

# **Efficient Parallelization of Multibody Systems Incorporating Co-Simulation Techniques**

Vom Fachbereich Maschinenbau  
an der Technischen Universität Darmstadt  
zur Erlangung des Grades  
eines Doktor-Ingenieurs (Dr.-Ing.)  
genehmigte

## **Dissertation**

vorgelegt von

**Jan Kraft, M.Sc.**

aus Ansbach

Berichterstatter: Prof. Dr.-Ing. Bernhard Schweizer

Mitberichterstatter: Prof. Dr. rer. nat. Michael Schäfer

Tag der Einreichung: 06.04.2021

Tag der mündlichen Prüfung: 30.06.2021

Darmstadt, 2021

D17

Jan Kraft, Efficient Parallelization of Multibody Systems Incorporating Co-Simulation Techniques  
Darmstadt, Technische Universität Darmstadt  
Jahr der Veröffentlichung der Dissertation auf TUprints: 2021  
Tag der mündlichen Prüfung: 30.06.2021

Veröffentlicht unter CC BY-SA 4.0 International  
<https://creativecommons.org/licences/>

## Abstract

Co-simulation methods can be used advantageously in the field of multi-disciplinary simulations. Another applicability of co-simulation methods is the parallelization of large monodisciplinary dynamical models. This work focuses on the reduction of computation time that can be achieved in the simulation of multibody systems by partitioning a monolithic model into a set of coupled subsystems. The connection between the subsystems can be described in various ways. In this work, different subsystems are coupled by nonlinear constitutive equations (applied force coupling approach). Information (i.e. coupling variables) is only exchanged between the subsystems at distinct communication-time points (macro-time points). Within each macro-time step, the unknown coupling variables are approximated by extrapolation polynomials. The essential point is that the subsystems are integrated independently of each other between the macro-time points. If a Jacobi type co-simulation scheme is used, all subsystems can be solved in parallel.

A main drawback of many co-simulation implementations is that they are based on an equidistant communication-time grid. Using a constant macro-step size may in many practical applications be not very efficient with respect to computation time, especially in connection with highly nonlinear models or in context with models with strongly varying physical parameters. In this work, explicit and implicit co-simulation approaches which incorporate a macro-step size and order control algorithm, are presented. Numerical examples show the benefit of this implementation and the significant reduction in computation time compared to an implementation with an equidistant communication-time grid. In addition, a comparison between a co-simulation model and a monolithic model demonstrates the great computation time reduction which can be achieved due to the parallelization and the multirate character of the proposed co-simulation methods.

The co-simulation approaches are fully parallelized by a hybrid MPI and OpenMP implementation. The resulting computation time of the implemented co-simulation approaches is analyzed in detail. The influence of various simulation parameters on the computation time is studied and sources of computational overhead are identified.

# Contents

<b>Abstract</b>	<b>3</b>
<b>1. Introduction</b>	<b>6</b>
1.1. Literature Review . . . . .	6
1.2. Motivation . . . . .	10
1.3. Outline . . . . .	13
<b>2. Co-Simulation of Mechanical Systems</b>	<b>14</b>
2.1. Theoretical Background: Decomposition of a Mechanical System into Subsystems Using an Applied Force Coupling Technique . . . . .	14
2.1.1. Force/Force-Decomposition . . . . .	16
2.1.2. Force/Displacement-Decomposition . . . . .	16
2.1.3. Displacement/Displacement-Decomposition . . . . .	18
2.2. Co-Simulation Test Model: Oscillator Chain . . . . .	18
2.3. Decomposition of the Test Model . . . . .	21
2.4. Co-Simulation Schemes . . . . .	23
2.4.1. Explicit Co-Simulation Method . . . . .	24
2.4.2. Implicit Co-Simulation Method . . . . .	26
2.4.3. Waveform Relaxation Method . . . . .	30
<b>3. Macro-Step Size and Order Control Algorithm</b>	<b>32</b>
3.1. Error Estimators for Co-Simulation Approaches . . . . .	32
3.1.1. Method e1: Local Extrapolation ( <i>exLE</i> ) . . . . .	32
3.1.2. Method e2: Milne Device ( <i>exMD</i> ) . . . . .	34
3.1.3. Method e3: Local Extrapolation Based on the Coupling Variables ( <i>exCV</i> ) . . . . .	36
3.1.4. Method i1: Milne Device ( <i>imMD</i> ) . . . . .	39
3.1.5. Method i2: Local Extrapolation Based on the Coupling Variables ( <i>imCV</i> ) . . . . .	39
3.2. Local Error Test . . . . .	40
3.3. Order Control Algorithm . . . . .	41
3.4. Macro-Step Size Controller . . . . .	44
<b>4. On the Computational Efficiency of Co-Simulation Approaches</b>	<b>48</b>
4.1. Parallel Implementation of the Co-Simulation Approaches . . . . .	50
4.2. Subsystem Solver . . . . .	50
4.3. Micro-Step Size Limitation . . . . .	50
4.4. Relation Between Macro-Step Size and Computation Time . . . . .	55
4.5. Differences in Subsystem Computation Times . . . . .	58
<b>5. Numerical Studies</b>	<b>63</b>
5.1. Convergence Analysis . . . . .	64
5.2. Comparison of Different Error Estimators . . . . .	68
5.3. Safety Factor . . . . .	86
5.4. Error Tolerances . . . . .	89

5.5. Asymmetric Coupling Properties . . . . .	96
5.6. Number of Subsystems . . . . .	99
5.7. Scaling Order of the Computation Time . . . . .	103
5.8. Co-Simulation Model with Additional Forces at the Coupling Elements . . . . .	105
<b>6. Conclusions</b>	<b>114</b>
<b>A. Approximation of the Interface Jacobian</b>	<b>116</b>
A.1. Linear Systems . . . . .	116
A.2. Nonlinear Systems . . . . .	121
A.3. Numerical Examples . . . . .	125
A.4. Conclusions on the Approximation of the Interface Jacobian . . . . .	134
<b>B. Convergence Behavior of Explicit Co-Simulation Approaches with Feed-Through</b>	<b>136</b>
<b>C. Numerical Stability of Explicit Co-Simulation Methods</b>	<b>139</b>
C.1. Linear Test Model . . . . .	140
C.2. Result Overview . . . . .	140
C.3. Stability Plots . . . . .	144
<b>Bibliography</b>	<b>157</b>

## 1. Introduction

The demand for more detailed and more complex simulation models leads to an increase in required computational power and to the need for more efficient solution strategies. This work deals with the subject of the simulation of mechanical models, or more precisely, with the subject of solving multibody systems efficiently in time domain by applying co-simulation techniques.

Considering a system that is decomposed into an arbitrary number of coupled subsystems, a numerical simulation can be accomplished in different ways [VKV04]. One possibility to classify solution methods for coupled systems is to distinguish between strong coupling approaches and weak coupling approaches.

Applying a strong coupling approach, the governing equations of the overall system are formulated as a set of coupled differential-algebraic equations, which is discretized and solved with a single solver. Using a weak coupling or co-simulation approach, each subsystem is treated as a separate system with its own set of equations and its own solving method. The interaction between the subsystems is considered only at certain communication- or macro-time points by evaluating appropriate coupling conditions and by exchanging coupling variables. Between the macro-time points, the coupling variables are approximated in the subsystems. A common way is the approximation by inter-/extrapolation polynomials. The formulation of the coupling conditions between two (or several) subsystems is problem-dependent.

By applying a co-simulation approach, which may be understood as a "second level" of time discretization, additional sources of numerical instabilities and errors are introduced and have to be considered carefully. Using a co-simulation method, however, also entails several advantages:

- Each subsystem can be computed by a specialized subsystem solver. This is especially interesting for the simulation of multiphysical systems.
- The co-simulation interface does not need full access to the subsystem equations, only the input and output variables have to be exchanged. Assuming that different groups are working together to simulate a complex model, the intellectual properties of each group do not have to be shared, because each subsystem can be treated as a black box system.
- Because of the modularity of a co-simulation model, the computation can be carried out on a distributed computing system. A distributed computation may be necessary due to the memory requirements of very large-scale problems, for example. In addition, computation time may be significantly reduced by making use of a parallelized co-simulation implementation.

### 1.1. Literature Review

Co-simulation or solver coupling methods have been used advantageously for the simulation of multiphysical models. The following collection of examples shows only an extract and is focused on models containing multibody subsystems:

- mechanical systems coupled with hydraulic systems [NCDL11, SS11, Sch15]
- mechatronic systems [SBC<sup>+</sup>07, PKB08, FSU10]

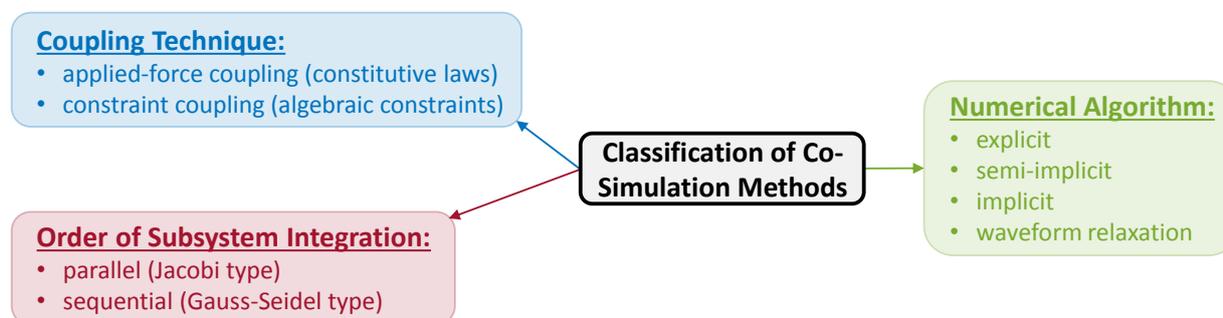
- multibody systems coupled with elastic structures [AMA<sup>+</sup>18, APP<sup>+</sup>12, APRP09, PA12, BS11a]
- multibody systems coupled with fluid dynamic systems [AMM13, Now18]
- multibody systems coupled with control tools [VKV04, GGM10]
- particle simulations coupled with elastic structures and multibody systems [SON<sup>+</sup>17, RSP<sup>+</sup>17]
- virtual prototyping of vehicle systems [DSN12, ZEK<sup>+</sup>14].

An overview on different available co-simulation methods and their applications can be found, for example, in [GTB<sup>+</sup>17] and in [GTB<sup>+</sup>18].

Another important field of application for co-simulation methods concerns the multirate integration of multibody systems [SM11b, VM11, Arn07, GnNLG11]. The idea of multirate methods is that different parts of a model are solved with different time step sizes. This allows an efficient simulation of models containing subsystems with different time scales.

### Classification of Co-Simulation Methods

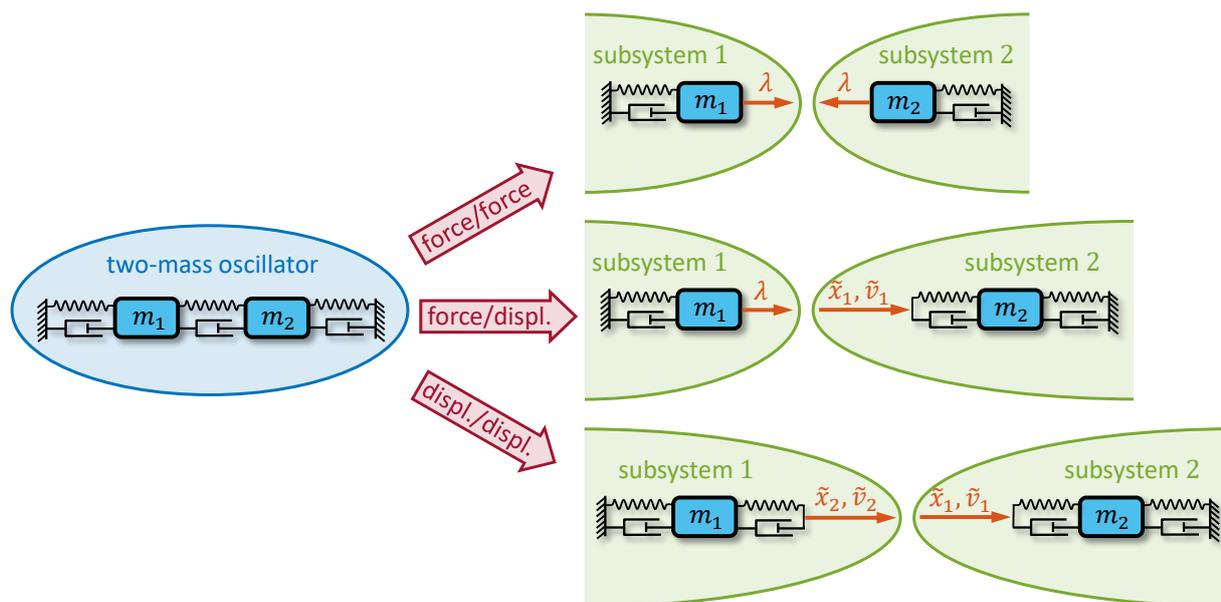
Co-simulation methods can be classified in different ways. An overview of common criteria is depicted in Fig. 1.1. One aspect is the coupling technique. Considering mechanical systems, the connection between two (or several) subsystems can be described either by algebraic constraint equations or by constitutive equations. The so called constraint coupling occurs, when the subsystems are connected by joints. Examples for a coupling by constraint equations can be found in [SM10, SL15, SL14a, GA04, KS00, SBAS17]. If the subsystems are connected by forces or torques which are represented by constitutive laws, it is referred to as applied-force coupling. Examples of co-simulation with applied force coupling can be found in [SLL15, SLLM15]. The coupling techniques can be subdivided further by the physical properties of the coupling variables into force/force-, force/displacement- and displacement/displacement-coupling [BS10a, SL14b] as shown exemplarily in Fig. 1.2 for the two-mass oscillator.



**Figure 1.1:** Common criteria for the classification of co-simulation approaches.

A second possibility to classify co-simulation approaches is the order in which the subsystems are solved. It can be distinguished between Jacobi type approaches and Gauss-Seidel type approaches [Arn07, SA12]. If all subsystems are integrated in parallel, the method is called a Jacobi type co-simulation scheme [FU09]. A sequential integration of the subsystems, for example in a

master-slave scheme, is called a Gauss-Seidel type co-simulation [Arn10]. The preferred subsystem integration order depends on the considered problem. If a large monodisciplinary model is split into a set of coupled subsystems and if the objective is to reduce the computation time by parallelization, then a Jacobi type co-simulation scheme will be more efficient than integrating the subsystems sequentially. Considering a complex multiphysical model which is simulated with different solvers, one of the subsystems may possibly dominate the computational effort so that the time spent on the integration of the remaining subsystems might be negligible. In this case the preferred approach would be a Gauss-Seidel type co-simulation because it usually shows better numerical stability properties than the parallel Jacobi scheme [Sch15].



**Figure 1.2:** Decomposition of the two-mass oscillator with an applied force coupling approach: force/force-, force/displacement- and displacement/displacement-decomposition.

A third aspect to classify co-simulation approaches is the numerical method which is used to solve the coupled problem. The classification is adopted from classical numerical time integration methods. Co-simulation procedures are separated into explicit and implicit schemes. Implicit schemes can be subdivided further into full-implicit and semi-implicit methods [SL14a, SL14b, SLLM15, Lu15]. Assuming the implicit computation is carried out in a predictor/corrector scheme, semi-implicit means that the corrector step is accomplished only once within each co-simulation step. Applying a full-implicit co-simulation approach, the corrector iteration is stopped when a user-defined convergence condition is fulfilled.

Explicit co-simulation approaches [FSU10, GnNLG11, SLL15, LYL<sup>+</sup>20] have the advantage that a repetition of macro-time steps is not required if an equidistant communication-time grid is used. This is an important issue if commercial simulation tools are coupled, which may not support (an efficient) reinitialization of the subsystem solver at certain macro-time points. In addition, explicit co-simulation methods typically produce less computational overhead. Implicit methods usually require macro-step repetitions and possibly need additional subsystem information (e.g. interface Jacobians). The advantage of implicit methods is their improved stability behavior [SLL15].

Waveform relaxation methods [WSVOR85, Gea91] represent another technique to solve coupled systems. While waveform relaxation methods are commonly used in connection with elec-

tronic circuit simulations [LRSV82, Cha89, MGS<sup>+</sup>17], these methods are not very often applied for simulating mechanical systems.

### Prediction and Approximation of the Coupling Variables

Depending on the considered co-simulation scheme, the values of the coupling variables have to be predicted at the next macro-time point for all subsystems (Jacobi type co-simulation) or only for one subsystem (Gauss-Seidel type co-simulation). In addition, the coupling variables have to be approximated within the subsystems between the macro-time points. The standard way for both tasks is an approximation by extra-/interpolation polynomials [Arn09, GnNLG11, SLL15]. The approximation polynomials are typically generated by using the values of the coupling variables at the previous macro-time points as supporting points. An alternative approach, presented in [BS19], suggests to use data points within the previous macro-time step for the generation of the approximation polynomials. A requirement for this method is that the co-simulation interface has access to the subsystem data.

A co-simulation approach with context based approximation polynomials is presented in [BKEFDF<sup>+</sup>17]. The polynomials are fitted through a defined number of previous macro-time points. The number of supporting points, the weighting factors and the polynomial degree are selected according to a heuristic scheme as a function of the coupling signal. It is stated that the suggested approach can be used advantageously in the case of discontinuous coupling signals. The generation of approximation polynomials by a linear least square fit through previous values is also supported in the MSC ADAMS CO-SIMULATION INTERFACE [Sof14].

Another approach, suggested in [LLSS17], predicts the values of the coupling variables based on an extrapolation and integration of the accelerations of the coupling bodies. Apparently, this method has a positive effect on the numerical stability of the considered explicit co-simulation approach compared to the direct extrapolation of the coupling variables. However, an update of the acceleration variables is required at the end of each macro-time step. To compute the updated values of the accelerations, the equations of motion of the subsystems have to be accessed. The approach is extended by using relaxation techniques in [LYL<sup>+</sup>20].

In [PGKT18] an interface model is used to estimate the values of the coupling variables within certain subsystems. The particular application is a multibody system with hydraulic components. The interface model consists of a reduced order model of the multibody system. Within the hydraulic subsystem, which requires smaller time steps than the multibody subsystem, the interface model is evaluated to estimate the coupling variables between the macro-time points.

A frequently observed issue when carrying out co-simulations are discontinuities in the approximation polynomials of the coupling variables at the macro-time points. Discontinuities may be introduced by the approximation technique or by any kind of update process which is carried out at the end of a macro-time step. Using a multistep method (e.g. BDF-method) as subsystem solver, the integration order and the subsystem solver step size may significantly be reduced due to the discontinuities. Large discontinuities may entail a solver reinitialization. As a consequence, the overall efficiency of the co-simulation is reduced. Approaches for a continuous approximation of the coupling variables are presented in [Bus16] and extended in [Bus19]. An alternative approach to deal with jumps in the coupling variables is presented in [AFk16]. The history data arrays of the subsystem solver (BDF-method) are modified to achieve a better prediction in the

case of discontinuous coupling signals. The modified predictor prevents the solver from reducing the integration order and the solver step size.

### **Error Estimation and Adaptive Macro-Step Size**

A challenging subject is the development and implementation of a macro-step size controller for co-simulation methods. If commercial simulation software tools are involved, the repetition of macro-steps is likely to be impossible or at least very costly, because a reinitialization of the sub-system solver at certain macro-time points is often not provided. The *Functional Mock-up Interface* (FMI 2.0) [FMI14], a standard interface definition for simulator coupling, specifies functions to save and restore internal solver states which would enable a repetition of the macro-step. If solver reinitialization is not provided, two major problems arise: Commonly used error estimation techniques are based on carrying out a time step multiple times, which requires solver reinitialization. Furthermore, if a macro-step cannot be repeated, there is no obvious way of proceeding if the estimated error of the macro-step exceeds the defined tolerances.

However, concepts of co-simulations with an adaptive macro-step size, which work without solver reinitialization, have been published. In [BS11b], an error estimator based on an explicit predictor/corrector approach is suggested. For the macro-step size selection, a PI-controller is used. A similar approach is applied in the MSC ADAMS CO-SIMULATION INTERFACE [Sof14]. The error is also estimated by the difference between the predicted and the updated coupling variables. If the estimated error exceeds the defined tolerances, a warning message is displayed but the co-simulation is continued nevertheless. Other approaches for the error estimation are based on the idea of energy conservation. In [BWZH13], an explicit co-simulation approach is suggested, where the errors of the coupling signals are partly compensated by adding a correction term based on an energy calculation in the coupling element. The idea of energy conservation is also used in [SKSP17, SP19] and in [RGN19] by introducing energy residuals as a measurement for the coupling errors within a non-iterative co-simulation implementation. The estimated errors are then used for a macro-step size controller; however, the macro-steps are not repeated.

Co-simulation approaches with a variable communication-time grid that require macro-step repetitions have also been developed, despite the fact that their compatibility with commercial software tools may not be given yet. In [ACS13], an error estimator based on *Richard Extrapolation* is suggested. It is indicated that the *Richard Extrapolation* based method is a reliable way to estimate the local error of a co-simulation, as long as there are no subsystems with direct feed-through involved. An error estimator based on the *Milne Device* approach is presented in [MKL<sup>+</sup>17] for a semi-implicit co-simulation method and in [MKLS19] for an explicit co-simulation method. In combination with the semi-implicit co-simulation method, the error is estimated by comparing variables of the predictor and the corrector step. The error estimation for the explicit co-simulation method is achieved by comparing the results obtained by subsystem integrations with different predictor polynomials.

## **1.2. Motivation**

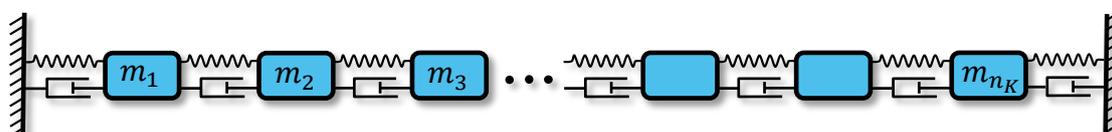
Due to the increasing availability of multi-core and cluster computers, parallel computing is an important and highly relevant topic in all fields of simulation-based engineering [DM06, VHK<sup>+</sup>16],

including the field of multibody system dynamics [CCMB00, NTM<sup>+</sup>12, NSMH14].

The aim of this work is to increase the efficiency of the simulation of large-scale mechanical systems – concretely, multibody systems are considered – by parallelizing the computation. The parallelization is achieved by decomposing an overall model into subsystems which are coupled and computed by using co-simulation techniques. The application of co-simulation methods is a highly promising approach to reduce the computation time of multibody models, mainly because of two reasons: frequently, the computation time does not scale linearly with the number of degrees of freedom so that the efficiency is strongly improved by a parallelization. Furthermore, the multirate character of the co-simulation can be used advantageously. The scaling of the computation time of a multibody simulation with respect to the number of degrees of freedom of the model depends on different aspects [GGLC10]:

- type of coordinates (e.g. absolute or relative coordinates) which are used to formulate the equations of motion,
- applied numerical integration scheme,
- sparsity of the system matrices,
- special features of the model (e.g. contact search algorithm),
- computational aspects (e.g. memory management).

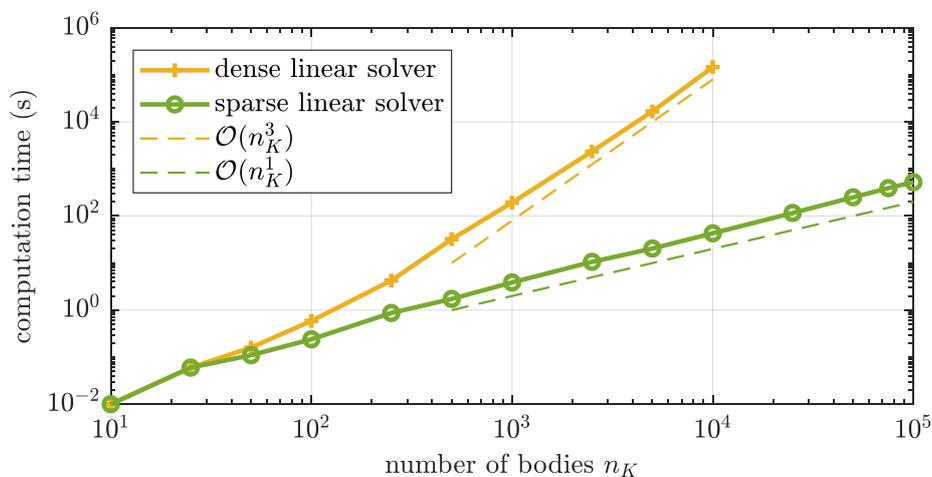
Typically the scaling order is between one and three. A linear scaling of the computation time can be achieved for example if a set of relative coordinates is used to formulate the equations of motion in combination with a recursive so called  $\mathcal{O}(n)$ -algorithm [Fea83, SV96, MS07]. This method is tailored for tree-structured multibody systems which often occur in the field of robot dynamics. A third order scaling may occur, for example, when the equations of motion are formulated in absolute coordinates and solved with an implicit integration scheme. Then, a system of nonlinear equations has to be solved in each time step. If the system is solved with a Newton method and the resulting linear system in each Newton iteration is solved with a direct method (e.g. LU-decomposition) without exploiting the sparsity of the matrices, a third order scaling of the computation time occurs [EEHJ96].



**Figure 1.3:** Chain-structured multibody system with  $n_K$  point masses.

The influence of the solver for the linear system of equations on the computation time is illustrated in Fig. 1.4. A chain-structured multibody system of point masses which are connected by nonlinear spring/damper-elements, as shown in Fig. 1.3, is considered and computed with the (BDF-method based) *IDA* solver of the *SUNDIALS* package [HBG<sup>+</sup>05]. Each mass has one translational degree of freedom in horizontal direction. The size of the model is increased successively by adding further masses to the chain. The resulting computation time of the simulation is shown in Fig. 1.4 for two different linear solvers, or more precisely for two different representations of the Jacobian matrix: a direct dense linear solver (yellow line) and a direct sparse linear solver (*KLU*

[DPN10]) (green line). Because of the chain structure of the system, the matrices are very sparse. A linear scaling of the computation time can be achieved by applying the sparse-matrix solver. If the dense linear solver is used, a cubic scaling order occurs.



**Figure 1.4:** Influence of the linear solver used within the Newton iterations on the computation time of a multibody simulation with a BDF-type solver.

Most multibody software codes use sparse matrix techniques [OCC77, AFF06] and have very efficient numerical algorithms which may execute the computation in parallel on multicore computers. Therefore, a general prediction of the scaling order of the computation time with respect to the number of degrees of freedom may be difficult. The important point is that the reduction of computation time, which can be achieved by a parallel co-simulation, increases as the scaling order increases. For an explicit co-simulation approach, the computation time can be estimated by the relation

$$T_{\text{cosim}} \approx \frac{T_{\text{monolith}}}{n_s^P} + \text{overhead}.$$

The estimation is based on the assumption that the overall (monolithic) model is split into  $n_s$  subsystems of equal size. The scaling order of the computation time with respect to the number of degrees of freedom is denoted with  $P$ .

