly improved.

5.4 Outlook

The main goal for the future is to make the presented coupling method available for a broader audience and therefore the implementation in the co-simulation platform Model.CONNECT$^{\text{TM}}$, see [27], will be in focus. This topic can be seen from two different viewpoints, on the one hand the implementation of the coupling method into the existing framework. On the other hand how the user should be able to interact or adjust the method, which and how the parameters and settings should be adjusted automatically. For example the question on which part of the co-simulation the Model-based Pre-Step Stabilization Method should be applied has to be answered. In any case it will not be necessary to couple every signal or even every subsystem with this enhanced coupling method, zero-order-hold coupling will be sufficiently for some signals for example. The usability of the Model-based Pre-Step Stabilization Method will be the main key to integrate the coupling method in Model.CONNECT$^{\text{TM}}$ successfully.

Abbreviations

FMI	Functional Mock-Up Interface
FMU	Functional Mock-Up Unit
CI/CD	Continuous Integration and Continuous Development
sbc	Signal based coupling
mbc	Model based coupling
NEPCE	Nearly Energy Preserving Coupling Element
ZOH	Zero-order-hold coupling
FOH	First-order-hold coupling
MOESP	Multivariable Output Error Stace Space
RLS	Recursive Least Squares
SISO	Single input single output
MISO	Multiple input single output
MIMO	Multiple input multiple output
QA	Quality Assessment
SI	System Identification
SS_A	Sum of squares for the main Effect according A
SS_{AB}	Sum of squares for the interaction Effect according A and B
DoF_A	Degree of freedom according A
DoF_{AB}	Degree of freedom according the combination of A and B
MS_A	Mean sum of squares for the main effect according A
MS_{AB}	Mean sum of squares for the interaction effect according A and B
DoE	Design of experiment
BIBO	Bounded-input bounded-output
MBCorr	Model-based Corrector

List of Figures

List of Tables

Bibliography

[1] R. Kübler and W. Schiehlen, “Two methods of simulator coupling.” Taylor and Francis, 2000. (cited on pages 2, 14 and 94.)

[2] T. Blochwitz and et al., “Functional mockup interface 2.0: The standard for tool independent exchange of simulation models.” 9th International Modelica Conference Munich, 2012. (cited on pages 2, 7 and 9.)

[3] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, 2nd ed. New York: Springer Verlag, 1993. (cited on pages 4 and 29.)

[4] M. Arnold, “Multi-rate time integration for large scale multibody system models.” IUTAM Symposium on Multiscale Problems in Multibody System Contacts, 2006. (cited on pages 5 and 6.)

[5] T. Haid, G. Stettinger, D. Watzenig, and M. Benedikt, “A model-based corrector approach for explicit co-simulation using subspace identification.” The 5th Joint International Conference on Multibody System Dynamics, 2018. (cited on pages 5, 6, 27, 43 and 96.)

[6] M. Benedikt, D. Watzenig, and A. Hofer, “Modelling and analysis of the noniterative coupling process for co-simulation.” TaylorFrancis, 2013. (cited on pages 5, 43 and 96.)

[7] M. Benedikt and A. Hofer, “Guidelines for the application of a coupling method for non-iterative co-simulation.” IEEE, 8th Congress on Modelling and Simulation (EUROSIM), Cardiff Wales, 2013. (cited on pages 5 and 96.)

[8] M. Benedikt and E. Drenth, *Relaxing Stiff System Integration by Smoothing Techniques for Non-iterative Co-simulation*, 01 2019, pp. 1–25. (cited on pages 5 and 17.)

[9] S. Sadjina, L. Kyllingstad, S. Skjong, and E. Pedersen, “Energy conservation and power bonds in co-simulations: non-iterative adaptive step size control and error estimation.” arXiv preprint arXiv:1602.06434, 2016. (cited on page 5.)

[10] M. Arnold, “Numerical stabilization of co-simulation techniques, the ode case.” Martin Luther University Halle-Wittenberg NWF II-Institute of Mathematics, 2011. (cited on page 6.)

[11] S. Sicklinger, V. Belsky, and B. Engelmann, “Interface-jacobian based co-simulation.” NAFEMS World Congress 2013, 2013. (cited on pages 6 and 9.)