The second advantage of an co-simulation implementation lies in the multirate integration of the subsystems [GnNLG11, SM11a, PGKT18]. Due to the weak coupling approach, each subsystem is integrated with an independent solver with an individual subsystem solver step size. The benefit of the multirate character can be illustrated by considering special large-scale multibody systems. The major part of the considered model is assumed to be non-stiff, therefore it can be integrated with a rather large solver step size. A small part of the model has very stiff properties and has therefore to be integrated with a small solver step size and a low integration order. If the overall system is solved in a monolithic fashion, the small solver step size has to be used for the whole model to obtain a stable and accurate simulation, resulting in a rather long computation time. Applying a co-simulation method, it is possible to separate the stiff part of the model within a small subsystem and the non-stiff part in another subsystem. In this case, the solver step size is reduced only within the stiff subsystem, while the remaining part of the model is integrated with a larger solver step size. The resulting computation time will be significantly reduced compared to the monolithic approach.

In addition, assuming that the considered overall system is represented by a DAE-system, it might be possible that some of the subsystems have a lower index than the overall system [CI94].

### 1.3. Outline

The theoretical background of the application of co-simulation methods to simulate mechanical systems is described in Section 2. The decomposition process of a multibody system into a set of coupled subsystems is explained in Section 2.1. In Section 2.4, three different co-simulation approaches are explained: the classical explicit method, the classical implicit method and a waveform relaxation approach.

Section 3 contains the detailed description of a macro-step size and order control algorithm for the considered co-simulation approaches. Five different local error estimators based on either *local extrapolation* or *Milne device* are presented in Section 3.1, an order selection strategy based on the scaled derivative norm of the coupling variables is explained in Section 3.3 and the macro-step size controller based on the estimated local error is described in Section 3.4.

The computational efficiency of the considered co-simulation approaches is analyzed in Section 4. Section 4.1 contains a description of the parallel hybrid MPI and OpenMP implementation including flowcharts of the explicit and the implicit co-simulation approaches. Different effects of the micro-step size (subsystem solver step size) limitation and the macro-step size selection on the overall computation time of the co-simulation model are explained in Section 4.3 and Section 4.4.

Section 5 contains a collection of results of numerical studies carried out with the chain structured multibody system introduced in Section 2.2 which is analyzed for different parametrizations. The accuracy of the co-simulation methods is examined and the convergence behavior is investigated. In addition, the influence of various co-simulation parameters on the overall computation time is examined. In Section 5.2, the different error estimators are compared in detail; the benefit of the macro-step size controller compared to a co-simulation approach with an equidistant macro-time grid is demonstrated in Section 5.8.

Finally, the results are summarized and the work is concluded in Section 6.

In Appendix A, alternative approaches for the approximation of the interface Jacobian, which is required for the implicit co-simulation approach, are studied. The influence of feed-through subsystems on the convergence behavior of different explicit co-simulation approaches is described in Appendix B. A detailed comparison of the numerical stability of different explicit co-simulation approaches is documented in Appendix C.

## 2. Co-Simulation of Mechanical Systems

The basic idea of co-simulation is to split an overall system into a set of coupled subsystems. The theoretical background of the decomposition of a mechanical system into a set of coupled subsystems by using an applied force coupling approach is described in Section 2.1, following the approach of [MKS21] and [SL14a]. In Section 2.2, a chain-structured multibody system is introduced. The decomposition of this reference multibody system is described in Section 2.3. The explicit and the implicit co-simulation methods used in this work are explained in Section 2.4 on the basis of this multibody model. The same type of model will be used later on for the numerical studies in Section 5.

### 2.1. Theoretical Background: Decomposition of a Mechanical System into Subsystems Using an Applied Force Coupling Technique

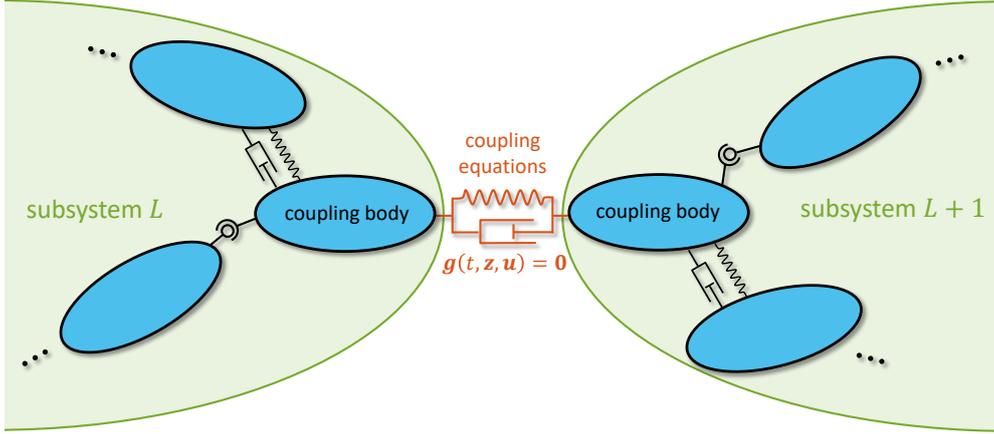
Applying a co-simulation approach, the overall model has firstly to be decomposed into a certain, user-defined number of subsystems. To describe and define the coupling between the subsystems, coupling equations and appropriate coupling variables have to be introduced. Here, only the case that the subsystems are coupled by applied forces (applied-force coupling) is considered, i.e. the coupling between the subsystems is described by constitutive laws. Alternatively, algebraic constraint equations and corresponding reaction forces/torques (Lagrange multipliers) can be used to connect the subsystems (constraint coupling), which is not considered here. It should be noted that in the current work, the term *coupling variables* refers to the subsystem input variables. In contrast to this definition, other authors use the notion *coupling variables* to specify the subsystem input and output variables. Since only the input variables have to be approximated within a co-simulation approach, the former definition is preferred here.

Within a co-simulation approach, a communication time grid (macro-time grid) is introduced by defining macro-time points  $T_N$ . The subsystems are integrated independently between the macro-time points; the coupling variables are only exchanged between the subsystems at the macro-time points. Between the macro-time points, the coupling variables have to be approximated in order to carry out the subsystem integrations. Different approximation techniques have been suggested and analyzed in literature. Here, polynomial inter-/extrapolation is used for approximating the coupling variables. To describe the decomposition into subsystems, a general mechanical system is considered, which is described by the subsequent first-order DAE system (differential-algebraic system of equations)

$$\mathbf{B}(t, \mathbf{z})\dot{\mathbf{z}} = \mathbf{f}(t, \mathbf{z}) . \quad (2.1)$$

In the above equation, the vector  $\mathbf{z}$  collects the position variables  $\mathbf{q}$ , the velocity variables  $\mathbf{v}$  and the Lagrange multipliers  $\boldsymbol{\mu}$ .  $\mathbf{B}$  represents a symmetric square matrix (in the ODE case,  $\mathbf{B}$  is invertible). The overall system is partitioned into  $n_s$  subsystems, see Fig. 2.1. The subsystem index is indicated by a superscript. The states and multipliers of an arbitrary subsystem  $L \in \{1, \dots, n_s\}$  are collected in the vector  ${}^L\mathbf{z} = [{}^L\mathbf{q}^\top, {}^L\mathbf{v}^\top, {}^L\boldsymbol{\mu}^\top]^\top$ . The equations of motion of subsystem  $L$  can be written as

$${}^L\mathbf{B}(t, {}^L\mathbf{z}) {}^L\dot{\mathbf{z}} = {}^L\mathbf{f}(t, {}^L\mathbf{z}) . \quad (2.2)$$



**Figure 2.1:** Decomposition into subsystems using an applied-force coupling technique.

For the definition of a co-simulation model, the coupling of the subsystems has to be specified. Therefore, coupling variables  $\mathbf{u}$  are introduced. The definition of the coupling variables, which are implicitly defined by the coupling conditions  $\mathbf{g} := \mathbf{u} - \varphi(t, \mathbf{z}, \mathbf{u}) = \mathbf{0}$ , depends on the physical connection of the subsystems. The subsystem coupling is, on the one hand, defined by the constitutive laws of the applied forces, which describe the connection between the coupling bodies. On the other hand, the decomposition technique has to be specified. Topologically, three different decomposition techniques can be distinguished, namely force/force-decomposition, force/displacement-decomposition, and displacement/displacement-decomposition [SL14b, SLL16, WMH03]. It should be mentioned that in case of an applied force coupling approach, which is considered in this work, the coupling variables can be expressed as functions of  $\mathbf{q}$  and  $\mathbf{v}$  only, i.e. the coupling functions read  $\mathbf{g} := \mathbf{u} - \varphi(t, \mathbf{q}, \mathbf{v}, \mathbf{u}) = \mathbf{0}$ . With the help of the coupling variables, the equations of motion of subsystem  $L$  can be written as

$${}^L \mathbf{B}(t, {}^L \mathbf{z}) {}^L \dot{\mathbf{z}} = {}^L \mathbf{f}^{co}(t, {}^L \mathbf{z}, \mathbf{u}) . \quad (2.3)$$

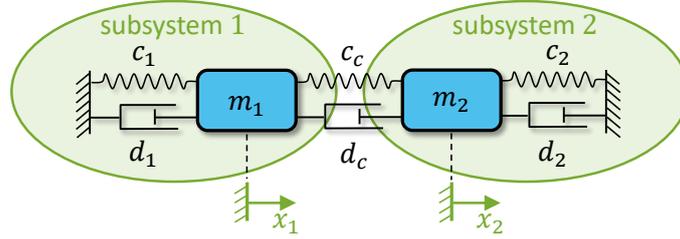
Arranging the right hand sides of all subsystem equations into the vector  $\mathbf{f}^{co}(t, \mathbf{z}, \mathbf{u}) = \left[ {}^1 \mathbf{f}^{co}(t, {}^1 \mathbf{z}, \mathbf{u})^\top, \dots, {}^{n_s} \mathbf{f}^{co}(t, {}^{n_s} \mathbf{z}, \mathbf{u})^\top \right]^\top$ , the decomposed co-simulation system can be written as

$$\begin{aligned} \mathbf{B}(t, \mathbf{z}) \dot{\mathbf{z}} &= \mathbf{f}^{co}(t, \mathbf{z}, \mathbf{u}) \\ \mathbf{g}(t, \mathbf{z}, \mathbf{u}) &= \mathbf{0} \end{aligned} \quad (2.4)$$

with  $\mathbf{z} = \left[ {}^1 \mathbf{z}^\top, \dots, {}^{n_s} \mathbf{z}^\top \right]^\top$ . Applying a co-simulation approach, i.e. a weak coupling approach, the coupling conditions are only considered and enforced at the macro time points  $T_N$ , i.e. only  $\mathbf{g}(T_N, \mathbf{z}_N, \mathbf{u}_N) = \mathbf{0}$  is fulfilled. Between the macro-time points, the coupling variables are approximated, e.g. by means of polynomials. Therefore, information has to be exchanged between the subsystems only at the macro-time points. As a consequence, the subsystems can be integrated independently of each other between the macro-time points.

From the topological point of view, basically three different decomposition techniques can be distinguished: force/force-, force/displacement- and displacement/displacement-decomposition. These three fundamental approaches are shortly described with the help of a two-mass oscillator

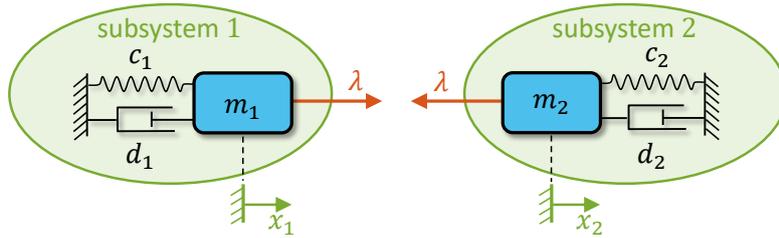
(masses  $m_1, m_2$ , spring constants  $c_1, c_2$ , damping coefficients  $d_1, d_2$ , coupling-spring  $c_c$ , coupling-damper  $d_c$ ), see Fig. 2.2. Generalization to arbitrary mechanical systems is described in [SL14b], for instance.



**Figure 2.2:** Linear two-mass oscillator: interpretation as two coupled single-mass oscillators.

### 2.1.1. Force/Force-Decomposition

Applying a force/force-coupling approach, the overall system is decomposed into two subsystems so that both subsystems are force-driven single-mass oscillators, see Fig. 2.3. The single-mass oscillators are excited by the coupling force (coupling variable)  $\lambda$ , which can be expressed as a function of the subsystem states. The physical force law for  $\lambda$  is implicitly defined by the coupling condition  $g$ .



**Figure 2.3:** Linear two-mass oscillator: force/force-decomposition approach.

The decomposed system is described by the following semi-explicit index-1 DAE system

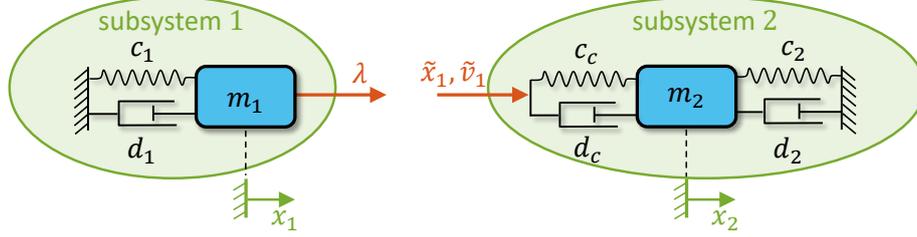
$$\begin{aligned} \begin{bmatrix} \dot{x}_1 \\ \dot{v}_1 \end{bmatrix} &= \begin{bmatrix} v_1 \\ -\frac{c_1}{m_1}x_1 - \frac{d_1}{m_1}v_1 + \frac{\lambda}{m_1} \end{bmatrix} && \text{(subsystem 1)} \\ \begin{bmatrix} \dot{x}_2 \\ \dot{v}_2 \end{bmatrix} &= \begin{bmatrix} v_2 \\ -\frac{c_2}{m_2}x_2 - \frac{d_2}{m_2}v_2 - \frac{\lambda}{m_2} \end{bmatrix} && \text{(subsystem 2)} \\ g &:= \lambda - c_c(x_2 - x_1) - d_c(v_2 - v_1) = 0 && \text{(coupling condition)}. \end{aligned} \quad (2.5)$$

Obviously, the coupling vector is given by the one-dimensional vector  $\mathbf{u} = [\lambda]$  and the one-dimensional coupling function by  $\varphi = [c_c(x_2 - x_1) + d_c(v_2 - v_1)]$ . The output vector of subsystem 1 is  $\mathbf{y}_1 = [x_1, v_1]^T$  and the output vector of subsystem 2 is given by  $\mathbf{y}_2 = [x_2, v_2]^T$ .

### 2.1.2. Force/Displacement-Decomposition

Applying a force/displacement-coupling approach, the overall system is decomposed into two subsystems so that subsystem 1 is a force-driven and subsystem 2 a base-point excited single-mass oscillator, see Fig. 2.4. In contrast to the force/force-coupling approach, the coupling force  $\lambda$  is re-

placed in subsystem 2 by means of the coupling condition (i.e. by means of the physical force law). Since the state variables  $x_1$  and  $v_1$  are unknown in subsystem 2, they are replaced in the equations of motion by the additional coupling variables  $\tilde{x}_1$  and  $\tilde{v}_1$ . Due to these additional coupling variables, two additional coupling conditions have to be added.



**Figure 2.4:** Linear two-mass oscillator: force/displacement-decomposition approach.

The decomposed system is described by the following semi-explicit index-1 DAE system

$$\begin{aligned}
 \begin{bmatrix} \dot{x}_1 \\ \dot{v}_1 \end{bmatrix} &= \begin{bmatrix} v_1 \\ -\frac{c_1}{m_1}x_1 - \frac{d_1}{m_1}v_1 + \frac{\lambda}{m_1} \end{bmatrix} && \text{(subsystem 1)} \\
 \begin{bmatrix} \dot{x}_2 \\ \dot{v}_2 \end{bmatrix} &= \begin{bmatrix} v_2 \\ -\frac{c_2}{m_2}x_2 - \frac{d_2}{m_2}v_2 - \frac{c_c}{m_2}(x_2 - \tilde{x}_1) - \frac{d_c}{m_2}(v_2 - \tilde{v}_1) \end{bmatrix} && \text{(subsystem 2)} \\
 \mathbf{g} &:= \begin{bmatrix} \lambda - c_c(x_2 - x_1) - d_c(v_2 - v_1) \\ \tilde{x}_1 - x_1 \\ \tilde{v}_1 - v_1 \end{bmatrix} = \mathbf{0} && \text{(coupling conditions)}.
 \end{aligned} \tag{2.6}$$

*Remark on an alternative implementation with feed-through:* In the above approach according to Eq. (2.6), the coupling vector is given by the 3-dimensional vector  $\mathbf{u} = [\lambda, \tilde{x}_1, \tilde{v}_1]^\top$  and the 3-dimensional coupling function by  $\varphi = [c_c(x_2 - x_1) + d_c(v_2 - v_1), x_1, v_1]^\top$ . Furthermore, the output vector of subsystem 1 is  $\mathbf{y}_1 = [x_1, v_1]^\top$  and the corresponding output vector of subsystem 2 reads  $\mathbf{y}_2 = [x_2, v_2]^\top$ . Since the outputs do not depend explicitly on their inputs (coupling variables), there is no feed-through. However, the above formulation requires a co-simulation master or co-simulation administrator, which calculates the coupling force  $\lambda$  with the help of the subsystem outputs, since  $\lambda$  is required as input for subsystem 1. In practical applications, a closed formula for  $\lambda$  as a function of the subsystem outputs – e.g.  $\lambda = c_c(x_2 - x_1) + d_c(v_2 - v_1)$  in the above example – might not be available and  $\lambda$  may only be accessible as a simulation result of subsystem 2. Then, the above approach has to be modified resulting in the following decomposed system

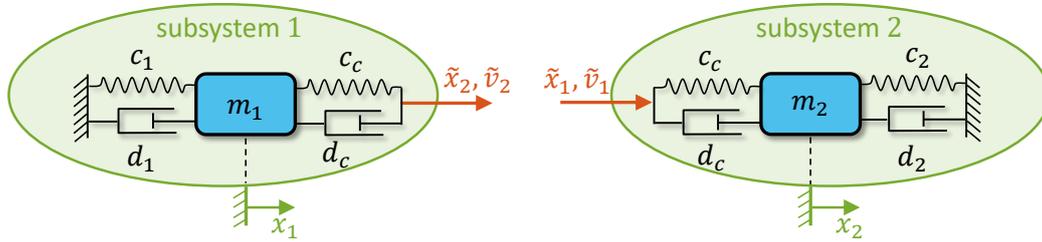
$$\begin{aligned}
 \begin{bmatrix} \dot{x}_1 \\ \dot{v}_1 \end{bmatrix} &= \begin{bmatrix} v_1 \\ -\frac{c_1}{m_1}x_1 - \frac{d_1}{m_1}v_1 + \frac{\lambda}{m_1} \end{bmatrix} && \text{(subsystem 1)} \\
 \begin{bmatrix} \dot{x}_2 \\ \dot{v}_2 \end{bmatrix} &= \begin{bmatrix} v_2 \\ -\frac{c_2}{m_2}x_2 - \frac{d_2}{m_2}v_2 - \frac{c_c}{m_2}(x_2 - \tilde{x}_1) - \frac{d_c}{m_2}(v_2 - \tilde{v}_1) \end{bmatrix} && \text{(subsystem 2)} \\
 \mathbf{g} &:= \begin{bmatrix} \lambda - c_c(x_2 - \tilde{x}_1) - d_c(v_2 - \tilde{v}_1) \\ \tilde{x}_1 - x_1 \\ \tilde{v}_1 - v_1 \end{bmatrix} = \mathbf{0} && \text{(coupling conditions)}.
 \end{aligned} \tag{2.7}$$

In the modified implementation according to Eq. (2.7), the input vector  $\mathbf{u} = [\lambda, \tilde{x}_1, \tilde{v}_1]^\top$  and the

output vector  $\mathbf{y}_1 = [x_1, v_1]^T$  of subsystem 1 remain unchanged. However, the output vector of subsystem 2 is now given by the 1-dimensional vector  $\mathbf{y}_2 = [c_c(x_2 - \tilde{x}_1) + d_c(v_2 - \tilde{v}_1)]$ . Obviously, the output vector  $\mathbf{y}_2$  depends explicitly on the inputs  $\tilde{x}_1, \tilde{v}_1$  so that a feed-through is generated. Note that a feed-through may reduce the convergence order of the co-simulation [Bus12], as described in Appendix B.

### 2.1.3. Displacement/Displacement-Decomposition

Applying the displacement/displacement-coupling approach, the overall system is decomposed into two subsystems so that both subsystems are base-point excited single-mass oscillators, see Fig. 2.5. Therefore, the coupling spring/damper-system is duplicated. The coupling variable  $\lambda$  is replaced in both subsystems by means of the coupling condition. As a consequence, additional coupling variables have to be defined and additional coupling conditions have to be formulated.



**Figure 2.5:** Linear two-mass oscillator: displacement/displacement-decomposition approach.

The decomposed system is mathematically defined by the following semi-explicit index-1 DAE system

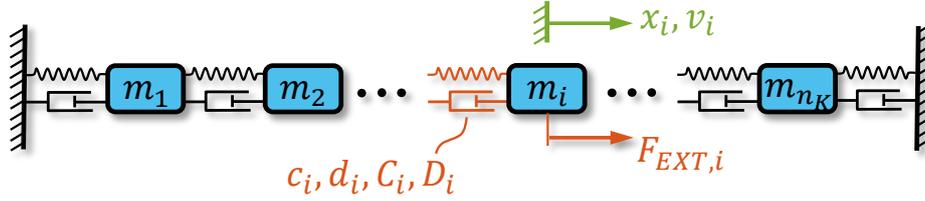
$$\begin{aligned}
 \begin{bmatrix} \dot{x}_1 \\ \dot{v}_1 \end{bmatrix} &= \begin{bmatrix} v_1 \\ -\frac{c_1}{m_1}x_1 - \frac{d_1}{m_1}v_1 + \frac{c_c}{m_1}(\tilde{x}_2 - x_1) + \frac{d_c}{m_1}(\tilde{v}_2 - v_1) \end{bmatrix} && \text{(subsystem 1)} \\
 \begin{bmatrix} \dot{x}_2 \\ \dot{v}_2 \end{bmatrix} &= \begin{bmatrix} v_2 \\ -\frac{c_2}{m_2}x_2 - \frac{d_2}{m_2}v_2 - \frac{c_c}{m_2}(x_2 - \tilde{x}_1) - \frac{d_c}{m_2}(v_2 - \tilde{v}_1) \end{bmatrix} && \text{(subsystem 2)} \\
 \mathbf{g} := \begin{bmatrix} \tilde{x}_2 - x_2 \\ \tilde{v}_2 - v_2 \\ \tilde{x}_1 - x_1 \\ \tilde{v}_1 - v_1 \end{bmatrix} &= \mathbf{0} && \text{(coupling conditions).}
 \end{aligned} \tag{2.8}$$

Here, the 4-dimensional coupling vector is given by the vector  $\mathbf{u} = [\tilde{x}_2, \tilde{v}_2, \tilde{x}_1, \tilde{v}_1]^T$  and the 4-dimensional coupling function by  $\varphi = [x_2, v_2, x_1, v_1]^T$ .

## 2.2. Co-Simulation Test Model: Oscillator Chain

The multibody model, which will be used to explain the different co-simulation approaches, is shown in Fig. 2.6. The same model will be used to examine and to compare the co-simulation implementations later on. The model consists of a chain of  $n_K$  point masses, which are connected by nonlinear spring/damper-elements. Each mass has one translational degree of freedom in horizontal direction. The main advantage of this chain-structured multibody system is that it can be scaled with respect to the degree of freedom very easily by adding further masses  $m_i$  to the

chain. It can also be split into any desired number of subsystems  $n_s$  by cutting the model through certain spring/damper-elements.



**Figure 2.6:** Co-simulation test model: chain-structured multibody system.

The constitutive equation of the spring/damper-elements is given by

$$F_{SD,i} = c_i (x_i - x_{i-1}) + d_i (v_i - v_{i-1}) + C_i \operatorname{sgn}(x_i - x_{i-1}) |x_i - x_{i-1}|^{e_x} + D_i \operatorname{sgn}(v_i - v_{i-1}) |v_i - v_{i-1}|^{e_v}. \quad (2.9)$$

The linear and nonlinear stiffness and damping coefficients  $c_i, C_i, d_i, D_i$  with index  $i = 1, \dots, n_K + 1$  can be chosen individually for each element. For the sake of clear representation the exponents are set to  $e_x = e_v = 3$  within the following chapter so that Eq. (2.9) is simplified to

$$F_{SD,i} = c_i (x_i - x_{i-1}) + d_i (v_i - v_{i-1}) + C_i (x_i - x_{i-1})^3 + D_i (v_i - v_{i-1})^3. \quad (2.10)$$

In addition, various external forces can be applied to each mass  $m_i$ :

- harmonic excitation

$$F_{HAR,i} = \Delta F_{H,i} \sin(\Omega_{H,i} t + \varphi_{H,i}) \quad (2.11)$$

- contact (penalty approach)

$$F_{CON,i} = A_C \exp(B_C x_i) \quad (2.12)$$

- impulse shaped force

$$F_{IMP,i} = \Delta F_{I,i} \frac{1}{2} \left[ \tanh\left(\frac{t - t_{I,i}}{\delta_I}\right) - \tanh\left(\frac{t - (t_{I,i} + \Delta t_{I,i})}{\delta_I}\right) \right] \quad (2.13)$$

- modified sinus force

$$F_{SIN,i} = \Delta F_{S,i} (\sin(\Omega_{S,i} t + \varphi_{S,i}))^{A_S}. \quad (2.14)$$

The parameters of the different external forces are listed in Table 2.1. Figure 2.7 shows schematically an arbitrary impulse shaped force  $F_{IMP}$  and an arbitrary modified sinus force  $F_{SIN}$ . Contact forces are modeled here by a penalty approach. Equation (2.12) describes, for instance, a contact of mass  $m_i$  with a rigid wall at  $x_i = 0$  with the penalty parameters  $|A_C| \ll 1 \text{ N}$  and  $B_C \gg 1 \text{ m}^{-1}$ .

The resulting equations of motion of the test model, written as a first order ODE-system with the displacements  $x_i$  and the velocities  $v_i$ , are given by

$$\begin{bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_i \\ \vdots \\ \dot{x}_{n_K} \\ \dot{v}_1 \\ \vdots \\ \dot{v}_i \\ \vdots \\ \dot{v}_{n_K} \end{bmatrix} = \begin{bmatrix} v_1 \\ \vdots \\ v_i \\ \vdots \\ v_{n_K} \\ \frac{1}{m_1} (-F_{SD,1} + F_{SD,2} + F_{EXT,1}) \\ \vdots \\ \frac{1}{m_i} (-F_{SD,i} + F_{SD,i+1} + F_{EXT,i}) \\ \vdots \\ \frac{1}{m_{n_K}} (-F_{SD,n_K} + F_{SD,n_K+1} + F_{EXT,n_K}) \end{bmatrix}. \quad (2.15)$$

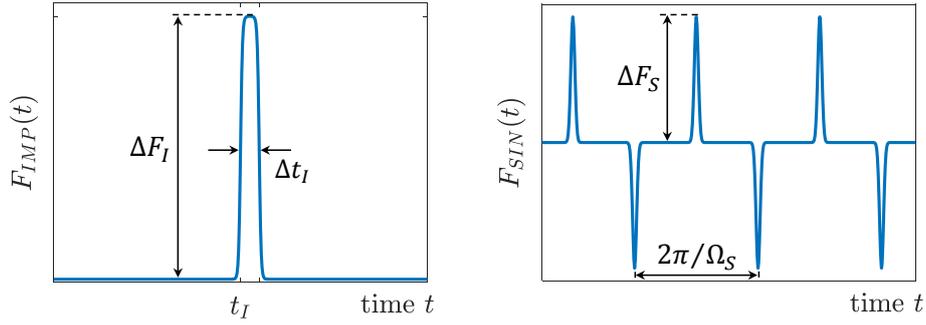
$F_{EXT,i}$  collects all external forces (besides the spring/damper-forces) acting on the mass  $m_i$ . The chain is assumed to be fixed at both ends, i. e.  $x_0 = v_0 = x_{n_K+1} = v_{n_K+1} = 0$ .

**Table 2.1:** Physical test model parameters.

Symbol	Unit	Parameter
$t_{sim}$	s	simulation time
$n_K$		number of bodies (= degree of freedom)
$m_i$	kg	mass
$c_i$	N/m	linear stiffness coefficient
$d_i$	Ns/m	linear damping coefficient
$C_i$	N/m <sup>3</sup>	nonlinear stiffness coefficient
$D_i$	Ns <sup>3</sup> /m <sup>3</sup>	nonlinear damping coefficient
$\Omega_{H,i}$	1/s	angular frequency of harmonic excitation
$\varphi_{H,i}$		phase shift of harmonic excitation
$\Delta F_{H,i}$	N	amplitude of harmonic excitation
$A_C$	N	contact parameter 1 ( $ A_C  \ll 1$ )
$B_C$	1/m	contact parameter 2 ( $B_C \gg 1$ )
$\Delta t_{I,i}$	s	impulse duration
$t_{I,i}$	s	start time of impulse shaped force
$\Delta F_{I,i}$	N	force amplitude
$\delta_I$	s	steepness parameter ( $\delta_I \ll 1$ )
$\Omega_{S,i}$	1/s	angular frequency of modified sinus force
$\varphi_{S,i}$		phase shift of modified sinus force
$\Delta F_{S,i}$	N	amplitude of modified sinus force
$A_S$		modification exponent

Although the multibody model is not meant to be a direct representation of any physical structure, similar models are used for example to describe longitudinal wave propagation in one-dimensional structures [Jen03]. Other possible applications are the simulation of oilwell drillstrings [KS12, TWY<sup>+</sup>16, AHSS03, Jan91, TZL19, YC00] or the modeling of quasi-one-dimensional lat-

tices [DML69].



**Figure 2.7:** External forces: impulse shaped force (left) and modified sinus force (right).

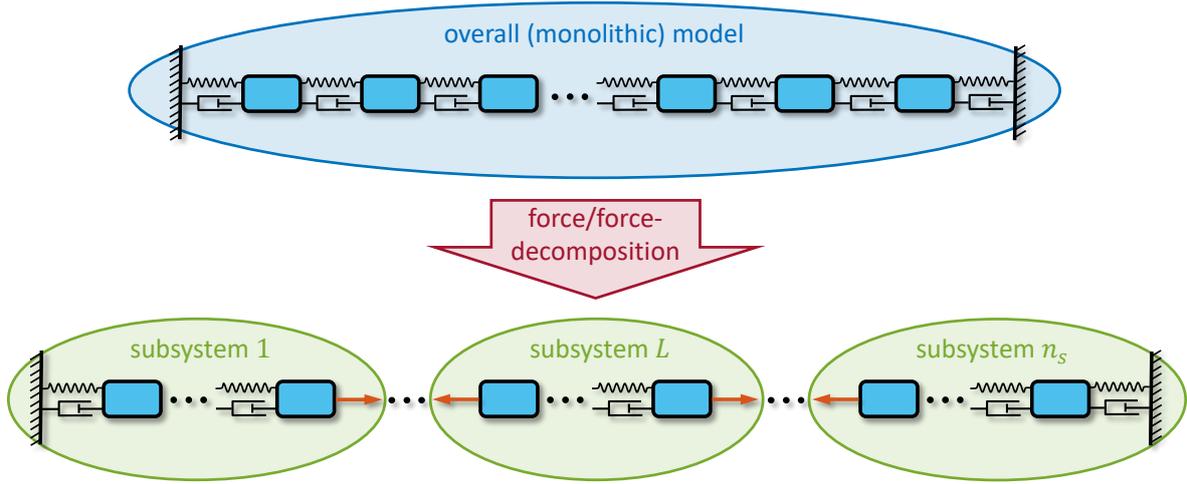
### 2.3. Decomposition of the Test Model

The overall system is split into a set of  $n_s$  coupled subsystems by a force/force-decomposition approach, following the description of Section 2.1.1. This is achieved by cutting the chain through certain spring/damper-elements, as shown in Fig. 2.8, and by using the corresponding nonlinear spring/damper-forces as coupling variables  $\mathbf{u} = [\lambda_1, \dots, \lambda_{n_c}]^T$ . The number of coupling variables  $n_c = n_s - 1$  is equal to the number of coupling elements. The number of subsystems  $n_s$  is arbitrary, but typically it is chosen significantly smaller than the number of degrees of freedom  $n_K$  of the overall system.

The equations of motion

$$\begin{bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_i \\ \vdots \\ \dot{x}_{n_{K,L}} \\ \dot{v}_1 \\ \dot{v}_2 \\ \vdots \\ \dot{v}_i \\ \vdots \\ \dot{v}_{n_{K,L}-1} \\ \dot{v}_{n_{K,L}} \end{bmatrix} = \begin{bmatrix} v_1 \\ \vdots \\ v_i \\ \vdots \\ v_{n_{K,L}} \\ \frac{1}{m_1} (-\lambda_{L-1} + F_{SD,2} + F_{EXT,1}) \\ \frac{1}{m_2} (-F_{SD,2} + F_{SD,3} + F_{EXT,2}) \\ \vdots \\ \frac{1}{m_i} (-F_{SD,i} + F_{SD,i+1} + F_{EXT,i}) \\ \vdots \\ \frac{1}{m_{n_{K,L}-1}} (-F_{SD,n_{K,L}-1} + F_{SD,n_{K,L}} + F_{EXT,n_{K,L}-1}) \\ \frac{1}{m_{n_{K,L}}} (-F_{SD,n_{K,L}} + \lambda_L + F_{EXT,n_{K,L}}) \end{bmatrix} \quad (2.16)$$

of an arbitrary subsystem  $L$  with  $n_{K,L}$  bodies (Fig. 2.9) are of the same structure as Eq. (2.15). The difference is that the spring/damper-forces  ${}^L F_{SD,1}$  and  ${}^L F_{SD,n_{K,L}+1}$  acting on the bodies at the left and right subsystem boundary (coupling bodies) are substituted by the coupling forces  $\lambda_{L-1}$  and  $\lambda_L$ . Note that the subsystem index  $L$  is omitted in Eq. (2.16) for reasons of clear representation.

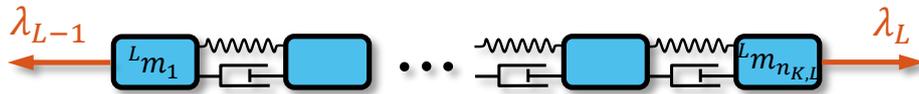


**Figure 2.8:** Force/force-decomposition of the chain-structured multibody system.

The equations of motion of all subsystems can be written as

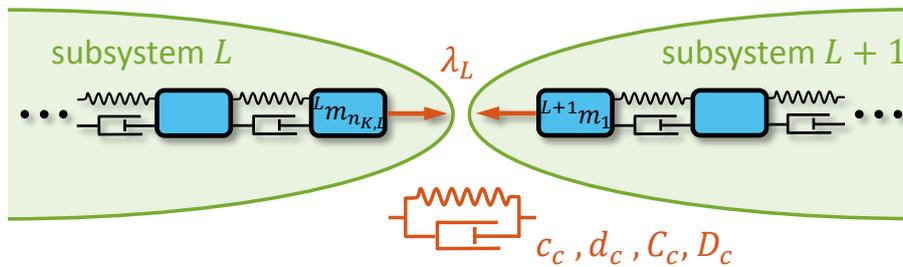
$$\begin{bmatrix} {}^1\dot{z} \\ \vdots \\ {}^L\dot{z} \\ \vdots \\ {}^{n_s}\dot{z} \end{bmatrix} = \begin{bmatrix} {}^1f^{co}(t, {}^1z, \mathbf{u}) \\ \vdots \\ {}^L f^{co}(t, {}^Lz, \mathbf{u}) \\ \vdots \\ {}^{n_s} f^{co}(t, {}^{n_s}z, \mathbf{u}) \end{bmatrix} \quad (2.17)$$

with the subsystem states  ${}^Lz = [{}^Lx_1, \dots, {}^Lx_{n_{K,L}}, {}^Lv_1, \dots, {}^Lv_{n_{K,L}}]^T$  (subsystem index  $L = 1, \dots, n_s$ ) and the coupling variables  $\mathbf{u} = [\lambda_1, \dots, \lambda_{n_c}]^T$ .



**Figure 2.9:** Arbitrary subsystem  $L$  with coupling forces  $\lambda_{L-1}$  and  $\lambda_L$ .

The subsystems are solved with the *IDA* solver from the SUNDIALS (Suite of Nonlinear and Differential/Algebraic Equation Solvers) package [HBG<sup>+</sup>05]. A detailed description of the subsystem solver is given in Section 4.2.



**Figure 2.10:** Connection of subsystem  $L$  and subsystem  $L + 1$  by the coupling force  $\lambda_L$ .

The coupling condition  $g_L$  between two adjacent subsystems  $L$  and  $L + 1$  (assuming that the last

body  $n_{K,L}$  of subsystem  $L$  is coupled with body 1 of subsystem  $L + 1$ ) is defined by

$$g_L := \lambda_L - {}^L F_{SD,c} = 0 \quad (2.18)$$

with the coupling force  $\lambda_L$  and the constitutive law of the coupling element, which reads

$$\begin{aligned} {}^L F_{SD,c} = & {}^L c_c ({}^{L+1}x_1 - {}^L x_{n_{K,L}}) + {}^L d_c ({}^{L+1}v_1 - {}^L v_{n_{K,L}}) \\ & + {}^L C_c ({}^{L+1}x_1 - {}^L x_{n_{K,L}})^3 + {}^L D_c ({}^{L+1}v_1 - {}^L v_{n_{K,L}})^3 . \end{aligned} \quad (2.19)$$

It should be mentioned that the index  $c$  in Eq. (2.19) refers to the spring/damper-element, which connects the two neighboring subsystems. All coupling conditions are collected in the vector

$$\mathbf{g}(\mathbf{u}, \boldsymbol{\varphi}(\mathbf{y})) := \mathbf{u} - \boldsymbol{\varphi}(\mathbf{y}) = \begin{bmatrix} \lambda_1 - {}^1 F_{SD,c} \\ \vdots \\ \lambda_L - {}^L F_{SD,c} \\ \vdots \\ \lambda_{n_c} - {}^{n_c} F_{SD,c} \end{bmatrix} = \mathbf{0} . \quad (2.20)$$

For the force/force-decomposition approach used here, the coupling function  $\boldsymbol{\varphi} = \boldsymbol{\varphi}(\mathbf{y})$  is a function of the subsystem output vector  $\mathbf{y} = [\mathbf{y}_1^T, \dots, \mathbf{y}_L^T, \dots, \mathbf{y}_{n_s}^T]^T$  with  $\mathbf{y}_L = [{}^L x_1, {}^L x_{n_{K,L}}, {}^L v_1, {}^L v_{n_{K,L}}]^T$  and does not depend on the coupling variables  $\mathbf{u}$  explicitly (no feed-through). The subsystem output variables  $\mathbf{y} \subset \mathbf{z}$  are a subset of the state variables, namely the states of the coupling bodies.

The coupling variables are exchanged between the subsystems only at certain communication-time or macro-time points  $[T_0, T_1, \dots, T_N, T_{N+1}, \dots]$ . In this work, the macro-step size  $H$  is assumed to be variable, but identical for all subsystems. The subsystem solver step size (micro-step size  $h_{mic}$ ) is also variable, but individual for each subsystem.

## 2.4. Co-Simulation Schemes

Applying a co-simulation approach, i.e. a weak coupling approach, the coupling conditions are only considered and enforced at the macro-time points  $T_N$ , i.e.  $\mathbf{g}(T_N, \mathbf{z}_N, \mathbf{u}_N) = \mathbf{0}$ . Between the macro-time points the coupling variables are approximated. Here, the approximation is accomplished with piecewise polynomials; the supporting points are defined at the macro-time points. Therefore, information has to be exchanged between the subsystems only at the macro-time points. As a consequence, the subsystems are integrated independently between the macro-time points.

For the macro-step  $T_N \rightarrow T_{N+1}$ , the approximation polynomials of degree  $\kappa$  (local approximation order  $\kappa + 1$ , i.e.  $\mathcal{O}(H^{\kappa+1})$ ) for the coupling variables  $\mathbf{u}$  are denoted by  $\mathbf{p}_{N+1}(t)$ . Within a co-simulation approach, system (2.1) is therefore replaced by the weakly coupled co-simulation system

$$\mathbf{B}(t, \mathbf{z})\dot{\mathbf{z}} = \mathbf{f}^{co}(t, \mathbf{z}, \mathbf{p}_{N+1}(t)) . \quad (2.21)$$

Integrating Eq. (2.21) from  $T_N$  to  $T_{N+1}$  with the initial conditions  $\mathbf{z}_N$  yields the new variables  $\mathbf{z}_{N+1}$ . Using an explicit co-simulation approach, the extrapolated coupling variables  $\mathbf{p}_{N+1}(T_{N+1})$  do in general not satisfy the coupling conditions at the macro-time point  $T_{N+1}$ , i.e.  $\mathbf{p}_{N+1}(T_{N+1}) -$

$\varphi(T_{N+1}, \mathbf{z}_{N+1}, \mathbf{p}_{N+1}(T_{N+1})) \neq \mathbf{0}$ . Consistent coupling variables  $\mathbf{u}_{N+1}$  have to be computed by an update step, namely by solving the coupling equations

$$\mathbf{u}_{N+1} - \varphi(T_{N+1}, \mathbf{z}_{N+1}, \mathbf{u}_{N+1}) = 0 \quad (2.22)$$

for  $\mathbf{u}_{N+1}$ . This update will cause a (small) jump in the coupling variables, since  $\mathbf{p}_{N+1}(T_{N+1})$  is usually not equal to the updated variables  $\mathbf{u}_{N+1}$ .

Three different co-simulation schemes will be described in the following subsections, namely the classical explicit, the classical implicit and a waveform relaxation method. The application of the three approaches will be explained based on the multibody model introduced in Section 2.2. A detailed description of the explicit and the implicit co-simulation method based on a linear two-mass oscillator can be found in [SLL15].

### 2.4.1. Explicit Co-Simulation Method

To explain the classical explicit co-simulation method, an arbitrary macro-time step from  $T_N$  to  $T_{N+1} = T_N + H$  with the current macro-step size  $H$  is considered. The states  $\mathbf{z}_N = [x_1(T_N), \dots, x_{n_K}(T_N), v_1(T_N), \dots, v_{n_K}(T_N)]^T$  of all bodies and the coupling variables  $\mathbf{u}_N = [\lambda_1(T_N), \dots, \lambda_{n_c}(T_N)]^T$  at the macro-time point  $T_N$  are assumed to be known;  $n_c$  is the number of (scalar) coupling conditions. The values of the coupling variables within the interval  $[T_N, T_{N+1}]$  are unknown and have therefore to be approximated.

The first step is the prediction of the coupling variables  $\mathbf{u}_{N+1}^{pre} = [\lambda_{1,N+1}^{pre}, \dots, \lambda_{n_c,N+1}^{pre}]^T$  at  $T_{N+1}$ . In this work, polynomial extrapolation according to

$$\mathbf{u}_{N+1}^{pre} = \mathbf{P}_\kappa(T_{N+1}; [T_{N-\kappa}, \mathbf{u}_{N-\kappa}], \dots, [T_N, \mathbf{u}_N]) \quad (2.23)$$

is used to obtain the predicted values of the coupling variables.  $\mathbf{P}_\kappa(t; [T_{N-\kappa}, \mathbf{u}_{N-\kappa}], \dots, [T_N, \mathbf{u}_N])$  is a vector of interpolation polynomials of degree  $\kappa$  through the given  $\kappa + 1$  sampling points. As sampling points, the previous  $\kappa + 1$  macro-time points and the corresponding values of the coupling variables are used. In an alternative approach, a polynomial fit through a defined number  $n_{sp} > \kappa + 1$  of previous macro-time points are used to predict the coupling variables. The determination of  $\mathbf{u}_{N+1}^{pre}$  by a polynomial fit, for instance by means of a least squares approach, may slightly increase the numerical stability of the co-simulation method.

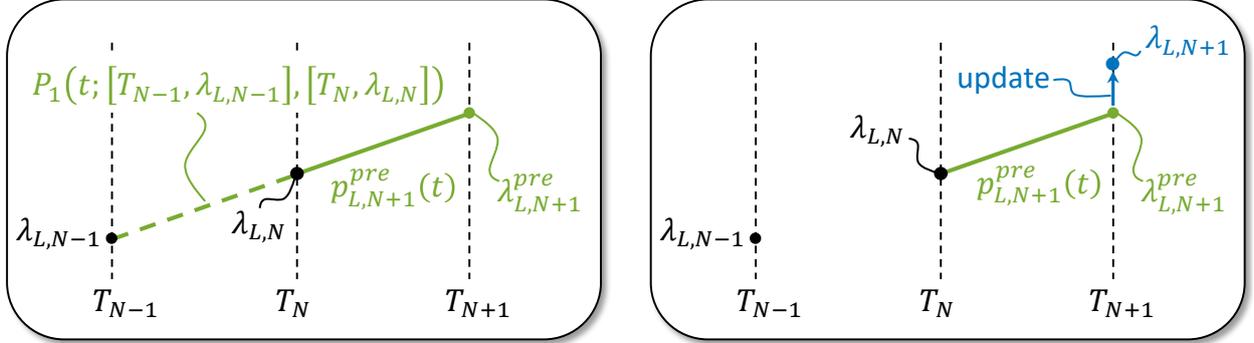
The next step is the generation of the approximation polynomials  $\mathbf{p}_{N+1}^{pre}(t) = [p_{1,N+1}^{pre}(t), \dots, p_{n_c,N+1}^{pre}(t)]^T$  for the coupling variables  $\mathbf{u}(t)$  within the interval  $[T_N, T_{N+1}]$ . The approximation polynomials

$$\mathbf{p}_{N+1}^{pre}(t) = \mathbf{P}_\kappa(t; [T_{N-\kappa+1}, \mathbf{u}_{N-\kappa+1}], \dots, [T_N, \mathbf{u}_N], [T_{N+1}, \mathbf{u}_{N+1}^{pre}]) \quad (2.24)$$

of degree  $\kappa$  are generated by using the values of the coupling variables at  $\kappa$  previous macro-time points and the predicted coupling variables  $\mathbf{u}_{N+1}^{pre}$  as supporting points. The process is visualized for an arbitrary coupling variable  $\lambda_L$  and linear polynomials in Fig. 2.11.

In fact, the polynomials used in Eq. (2.23) and Eq. (2.24) are identical for the case that  $\mathbf{u}_{N+1}^{pre}$  is predicted by extrapolation and  $\mathbf{p}_{N+1}^{pre}(t)$  is an interpolation polynomial of the same degree  $\kappa$ . In general, other approaches can be used to compute  $\mathbf{u}_{N+1}^{pre}$  then the polynomials will not be identical. It is desirable, that the approximation polynomials  $\mathbf{p}_N^{pre}(t)$  and  $\mathbf{p}_{N+1}^{pre}(t)$  of two consecutive macro-

steps merge into each other as smooth as possible at  $T_N$  to minimize discontinuities. However, there are no smoothness requirements for the prediction of the values  $\mathbf{u}_{N+1}^{pre}$ . Therefore, it might be useful to distinguish between the prediction of  $\mathbf{u}_{N+1}^{pre}$  and the generation of the approximation polynomials  $p_{N+1}^{pre}(t)$ .



**Figure 2.11:** Approximation of arbitrary coupling force  $\lambda_L$  with linear polynomials (left figure) and update of arbitrary coupling force  $\lambda_L$  (right figure).

The subsystems equations of motion Eq. (2.17) are integrated from  $T_N$  to  $T_{N+1}$  using the approximation polynomials  $p_{N+1}^{pre}(t)$ , which substitute the unknown coupling variables  $\mathbf{u}(t)$ . Note that all subsystems are integrated independently of each other so that all subsystem integrations can be carried out in parallel. After the subsystem integrations from  $T_N$  to  $T_{N+1}$  have been accomplished, updated coupling variables  $\mathbf{u}_{N+1}$  are computed. Therefore, the subsystem output variables  $\mathbf{y}_{N+1}$  at the new macro-time point  $T_{N+1}$ , which have been obtained by the subsystem integrations, are inserted into the coupling conditions  $\mathbf{g}(\mathbf{u}_{N+1}, \varphi(\mathbf{y}_{N+1}))$ , see Eq. (2.20). The coupling equations are solved for the updated coupling forces according to

$$\mathbf{u}_{N+1} = \varphi(\mathbf{y}_{N+1}) \quad (2.25)$$

or in particular

$$\begin{aligned} \lambda_{L,N+1} &= {}^L F_{SD,c}(\mathbf{y}_{N+1}) \\ &= {}^L C_c ({}^{L+1} x_{1,N+1} - {}^L x_{n_{K,L},N+1}) + {}^L d_c ({}^{L+1} v_{1,N+1} - {}^L v_{n_{K,L},N+1}) \\ &\quad + {}^L C_c ({}^{L+1} x_{1,N+1} - {}^L x_{n_{K,L},N+1})^3 + {}^L D_c ({}^{L+1} v_{1,N+1} - {}^L v_{n_{K,L},N+1})^3 \end{aligned} \quad (2.26)$$

with coupling index  $L = 1, \dots, n_c$ . Then the co-simulation is continued with the next macro-time step  $T_{N+1} \rightarrow T_{N+2}$ . The update process of an arbitrary coupling force  $\lambda_L$  is illustrated in Fig. 2.11 for the case of linear approximation polynomials.

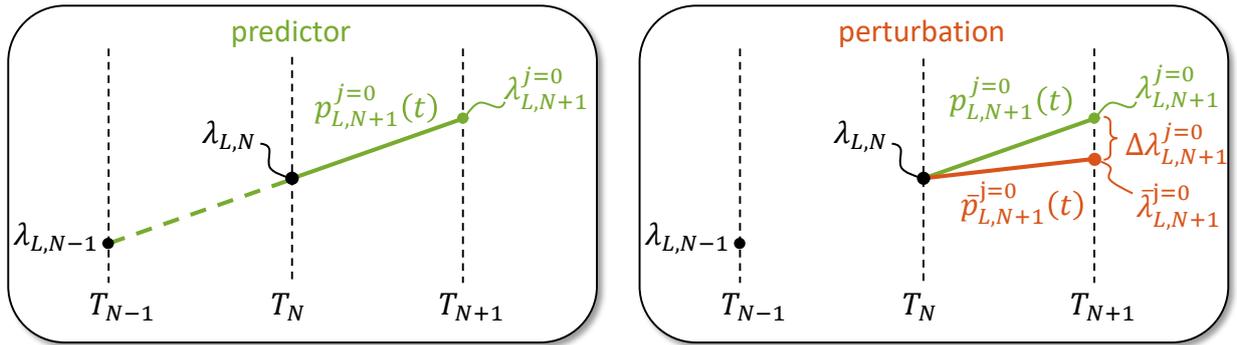
It should be mentioned that in practical applications, difficulties with the implementation may occur in connection with feed-through subsystems. Especially in connection with commercial simulation tools, it may be technically complicated or practically impossible to carry out the update step due to solver restrictions and reduced solver access. Then,  $\mathbf{u}_{N+1} = \varphi(T_{N+1}, \mathbf{z}_{N+1}, \mathbf{u}_{N+1})$  is replaced by  $\mathbf{u}_{N+1} = \varphi(T_{N+1}, \mathbf{z}_{N+1}, \mathbf{u}_{N+1}^{pre})$ . As a consequence, however, problems with the error estimator will arise if a macro-step size controller is used, see Section 3.1.3.

### 2.4.2. Implicit Co-Simulation Method

To explain the classical implicit co-simulation approach, we again consider a general macro-step from  $T_N$  to  $T_{N+1}$ . At the beginning of the macro-time step, the variables of the subsystems  $z_N$  and the coupling variables  $u_N$  are assumed to be known. The intention of the implicit approach is to determine (corrected) coupling variables  $u_{N+1}$  so that the coupling conditions  $g(u_{N+1}, \varphi(y_{N+1})) = 0$  (abbreviated by  $g(u_{N+1}, y_{N+1}) = 0$ ) are fulfilled at  $T_{N+1}$ . The subsystem output variables  $y_{N+1}$  (state variables of the coupling bodies) are obtained by integrating the subsystems with the approximation polynomials  $p_{N+1}(t) = P_\kappa(t; [T_{N-\kappa+1}, u_{N-\kappa+1}], \dots, [T_N, u_N], [T_{N+1}, u_{N+1}])$ .

The classical implicit co-simulation method is based on a predictor/corrector approach. The corrected coupling variables  $u_{N+1}$  are computed by Newton's method.

First, a predictor step is carried out to obtain starting values  $u_{N+1}^{j=0} := u_{N+1}^{pre}$  for the Newton iteration with the iteration index  $j$ . The predictor step is an explicit co-simulation step as described in Section 2.4.1. The predicted coupling variables  $u_{N+1}^{j=0}$  are obtained by extrapolation according to Eq. (2.23) and the approximation polynomials  $p_{N+1}^{j=0}(t) := p_{N+1}^{pre}(t)$  of degree  $\kappa$  are generated according to Eq. (2.24), see Fig 2.12 (left figure). The subsystems (2.17) are integrated from  $T_N$  to  $T_{N+1}$  using the approximation polynomials  $p_{N+1}^{j=0}(t)$  to substitute the unknown coupling variables  $u(t)$ .



**Figure 2.12:** Implicit co-simulation approach: predicted and perturbed approximation polynomials for arbitrary coupling force  $\lambda_L$  (linear approximation).

The resulting subsystem output variables  $y_{N+1}^{j=0}(u_{N+1}^{j=0})$  depend (implicitly) on the approximation polynomials  $p_{N+1}^{j=0}(t)$  and therefore on the predicted coupling variables  $u_{N+1}^{j=0}$ . Inserting  $y_{N+1}^{j=0}(u_{N+1}^{j=0})$  into the coupling equations  $g$  yields the relationship  $g(u_{N+1}^{j=0}, y_{N+1}^{j=0}(u_{N+1}^{j=0}))$ . In general, the coupling equations are not fulfilled, i.e.  $g(u_{N+1}^{j=0}, y_{N+1}^{j=0}(u_{N+1}^{j=0})) \neq 0$ . To find (corrected) coupling variables  $u_{N+1}$ , which fulfill the coupling equations  $g(u_{N+1}, y_{N+1}(u_{N+1})) = 0$ , a Newton iteration is carried out. Therefore, the relationship  $g(u_{N+1}, y_{N+1}(u_{N+1})) = 0$  is linearized by a Taylor series expansion.