[12] S. Sicklinger, V. Belsky, B. Engelmann, H. Elmqvist, H. Olsson, R. Wüchner, and K.-U. Bletzinger, “Interface jacobian-based co-simulation,” *International Journal for Numerical Methods in Engineering*, vol. 98, 05 2014. (cited on pages 6 and 9.)

[13] F. Cellier and E. Kofman, *Continuous System Simulation*, 1st ed. New York: Springer, 2010. (cited on pages 7, 16, 18 and 19.)

[14] T. Katayama, *Subspace Methods for System Identification*, 1st ed. London: Springer-Verlag London Limited, 2005. (cited on pages 7 and 27.)

[15] P. V. Overschee and D. B. Moor, *Subspace Identification for Linear Systems*, 1st ed. London: Kluwer Academic Publishers, 1996. (cited on pages 7 and 27.)

[16] P. Trnka, “Subspace identification methods,” 2005, [Czech Technical University in Prague; Department of Control Engineering]. (cited on pages 7 and 27.)

[17] L. Ljung, *System Identification: Theory for the User*, 2nd ed. Prentice Hall, New Jersey,: PTR Prentice Hall Information and System Sciences Series, 1999. (cited on pages 7, 31, 32, 34, 35 and 64.)

[18] M. H. Hayes, *Statistical Digital Signal Processing and Modeling*, 13th ed. Brisbane: John Wiley and Sons, 1996. (cited on pages 7, 31, 32 and 64.)

[19] C. Moler and C. Loan, “Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later,” *Society for Industrial and Applied Mathematics*, vol. 45, pp. 3–49, 03 2003. (cited on page 8.)

[20] S. A. Sicklinger, “Stabilized co-simulation of coupled problems including fields and signals,” Ph.D. dissertation, Technische Universität München, 2014. (cited on pages 9 and 43.)

[21] S. Genser and M. Benedikt, “Model-based pre-step stabilization method for non-iterative co-simulation.” IMSD Tecnico Lisboa, 2018. (cited on page 9.)

[22] S.Genser and M. Benedikt, “Extension of the model-based pre-step stabilization method for non-iterative co-simulation - including direct-feedthrough.” Proceedings of 8th International Conference on Simulation and Modeling Methodologies Technologies and Applications - Volume 1: SIMULTECH, 2018. (cited on pages 9 and 43.)

[23] S. Genser and M. Benedikt, “A pre-step stabilization method for non-iterative co-simulation and effects of interface-jacobians identification.” Advances in Intelligent Systems and Computing, Springer Verlag, 2018. (cited on pages 9, 13, 39 and 43.)

[24] A. Isidori, *Nonlinear Control Systems*, 3rd ed. London: Springer Verlag, 1995. (cited on page 9.)

[25] S. Genser, G. Stettinger, D. Watzenig, and C. Pötzsche, “Numerical analysis of exponential and bibs stability for linear discrete time-variant systems using bohl exponents.” MED 2020 - 28th Mediterranean Conference on Control and Automation, 2020. (cited on pages 10, 95, 101 and 106.)

[26] G. Golub and W. Kahan, “Calculating the singular values and pseudo-inverse of a matrix,” *Journal of the Society for Industrial and Applied Mathematics Series B Numerical Analysis*, vol. 2, no. 2, pp. 205–224, 1965. (cited on page 11.)

[27] AVL, “Model.CONNECT$^{\text{TM}}$, the neutral model integration and co-simulation platform connecting virtual and real components,” http://www.avl.com/-/model-connect-, 2018, [Online; accessed 03.11.2020]. (cited on pages 12 and 156.)

[28] G. Stettinger, J. Zehetner, M. Benedikt, and N. Thek, “Extending co-simulation to the real-time domain,” in *SAE Technical Paper*. SAE International, 04 2013. (cited on page 12.)

[29] M. Busch, “Zur effizienten kopplung von simulationsprogrammen,” Ph.D. dissertation, University Kassel, 2012. (cited on pages 14, 43 and 94.)

[30] B. Friedland, *Control System Design: An Introduction to State-space Methods*, 1st ed. Mineola, New York: Dover Publications, 1986. (cited on page 21.)

[31] J. W. Demmel, *Applied Numerical Linear Algebra*, 1st ed. Philadelphia: SIAM, 1997. (cited on pages 25 and 29.)