For the first corrector iteration step, the predictor point  $[u_{N+1}^{j=0}, y_{N+1}^{j=0}(u_{N+1}^{j=0})]$  is used as expansion point for the Taylor series expansion and the linearized coupling equations

$$g^{lin}(u_{N+1}, y_{N+1}(u_{N+1})) := g(u_{N+1}^{j=0}, y_{N+1}^{j=0}(u_{N+1}^{j=0})) + \left. \frac{dg}{du_{N+1}} \right|_{u_{N+1}^{j=0}} \cdot (u_{N+1} - u_{N+1}^{j=0}) = 0 \quad (2.27)$$

are obtained. For example, the linearized coupling condition  $g_L^{lin}$  for the connection of subsystem  $L$  with subsystem  $L + 1$  reads

$$\begin{aligned}
 g_L^{lin} := & g_L \left( \lambda_{L,N+1}^{j=0}, \mathbf{y}_{L,N+1}^{j=0} \left( \lambda_{L-1,N+1}^{j=0}, \lambda_{L,N+1}^{j=0} \right), \mathbf{y}_{L+1,N+1}^{j=0} \left( \lambda_{L,N+1}^{j=0}, \lambda_{L+1,N+1}^{j=0} \right) \right) \\
 & + \frac{dg_L}{d\lambda_{L-1}} \Big|_{\lambda_{L-1,N+1}^{j=0}} \left( \lambda_{L-1,N+1} - \lambda_{L-1,N+1}^{j=0} \right) \\
 & + \frac{dg_L}{d\lambda_L} \Big|_{\lambda_{L,N+1}^{j=0}} \left( \lambda_{L,N+1} - \lambda_{L,N+1}^{j=0} \right) \\
 & + \frac{dg_L}{d\lambda_{L+1}} \Big|_{\lambda_{L+1,N+1}^{j=0}} \left( \lambda_{L+1,N+1} - \lambda_{L+1,N+1}^{j=0} \right) = 0.
 \end{aligned} \tag{2.28}$$

With the abbreviations

$$\begin{aligned}
 {}^L x_{n_{K,L}}^{j=0} & := {}^L x_{n_{K,L}} \left( \lambda_{L-1,N+1}^{j=0}, \lambda_{L,N+1}^{j=0} \right) ; & {}^L v_{n_{K,L}}^{j=0} & := {}^L v_{n_{K,L}} \left( \lambda_{L-1,N+1}^{j=0}, \lambda_{L,N+1}^{j=0} \right) \\
 {}^{L+1} x_1^{j=0} & := {}^{L+1} x_1 \left( \lambda_{L,N+1}^{j=0}, \lambda_{L+1,N+1}^{j=0} \right) ; & {}^{L+1} v_1^{j=0} & := {}^{L+1} v_1 \left( \lambda_{L,N+1}^{j=0}, \lambda_{L+1,N+1}^{j=0} \right)
 \end{aligned}$$

and

$$S_x := {}^L c_c + 3 {}^L C_c \left( {}^{L+1} x_{1,N+1}^{j=0} - {}^L x_{n_{K,L},N+1}^{j=0} \right)^2 ; \quad S_v := {}^L d_c + 3 {}^L D_c \left( {}^{L+1} v_{1,N+1}^{j=0} - {}^L v_{n_{K,L},N+1}^{j=0} \right)^2$$

and omitting the index  $N + 1$  for the reason of a clear representation, we obtain

$$\begin{aligned}
 g_L^{lin} = & \lambda_L^{j=0} - \left[ {}^L c_c \left( {}^{L+1} x_1^{j=0} - {}^L x_{n_{K,L}}^{j=0} \right) + {}^L d_c \left( {}^{L+1} v_1^{j=0} - {}^L v_{n_{K,L}}^{j=0} \right) \right. \\
 & \left. + {}^L C_c \left( {}^{L+1} x_1^{j=0} - {}^L x_{n_{K,L}}^{j=0} \right)^3 + {}^L D_c \left( {}^{L+1} v_1^{j=0} - {}^L v_{n_{K,L}}^{j=0} \right)^3 \right] \\
 & + \left[ \frac{\partial {}^L x_{n_{K,L}}}{\partial \lambda_{L-1}} \Big|_{\lambda_{L-1}^{j=0}} S_x + \frac{\partial {}^L v_{n_{K,L}}}{\partial \lambda_{L-1}} \Big|_{\lambda_{L-1}^{j=0}} S_v \right] \left( \lambda_{L-1} - \lambda_{L-1}^{j=0} \right) \\
 & + \left[ 1 - \frac{\partial \left( {}^{L+1} x_1 - {}^L x_{n_{K,L}} \right)}{\partial \lambda_L} \Big|_{\lambda_L^{j=0}} S_x - \frac{\partial \left( {}^{L+1} v_1 - {}^L v_{n_{K,L}} \right)}{\partial \lambda_L} \Big|_{\lambda_L^{j=0}} S_v \right] \left( \lambda_L - \lambda_L^{j=0} \right) \\
 & + \left[ - \frac{\partial {}^L x_1}{\partial \lambda_{L+1}} \Big|_{\lambda_{L+1}^{j=0}} S_x - \frac{\partial {}^L v_1}{\partial \lambda_{L+1}} \Big|_{\lambda_{L+1}^{j=0}} S_v \right] \left( \lambda_{L+1} - \lambda_{L+1}^{j=0} \right) = 0.
 \end{aligned} \tag{2.29}$$

The partial derivatives  $\frac{\partial \mathbf{y}_{N+1}}{\partial \mathbf{u}_{N+1}} \Big|_{\mathbf{u}_{N+1}^{j=0}}$  (also called interface Jacobian) of the subsystem output variables with respect to the coupling variables, which appear in the linearized coupling conditions (2.29), are numerically computed by finite differences

$$\begin{aligned}
 \frac{\partial {}^L x_{n_{K,L}}}{\partial \lambda_{L-1}} \Big|_{\lambda_{L-1}^{j=0}} & \approx \frac{{}^L x_{n_{K,L}} \left( \bar{\lambda}_{L-1}^{j=0}, \lambda_L^{j=0} \right) - {}^L x_{n_{K,L}}^{j=0}}{\bar{\lambda}_{L-1}^{j=0} - \lambda_{L-1}^{j=0}} ; & \frac{\partial {}^L v_{n_{K,L}}}{\partial \lambda_{L-1}} \Big|_{\lambda_{L-1}^{j=0}} & \approx \frac{{}^L v_{n_{K,L}} \left( \bar{\lambda}_{L-1}^{j=0}, \lambda_L^{j=0} \right) - {}^L v_{n_{K,L}}^{j=0}}{\bar{\lambda}_{L-1}^{j=0} - \lambda_{L-1}^{j=0}} \\
 \frac{\partial {}^L x_{n_{K,L}}}{\partial \lambda_L} \Big|_{\lambda_L^{j=0}} & \approx \frac{{}^L x_{n_{K,L}} \left( \lambda_{L-1}^{j=0}, \bar{\lambda}_L^{j=0} \right) - {}^L x_{n_{K,L}}^{j=0}}{\bar{\lambda}_L^{j=0} - \lambda_L^{j=0}} ; & \frac{\partial {}^L v_{n_{K,L}}}{\partial \lambda_L} \Big|_{\lambda_L^{j=0}} & \approx \frac{{}^L v_{n_{K,L}} \left( \lambda_{L-1}^{j=0}, \bar{\lambda}_L^{j=0} \right) - {}^L v_{n_{K,L}}^{j=0}}{\bar{\lambda}_L^{j=0} - \lambda_L^{j=0}}
 \end{aligned}$$

$$\begin{aligned} \left. \frac{\partial^{L+1} x_1}{\partial \lambda_L} \right|_{\lambda_L^{j=0}} &\approx \frac{L+1 x_1 \left( \bar{\lambda}_L^{j=0}, \lambda_{L+1}^{j=0} \right) - L+1 x_1^{j=0}}{\bar{\lambda}_L^{j=0} - \lambda_L^{j=0}} ; & \left. \frac{\partial^{L+1} v_1}{\partial \lambda_L} \right|_{\lambda_L^{j=0}} &\approx \frac{L+1 v_1 \left( \bar{\lambda}_L^{j=0}, \lambda_{L+1}^{j=0} \right) - L+1 v_1^{j=0}}{\bar{\lambda}_L^{j=0} - \lambda_L^{j=0}} \\ \left. \frac{\partial^{L+1} x_1}{\partial \lambda_{L+1}} \right|_{\lambda_{L+1}^{j=0}} &\approx \frac{L+1 x_1 \left( \lambda_L^{j=0}, \bar{\lambda}_{L+1}^{j=0} \right) - L+1 x_1^{j=0}}{\bar{\lambda}_{L+1}^{j=0} - \lambda_{L+1}^{j=0}} ; & \left. \frac{\partial^{L+1} v_1}{\partial \lambda_{L+1}} \right|_{\lambda_{L+1}^{j=0}} &\approx \frac{L+1 v_1 \left( \lambda_L^{j=0}, \bar{\lambda}_{L+1}^{j=0} \right) - L+1 v_1^{j=0}}{\bar{\lambda}_{L+1}^{j=0} - \lambda_{L+1}^{j=0}} . \end{aligned}$$

Therefore, the subsystems are integrated with perturbed approximation polynomials

$$\bar{p}_{N+1}^{j=0}(t) = \mathbf{P}_\kappa \left( t; [T_{N-\kappa+1}, \mathbf{u}_{N-\kappa+1}], \dots, [T_N, \mathbf{u}_N], [T_{N+1}, \bar{\mathbf{u}}_{N+1}^{j=0}] \right), \quad (2.30)$$

see Fig 2.12 (right figure). The perturbed coupling variables  $\bar{\mathbf{u}}_{N+1}^{j=0} = \mathbf{u}_{N+1}^{j=0} + \Delta \mathbf{u}_{N+1}^{j=0}$  are obtained by adding user defined perturbations  $\Delta \mathbf{u}_{N+1}^{j=0} = [\Delta \lambda_{1,N+1}^{j=0}, \dots, \Delta \lambda_{n_c,N+1}^{j=0}]^T$  to the predicted coupling variables. In the above equations a special abbreviation is used:  $L+1 x_1 \left( \lambda_L^{j=0}, \bar{\lambda}_{L+1}^{j=0} \right)$  means, for instance, the displacement of body 1 of subsystem  $L+1$  at  $T_{N+1}$  that is obtained by an integration of subsystem  $L+1$  from  $T_N$  to  $T_{N+1}$  using the approximation polynomial  $p_{L,N+1}^{j=0}(t)$  for the coupling force  $\lambda_L(t)$  and the perturbed approximation polynomial  $\bar{p}_{L+1,N+1}^{j=0}(t)$  for the coupling force  $\lambda_{L+1}(t)$ . It should be stressed that the two additional subsystem integrations (one per coupling variable), which are required to numerically compute the partial derivatives (interface Jacobian) by finite differences, are carried out in parallel to the obligatory subsystem integrations. In Appendix A, alternative approaches to numerically compute or to approximate the interface Jacobian are discussed.

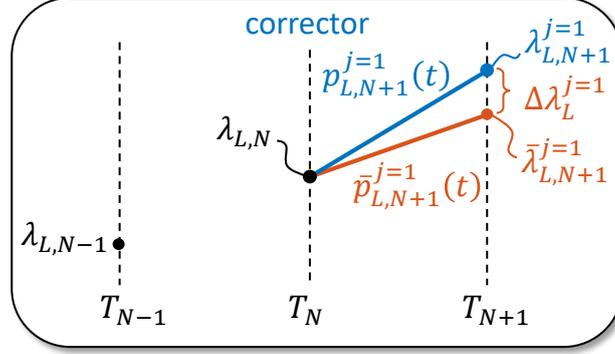
*Remark on the perturbations  $\Delta \mathbf{u}$ :* The choice of adequate perturbations is a challenging matter because it affects the quality of the computed interface Jacobian and therefore the robustness and also the efficiency of the implicit co-simulation. An inappropriate approximation of the interface Jacobian results in an increased number of corrector steps because of the reduced convergence rate of the Newton iteration. Numerical studies with the multibody model introduced in Section 2.2 show that rather large perturbations have a positive influence on the robustness of the simulation. The computational efficiency in contrast suffers from large perturbations, because these have a negative affect on the smoothness of the perturbed approximation polynomials at the macro-time points. Therefore subsystem integrations with the perturbed approximation polynomials require more subsystem solver steps. However, good results have been achieved with the following choice for the  $L = 1, \dots, n_c$  coupling variables:

- predictor step ( $j = 0$ ):  $\Delta \lambda_{L,N+1}^{j=0} = \max \left( \left| \lambda_{L,N} - \lambda_{L,N}^{j=0} \right|, \Delta \lambda_{min} \right)$
- corrector steps ( $j > 0$ ):  $\Delta \lambda_{L,N+1}^j = \max \left( \left| \lambda_{L,N+1}^j - \lambda_{L,N+1}^{j-1} \right|, \Delta \lambda_{min} \right)$ .

The vectors  $\mathbf{u}_N^{j=0} = [\lambda_{1,N}^{j=0}, \dots, \lambda_{n_c,N}^{j=0}]^T$  and  $\mathbf{u}_N = [\lambda_{1,N}, \dots, \lambda_{n_c,N}]^T$  collect the predicted and the converged coupling variables of the previous macro-step. In each case, the perturbations  $\Delta \lambda_{L,N+1}^j$  are limited by an absolute lower bound  $\Delta \lambda_{min}$  to avoid a division by small values.

Solving the linear system (2.27) for  $\mathbf{u}_{N+1}$  yields the corrected coupling variables  $\mathbf{u}_{N+1}^{j=1}$ . The iteration is continued, using  $[\mathbf{u}_{N+1}^{j=1}, \mathbf{y}_{N+1}^{j=1}(\mathbf{u}_{N+1}^{j=1})]$  as expansion point for the Taylor series expansion and the above described procedure is repeated. The subsystems are integrated with the corrected

approximation polynomials  $p_{N+1}^{j=1}(t) = P_\kappa \left( t; [T_{N-\kappa+1}, \mathbf{u}_{N-\kappa+1}], \dots, [T_{N+1}, \mathbf{u}_{N+1}^{j=1}] \right)$  and in parallel with the recomputed perturbed approximation polynomials  $\bar{p}_{N+1}^{j=1}(t) = P_\kappa \left( t; [T_{N-\kappa+1}, \mathbf{u}_{N-\kappa+1}], \dots, [T_{N+1}, \bar{\mathbf{u}}_{N+1}^{j=1}] \right)$  to obtain the subsystem output variables  $\mathbf{y}_{N+1}^{j=1}$  and to approximate the interface Jacobian  $\left. \frac{\partial \mathbf{y}_{N+1}}{\partial \mathbf{u}_{N+1}} \right|_{\mathbf{u}_{N+1}^{j=1}}$ , see Fig. 2.13. The resulting system of linearized coupling equations is again solved for  $\mathbf{u}_{N+1} =: \mathbf{u}_{N+1}^{j=2}$ .



**Figure 2.13:** Implicit co-simulation approach: corrected and perturbed approximation polynomials for arbitrary coupling force  $\lambda_L$  (linear approximation).

The iteration is terminated, if a convergence criterion is fulfilled, see Fig. 2.14. The convergence criterion is defined by the relationship

$$\frac{R^j}{1 - R^j} \|\delta_u^j\|_{WRMS} < \tau, \quad (2.31)$$

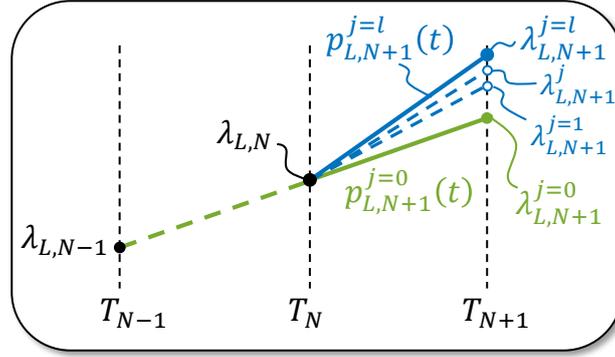
as suggested in [Sha80]. Note that the index  $N + 1$  has been removed from the variables  $R$  and  $\delta_u$  for the reason of a clear representation. The coupling variables  $\mathbf{u}_{N+1}^j$  are assumed to be converged, if the difference  $\delta_u^j = \mathbf{u}_{N+1}^j - \mathbf{u}_{N+1}^{j-1}$  of the coupling variables of two consecutive iteration steps, measured in the weighted root-mean-square norm

$$\|\delta_u^j\|_{WRMS} = \sqrt{\frac{1}{n_c} \sum_{L=1}^{n_c} (\delta_{u,L}^j W_{L,N+1})^2} \quad (2.32)$$

and scaled with a factor depending on the approximated convergence rate  $R^j$ , is smaller than a user defined limit  $\tau$ . The multiplicative weights  $\mathbf{W}_{N+1} = [W_{1,N+1}, \dots, W_{n_c,N+1}]^T$  used to compute the norm are based on the values of the coupling variables and on user defined relative and absolute error tolerances, namely  $W_{L,N+1} = 1 / \left( \text{atol}_L + \text{rtol} \left| \lambda_{L,N+1}^{j=0} \right| \right)$  for index  $L = 1, \dots, n_c$ . The tolerances are typically identical to the tolerances used for the macro-step size controller, at least in the case, where the macro-step size controller is based on an error estimator that estimates the error of the coupling variables. If different types of variables are used for the definition of the convergence criterion and for the error estimation, then it may be necessary to use also different tolerances for each task. The convergence rate is approximated according to

$$R^j = \frac{\|\delta_u^j\|_{WRMS}}{\|\delta_u^{j-1}\|_{WRMS}}. \quad (2.33)$$

The value  $\tau < 1$  must be chosen small enough to ensure that the error of the nonlinear iteration does not interfere with the local error test of the macro-step size controller. On the other hand, a too small value unnecessarily increases the number of Newton iterations and therefore the computation time. A common value used for ODE solvers is  $\tau = 0.1$  [Sha80]. For most simulations carried out in the scope of this work, setting  $\tau = 0.33$  lead to a slightly better performance of the implicit co-simulation. If the co-simulation fails in connection with highly nonlinear models, the problem may be solved by reducing the value of  $\tau$  drastically, e.g.  $\tau = 0.01$ .



**Figure 2.14:** Implicit co-simulation approach: corrector iteration for arbitrary coupling force  $\lambda_L$  (linear approximation polynomials).

In an alternative approach, the convergence criterion (2.31) is formulated in terms of the subsystem output variables  $\mathbf{y}_{N+1}^j$  instead of the coupling variables  $\mathbf{u}_{N+1}^j$ . Therefore, the difference  $\delta_u^j$  is replaced by  $\delta_y^j = \mathbf{y}_{N+1}^j - \mathbf{y}_{N+1}^{j-1}$  (and  $\delta_u^{j-1}$  by  $\delta_y^{j-1}$ ) in Eq. (2.31) and Eq. (2.33). The tolerances may have to be adjusted, depending on the order of magnitude of the considered variables.

Within this work the convergence criterion is always formulated in terms of the variables that are used for the error estimation of the macro-step size controller. If the error estimator described in Section 3.1.5 is applied, the convergence criterion (2.31) is used. In case of an estimation of the error of the state variables of the coupling bodies (Section 3.1.4), the convergence criterion is also stated in terms of the subsystem output variables, which are by definition the state variables of the coupling bodies.

The values  $\mathbf{u}_{N+1}^{j=l}$  and  $\mathbf{z}_{N+1}^{j=l}$  obtained by the final iteration  $l$  are used as corrected variables  $\mathbf{u}_{N+1}$  and  $\mathbf{z}_{N+1}$ . The next macro-step from  $T_{N+1} \rightarrow T_{N+2}$  is carried out using these corrected variables.

### 2.4.3. Waveform Relaxation Method

An alternative iterative co-simulation method is based on a waveform relaxation approach. A general macro-step from  $T_N$  to  $T_{N+1}$  is carried out in the same way as in the explicit co-simulation method described in Section 2.4.1. The predicted coupling variables  $\mathbf{u}_{N+1}^{j=0} := \mathbf{u}_{N+1}^{pre}$  are obtained by extrapolation according to Eq. (2.23) and the approximation polynomials  $\mathbf{p}_{N+1}^{j=0}(t) := \mathbf{p}_{N+1}^{pre}(t)$  of degree  $\kappa$  are generated according to Eq. (2.24).

Then each subsystem is integrated separately and the resulting subsystem output variables  $\mathbf{y}_{N+1}^{j=0}$  are used to compute updated coupling variables  $\mathbf{u}_{N+1}^{j=1}$  according to Eq. (2.26). Within the next iteration step, the updated coupling variables are used to obtain improved approximation

polynomials according to

$$\mathbf{p}_{N+1}^{j=1}(t) = \mathbf{P}_\kappa \left( t; [T_{N-\kappa+1}, \mathbf{u}_{N-\kappa+1}], \dots, [T_N, \mathbf{u}_N], [T_{N+1}, \mathbf{u}_{N+1}^{j=1}] \right). \quad (2.34)$$

The integration of the subsystems from  $T_N$  to  $T_{N+1}$  is repeated with the improved approximation polynomials  $\mathbf{p}_{N+1}^{j=1}(t)$ . The iteration is terminated, if a convergence criterion is fulfilled, as described above in Section 2.4.2 for the implicit co-simulation approach. The next macro-step from  $T_{N+1} \rightarrow T_{N+2}$  is carried out using the corrected variables  $\mathbf{u}_{N+1} := \mathbf{u}_{N+1}^{j=l}$  and  $z_{N+1} := z_{N+1}^{j=l}$ , with  $l$  being the index of the final iteration step.

The method can be modified by using a relaxation parameter  $\omega \in (0, 1)$ , yielding the approximation polynomials

$$\mathbf{p}_{N+1}^j(t) = \mathbf{P}_\kappa \left( t; [T_{N-\kappa+1}, \mathbf{u}_{N-\kappa+1}], \dots, [T_N, \mathbf{u}_N], [T_{N+1}, \left( (1-\omega)\mathbf{u}_{N+1}^j + \omega\mathbf{u}_{N+1}^{j-1} \right)] \right). \quad (2.35)$$

The advantage of the waveform method is, that it does not require the computation of the interface Jacobian. Numerical tests indicate, however, that the implicit co-simulation method described within the previous subsection shows a clearly better overall performance. The Newton method converges with significantly fewer corrector steps and also shows superior numerical stability properties.

### 3. Macro-Step Size and Order Control Algorithm

The algorithm for macro-step size and order control considered here is carried out in three stages. It is a straightforward generalization of a step size and order controller well-established in classical time integration schemes [SG75, Ske77, BCP96, ESF98]. The first stage is to check, whether the macro-step is accepted or if the estimated error is too large so that the macro-step has to be repeated. The process of estimating the local error of a co-simulation step is described in Section 3.1 for the explicit co-simulation approach and for the implicit co-simulation approach. The second stage consists of the order selection algorithm (Section 3.3) and the third stage contains the calculation of the optimal step size for the next macro-step (Section 3.4). In this context, the term *order* refers to the polynomial degree  $\kappa$  of the approximation polynomials of the coupling variables.

#### 3.1. Error Estimators for Co-Simulation Approaches

For deriving an error estimator, the co-simulation results of a macro-step have to be compared with a numerically generated reference solution. Based on these two solutions, an error estimator for the macro-step is constructed. Roughly speaking, the numerical error of a co-simulation has two components, namely the numerical error generated by the subsystem integration with the subsystem solver and the numerical error produced by the co-simulation approach, i.e. by the polynomial approximation of the coupling variables. In the subsequent analysis, it is generally assumed that the error introduced by the numerical subsystem integration is much smaller than the error produced by the co-simulation. In practical applications, this hypothesis is often justified and valid, especially for the case that the subsystems are solved with small error tolerances so that the micro-time step sizes of the subsystem integrators are smaller than the macro-step size.

#### Error Estimators for the Explicit Co-Simulation Approach

In the following subsections, three different error estimators for the explicit co-simulation method described in Section 2.4.1 are presented. The first and third error estimator are based on the local extrapolation technique [Sha73, SW81] and the second estimator is based on the Milne device approach [Mil26].

##### 3.1.1. Method e1: Local Extrapolation (exLE)

A general macro-step  $T_N \rightarrow T_{N+1} = T_N + H$  of the explicit co-simulation scheme described in Section 2.4.1 is considered. Using approximation polynomials  $\mathbf{p}_{N+1}^{pre}(t)$  of degree  $\kappa$  according to Eq. (2.24) with the predicted (extrapolated) values of the coupling variables  $\mathbf{u}_{N+1}^{pre}$  according to Eq. (2.23), the coupling variables  $\mathbf{u}_{N+1}^{pre}$  are extrapolated with order  $\mathcal{O}(H^{\kappa+1})$ . Since the coupling variables and therefore the accelerations are approximated with order  $\mathcal{O}(H^{\kappa+1})$ , the velocities  $\mathbf{v}_{N+1}$  will converge with order  $\mathcal{O}(H^{\kappa+2})$  and the position variables  $\mathbf{q}_{N+1}$  with order  $\mathcal{O}(H^{\kappa+3})$ , i.e.

$$\begin{aligned} \mathbf{q}_{N+1} &= \mathbf{q}(T_{N+1}) + \mathcal{O}(H^{\kappa+3}) \\ \mathbf{v}_{N+1} &= \mathbf{v}(T_{N+1}) + \mathcal{O}(H^{\kappa+2}) \end{aligned} \quad (3.1)$$

where  $\mathbf{q}(T_{N+1})$  and  $\mathbf{v}(T_{N+1})$  denote the analytical solutions with  $\mathbf{q}(T_N) = \mathbf{q}_N$  and  $\mathbf{v}(T_N) = \mathbf{v}_N$ . Note that these error bounds are valid for the case that numerical errors due to the subsystem integration can be neglected, e.g. for the case that the subsystems are integrated analytically or numerically with very tight error tolerances.

To calculate an error estimate for  $\mathbf{q}_{N+1}$  and  $\mathbf{v}_{N+1}$ , comparative solutions  $\hat{\mathbf{q}}_{N+1}$  and  $\hat{\mathbf{v}}_{N+1}$  are required. Using the local extrapolation technique for generating an error estimator, two solutions with different convergence orders are compared. Therefore, the considered macro-step  $T_N \rightarrow T_{N+1}$  is carried out twice; the comparative solution is calculated with the same initial conditions ( $\hat{\mathbf{z}}_N = \mathbf{z}_N$ ):

- The first co-simulation step (actual simulation) is carried out with the approximation polynomials  $\mathbf{p}_{N+1}^{pre}(t)$  of degree  $\kappa$  according to Eq. (2.24) and yields  $\mathbf{q}_{N+1}$  and  $\mathbf{v}_{N+1}$ .
- The second co-simulation step (comparative solution) is accomplished with the approximation polynomials

$$\hat{\mathbf{p}}_{N+1}^{pre}(t) = \mathbf{P}_{\kappa+1}(t; [T_{N-\kappa}, \mathbf{u}_{N-\kappa}], \dots, [T_N, \mathbf{u}_N], [T_{N+1}, \hat{\mathbf{u}}_{N+1}^{pre}]) \quad (3.2)$$

with

$$\hat{\mathbf{u}}_{N+1}^{pre} = \mathbf{P}_{\kappa+1}(T_{N+1}; [T_{N-\kappa-1}, \mathbf{u}_{N-\kappa-1}], [T_{N-\kappa}, \mathbf{u}_{N-\kappa}], \dots, [T_N, \mathbf{u}_N]) \quad (3.3)$$

of degree  $\kappa + 1$  and gives  $\hat{\mathbf{q}}_{N+1}$  and  $\hat{\mathbf{v}}_{N+1}$ .

For the solution of the first integration, the local errors of the position variable  $q_i$  and velocity variable  $v_i$  are given by

$$\begin{aligned} e_{i,N+1}^{pos} &= |q_{i,N+1} - q_i(T_{N+1})| \\ e_{i,N+1}^{vel} &= |v_{i,N+1} - v_i(T_{N+1})|. \end{aligned} \quad (3.4)$$

Replacing in Eq. (3.4) the analytical solutions with the help of  $\hat{q}_{i,N+1} = q_i(T_{N+1}) + \mathcal{O}(H^{\kappa+4})$  and  $\hat{v}_{i,N+1} = v_i(T_{N+1}) + \mathcal{O}(H^{\kappa+3})$ , we simply obtain

$$\begin{aligned} e_{i,N+1}^{pos} &= |q_{i,N+1} - q_i(T_{N+1})| \\ &= |q_{i,N+1} - (\hat{q}_{i,N+1} + \mathcal{O}(H^{\kappa+4}))| \\ &= |q_{i,N+1} - \hat{q}_{i,N+1}| + \mathcal{O}(H^{\kappa+4}) \\ &= \varepsilon_{i,N+1}^{pos} + \mathcal{O}(H^{\kappa+4}) \\ e_{i,N+1}^{vel} &= |v_{i,N+1} - v_i(T_{N+1})| \\ &= |v_{i,N+1} - (\hat{v}_{i,N+1} + \mathcal{O}(H^{\kappa+3}))| \\ &= |v_{i,N+1} - \hat{v}_{i,N+1}| + \mathcal{O}(H^{\kappa+3}) \\ &= \varepsilon_{i,N+1}^{vel} + \mathcal{O}(H^{\kappa+3}). \end{aligned} \quad (3.5)$$

Thus, the local error of the co-simulation in the macro-step from  $T_N \rightarrow T_{N+1}$  can be estimated by the difference of the two solutions, i.e.

$$\begin{aligned} \varepsilon_{i,N+1}^{pos} &= |q_{i,N+1} - \hat{q}_{i,N+1}| \\ \varepsilon_{i,N+1}^{vel} &= |v_{i,N+1} - \hat{v}_{i,N+1}|, \end{aligned} \quad (3.6)$$

where the estimated errors  $\varepsilon_{i,N+1}^{pos}$  and  $\varepsilon_{i,N+1}^{vel}$  converge to the local errors  $e_{i,N+1}^{pos}$  and  $e_{i,N+1}^{vel}$  with order  $\mathcal{O}(H^{\kappa+4})$  and  $\mathcal{O}(H^{\kappa+3})$ , respectively.

The error estimates  $\varepsilon_{i,N+1}^{pos}$  and  $\varepsilon_{i,N+1}^{vel}$  according to Eq. (3.6) can be used to monitor the local error of all state variables. Alternatively, only the states of the coupling bodies may be monitored, which is done in this work.

### 3.1.2. Method e2: Milne Device (exMD)

Next, an error estimator is constructed on the basis of a Milne device approach [Mil26]. Therefore, two different solutions with the same convergence order but different leading error terms are compared. Two parallel simulation steps from  $T_N$  to  $T_{N+1}$  are executed with the same initial conditions ( $\hat{z}_N = z_N$ ):

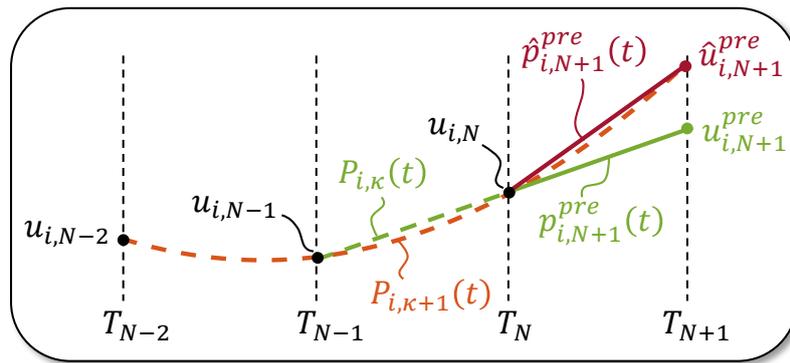
- The first co-simulation step (actual simulation) is carried out with the approximation polynomials  $\hat{p}_{N+1}^{pre}(t)$  of degree  $\kappa$  according to Eq. (2.24) and yields  $\mathbf{q}_{N+1}$  and  $\mathbf{v}_{N+1}$ .
- The second co-simulation step (comparative solution) is accomplished with the approximation polynomials

$$\hat{\mathbf{p}}_{N+1}^{pre}(t) = \mathbf{P}_{\kappa}(t; [T_{N-\kappa+1}, \mathbf{u}_{N-\kappa+1}], \dots, [T_N, \mathbf{u}_N], [T_{N+1}, \hat{\mathbf{u}}_{N+1}^{pre}]) \quad (3.7)$$

with

$$\hat{\mathbf{u}}_{N+1}^{pre} = \mathbf{P}_{\kappa+1}(T_{N+1}; [T_{N-\kappa-1}, \mathbf{u}_{N-\kappa-1}], [T_{N-\kappa}, \mathbf{u}_{N-\kappa}], \dots, [T_N, \mathbf{u}_N]) \quad (3.8)$$

and gives  $\hat{\mathbf{q}}_{N+1}$  and  $\hat{\mathbf{v}}_{N+1}$ . Note that the predicted values of the coupling variables  $\hat{\mathbf{u}}_{N+1}^{pre}$  are obtained by an extrapolation of degree  $\kappa + 1$ ; however, the approximation polynomials  $\hat{p}_{N+1}^{pre}(t)$  are of degree  $\kappa$ . The procedure is illustrated in Fig. 3.1 for an arbitrary coupling variable  $u_i$  and linear approximation polynomials.



**Figure 3.1:** Milne device approach in combination with the explicit co-simulation scheme (exMD): approximation polynomials for the arbitrary coupling variable  $u_i$  of degree  $\kappa = 1$ .

Since  $\mathbf{q}_{N+1}$  and  $\hat{\mathbf{q}}_{N+1}$  are both calculated with polynomials of degree  $\kappa$ , they both show a local convergence with order  $\mathcal{O}(H^{\kappa+3})$ . It can be shown (see [MKS21]) that the local errors of  $\mathbf{q}_{N+1}$

and  $\hat{q}_{N+1}$  can be expressed as

$$\mathbf{q}_{N+1} - \mathbf{q}(T_{N+1}) = H^2 C_{pos,N+1}^{\kappa+1} \mathbf{A}_N^{co} \Delta \mathbf{p}_{N+1} + \mathcal{O}(H^{\kappa+4}) \quad (3.9)$$

$$\hat{\mathbf{q}}_{N+1} - \mathbf{q}(T_{N+1}) = H^2 \left( C_{pos,N+1}^{\kappa+1} - C_{pos,N+1}^{\kappa} \right) \mathbf{A}_N^{co} \Delta \mathbf{p}_{N+1} + \mathcal{O}(H^{\kappa+4}) . \quad (3.10)$$

In the above equation,  $H = T_{N+1} - T_N$  denotes the current macro-step size.  $C_{pos,N+1}^{\kappa}$  and  $C_{pos,N+1}^{\kappa+1}$  denote error constants of the co-simulation, which are derived in [MKS21]. The error constants are calculated by special integrals of Lagrange basis polynomials  $L^{\kappa}(t)$ , namely by

$$C_{pos,N+1}^{\kappa} = \frac{1}{H^2} \int_{T_N}^{T_{N+1}} \int_{T_N}^{\tau} L^{\kappa}(t) dt d\tau \quad \text{with} \quad L^{\kappa}(t) = \prod_{j=0}^{\kappa-1} \frac{t - T_{N-j}}{T_{N+1} - T_{N-j}} . \quad (3.11)$$

Hence,  $C_{pos,N+1}^{\kappa}$  and  $C_{pos,N+1}^{\kappa+1}$  only depend on the polynomial degree  $\kappa$  and on the ratio of the previous macro-step sizes. The matrix  $\mathbf{A}_N^{co}$  can be interpreted as a special Jacobian matrix, which does not explicitly depend on the macro-step size  $H$ . The vector  $\Delta \mathbf{p}_{N+1}$  can be interpreted as the difference of two polynomials, which are evaluated at  $T_{N+1}$ . While a direct calculation of the leading error term  $H^2 C_{pos,N+1}^{\kappa+1} \mathbf{A}_N^{co} \Delta \mathbf{p}_{N+1}$  in Eq. (3.9) is impossible, the leading error term can be determined with the help of the two solutions  $\mathbf{q}_{N+1}$  and  $\hat{\mathbf{q}}_{N+1}$ . Subtracting Eq. (3.10) from Eq. (3.9) and multiplication with  $C_{pos,N+1}^{\kappa+1}/C_{pos,N+1}^{\kappa}$  yields

$$\begin{aligned} \mathbf{q}_{N+1} - \mathbf{q}(T_{N+1}) - (\hat{\mathbf{q}}_{N+1} - \mathbf{q}(T_{N+1})) &= H^2 C_{pos,N+1}^{\kappa} \mathbf{A}_N^{co} \Delta \mathbf{p}_{N+1} + \mathcal{O}(H^{\kappa+4}) \\ \Rightarrow \frac{C_{pos,N+1}^{\kappa+1}}{C_{pos,N+1}^{\kappa}} (\mathbf{q}_{N+1} - \hat{\mathbf{q}}_{N+1}) &= H^2 C_{pos,N+1}^{\kappa+1} \mathbf{A}_N^{co} \Delta \mathbf{p}_{N+1} + \mathcal{O}(H^{\kappa+4}) . \end{aligned} \quad (3.12)$$

Comparing Eq. (3.12) and Eq. (3.9), the local error  $e_{i,N+1}^{pos} = |q_{i,N+1} - \hat{q}_i(T_{N+1})|$  of the actual co-simulation step can be estimated by

$$\varepsilon_{i,N+1}^{pos} = \frac{C_{pos,N+1}^{\kappa+1}}{C_{pos,N+1}^{\kappa}} |q_{i,N+1} - \hat{q}_i(T_{N+1})| , \quad (3.13)$$

which converges to the local error  $e_{i,N+1}^{pos}$  with order  $\mathcal{O}(H^{\kappa+4})$ .

Accordingly, an error estimate  $\varepsilon_{i,N+1}^{vel}$  for the velocity variables can be derived, see [MKS21]. The estimated error

$$\varepsilon_{i,N+1}^{vel} = \frac{C_{vel,N+1}^{\kappa+1}}{C_{vel,N+1}^{\kappa}} |v_{i,N+1} - \hat{v}_i(T_{N+1})| \quad (3.14)$$

approximates the local error  $e_{i,N+1}^{vel} = |v_{i,N+1} - v_i(T_{N+1})|$  of the actual co-simulation with order  $\mathcal{O}(H^{\kappa+3})$ . The error constants for the velocities are defined by the integral

$$C_{vel,N+1}^{\kappa} = \frac{1}{H} \int_{T_N}^{T_{N+1}} L^{\kappa}(t) dt \quad \text{with} \quad L^{\kappa}(t) = \prod_{j=0}^{\kappa-1} \frac{t - T_{N-j}}{T_{N+1} - T_{N-j}} . \quad (3.15)$$

*Remark on the error constants:* usually, the ratios of the error constants  $\frac{C_{pos,N+1}^{\kappa+1}}{C_{pos,N+1}^{\kappa}}$  and  $\frac{C_{vel,N+1}^{\kappa+1}}{C_{vel,N+1}^{\kappa}}$  have to be recalculated in each macro-step, since the error constants depend on the ratios of the current and the previous macro-step sizes. Within a simplified implementation, universal error

constants based on the assumption of a constant macro-step size may be used instead of the exact error constants. The ratios of the universal error constants are collected in Table 3.1. Note that the effort of recomputing the exact error constants in each macro-step is very low, therefore the expectable reduction of computation time achieved by using the the simplified approach is also low.

**Table 3.1:** Error constant ratios for constant macro-step size.

$\kappa$	0	1	2	3	4	5
$C_{pos}^{\kappa+1}/C_{pos}^{\kappa}$	1/3	3/4	38/45	135/152	863/945	9625/10356
$C_{vel}^{\kappa+1}/C_{vel}^{\kappa}$	1/2	5/6	9/10	251/270	475/502	19087/19950

### 3.1.3. Method e3: Local Extrapolation Based on the Coupling Variables (exCV)

Within this error estimation approach, only the error of the coupling variables is considered. Regarding the explicit co-simulation approach, the approximation polynomials  $p_{N+1}^{pre}(t)$  of degree  $\kappa$  according to Eq. (2.24) with the predicted (extrapolated) values of the coupling variables  $\mathbf{u}_{N+1}^{pre}$  according to Eq. (2.23) are used for the integration of the subsystems from  $T_N$  to  $T_{N+1}$ . After the subsystem integration, updated coupling variables  $\mathbf{u}_{N+1}$  are calculated at  $T_{N+1}$  by inserting the new state variables  $\mathbf{q}_{N+1}, \mathbf{v}_{N+1}$  into the coupling equations, i.e. by solving the coupling conditions  $\mathbf{u}_{N+1} - \varphi(T_{N+1}, \mathbf{q}_{N+1}, \mathbf{v}_{N+1}, \mathbf{u}_{N+1}) = \mathbf{0}$  for  $\mathbf{u}_{N+1}$ .

Usually, the predicted coupling variables  $\mathbf{u}_{N+1}^{pre}$  are different from the updated coupling variables  $\mathbf{u}_{N+1}$  and the difference  $\mathbf{u}_{N+1}^{pre} - \mathbf{u}_{N+1}$  can be used to construct an error estimator based on the local extrapolation technique.

In the following, only explicit co-simulations without feed-through are considered. As mentioned above, the state variables  $\mathbf{q}_{N+1}$  and  $\mathbf{v}_{N+1}$  generally converge with orders  $\mathcal{O}(H^{\kappa+3})$  and  $\mathcal{O}(H^{\kappa+2})$ , respectively. Without feed-through, the updated coupling variables  $\mathbf{u}_{N+1} = \varphi(T_{N+1}, \mathbf{q}_{N+1}, \mathbf{v}_{N+1})$  can be calculated explicitly and converge at least with order  $\mathcal{O}(H^{\kappa+2})$  (order  $\mathcal{O}(H^{\kappa+3})$  for the case that the coupling forces only depend on the position variables, i.e. for the case that  $\mathbf{u}_{N+1} = \varphi(T_{N+1}, \mathbf{q}_{N+1})$ ).

The predicted coupling variables  $\mathbf{u}_{N+1}^{pre}$  show a convergence rate of  $\mathcal{O}(H^{\kappa+1})$ . Since the predicted and updated coupling variables show a different convergence rate, the local error  $e_{i,N+1}^u = |u_{i,N+1}^{pre} - u_i(T_{N+1})|$  of the predicted coupling variable  $u_{i,N+1}^{pre}$  can be estimated with the help of the updated coupling variable  $u_{i,N+1}$  according to

$$\begin{aligned}
 e_{i,N+1}^u &= \left| u_{i,N+1}^{pre} - u_i(T_{N+1}) \right| \\
 &= \left| u_{i,N+1}^{pre} - (u_{i,N+1} + \mathcal{O}(H^{\kappa+2})) \right| \\
 &= \left| u_{i,N+1}^{pre} - u_{i,N+1} \right| + \mathcal{O}(H^{\kappa+2}) \\
 &= \varepsilon_{i,N+1}^u + \mathcal{O}(H^{\kappa+2}),
 \end{aligned} \tag{3.16}$$

where the estimated error  $\varepsilon_{i,N+1}^u = |u_{i,N+1}^{pre} - u_{i,N+1}|$  converges with order  $\mathcal{O}(H^{\kappa+2})$  to the local error  $e_{i,N+1}^u$ .

Compared to the error estimators *exLE* and *exMD* of Sections 3.1.1 and 3.1.2 (see Eqs. (3.6), (3.13) and (3.14)), the error estimator according to Eq. (3.16) has several advantages and disadvantages.

- For the error estimators *exLE* and *exMD*, parallel simulations are required in order to generate a comparative solution. Parallel execution of the macro-steps is, however, not required for  $\varepsilon_{i,N+1}^u$  according to Eq. (3.16), which simplifies the implementation and also reduces the amount of required cores.
- A drawback of the error estimator *exCV* is that  $\varepsilon_{i,N+1}^u$  only monitors the error of the predicted coupling variables  $\mathbf{u}_{N+1}^{pre}$ , whereas the error estimators *exLE* and *exMD* estimate the error of the state variables  $\mathbf{q}_{N+1}$ ,  $\mathbf{v}_{N+1}$ .
- The error estimators *exLE* and *exMD* can be used to monitor the error of all subsystem states, i.e. not only the error of the states of the coupling bodies. It should, however, be mentioned again that the error estimators *exLE* and *exMD* may also be used to only monitor the states of the coupling bodies.

### Error Estimation for Explicit Co-Simulation Approaches in Connection with Feed-Through

The error bounds according to Eq. (3.1) are valid for explicit and implicit co-simulation approaches. Moreover, these error bounds hold for systems without feed-through – i.e. for the case that the subsystem output variables are not explicit functions of the coupling (input) variables – and also for systems with feed-through, see [MKS21]. As mentioned in Section 2.4.1, for practical reasons it might be necessary to replace the update equations  $\mathbf{u}_{N+1} = \varphi(T_{N+1}, \mathbf{z}_{N+1}, \mathbf{u}_{N+1})$  by  $\mathbf{u}_{N+1} = \varphi(T_{N+1}, \mathbf{z}_{N+1}, \mathbf{u}_{N+1}^{pre})$ . For explicit co-simulation models with feed-through subsystems, the leading error term will, however, change, if  $\mathbf{u}_{N+1} = \varphi(T_{N+1}, \mathbf{z}_{N+1}, \mathbf{u}_{N+1})$  is simplified by  $\mathbf{u}_{N+1} = \varphi(T_{N+1}, \mathbf{z}_{N+1}, \mathbf{u}_{N+1}^{pre})$ .

The reason therefore is the error produced by polynomial extrapolation/interpolation of the coupling variables. Roughly speaking, the approximation error of the coupling variables has two components. The first component reflects the error due to the polynomial approximation with a finite degree  $\kappa$  and only depends on the number of sampling points. This error component generally converges with order  $\mathcal{O}(H^{\kappa+1})$  for systems with feed-through and also for systems without feed-through. The structure of the first error component reads  $e_{i,N+1}^{u,1} = C H^{\kappa+1} u_i^{(\kappa+1)}(T_{N+1}) + \mathcal{O}(H^{\kappa+2})$  with the constant  $C$  and the derivative  $u_i^{(\kappa+1)} := \frac{d^{\kappa+1} u_i}{dt^{\kappa+1}}$ .

The second error component results from the error of the sampling points, which are not exact values, since they are a result of a co-simulation. Note that the second error component also occurs, if the subsystem are integrated analytically. Considering the explicit co-simulation approach for systems without feed-through, the second error component will converge with order  $\mathcal{O}(H^{\kappa+2})$ . For explicit co-simulation models with feed-through subsystems, the second error component only converges with order  $\mathcal{O}(H^{\kappa+1})$ . The structure of the second error component reads  $e_{i,N+1}^{u,2} = \sum_{j=0}^{\kappa} c_j \Delta u_{i,N-j} + \mathcal{O}(H^{\kappa+2})$ , with the error of the sampling points  $\Delta u_{i,N-j} := u_{i,N-j} - u_i(T_{N-j})$  and constants  $c_j$  (see [MKS21]).

Hence, regarding explicit co-simulation schemes in connection with feed-through, the leading error term contains two summands and has the structure  $e_{i,N+1}^u = e_{i,N+1}^{u,1} + e_{i,N+1}^{u,2} =$

$$C H^{\kappa+1} u_i^{(\kappa+1)}(T_{N+1}) + \sum_{j=0}^{\kappa} c_j \Delta u_{i,N-j} + \mathcal{O}(H^{\kappa+2}).$$

a) *Error estimator based on Milne device for systems with feed-through:*

By calculating the difference between the actual co-simulation and a comparative solution, which both have the same convergence order, an error estimation based on the Milne device approach according to Section 3.1.2 is not possible for systems with feed-through in connection with the simplified update  $\mathbf{u}_{N+1} = \varphi(T_{N+1}, \mathbf{z}_{N+1}, \mathbf{u}_{N+1}^{pre})$ , since the leading error term consist of two components.

b) *Error estimator based on local extrapolation for systems with feed-through:*

Error estimation with the local extrapolation idea of Section 3.1.1 is impossible for systems with feed-through in connection with the simplified update  $\mathbf{u}_{N+1} = \varphi(T_{N+1}, \mathbf{z}_{N+1}, \mathbf{u}_{N+1}^{pre})$ , since the two approximation polynomials –  $\mathbf{p}_{N+1}^{pre}(t)$  (Eq. (2.24)) of degree  $\kappa$  for the actual co-simulation and  $\hat{\mathbf{p}}_{N+1}^{pre}(t)$  (Eq. (3.2)) of degree  $\kappa + 1$  for the comparative solution – have the same approximation order  $\mathcal{O}(H^{\kappa+1})$ : although  $\hat{\mathbf{p}}_{N+1}^{pre}(t)$  is generated with  $\kappa + 2$  sampling points,  $\hat{\mathbf{u}}_{N+1}^{pre}$  only converges with order  $\mathcal{O}(H^{\kappa+1})$ , since the accuracy of the sampling points  $[T_{N-\kappa-1}, \mathbf{u}_{N-\kappa-1}], \dots, [T_N, \mathbf{u}_N]$  is only of order  $\mathcal{O}(H^{\kappa+1})$  for explicit parallel co-simulation systems with feed-through.

This problem can be solved, if consistent coupling variables are calculated at  $T_{N+1}$ , i.e. by solving  $\mathbf{u}_{N+1} = \varphi(T_{N+1}, \mathbf{z}_{N+1}, \mathbf{u}_{N+1})$  for  $\mathbf{u}_{N+1}$ . For the feed-through system of Eq. (2.7), for instance, consistent coupling variables are obtained by replacing  $\mathbf{u}_{1,N+1} = \mathbf{y}_{2,N+1} = [c_c(x_{2,N+1} - \tilde{x}_{1,N+1}) + d_c(v_{2,N+1} - \tilde{v}_{1,N+1})]$  by the consistent variables  $\mathbf{u}_{1,N+1} = \mathbf{y}_{2,N+1} = [c_c(x_{2,N+1} - x_{1,N+1}) + d_c(v_{2,N+1} - v_{1,N+1})]$ . In this simple example, the calculation of the force update is unproblematic, since the law for the coupling force is explicitly known so that new state variables  $\mathbf{q}_{N+1}, \mathbf{v}_{N+1}$  have only to be inserted into the constitutive law of the coupling force. In practical and complex applications, there might be no explicit expression available for the coupling force as a function of the subsystem states, since the coupling force is a result of the subsystem integration. Then, a static updated calculation at  $T_{N+1}$  would be necessary to generate consistent coupling forces at  $T_{N+1}$ , which may – from the technical point of view – be elaborately with commercial software tools.

*Summarizing:* For explicit co-simulation models with feed-through subsystems, where the simplified update  $\mathbf{u}_{N+1} = \varphi(T_{N+1}, \mathbf{z}_{N+1}, \mathbf{u}_{N+1}^{pre})$  is used, the presented error estimators cannot be used. In this case, usage of the presented error estimators is only possible, if an additional update step at  $T_{N+1}$  is carried out in order to get consistent coupling variables. In the general nonlinear case, the additional update may be interpreted as a nonlinear iteration of the subsystem output variables at  $T_{N+1}$  [KS00].

Using an implicit co-simulation approach for systems with feed-through does not entail problems with the error estimation, since consistent coupling variables at  $T_{N+1}$  are obtained through the iteration process.

### Error Estimators for the Implicit Co-Simulation Approach

In the following subsections, two different error estimators for the implicit co-simulation method described in Section 2.4.2 are presented. The first estimator is based on the Milne device approach [Mil26] and the second error estimator is based on the local extrapolation technique [Sha73,

SW81].

### 3.1.4. Method i1: Milne Device (*imMD*)

The implicit co-simulation approach of Section 2.4.2 is based on a predictor/corrector scheme. The difference between the predicted and corrected state variables can be used to generate an error estimator with the help of the Milne device approach. The state variables  $\mathbf{q}_{N+1}^{pre}$ ,  $\mathbf{v}_{N+1}^{pre}$  of the predictor step and also the corrected state variables  $\mathbf{q}_{N+1}$ ,  $\mathbf{v}_{N+1}$  converge with order  $\mathcal{O}(H^{\kappa+3})$  and order  $\mathcal{O}(H^{\kappa+2})$ , respectively. In [MKS21], it is shown that the local errors of  $\mathbf{q}_{N+1}^{pre}$  and  $\mathbf{q}_{N+1}$  can be expressed as

$$\begin{aligned} \mathbf{q}_{N+1}^{pre} - \mathbf{q}(T_{N+1}) &= H^2 C_{pos,N+1}^{\kappa} \mathbf{A}_N^{co} \Delta \mathbf{p}_{N+1} + \mathcal{O}(H^{\kappa+4}) \\ \mathbf{q}_{N+1} - \mathbf{q}(T_{N+1}) &= H^2 \left( C_{pos,N+1}^{\kappa+1} - C_{pos,N+1}^{\kappa} \right) \mathbf{A}_N^{co} \Delta \mathbf{p}_{N+1} + \mathcal{O}(H^{\kappa+4}) . \end{aligned} \quad (3.17)$$

$C_{pos,N+1}^{\kappa}$  and  $C_{pos,N+1}^{\kappa+1}$  are the same error constants as in Section 3.1.2.

Applying a very similar calculation as in Section 3.1.2, an estimate  $\varepsilon_{i,N+1}^{pos}$  for the local error  $e_{i,N+1}^{pos} = |q_{i,N+1} - q_i(T_{N+1})|$  can be derived, which monitors the error of the corrected variables. From Eqs. (3.17) one simply obtains the error estimate

$$\varepsilon_{i,N+1}^{pos} = \left( 1 - \frac{C_{pos,N+1}^{\kappa+1}}{C_{pos,N+1}^{\kappa}} \right) \left| q_{i,N+1}^{pre} - q_{i,N+1} \right| , \quad (3.18)$$

which converges to the local error  $e_{i,N+1}^{pos}$  with order  $\mathcal{O}(H^{\kappa+4})$ .

On velocity level, one obtains the error estimate

$$\varepsilon_{i,N+1}^{vel} = \left( 1 - \frac{C_{vel,N+1}^{\kappa+1}}{C_{vel,N+1}^{\kappa}} \right) \left| v_{i,N+1}^{pre} - v_{i,N+1} \right| , \quad (3.19)$$

which approximates the local error  $e_{i,N+1}^{vel} = |v_{i,N+1} - v_i(T_{N+1})|$  of the corrected velocity variables with order  $\mathcal{O}(H^{\kappa+3})$ . The error constants  $C_{vel,N+1}^{\kappa}$  and  $C_{vel,N+1}^{\kappa+1}$  are the same as in Section 3.1.2 and are defined by Eq. (3.11).