[32] A. Ben-Israel and T. N. Greville, *Generalized Inverses - Theory and Applications*, 2nd ed. New York: Springer-Verlag, 2003. (cited on page 25.)

[33] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*, 1st ed. New York: Academic Press, 1982. (cited on page 25.)

[34] J. Lunze, *Regelungstechnik 2 Mehrgrößensysteme, Digitale Regelung*, 6th ed. Bochum: Springer Verlag, 2010. (cited on pages 31 and 33.)

[35] M. Green and J. B. Moore, “Persistence of excitation in linear systems,” in *1985 American Control Conference*, 1985, pp. 412–417. (cited on page 34.)

[36] S. Genser, G. Stettinger, and F. Holzinger, “Identification of informative time frames for system identification purposes - a time-varying order of excitation assessment approach,” 5 2019, 15th International Conference of Computational Methods in Sciences and Engineering ; Conference date: 01-05-2019 Through 05-05-2019. [Online]. Available: https://www.iccmse.org/ (cited on page 35.)

[37] S. Genser, F. Holzinger, and G. Stettinger, “Computation of the ideal step-size - a constant and adaptive approach,” 5 2019, 15th International Conference of Computational Methods in Sciences and Engineering ; Conference date: 01-05-2019 Through 05-05-2019. [Online]. Available: https://www.iccmse.org/ (cited on page 35.)

[38] G. Stettinger, M. Horn, M. Benedikt, and J. Zehetner, "A model-based approach for prediction-based interconnection of dynamic systems." 53rd IEEE Conference on Decision and Control, 2014. (cited on page 41.)

[39] D. C. Montgomery, *Design and Analysis of Experiments*, 8th ed. Arizona State University: John Wiley and Sons, 2013. (cited on pages 47 and 50.)

[40] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I*, 2nd, Ed. Springer-Verlag Berlin Heidelberg, 1993. (cited on pages 49, 120 and 123.)

[41] G. Ludyk, *Stability of Time-Variant Discrete Time Systems.* Springer Fachmedien Wiesbaden GmbH, 1985. (cited on pages 95, 101 and 102.)

[42] K. Przyluski, "Remarks on the stability of linear infinite-dimensional discrete-time systems." Journal of Differential Equations, 1988. (cited on page 106.)

[43] G. Padfield, "Theoretical dynamic modelling of helicopter flight dynamics." Flight Dynamics Divison, Royal Aerospace Establishment, Bedford, UK, 1981. (cited on pages 115 and 117.)

[44] I. P. S. Skogestad, *Multivariable Feedback Control: Analysis and Design*, 2nd ed. John Wiley and Sons, 2001. (cited on pages 115, 118, 120, 121, 122 and 165.)

[45] I. P. A. Yue, "Improvement of helicopter handling qualities using h^∞-optimisation." IEE Proceedings D - Control Theory and Applications, vol. 137, no. 3, pp. 115-129, May 1990, doi: 10.1049/ip-d.1990.0016., 1990. (cited on pages 115, 120 and 121.)

[46] I. Postlethwaite, N. P. Foster, and D. J. Walker, "Rotorcraft control law design for rejection of atmospheric turbulence," in *1994 International Conference on Control - Control '94.*, vol. 2, 1994, pp. 1284–1289 vol.2. (cited on pages 115 and 118.)

[47] N. P. Foster, "H-infinity design for gust turbulence rejection," https://folk.ntnu.no/skoge/book/1st_edition/matlab_m/Sec12_2.m, accessed: 2020-09-21. (cited on pages 118, 119 and 121.)

[48] K. Glover and J. C. Doyle, "A state space approach to h^∞ optimal control," vol. Springer, Berlin, Heidelberg, no. Lecture Notes in Control and Information Sciences, vol 135., 1989, pp. 412–417. (cited on page 120.)

[49] J. C. Doyle, K. Glover, P. P. Khargonekar, and B. Francis, "State-space solutions to standard h^2 and h^∞ control problems." IEEE Transactions on Automatic Control, 34 (8). pp. 831-847., 1989. (cited on page 120.)

[50] D. Garcia, "Robust smoothing of gridded data in one and higher dimensions with missing values." Elsevier, 2010. (cited on page 128.)