### 3.1.5. Method i2: Local Extrapolation Based on the Coupling Variables (*imCV*)

Within this error estimation approach, only the error of the coupling variables is considered. Applying the implicit co-simulation scheme described in Section 2.4.2, the approximation polynomials  $\mathbf{p}_{N+1}^{pre}(t)$  of degree  $\kappa$  according to Eq. (2.24) are used for the subsystem integration within the predictor step. Hence, the predicted coupling variables  $\mathbf{u}_{N+1}^{pre}$  converge with order  $\mathcal{O}(H^{\kappa+1})$ .

The corrected state variables  $\mathbf{q}_{N+1}$  and  $\mathbf{v}_{N+1}$  will converge with orders  $\mathcal{O}(H^{\kappa+3})$  and  $\mathcal{O}(H^{\kappa+2})$  for both systems with feed-through and systems without feed-through. Hence, the local error  $e_{i,N+1}^u = \left| u_{i,N+1}^{pre} - u_i(T_{N+1}) \right|$  of the predicted coupling variables  $\mathbf{u}_{N+1}^{pre}$  can be estimated with the

help of the corrected coupling variables  $\mathbf{u}_{N+1}$ . Therefore, one obtains

$$\begin{aligned}
 e_{i,N+1}^u &= \left| u_{i,N+1}^{pre} - u_i(T_{N+1}) \right| \\
 &= \left| u_{i,N+1}^{pre} - (u_{i,N+1} + \mathcal{O}(H^{\kappa+2})) \right| \\
 &= \left| u_{i,N+1}^{pre} - u_{i,N+1} \right| + \mathcal{O}(H^{\kappa+2}) \\
 &= \varepsilon_{i,N+1}^u + \mathcal{O}(H^{\kappa+2}) ,
 \end{aligned} \tag{3.20}$$

where the error estimate  $\varepsilon_{i,N+1}^u = \left| u_{i,N+1}^{pre} - u_{i,N+1} \right|$  converges with order  $\mathcal{O}(H^{\kappa+2})$  to the local error  $e_{i,N+1}^u$ .

Compared to the error estimator of Section 3.1.4, the estimator according to Eq. (3.20) has different benefits and drawbacks, see the corresponding discussion in Section 3.1.3. A detailed comparison of the five error estimators based on a numerical study of the two-mass oscillator with different parameterizations can be found in Section 5.2.

### 3.2. Local Error Test

After the macro-step  $T_N \rightarrow T_{N+1}$  is accomplished, the local error of the step is estimated with one of the methods described in Section 3.1. Regardless of the used method, the error estimate  $\varepsilon_{N+1}$  measured in the weighted root mean square norm has to pass the local error test

$$\|\varepsilon_{N+1}\|_{WRMS} = \sqrt{\frac{1}{n_\varepsilon} \sum_{i=1}^{n_\varepsilon} (\varepsilon_{i,N+1} W_{i,N+1})^2} \leq 1 . \tag{3.21}$$

The weights  $W_{i,N+1}$  depend on the defined relative and absolute error tolerances  $\text{rtol}$  and  $\text{atol}_i$  and on the magnitude of the  $n_\varepsilon$  variables, which are considered for the error estimation. Within the scope of this work, basically two concepts are used:

- 1) The error of the state variables of the coupling bodies is estimated (*exLE*, *exMD*, *imMD*).
- 2) The error of the coupling variables is estimated (*exCV*, *imCV*).

Considering a co-simulation of the chain-structured multibody system introduced in Section 2.2, there are  $n_c$  coupling variables and  $2n_c$  coupling bodies.

Using concept 1), the error analysis is carried out on position level and on velocity level separately. This is reasonable, because the considered co-simulation methods have different convergence orders on position and on velocity level and the error estimates are used later on to control the macro-step size. The local error test for concept 1) reads

$$\max \left( \|\varepsilon_{N+1}^{pos}\|_{WRMS}, \|\varepsilon_{N+1}^{vel}\|_{WRMS} \right) \leq 1 \tag{3.22}$$

with the weights  $W_{i,N+1}^{pos} = 1/(\text{atol}_i^{pos} + \text{rtol} |x_{i,N+1}|)$  and  $W_{i,N+1}^{vel} = 1/(\text{atol}_i^{vel} + \text{rtol} |v_{i,N+1}|)$ . The index  $i = 1, \dots, 2n_c$  runs over all coupling bodies. It should be mentioned that the value  $\text{atol}_i$  can be chosen individually for each component, while  $\text{rtol}$  is identical for all components. As for numerical ODE solvers, the choice of adequate tolerances is problem dependent and is important for an efficient simulation.

Applying concept 2), the local error test reads

$$\|\epsilon_{N+1}^u\|_{WRMS} \leq 1 \quad (3.23)$$

with the weights  $W_{i,N+1}^u = 1 / (\text{atol}_i^u + \text{rtol} |u_{i,N+1}|)$ . The index  $i = 1, \dots, n_c$  runs over all coupling variables.

Depending on whether the error test is passed, i.e. condition (3.22) or (3.23) is fulfilled or not fulfilled, the macro-step to be carried out next is either the next macro-step  $T_{N+1} \rightarrow T_{N+2}$  or a repetition of the current macro-step  $T_N \rightarrow T_{N+1}$ . To keep the notation clear, the macro-step carried out next is denoted by  $T_M \rightarrow T_{M+1}$  with

$$T_M \rightarrow T_{M+1} := \begin{cases} T_{N+1} \rightarrow T_{N+2} & \|\epsilon_{N+1}\|_{WRMS} \leq 1 \\ T_N \rightarrow T_{N+1} & \|\epsilon_{N+1}\|_{WRMS} > 1 \end{cases}. \quad (3.24)$$

Shortly speaking, the index  $M$  is set to  $M := N + 1$  if the local error test is passed and to  $M := N$  if the local error test is failed.

### 3.3. Order Control Algorithm

The second task of the macro-step size and order control algorithm consists of the selection of the polynomial degree  $\kappa$  of the approximation polynomials (order selection) for the next macro-step  $T_M \rightarrow T_{M+1}$ . As already mentioned, depending on whether the current macro-step was accepted or discarded, the next macro-step is either a new macro-step or a repetition of the current macro-step.

Considering variable-order BDF solvers, the selection of the method order is made based on the leading order term in the remainder of a Taylor series expansion. The term is used as an estimation of the local truncation error. The local truncation error of a  $k^{\text{th}}$ -order method depends on the  $k + 1^{\text{st}}$  derivative of the solution vector  $z$  and the on solver step size  $h$ , according to  $\text{LTE}(k) = c_{k+1}h^{k+1}z^{(k+1)} + \mathcal{O}(h^{k+2})$  [SG75, ESF98]. The error constant  $c_{k+1}$  is method dependent. The method order for the next solver step is selected in order to maximize the step size. Therefore, the terms  $\|\text{LTE}(k-1)\| \approx \|c_k h^k z^{(k)}\|$ ,  $\|\text{LTE}(k)\| \approx \|c_{k+1} h^{k+1} z^{(k+1)}\|$  and  $\|\text{LTE}(k+1)\| \approx \|c_{k+2} h^{k+2} z^{(k+2)}\|$  are estimated. The order which yields the lowest estimated truncation error and therefore allows the largest step size, is used for the next step [RH93]. In addition, if a method of order  $k > 2$  is used, an algorithm to detect instabilities may be carried out. An illustrative explanation for the instability detection strategy is, that if the simulation becomes unstable, oscillations with high frequencies and growing amplitudes are expected to arise. As a consequence, the local truncation error oscillates rapidly with increasing magnitude when instabilities occur [Ske77].

An alternative order control strategy, which is implemented in the *IDA* solver [HBG<sup>+</sup>05], is presented in [BCP96]. The selection of the method order is also made based on the leading order term in the remainder of a Taylor series expansion, but the term is scaled to be independent of the error constant. Within this control algorithm, the terms  $\|h^{k-1} z^{(k-1)}\|$ ,  $\|h^k z^{(k)}\|$  and  $\|h^{k+1} z^{(k+1)}\|$  are compared. If the terms fail to form a decreasing sequence, i.e. if the condition  $\|h^{k-1} z^{(k-1)}\| > \|h^k z^{(k)}\| > \|h^{k+1} z^{(k+1)}\|$  is not fulfilled, the order is decreased to  $k - 1$ . If the

condition is fulfilled and if also  $\|h^{k+1}z^{(k+1)}\| > \|h^{k+2}z^{(k+2)}\|$ , the order is increased to  $k + 1$ . This order control strategy lowers the order sooner in the case of instabilities than the above described order control algorithm [BCP96].

The order control algorithm implemented here is an adaption of the order control strategy described in [BCP96] to the co-simulation environment. Within each macro-step, approximation polynomials  $P_j = P_j(t; [T_{M-j}, \mathbf{u}_{M-j}], \dots, [T_M, \mathbf{u}_M])$  of different degrees  $j = 1, \dots, \kappa + 1$  are considered. Next, the corresponding scaled derivative norm according to  $\text{SDN}(j) := \|H^j P_j^{(j)}\|$  of the polynomials is computed. The upper index in brackets denotes a differentiation with respect to the time, i.e.  $P_j^{(j)} = d^j P_j / dt^j$ ;  $H = T_{N+1} - T_N$  is the current macro-step size. The condition

$$\|H^1 P_1^{(1)}\| > \dots > \|H^{\kappa-1} P_{\kappa-1}^{(\kappa-1)}\| > \|H^\kappa P_\kappa^{(\kappa)}\| > \|H^{\kappa+1} P_{\kappa+1}^{(\kappa+1)}\| \quad (3.25)$$

must be fulfilled to continue the co-simulation with the current degree  $\kappa$  of the approximation polynomials. If the condition is not fulfilled, the polynomial degree is decreased to  $\kappa - 1$ . In addition, if condition (3.25) is fulfilled and if at least  $\kappa + 1$  consecutive macro-time steps have been carried out successfully with degree  $\kappa$ , the term  $\|H^{\kappa+2} P_{\kappa+2}^{(\kappa+2)}\|$  is computed. The polynomial degree is increased to  $\kappa + 1$ , if  $\|H^{\kappa+1} P_{\kappa+1}^{(\kappa+1)}\| > \|H^{\kappa+2} P_{\kappa+2}^{(\kappa+2)}\|$ . Listing 3.1 shows the general order control strategy in pseudo code.

```

{...}      /* compute SDN[1], SDN[2], ..., SDN[κ+1] */
κnew = κ;

/* decreasing the order if necessary */
while (κnew > 0 && NOT((SDN[κnew+1] < SDN[κnew]) < ... < SDN[1]))
    κnew = κnew - 1;

/* an increase of κ is only taken into consideration if the κ+1 directly
   preceding macro-steps have been carried out with order κ successfully */
if ((κ < κmax) && (Nsucc > κ) && (κnew == κ)) {
    {...}      /* compute SDN[κ+2] */
    /* increasing the order if possible */
    if (SDN[κnew+2] < SDN[κnew+1])
        κnew = κnew + 1;
}
κ = κnew;
    
```

**Listing 3.1:** General order control strategy.

The actual order control algorithm used for the simulations in this work is inspired by the order control strategy in *IDA* [HBG<sup>+</sup>05] and selects the polynomial degree  $\kappa$  according to the following scheme:

- Check if order has to be decreased:
  - If  $\kappa = 1$  and  $\text{SDN}(1) \leq \frac{1}{2}\text{SDN}(2)$ :
 
$$\kappa_{\text{new}} = 0$$
  - If  $\kappa > 1$  and  $\max(\text{SDN}(\kappa), \text{SDN}(\kappa - 1)) \leq \text{SDN}(\kappa + 1)$ :
 
$$\kappa_{\text{new}} = \kappa - 1$$

- Else:
  - $\kappa_{new} = \kappa$
  - If the number of consecutive failed steps  $N_{fail} > 3$ :
    - $\kappa_{new} = \min(\kappa_{new}, 1)$
  - If the number of consecutive failed steps  $N_{fail} > 5$ :
    - $\kappa_{new} = 0$
- If the number  $N_{succ}$  of consecutive successful macro-steps with constant order  $\kappa$  and without reducing the macro-step size fulfills  $N_{succ} > \kappa$ , then  $SDN(\kappa + 2)$  is computed and an adjustment of the order is considered:
  - If  $\kappa = 0$  and  $SDN(2) < \frac{1}{2}SDN(1)$ :
    - $\kappa_{new} = 1$
  - If  $\kappa > 0$  and  $SDN(\kappa) \leq \min(SDN(\kappa + 1), SDN(\kappa + 2))$ :
    - $\kappa_{new} = \kappa - 1$
  - Else if  $\kappa > 0$  and  $SDN(\kappa + 2) < SDN(\kappa + 1)$ :
    - $\kappa_{new} = \kappa + 1$

Listing 3.2 shows the implemented order control algorithm as pseudo code.

```

{...}      /* compute SDN[κ-1], SDN[κ], SDN[κ+1] */
κnew = κ;

/* decreasing the order if necessary */
if (κnew == 1 && SDN[1] <= SDN[2] / 2.)
    κnew = 0;
else if (κnew > 1 && max(SDN[κnew], SDN[κnew-1]) <= SDN[κnew+1])
    κnew = κnew - 1;

if (Nfail > 3)
    κnew = min(1, κnew);
if (Nfail > 5)
    κnew = 0;

/* increasing the order if possible */
if ((κ < κmax) && (Nsucc > κ) && (κnew == κ)) {
    {...}      /* compute SDN[κ+2] */
    if (κnew == 0 && SDN[2] < SDN[1] / 2.)
        κnew = 1;
    else if (κnew > 0) {
        if (SDN[κnew] <= min(SDN[κnew+1], SDN[κnew+2]))
            κnew = κnew - 1;
        else if (SDN[κnew+2] < SDN[κnew+1])
            κnew = κnew + 1;
    }
}
κ = κnew;

```

**Listing 3.2:** Order control strategy adapted from *IDA* [HBG<sup>+</sup>05].

### 3.4. Macro-Step Size Controller

After the polynomial degree  $\kappa$  for the next macro-step has been determined, the algorithm for the macro-step size selection is executed. The macro-step size controller computes a (positive, real-valued) scaling factor  $r$  for calculating the new macro-step size, i.e.  $H_{new} = rH$ , for the next macro-step  $T_M \rightarrow T_{M+1}$  (see Eq. (3.24)). The calculation is based on the estimated error  $\varepsilon_{N+1}$ . The idea is to select an optimal value for the new macro-step size  $H_{new}$  so that if the accomplished macro-step  $T_N \rightarrow T_{N+1}$  had been carried out with  $H_{new}$ , the resulting estimated error  $\varepsilon_{N+1}$  would have fulfilled the local error condition (3.21) exactly, i.e.  $\|\varepsilon_{N+1}\|_{WRMS} = 1$ .

The determination of the macro-step size for the next macro-step  $T_M \rightarrow T_{M+1}$  based on the estimated error  $\varepsilon_{N+1}$  is only appropriated, if the next macro-step  $T_M \rightarrow T_{M+1}$  is carried out with the same polynomial degree  $\kappa$  as the current macro-step  $T_N \rightarrow T_{N+1}$ . Therefore two cases have to be distinguished:

- 1) The two consecutive macro-steps  $T_N \rightarrow T_{N+1}$  and  $T_M \rightarrow T_{M+1}$  are carried out with the same polynomial degree.
- 2) The polynomial degree is changed by the order control algorithm after the macro-step  $T_N \rightarrow T_{N+1}$  has been accomplished.

Considering case 1), the error estimate  $\varepsilon_{N+1}$  (see Section 3.1), which has been computed for the local error test (3.21), is also utilized for the calculation of the new macro-step size.

In case 2), the error estimate  $\varepsilon_{N+1}$  cannot be used to calculate the step size of the next macro-step. Therefore, the alternative error estimate

$$\begin{aligned} \tilde{\varepsilon}_{M+1}^u &= \mathbf{P}_{\kappa+1}(T_{M+1}) - \mathbf{P}_{\kappa}(T_{M+1}) \\ &= \frac{1}{(\kappa+1)!} \prod_{j=1}^{\kappa+1} (T_{M+1} - T_{M+1-j}) \mathbf{P}_{\kappa+1}^{(\kappa+1)} \end{aligned} \quad (3.26)$$

for the next macro-step  $T_M \rightarrow T_{M+1}$  has to be computed *a priori*. The two vectors  $\mathbf{P}_{\kappa+1} = \mathbf{P}_{\kappa+1}(t; [T_{M-\kappa-1}, \mathbf{u}_{M-\kappa-1}], \dots, [T_M, \mathbf{u}_M])$  and  $\mathbf{P}_{\kappa} = \mathbf{P}_{\kappa}(t; [T_{M-\kappa}, \mathbf{u}_{M-\kappa}], \dots, [T_M, \mathbf{u}_M])$  collect the approximation polynomials of degrees  $\kappa$  and  $\kappa+1$ . The upper index in brackets denotes a differentiation with respect to the time, i.e.  $\mathbf{P}_{\kappa+1}^{(\kappa+1)} = d^{\kappa+1} \mathbf{P}_{\kappa+1} / dt^{\kappa+1}$ . Note that the quantity which is estimated by  $\tilde{\varepsilon}_{M+1}^u = e_{M+1}^u + \mathcal{O}(H^{\kappa+2})$  is the local error of the approximation polynomials (predictor) of the coupling variables.

*Remark:* The error estimate  $\tilde{\varepsilon}_{M+1}^u$  is only used if the approximation order has changed, in order to calculate the new macro-step size. After the macro-step  $T_M \rightarrow T_{M+1}$  has been accomplished with this new step size, the error of the macro-step is again estimated with the *a posteriori* error estimator  $\varepsilon_{M+1}$  of Section 3.1. Hence, the acceptance of a macro-step (see Eq. (3.21)) is always based on an *a posteriori* error estimator. Note that the *a priori* estimator only monitors the error of the coupling variables. Using the error estimators *exLE*, *exMD*, *imMD*, however, the error of the state variables is monitored. As a consequence, the *a priori* estimator and the *a posteriori* estimator may be based on different variables.

The scaling factor  $r$  for the new (optimal) macro-step size  $H_{new}$  is obtained by one of the follow-

ing relations

$$\begin{aligned}
 r^{pos} &= \left( \Gamma \cdot \|\epsilon_{N+1}^{pos}\|_{WRMS} \right)^{\frac{-1}{\kappa+3}} ; & r^{vel} &= \left( \Gamma \cdot \|\epsilon_{N+1}^{vel}\|_{WRMS} \right)^{\frac{-1}{\kappa+2}} \\
 r^u &= \left( \Gamma \cdot \|\epsilon_{N+1}^u\|_{WRMS} \right)^{\frac{-1}{\kappa+1}} ; & \tilde{r}^u &= \left( \Gamma \cdot \|\tilde{\epsilon}_{M+1}^u\|_{WRMS} \right)^{\frac{-1}{\kappa+1}} ,
 \end{aligned} \tag{3.27}$$

depending on the computed error estimate. The exponents in Eq. (3.27) refer to the convergence order of the co-simulation method, which is  $\kappa + 3$  on position level and  $\kappa + 2$  on velocity level for the here considered explicit and implicit co-simulation approaches. The coupling variables converge with order  $\kappa + 1$ .  $\Gamma$  is a user-defined safety factor. Common values for the safety factor are  $\Gamma \in [2, 6]$ . The choice of the safety factor  $\Gamma$  is a trade-off between an increased number of macro-step repetitions caused by local error test fails for a low value of  $\Gamma$  and an unnecessary small macro-step size for a high value of  $\Gamma$ . A numerical study on the safety factor can be found in Section 5.3. Most of the simulations for this work have been carried out with  $\Gamma = 6$ .

If both, the error of the position variables and the error of the velocity variables is estimated (*exLE*, *exMD*, *imMD*), the scaling factor is determined (conservatively) according to  $r = \min(r^{pos}, r^{vel})$ . Summarizing, the scaling factor  $r$  is determined, depending on the employed error estimator and whether  $\kappa$  is changed or not, according to

$$r = \begin{cases} \min(r^{pos}, r^{vel}) & \text{error estimation of position/velocity variables (exLE, exMD, imMD)} \\ r^u & \text{error estimation of the coupling variables (exCV, imCV)} \\ \tilde{r}^u & \text{change in the order of the approx. polynomials } (\kappa_{new} \neq \kappa) . \end{cases} \tag{3.28}$$

In addition, depending on the number of consecutive failed macro-steps  $N_{fail}$ , the following limitations for the scaling factor  $r$  are used:

- $N_{fail} = 0$ :
  - If  $r \geq r_{max}$ :  $r = r_{max}$
  - If  $r < 1$ :  $r = \min(0.9, \max(r_{min}, r))$
  - Else:  $r = 1$
- $N_{fail} = 1$ :
  - $r = \min(0.9, \max(0.25, 0.9 \cdot r))$
- $N_{fail} > 1$  or corrector failed to converge (implicit co-simulation):
  - $r = 0.25$

There are upper and lower bounds  $r_{max}$  and  $r_{min}$  for the scaling of the macro-step size between two consecutive macro-steps, e.g.  $r_{max} = 2.0$  and  $r_{min} = 0.5$ . In addition, a so called *dead-zone* is defined to avoid many small changes in the macro-step size, i.e. the macro-step size is not changed if  $1.0 \leq r < r_{max}$ . ODE solvers based on multi-step methods typically use a dead-zone to produce sequences of steps with a constant step size, because step size changes may entail additional time consuming computations (e.g. refactorization of the Jacobian). For the co-simulation approaches considered here, additional computations in connection with macro-step size changes do not occur. However, numerical tests indicate that the use of a dead-zone tends

to reduce the number of failed macro-steps during the simulation process. The application of a dead-zone is – just like the safety factor  $\Gamma$  – a trade-off between increasing the macro-step size and decreasing the number of repeated macro-steps.

In case of a macro-step repetition ( $N_{fail} = 1$ ), the macro-step size is reduced at least by the factor 0.9. Moreover, the lower bound  $r_{min}$  is reduced to 0.25 and the calculated value of  $r$  is additionally scaled with 0.9. If the local error test is not passed on the second (or subsequent) attempt ( $N_{fail} > 1$ ), the scaling factor  $r$  is set to 0.25, because it is assumed that the computed value based on the estimated error is of poor quality. If a macro-step has to be repeated because the corrector iteration fails to converge (implicit co-simulation), the scaling factor is also set to  $r = 0.25$ . The implemented strategy for controlling the macro-step size is given in Listing 3.3 as pseudo code.

```

if ( $\kappa_{new} == \kappa$ )
  {...}      /* compute  $\varepsilon_{N+1}$  depending on the employed error estimator */
else
  {...}      /* compute  $\tilde{\varepsilon}_{M+1}^u$  according to Eq. (3.26) */

{...}      /* determine  $r$  according to Eq. (3.27) */

if ( $N_{fail} == 0$ ) {
  if ( $r \geq r_{max}$ )
     $r = r_{max}$ ;
  else if ( $r < 1.0$ )
     $r = \min(0.9, \max(r_{min}, r))$ ;
  else
     $r = 1.0$ ; /* dead-zone:  $1.0 \leq r < r_{max}$  */
}
if ( $N_{fail} == 1$ )
   $r = \min(0.9, \max(0.25, 0.9 * r))$ ;
if ( $N_{fail} > 1 \parallel corConvFail == 1$ )
   $r = 0.25$ ;

 $H_{new} = r * H$ ;

```

**Listing 3.3:** Macro-step size control strategy.

The co-simulation starts with a user defined initial macro-step size  $H_{init}$  and  $\kappa = 0$ . During an initial phase, the macro-step size is doubled and the polynomial degree  $\kappa$  is increased by 1 within each macro-step. The initial phase is terminated in any of the following cases: the maximum value of  $\kappa$  is reached, the local error test is not passed, or the macro-step size or  $\kappa$  have to be reduced.

In [GLS88], a PI-controller instead of an I-controller is suggested to adjust the step size of ODE solvers. However, many commonly used solvers, for example *ode15s* and *ode45* of the MATLAB *ODE suite* [SR97] or *IDA* and *cvode* from SUNDIALS [HBG<sup>+</sup>05] use an I-controller. Numerical tests with the implemented co-simulation approaches do not show any benefit of using a PI-controller. Therefore, only an I-controller is applied for adjusting the macro-step size.

An overview of the integration step size control strategies of different numerical solvers of the MATLAB *ODE suite* [SR97] and SUNDIALS [HBG<sup>+</sup>05] is given in Fig. 3.2. Note that the index  $\|\cdot\|_{WRMS}$  of the norm of the estimated local error  $\varepsilon$  is omitted in the figure. It turns out, that the

*IDA* solver (SUNDIALS) and *ode15i* (MATLAB) have almost the same step size control strategy. Another interesting observation is that the different solvers use different definitions of the safety factor. The last column shows the behavior of the solvers in the case of a convergence error of the nonlinear system solution.

	solver	$N_{fail} = 0$	$N_{fail} = 1$	$N_{fail} = 2$	$N_{fail} \geq 3$	conv. fail
matlab	ode45	$r = \frac{1.0}{1.25 \cdot \ \boldsymbol{\varepsilon}\ ^{\frac{1}{5}}}$ $r_{max} = 5.0$	$r = \frac{1.0}{1.25 \cdot \ \boldsymbol{\varepsilon}\ ^{\frac{1}{5}}}$ $r_{min} = 0.1$	$r = 0.5$	$r = 0.5$	
	ode15s	$r = \frac{1.0}{1.2 \cdot \ \boldsymbol{\varepsilon}\ ^{\frac{1}{k+1}}}$ $r_{max} = 10.0$	$r = \frac{1.0}{1.2 \cdot \ \boldsymbol{\varepsilon}\ ^{\frac{1}{k+1}}}$ $r_{min} = 0.1$	$r = 0.5$	$r = 0.5$	$r = 0.3$
	ode15i	$r = \frac{1.0}{(2.0 \cdot \ \boldsymbol{\varepsilon}\ )^{\frac{1}{k+1}}}$ $r_{min} = 0.5$ $r_{max} = 2.0$ dead-zone: (1.0, 2.0) if $r < 1$ : $r_{max} = 0.9$	$r = \frac{0.9}{(2.0 \cdot \ \boldsymbol{\varepsilon}\ )^{\frac{1}{k+1}}}$ $r_{min} = 0.25$	$r = 0.25$	$r = 0.25$	$r = 0.25$
sundials	cvode	$r = \frac{1.0}{(6.0 \cdot \ \boldsymbol{\varepsilon}\ )^{\frac{1}{k+1}}}$ $r_{max} = 10.0$ dead-zone: (1.0, 1.5)	$r = \frac{1.0}{(6.0 \cdot \ \boldsymbol{\varepsilon}\ )^{\frac{1}{k+1}}}$ $r_{max} = 10.0$ dead-zone: (1.0, 1.5)	$r = \frac{1.0}{(6.0 \cdot \ \boldsymbol{\varepsilon}\ )^{\frac{1}{k+1}}}$ $r_{max} = 0.2$	$r = \frac{1.0}{(6.0 \cdot \ \boldsymbol{\varepsilon}\ )^{\frac{1}{k+1}}}$ $r_{min} = 0.1$ $r_{max} = 0.2$	$r = 0.25$
	ida	$r = \frac{1.0}{(2.0 \cdot \ \boldsymbol{\varepsilon}\ )^{\frac{1}{k+1}}}$ $r_{min} = 0.5$ $r_{max} = 2.0$ dead-zone: (1.0, 2.0) if $r \leq 1$ : $r_{max} = 0.9$	$r = \frac{0.9}{(2.0 \cdot \ \boldsymbol{\varepsilon}\ )^{\frac{1}{k+1}}}$ $r_{min} = 0.25$ $r_{max} = 0.9$	$r = 0.25$	$r = 0.25$	$r = 0.25$

**Figure 3.2:** Computation of the scaling factor  $r$  of different numerical solvers of the MATLAB *ODE suite* ([SR97], [MAT17]) and SUNDIALS ([HBG<sup>+</sup>05], [HSR04], [HSC20]) depending on the number of consecutive local error test misses  $N_{fail}$ .

## 4. On the Computational Efficiency of Co-Simulation Approaches

Within the following section, the implementation of the co-simulation approaches introduced in Section 2.4 will be discussed. The primary focus will be on the question, how the overall computation time of a co-simulation model can be reduced. It should be mentioned that the term computation time always refers to the wall-clock time and not to the CPU time within this work. All considerations are made under the assumption that the co-simulation model is computed on a machine with a sufficient number of cores to fully parallelize the simulation. The numerical studies for this work were conducted on the *Lichtenberg high performance computer* of the TU Darmstadt.

**Table 4.1:** Required number of cores for full parallelization.

method	$n_{\text{cores}}$
<i>exH</i>	$n_s + 1$
<i>exLE</i>	$2n_s + 1$
<i>exMD</i>	$2n_s + 1$
<i>exCV</i>	$n_s + 1$
implicit	$n_s + \sum_{L=1}^{n_s} ({}^L n_u) + 1$
waveform	$n_s + 1$

The required numbers of cores for a full parallelization of the different co-simulation approaches are listed in Table 4.1. On the one hand, the number of cores depends on the considered co-simulation scheme. In connection with the explicit co-simulation method, the number of cores depends also on the applied error estimator, if a macro-step size controller is used. For a fully parallelized simulation of the explicit method in connection with an equidistant macro-time grid (*exH*) one core per subsystem and one core for the co-simulation interface are required. The same requirements are necessary for the explicit co-simulation with a controlled macro-step size, if the error estimator *exCV* is used. If an error estimator based on the state variables (*exLE* or *exMD*) is applied, then two cores per subsystem are necessary (see Section 3.1). A simulation with the implicit co-simulation approach requires, independent of the macro-step size controller,  $1 + {}^L n_u$  cores per subsystem, where  ${}^L n_u$  is the number of coupling variables of each subsystem  $L$ . The increased number of cores is necessary because of the numerical computation of the interface Jacobian by finite differences and the therefore required additional subsystem integrations with perturbed coupling variables (see Section 2.4.2). An alternative approach to approximate the interface Jacobian without additional subsystem integrations is discussed in Appendix A. If the waveform relaxation method is used, one core per subsystem and one core for the co-simulation interface are sufficient.

Applying a parallel co-simulation implementation, the computation time for large-scale systems is usually strongly reduced compared to the monolithic computation. The computation time of the explicit co-simulation approach can be estimated by

$$\begin{aligned}
 T_{\text{cosim}}^{\text{expl}} &= \frac{T_{\text{monolith}}}{n_s^P} \cdot [1 + F_{\text{solv}}(\bar{H}, \kappa, \dots)] \\
 &+ N_{\text{mac}} \cdot [O_{\text{dump}}(n_K/n_s) + O_{\text{cosim}}(n_s, n_c) + O_{\text{mpi}}(n_s, n_c)],
 \end{aligned} \tag{4.1}$$

where  $T_{\text{monolith}}$  denotes the computation time of the monolithic simulation and  $N_{\text{mac}}$  denotes the total number of macro-steps. The variables  $n_s$  and  $n_c$  denote the number of subsystems and the number of coupling variables.  $P$  represents the scaling order of the computation time of the multibody implementation with respect to the degree of freedom. For typical multibody systems,

the value of  $P$  is between one and three, depending on the formulation of the equations of motion and the solving strategy.

The factor  $F_{\text{solv}}$  describes the deviation of the subsystem integration time from the theoretical value  $T_{\text{monolith}}/n_s^P$ . In fact, deviations in both directions may occur. On the one hand, the average micro-step size of the subsystem integrations can be reduced because of the limitation by the macro-step size or because of jumps in the approximation polynomials of the coupling variables (see Section 4.4). In this case, the subsystem integration time will be longer than estimated ( $F_{\text{solv}} > 0$ ). On the other hand, the average subsystem solver step size can be larger than the integration step size of the monolithic computation because of multirate effects. Then  $F_{\text{solv}} < 0$  and the subsystem integration time will be shorter than estimated. The factor  $F_{\text{solv}}$  is influenced by many parameters, for example by the average macro-step size  $\bar{H}$ , by the degree  $\kappa$  of the approximation polynomials and by the structure and the parameterization of the considered model.

Numerical studies have shown that the subsystem solver characteristics (number of solver steps, number of Jacobian evaluations, number of nonlinear solver iterations, ...) may be better than the corresponding characteristics of the monolithic computation, but still the computation time of the subsystem solver takes longer than estimated ( $F_{\text{solv}} > 0$ ). This effect can be explained from a computational point of view: the memory management of the solver of the monolithic simulation is more efficient because the solver integrates the equations of motion from the start to the end without interruption. Therefore, the chances are good that the variables are stored in cache and can be accessed in a very fast way. Considering a co-simulation approach, the solver is stopped after each macro-step and other computations are carried out (e.g. solver workspace dump). As a consequence, the variables accessed by the solver may be transferred from cache to main memory and back again, when the solver resumes the subsystem integration. This process decreases the computational efficiency of the solver.

The different sources of computational overhead denoted by the colored  $O(\dots)$  terms in Eq. (4.1), e.g. data traffic between the co-simulation interface and the subsystems, will be explained within the following subsections. The estimation formula for  $T_{\text{cosim}}^{\text{expl}}$  implies the assumption that the overall system is split into equal-sized subsystems, so that the integration times for the different subsystems are similar. Neglecting the computational overhead, a simplified estimation for the computation time of the explicit co-simulation approach can be made according to

$$T_{\text{cosim}}^{\text{expl}} \approx \frac{T_{\text{monolith}}}{n_s^P}. \quad (4.2)$$

The computation time of the implicit co-simulation method can be estimated analogically by

$$T_{\text{cosim}}^{\text{impl}} = (1 + \bar{n}_{\text{cor}}) \cdot \left\{ \frac{T_{\text{monolith}}}{n_s^P} \cdot \left[ 1 + \tilde{F}_{\text{solv}}(\bar{H}, \kappa, \dots) \right] + N_{\text{mac}} \cdot \left[ \tilde{O}_{\text{dump}}(n_K/n_s) + \tilde{O}_{\text{cosim}}(n_s, n_c) + \tilde{O}_{\text{mpi}}(n_s, n_c) \right] \right\} \quad (4.3)$$

with the average number of corrector iterations per macro-step  $\bar{n}_{\text{cor}}$ . The simplified estimation of the computation time of the implicit co-simulation approach is

$$T_{\text{cosim}}^{\text{impl}} \approx (1 + \bar{n}_{\text{cor}}) \cdot \frac{T_{\text{monolith}}}{n_s^P} \quad (4.4)$$

assuming the overall system is decomposed into subsystems of equal size.

It should be mentioned that the numerical studies in this work are carried out with an efficient (subsystem) solver (*IDA* [HBG<sup>+</sup>05]), which exploits the sparsity of the system matrices of the considered multibody systems. The computation time of the monolithic simulations scales approximately linear with the number of degrees of freedom ( $P \approx 1$ ). Therefore, the resulting speed-up factors of the co-simulation approaches compared to the monolithic computation are rather conservative. Considering other models/solvers with  $P > 1$ , the speed-up factors would increase significantly, as demonstrated in Section 5.7.

#### 4.1. Parallel Implementation of the Co-Simulation Approaches

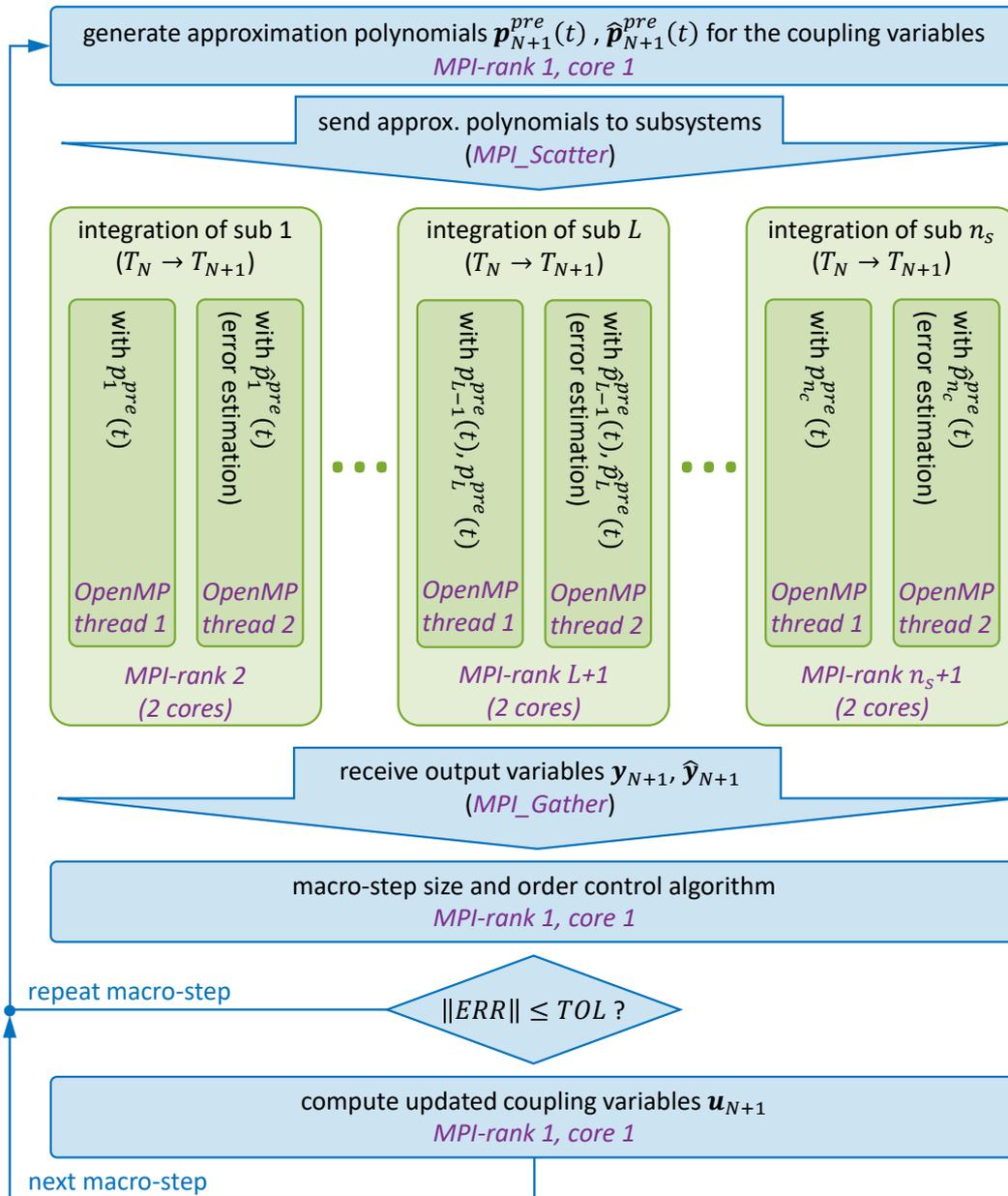
The implemented co-simulation interface and each subsystem are stand-alone C-programs. The communication between the interface and the subsystems is managed over the *Message Passing Interface v3.1 (MPI-3.1)*, in particular the INTEL MPI LIBRARY [int18] is used. Each subsystem is executed by a MPI-rank; if multiple integrations of the same subsystem within a macro-time step are required, e.g. to numerically compute the interface Jacobian or for the error estimation, these are parallelized via OpenMP [Ope15]. All parallel threads are synchronized at the macro-time points. Between the macro-time points, the subsystems are integrated independently of each other, since a weak coupling approach is considered here. A flowchart of the parallel implementation of the explicit co-simulation method with the macro-step size and order control algorithm is shown in Fig. 4.1, a corresponding flowchart of the implicit co-simulation method is shown in Fig. 4.2.

#### 4.2. Subsystem Solver

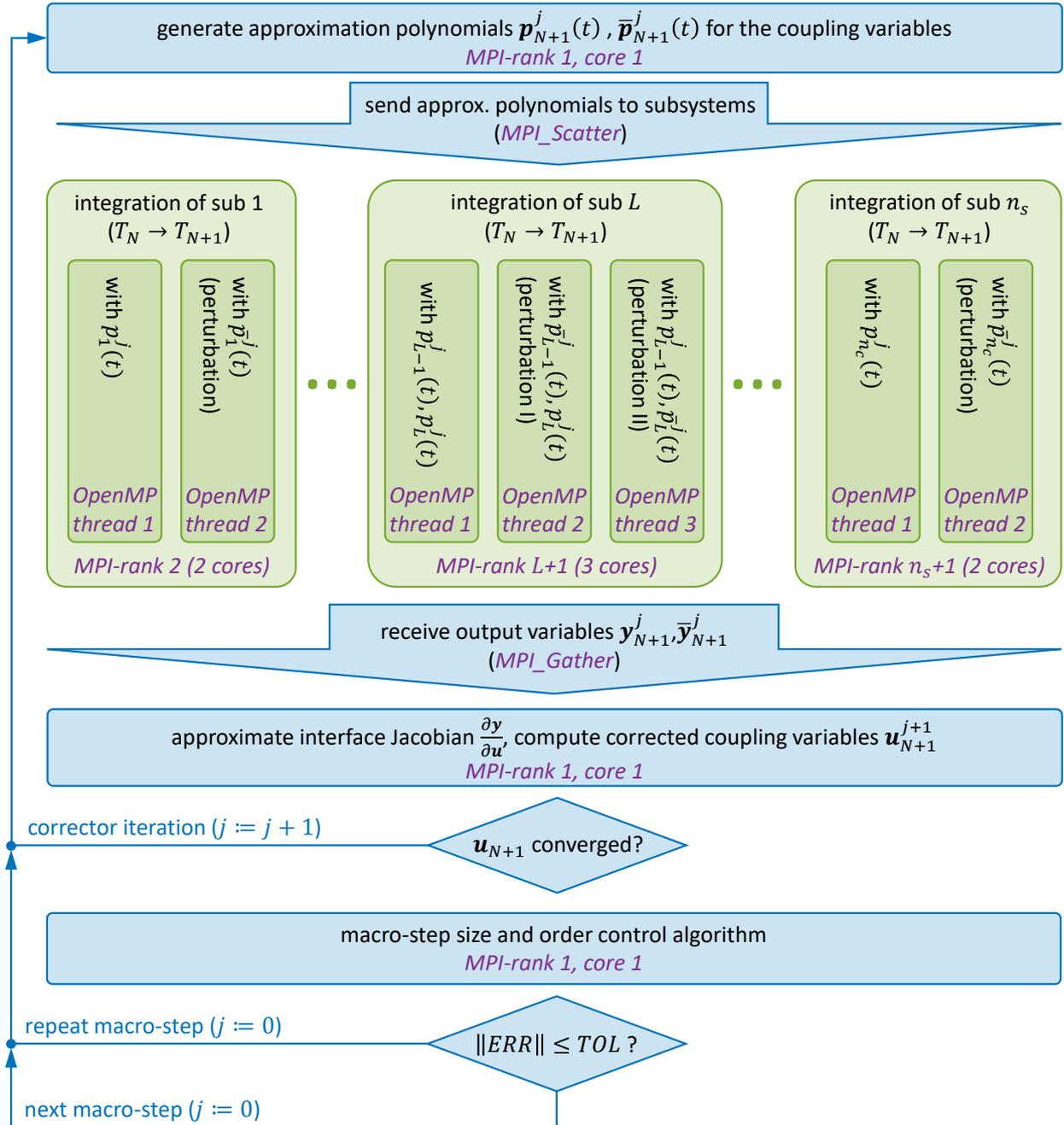
The subsystems are solved with the *IDA* solver from the SUNDIALS (Suite of Nonlinear and Differential/Algebraic Equation Solvers [HBG<sup>+</sup>05]) package. This implicit DAE solver is based on a variable-order variable-coefficient BDF implementation combined with either direct (sparse) or iterative methods for solving the linear system within the Newton iteration. For the present studies, the direct sparse linear solver (*KLU* [DPN10]) is used. The *IDA* solver is open-source and gives the user full access to all relevant data. This is an important point for implementing co-simulation methods which require a macro-step repetition, for example an implicit method or a co-simulation approach with variable macro-step size. Macro-step repetitions are implemented here by copying and reloading the internally used solver workspace (solver workspace dump). The solver workspace structure contains all relevant data used by the BDF method and also the data of the linear solver. This allows the solver to be restarted at a certain point without any initialization or additional starting procedures.

#### 4.3. Micro-Step Size Limitation

The micro-step size  $h_{mic}$  is determined by the subsystem solver and is therefore individual for each subsystem ( ${}^L h_{mic}$  for arbitrary subsystem  $L$ ). Most of the commonly used ODE/DAE solvers have a variable solver step size, i.e. at each integration step the local error is estimated and the step size is maximized under the constraint that the estimated error is kept below a certain



**Figure 4.1:** Flow chart of the parallel implementation of the explicit co-simulation method (*exLE* or *exMD*) with variable macro-step size based on the multibody system described in Section 2.2.

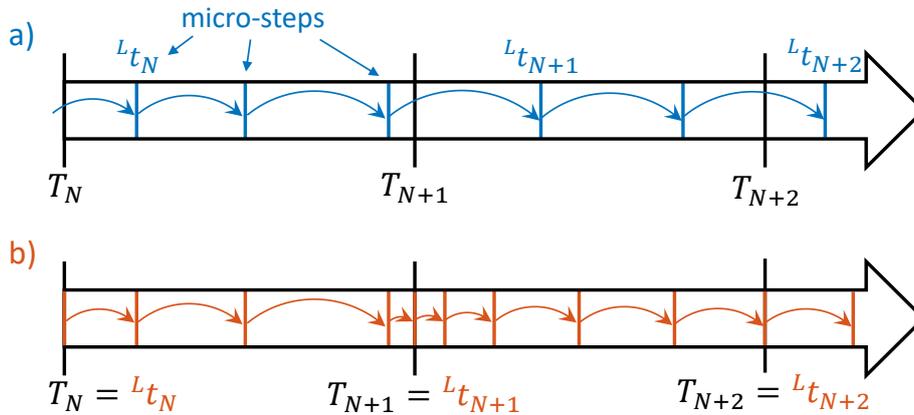


**Figure 4.2:** Flow chart of the parallel implementation of the implicit co-simulation method (*imCV*) with variable macro-step size based on the multibody system described in Section 2.2.

tolerance. Considering an arbitrary co-simulation step from  $T_N$  to  $T_{N+1}$ , a subsystem solver with a variable solver step size  $h_{mic}$  does in general not integrate to  $T_{N+1}$  exactly, but will stop as soon as it passes  $T_{N+1}$ . In general, there are two possible ways of proceeding, if a co-simulation model contains subsystems with a variable solver step size:

- a) Each subsystem solver is stopped, when the solver steps beyond a macro-time point ( ${}^L t_{N+1} \geq T_{N+1}$  for arbitrary subsystem  $L$ ); the values of the subsystem output variables  $\mathbf{y}_{N+1}$  at  $T_{N+1}$  are then approximated by interpolation polynomials. In addition, the restriction  $h_{mic} \leq H$  for the micro-step size is applied to prevent the subsystem solver from skipping macro-steps.
- b) Each subsystem solver is forced to stop the integration at the macro-time point exactly ( ${}^L t_{N+1} = T_{N+1}$ ).

The two options are shown schematically in Fig. 4.3 for an arbitrary subsystem  $L$ . The advantage of option a) is that the step size controller of the subsystem solver does not get affected by the co-simulation. It is also simple to implement and does not have any special requirements on the subsystem solver. The approximation of the subsystem output variables is carried out with the order that is currently used by the BDF method of the subsystem solver to obtain sufficiently accurate results. The restriction  $h_{mic} \leq H$  can also be extended by a multirate factor  $n$  according to  $h_{mic} \leq H/n$ , to ensure that each subsystem is evaluated at least  $n$  times within each macro-step. The disadvantage of this approach is that the subsystem solver starts the integration of the next macro-step  $T_{N+1} \rightarrow T_{N+2}$  not at  $T_{N+1}$  as intended, but at  ${}^L t_{N+1} \geq T_{N+1}$ . The interval  $[T_{N+1}, {}^L t_{N+1}] \subset [T_{N+1}, T_{N+2}]$  is therefore integrated by using outdated values of the coupling variables, namely with the approximation polynomials  $\mathbf{p}_{N+1}^{pre}(t)$  from the preceding macro-step  $T_N \rightarrow T_{N+1}$ . This results in a discontinuity in the approximation polynomials at  ${}^L t_{N+1}$  and is an additional source of numerical errors. However, numerical studies carried out with the multibody model introduced in Section 2.2 suggest that option a) including the restriction  $h_{mic} \leq H$  is a good compromise for explicit co-simulation implementations with an equidistant macro-time grid.



**Figure 4.3:** Micro-step restrictions: a) micro-step size limited by the macro-step size  $h_{mic} \leq H$ , b) integration is stopped at each macro-time point exactly.

For co-simulation approaches using a macro-step size controller, which is based on an error estimator or if macro-step repetitions are permitted, the restriction  $h_{mic} \leq H$  is not sufficient to ensure a robust co-simulation process with accurate results. The fact that the subsystem solver

does not start each macro-step at its actual starting point may cause several problems. The most obvious one appears in the case of macro-step repetitions. Assuming the last micro-step in subsystem  $L$  within the macro-step  $T_N \rightarrow T_{N+1}$  is carried out with a very large micro-step size ( ${}^L h_{mic} \approx H$ ), then the subsystem solver is stopped at the time point  ${}^L t_{N+1}$  which can be close to  $T_{N+2}$ . If the next macro-step  $T_{N+1} \rightarrow T_{N+2}$  does not pass the local error test (Eq. (3.22)) and will therefore be repeated with a reduced macro-step size, the starting point  ${}^L t_{N+1}$  of the integration of subsystem  $L$  lies potentially out of the new macro-time interval  $T_{N+1} \rightarrow T_{N+2}^* < T_{N+2}$  with  ${}^L t_{N+1} > T_{N+2}^*$ . Subsystem  $L$  is then not reevaluated and the subsystem states remain unchanged. In addition, the error estimators described in Section 3.1 may give inaccurate results if the evaluated interval differs from the actual macro-step.

Another issue is that solvers may assume that the limit  $h_{max}$  in the restriction  $h_{mic} \leq h_{max}$  of the solver step size is a constant value and is not changed during the integration process. Therefore, the solver checks the condition  $h_{mic} \leq h_{max}$  only when the solver step size is increased and not if  $h_{mic}$  remains constant between two or more solver steps. Considering a co-simulation approach with a variable macro-step size  $H$  and the restriction  $h_{mic} \leq H$ , it is possible that the macro-step size  $H$  is reduced by the macro-step size controller to a value, which is smaller than the current micro-step size of one of the subsystems. However, the subsystem solver does not check the condition  $h_{mic} \leq H$  within the following micro-step(s) and will therefore continue the integration with  $h_{mic} > H$ .

Restricting the subsystem solver step size according to option b) (exact stop) has the advantage that each subsystem is integrated as desired from  $T_N$  to  $T_{N+1}$ , but it has also several disadvantages: the essential point is that the employed subsystem solver has to provide the functionality to exactly stop the integration process at a certain time point. Even if this kind of functionality is supported, it reduces the efficiency of the solver significantly if an exact solver stop is enforced within each macro-step. Each time the subsystem solver is forced to stop at a certain time point, the step size controller of the solver is interrupted. This may lead to a significant increase in the number of micro-steps. The frequent changes in the micro-step size caused by the exact stop function can lead to additional expensive computational operations of the subsystem solver, for example reevaluations of the Jacobian matrix.

For a reliable co-simulation implementation where macro-step repetitions occur, it must be ensured that the subsystem solver integrates to the end  $T_{N+1}$  of each macro-step exactly. In addition, for reasons of computational efficiency, the interruption of the step size controller of the subsystem solver to achieve this exact integration stop should be as tentative as possible. Therefore, the following function to (carefully) adjust the micro-step size according to Listing 4.1 is implemented.

```
tLeft = TN+1 - t;           /* t: current subsystem solver position */
if (tLeft < 1.1 * hmic)
    hmic = tLeft;           /* stretch next micro-step */
else if (tLeft < 2. * hmic)
    hmic = tLeft / 2.;     /* reduce micro-step size */
```

**Listing 4.1:** Micro-step size adjustment function.

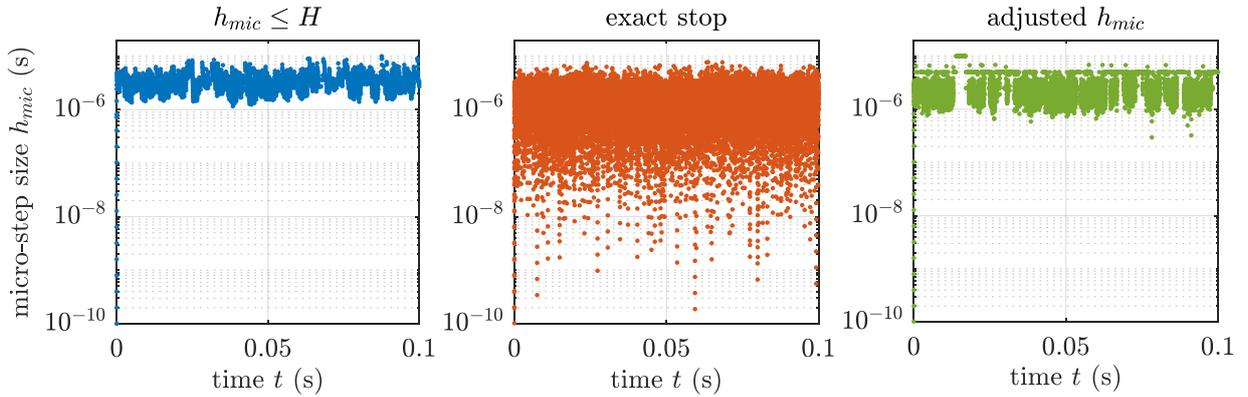
The function performs two different adjustments to the micro-step size:

- If the next planned micro-step reaches "almost" the next macro-time point  $T_{N+1}$ , the micro-

step size is increased by up to 10 % of its size so that it reaches  $T_{N+1}$ . This strategy avoids the situation that an additional (very small) micro-step will be necessary to reach  $T_{N+1}$ . Increasing the micro-step by up to 10 % is reasonable because most numerical solvers (including *IDA* [HBG<sup>+</sup>05]) use a safety factor within the error estimation, therefore chances that the stretched micro-step is rejected are still low.

- If the difference  $t_{Left} = T_{N+1} - t$  between the next macro-time point  $T_{N+1}$  and the current solver time  $t$  is less than  $2 h_{mic}$ , the micro-step size is reduced to  $t_{Left}/2$ . The idea is again to avoid a small final solver step at the end of a macro-step.

Small solver steps at the end of a macro-time step should be avoided because step size controllers of numerical solvers have typically a restriction for increasing the solver step size. A common restriction is  $h_{mic}^{new} \leq 2 h_{mic}$ . Therefore, if the solver step size has to be reduced significantly, only for the purpose to exactly hit  $T_{N+1}$ , it will take a number of steps to recover the original micro-step size. The micro-step size adjustment suggested in Listing 4.1 is based on heuristic assumptions. However, numerical tests indicate that it performs best from the alternatives mentioned above.



**Figure 4.4:** Micro-step size of an arbitrary subsystem for different limitations: micro-step size limited by the macro-step size  $H$ , integration is stopped at each macro-time point exactly and adjusted micro-step size according to Listing 4.1.

Figure 4.4 shows exemplarily the micro-step size of an arbitrary subsystem solver of an explicit co-simulation carried out with the three different micro-step size restrictions and the following parameters: macro-step size  $H = 1.0e-5$  s is constant, the degree of the approximation polynomials is  $\kappa = 2$ , the model consists of  $n_K = 30$  masses and is split into  $n_s = 2$  subsystems of equal size. As can be seen, the micro-step size of the co-simulation, in which the subsystem solver is stopped exactly at each macro-time point shows large fluctuations. The average micro-step size is therefore smaller than for the co-simulation where the micro-step size is restricted by  $h_{mic} \leq H$ . The application of the micro-step size adjustment function results in a still smaller average micro-step size than using the limitation  $h_{mic} \leq H$ , but the fluctuations are reduced significantly compared to the unadjusted exact stopping method.

#### 4.4. Relation Between Macro-Step Size and Computation Time

Considering numerical time integration methods, the numerical error decreases and the computational effort increases when the solver step size is reduced. For co-simulation methods, the

numerical error also decreases with the macro-step size, but the relation between the computation time and the macro-step size is not necessarily as obvious as for classical time integration methods. To get a better understanding of the influence of the macro-step size on the computational efficiency, the computation time of the main processes which are executed during a co-simulation is analyzed. Assuming that the co-simulation is carried out in parallel, the overall computation time consists of basically five parts:

- subsystems:
  - 1) subsystem integration time (the slowest subsystem integration process is relevant)
  - 2) additional subsystem computations due to the co-simulation approach (save/reload of subsystem solver workspace in connection with macro-step repetitions)
  - 3) generation of output file
- co-simulation interface:
  - 4) synchronization and data exchange between the subsystems and the co-simulation interface (MPI data traffic)
  - 5) co-simulation method specific computations of the co-simulation interface (e.g. step size controller, computation of corrected coupling variables).

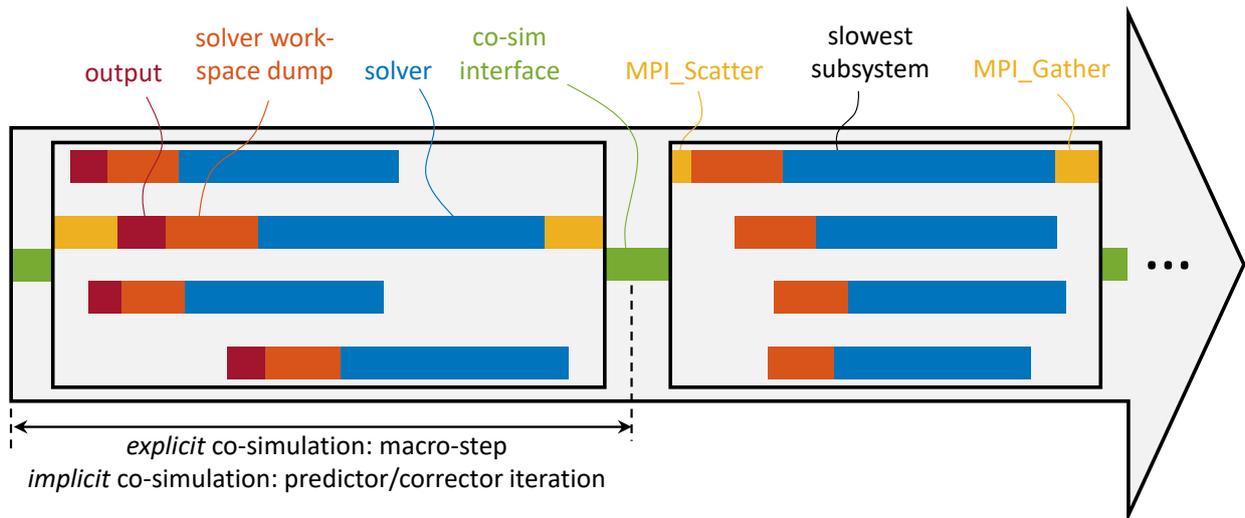
The segmentation of the overall computation time is schematically visualized in Fig. 4.5. Considering an iterative (implicit, waveform relaxation) co-simulation scheme, each predictor/corrector step is, from the computational point of view, similar to a macro-step of an explicit co-simulation scheme.

All co-simulation methods considered here, except for the explicit approach with an equidistant macro-time grid, require the ability to repeat macro-steps. Therefore, the solver workspace of each subsystem solver is copied at the beginning of each macro-step (solver workspace dump). In case of a macro-step repetition, the solver workspace dump created before is used for a quick reinitialization of the subsystem solver. The computation time spent for dumping and loading the subsystem solver workspace increases, as the subsystem size increases.

The output file generation is not further considered here. The output step size is typically much larger than the average macro-step size (e.g.  $H_{out} \approx 100 \cdot \bar{H}$ ), therefore the time spent on writing output files is typically negligible.

The time delay, which arises from sending data from the co-simulation interface to the subsystems and reverse, varies slightly from one subsystem to each other. To make a clear statement about the overhead which is created by this data traffic, the following definition is made. In each macro-step the subsystem that requires the longest computation time ("slowest subsystem") is identified. The time spent on sending data from the co-simulation interface to this particular subsystem plus the time spent on receiving data from this subsystem is defined as computation time for MPI data traffic.

The computation time of the co-simulation interface is typically small compared to the subsystem integration time. However, if the number of coupling variables is comparable with the number of subsystem state variables and if the average macro-step size is in the same range as the average subsystem solver-step size, then the computational effort of the co-simulation interface may play a significant role.



**Figure 4.5:** Schematic visualization of the main parts of the overall computation time of an arbitrary co-simulation with four subsystems.

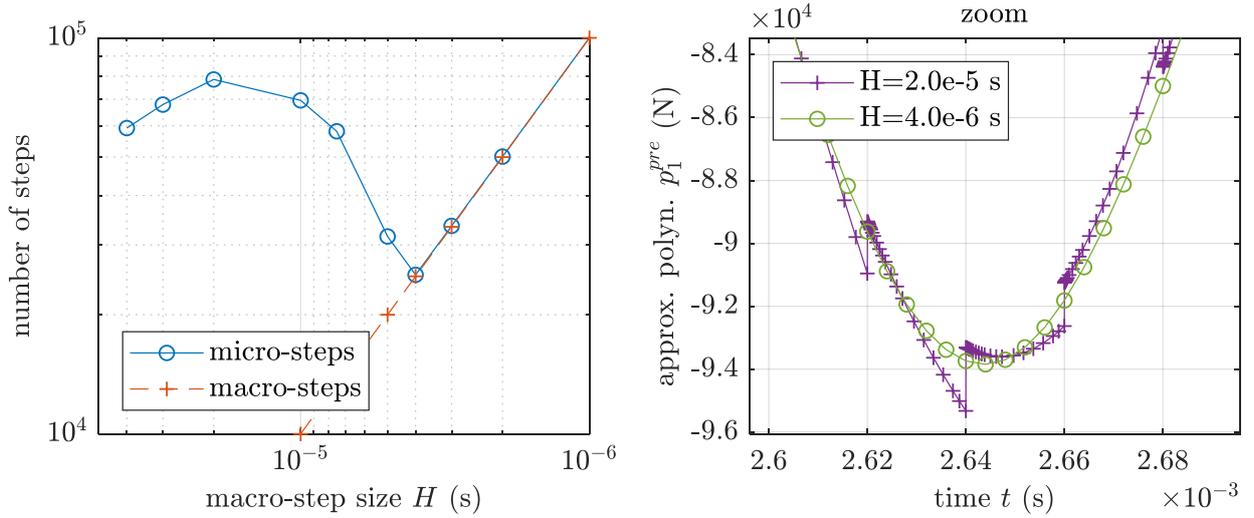
The macro-step size affects all above mentioned parts of the overall computation time of a co-simulation approach, except the output file generation. On the one hand, a larger macro-step size decreases

- the number of calls to the co-simulation interface,
- the number of synchronization points,
- the number of save/reload processes of the subsystem solver workspace,
- the data transfer between the co-simulation interface and the subsystems.

As a consequence, the computation time is reduced. On the other hand, a larger macro-step size increases the discontinuities in the approximation polynomials of the coupling variables at the macro-time points. Because of these discontinuities, the subsystem solver potentially reduces the micro-step size at the beginning of each macro-step (see Fig. 4.6). As a result, the computation time spent on the subsystem integration increases. The effect of the discontinuities occurs especially, when an explicit co-simulation approach is applied. Using an implicit co-simulation approach, the coupling variables are  $C^0$ -continuous at the macro-time points ( $\kappa > 0$ ) because of the corrector iteration. However, there are still jumps in the derivatives. Approaches to reduce the discontinuities in the coupling variables by using modified approximation polynomials have been suggested in literature (e.g. [Bus19]); the proposed methods are, however, in general not compatible with the error estimators described in Section 3.1.

A small macro-step size on the other hand, restricts the micro-step size (see Section 4.3) and therefore also increases the number of micro-steps and the overall computation time.

For the determination of an appropriate macro-step size, it is necessary to identify the dominating part of the overall computation time. If the bottleneck is the data transfer – this may be the case if the number of subsystems or coupling variables is large – then the macro-step size should be chosen as large as possible. If the dominating factor of the computation time are the subsystem integration processes, which is typically the case when the number of coupling variables is small



**Figure 4.6:** Arbitrary explicit co-simulation with fixed macro-step size: average number of micro-steps depending on the macro-step size  $H$  (left figure). Discontinuities in the approximation polynomials resulting in an increased number of micro-steps (right figure).

compared to the number of state variables of each subsystem, an appropriate macro-step size is a compromise between small discontinuities and the limitation of the micro-step size.

It should be noted that the macro-step size control algorithm described in Section 3.4 tries to maximize the macro-step size under the constraint of keeping the estimated errors below the user defined error tolerances. The above discussed effect, that in certain cases a smaller macro-step size allows a more efficient computation, may not be captured within the macro-step size control algorithm. Therefore, it is advisable to simulate a particular model with different error tolerances to find the most efficient configuration of the macro-step size controller.

It is also important to notice that the computation time of the subsystem solver may differ between the subsystems within each macro-step. As shown within the following subsection, the magnitude of these differences in the subsystem computation times is also influenced by the macro-step size.

#### 4.5. Differences in Subsystem Computation Times

The computation time of each subsystem solver within each macro-step varies between the subsystems. Even if all subsystems have similar resulting computation times for the overall simulation, as in the following test case, the subsystem computation times within each macro-step may differ. The effect of these "local" differences in the computation times of the subsystems on the overall computation time may also depend on the choice of the macro-step size. To measure the effect of these local differences in computation time, the idle-time  $T^{\text{idle}}$  is defined according to

$$\begin{aligned}
 T^{\text{solv}} &:= \max_{L \in [1, n_s]} \left( \sum_{N=1}^{N_{\text{mac}}} T_{L,N}^{\text{solv}} \right) \\
 T^{\text{idle}} &:= \sum_{N=1}^{N_{\text{mac}}} \left( \max_{L \in [1, n_s]} T_{L,N}^{\text{solv}} \right) - T^{\text{solv}}.
 \end{aligned} \tag{4.5}$$

In Eq. (4.5),  $N_{mac}$  is the number of macro-steps and  $T_{L,N}^{solv}$  is the computation time of the solver of subsystem  $L$  in macro-step  $N$ .  $T^{idle}$  is defined as the difference between the sum of the slowest subsystem integration processes within each macro-step and the slowest overall subsystem solver. For instance, if always the same subsystem has the slowest integration process within every macro-step then  $T^{idle} = 0$ .

For the analysis of the local computation time differences, a homogenous test model is generated. This test model is the chain-structured multibody system described in Section 2.2, which is parameterized according to Table 4.2 (all subsystems have identical parameters; the initial conditions chosen randomly within the given intervals).

Figure 4.7 shows a detailed view (excerpt) of the computation time of an explicit co-simulation of the model, simulated with three different constant macro-step sizes  $H$ . The blue bars show the computation time of each subsystem in each macro-step. The macro-steps are indicated by the black lines. The red bars show the time spent on writing output files, which is only shown for the sake of completeness. The dark shaded fields mark the slowest subsystem integration process within each macro-step.

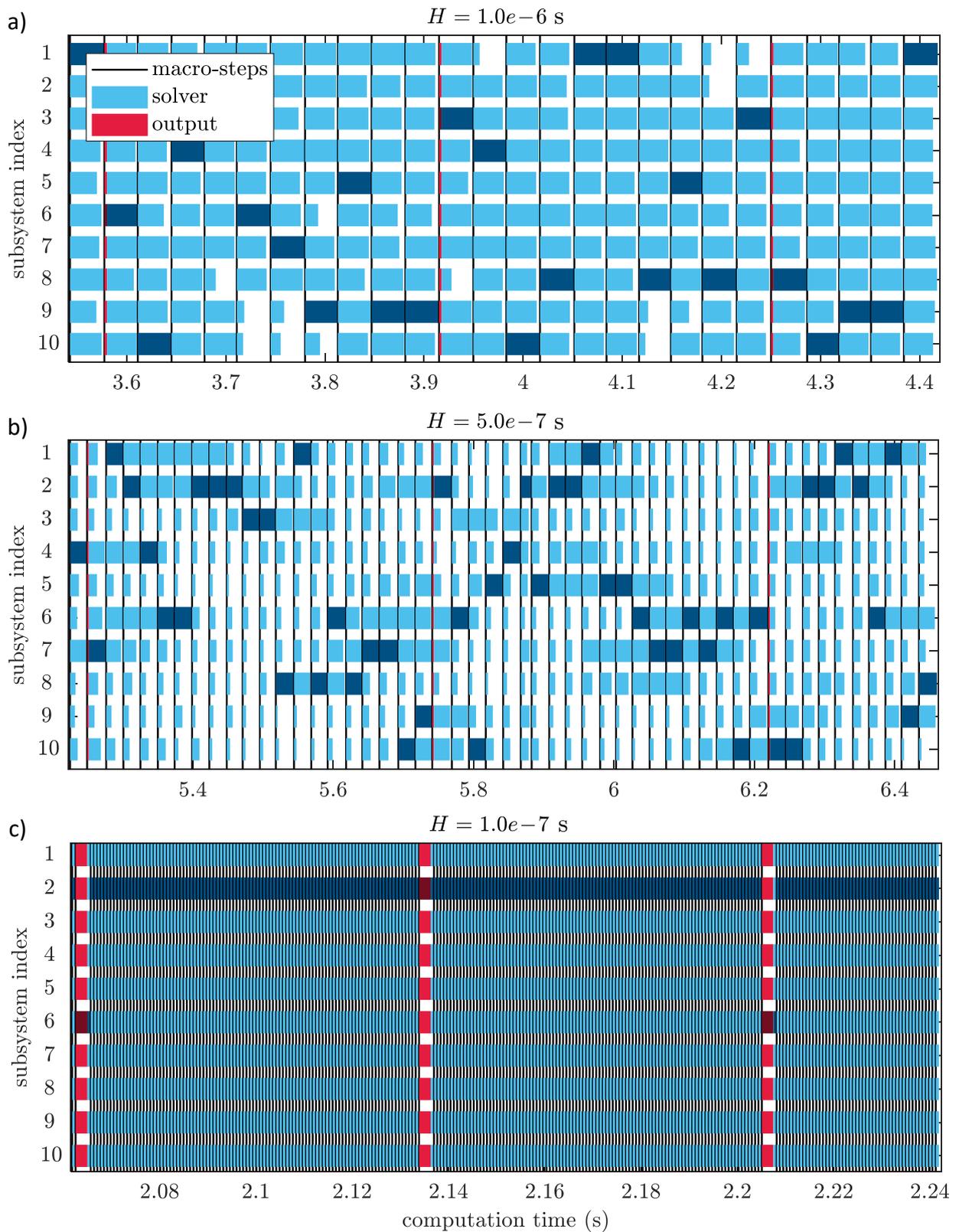
In Fig. 4.7a), the macro-step size  $H = 1.0e-6$  s is chosen relatively large, so that each subsystem solver performs many micro-steps (average:  $H/\bar{h}_{mic} \approx 16$ ) within each macro-step. The computation times of the subsystems within each macro-step are similar. Fig. 4.7b) shows results obtained with a reduced macro-step size of  $H = 5.0e-7$  s. The subsystem solver performs about 7 micro-steps (average value) within each macro-step. This results in relatively large differences in the subsystem computation times. Considering that each subsystem is assigned to one core, the hardware utilization is low, because the cores are idling a large proportion of the time. The macro-step size  $H = 1.0e-7$  s in Fig. 4.7c) is assumed to be smaller than the micro-step size  $h_{mic}$  that would be chosen by the step size controller of the subsystem solver. In this case the micro-step size is limited by the macro-step size due to the restriction  $h_{mic} \leq H$ . Each subsystem solver makes only one micro-step per macro-step. As a consequence, the computation time for all subsystems is almost equal. The price for the good hardware utilization is, however, the limited micro-step size and the increased number of macro-steps.

The same analysis is carried out again for a modified model (heterogeneous model): now, there are contacts according to Eq. (2.12) in subsystems 4 and 8. The contact parameters are set to  $A_C = -1.0e-2$  N and  $B_C = 1.0e7$  m<sup>-1</sup>. A detailed view of the resulting computation time of the heterogeneous model is shown in Fig. 4.8. As expected, the subsystems including the contacts clearly dominate the overall computation time.

The different contributions of the overall computation time are shown in Fig. 4.9 for the homogenous model and in Fig. 4.10 for the heterogeneous model. For both models and all considered macro-step sizes, the dominating parts are the subsystem solver time  $T^{solv}$  and the idle-time  $T^{idle}$ . The time for computations of the co-simulation interface (green), for the data traffic between the

**Table 4.2:** Test model parameters.

$n_K$	20 000
$n_s$	10
$t_{sim}$	$1.0e-3$ s
$m_i$	$5.0e-2$ kg
$c_i$	$2.5e9$ N/m
$d_i$	$2.5e2$ Ns/m
$C_i$	$1.0e17$ N/m <sup>3</sup>
$D_i$	$1.0e-1$ Ns <sup>3</sup> /m <sup>3</sup>
$x_{0,i}$	$[-1.0e-4, 1.0e-4]$ m
$v_{0,i}$	$[-1.0, 1.0]$ m/s
$\kappa$	1



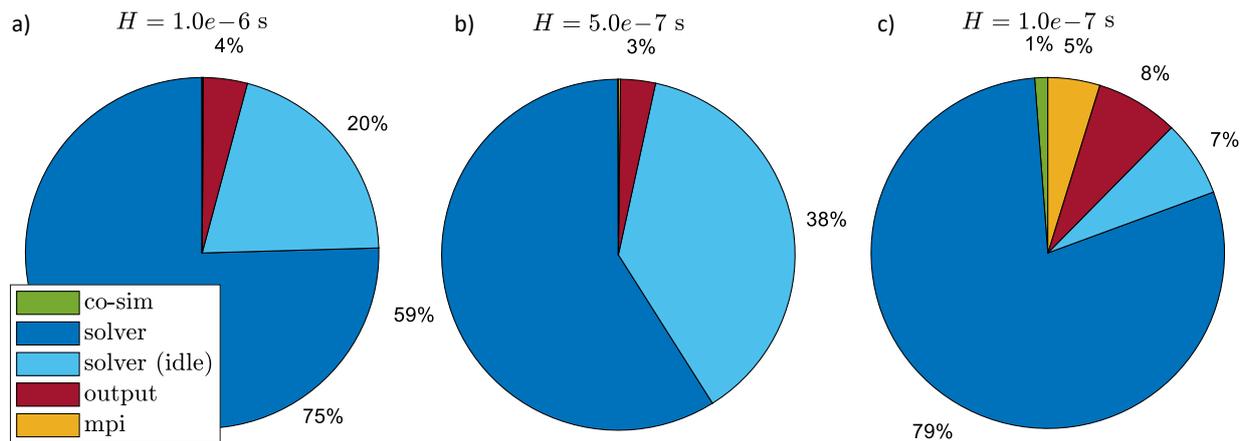
**Figure 4.7:** Subsystem computation times for different macro-step sizes (homogeneous system):  
a)  $H = 1.0e-6$  s, b)  $H = 5.0e-7$  s, c)  $H = 1.0e-7$  s.



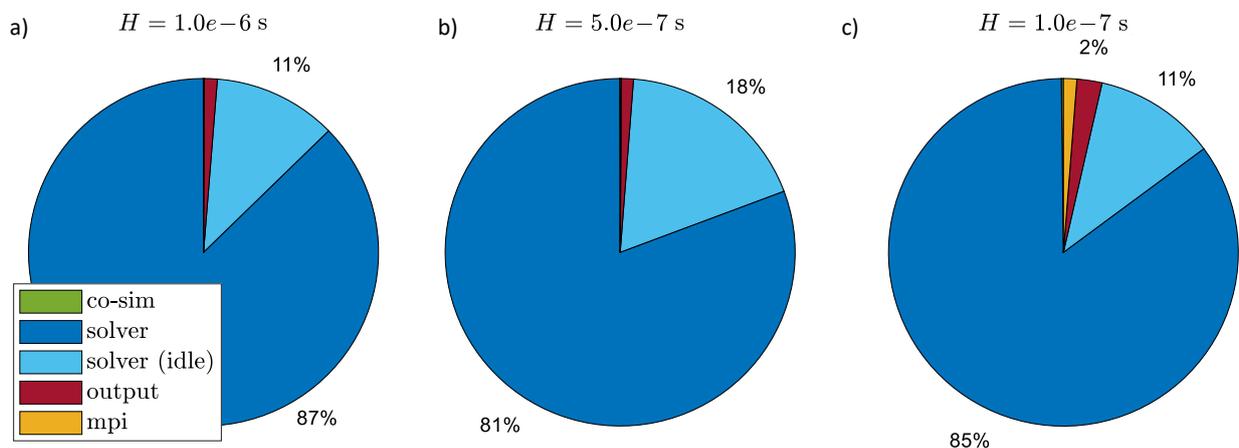
**Figure 4.8:** Subsystem computation times for different macro-step sizes (heterogeneous system, contacts in subsystems 4 and 8): a)  $H = 1.0e-6$  s, b)  $H = 5.0e-7$  s, c)  $H = 1.0e-7$  s.

co-simulation interface and the subsystems (yellow) and for writing output files (red) is considerably low for these particular co-simulations due to the rather small number of subsystem  $n_s = 10$ . A solver workspace dump is not required, since an explicit co-simulation method with a fixed macro-step size is used.

The interesting observation is the influence of the macro-step size on the idle-time. When the homogenous model is simulated with  $H = 5.0e-7$  s (Fig. 4.7b and Fig. 4.9b), the subsystems spend 38% of the time on waiting for each other, although they have almost identical overall subsystem computation times. For the heterogeneous model the idle-time is smaller, because the overall computation time is dominated by the subsystems with the contacts.



**Figure 4.9:** Homogeneous system: relative parts of the overall computation time for different macro-step sizes.



**Figure 4.10:** Heterogeneous system: relative parts of the overall computation time for different macro-step sizes.

*Remark on the accurate measuring of the computation time:* To carry out a detailed analysis of the computation time of a parallelized simulation it should be noted that the clocks of different computing nodes of a cluster computer are not necessarily synchronized. A method to synchronize the absolute computation time measurement on different nodes based on the round-trip-time (time required for sending and receiving a message from one MPI-rank to another) is described in [HSL10].

## 5. Numerical Studies

The following numerical studies are carried out with various parametrizations of the chain-structured multibody system introduced in Section 2.2. The parameter sets are chosen very diversely to cover a wide range of model attributes and to demonstrate different numerical effects. A detailed investigation of the macro-step size and order control algorithm of Section 3, including the different error estimators described in Section 3.1, is carried out in Sections 5.2 and 5.8. Also the explicit and the implicit co-simulation approaches are compared. In addition, the effect of various parameters of the co-simulation interface on the computational efficiency is analyzed. Finally, in order to point out the advantages of the co-simulation approaches, the computation time of co-simulations is compared to the computation time of a monolithic simulation of the same model.

If not defined differently, co-simulations are carried out using the following parameters for the macro-step size and order control algorithm: safety factor  $\Gamma = 6$ , macro-step size scaling factor limits  $[r_{min}, r_{max}] = [0.5, 1.5]$ , dead-zone  $[1.0, r_{max})$ , constant of the convergence criterion  $\tau = 0.33$ . The subsystem solver tolerances are set to  $rtol_{IDA} = 10^{-6}$ ,  $atol_{IDA}^{pos} = 10^{-3} \cdot rtol_{IDA}$ ,  $atol_{IDA}^{vel} = 1.0 \cdot rtol_{IDA}$ . The exponents in the constitutive equation (2.9) of the nonlinear spring/damper-elements are set to  $e_x = e_v = 3$ .

To compare the results of different simulations, the following definitions have to be introduced:

- *Monolithic simulation*: numerical computation of the respective overall model without a decomposition into subsystems. The monolithic simulation is carried out with the same solver (IDA [HBG<sup>+</sup>05]), which is used as subsystem solver for the co-simulation. The error tolerances are typically the same as for the subsystems ( $rtol_{IDA} = 10^{-6}$ ,  $atol_{IDA}^{pos} = 10^{-3} \cdot rtol_{IDA}$ ,  $atol_{IDA}^{vel} = 1.0 \cdot rtol_{IDA}$ ). The monolithic simulations are carried out to get a comparative value of the computation time of the respective model.
- *Global reference simulation*: monolithic simulation of the respective model with very stringent error tolerances ( $rtol_{IDA} = 10^{-10}$ ,  $atol_{IDA}^{pos} = 10^{-13}$ ,  $atol_{IDA}^{vel} = 10^{-10}$ ). The results of the reference simulation are used to compute a global error of the co-simulations and monolithic simulations. With the help of the computed error norms, the results of different (co-)simulations of the same model can be compared with regard to their accuracy.
- *Local reference simulation*: the reference simulation is reinitialized at each macro-step of the co-simulation by using the output values of the co-simulation as initial values. The results obtained by the local reference simulation are used to compute the local error of the co-simulation.

For the comparison of the accuracy of different simulation results, the *normalized root mean square error (NRMSE)* according to

$$nrsme(\mathbf{Z}_j) = \sqrt{\frac{\sum_{i=1}^{n_t} (\mathbf{Z}_{i,j}^{ref} - \mathbf{Z}_{i,j})^2}{\sum_{i=1}^{n_t} (\mathbf{Z}_{i,j}^{ref} - \bar{\mathbf{Z}}_j^{ref})^2}} \quad \text{with} \quad \bar{\mathbf{Z}}_j^{ref} = \frac{1}{n_t} \sum_{i=1}^{n_t} \mathbf{Z}_{i,j}^{ref} \quad (5.1)$$

is used. In Eq. (5.1),  $\mathbf{Z} \in \mathbb{R}^{n_t \times n_z}$  is the output matrix of a (co-)simulation with  $i = 1, \dots, n_t$  output points of  $j = 1, \dots, n_z$  output variables (e.g. states);  $\mathbf{Z}^{ref}$  is the output matrix of the corresponding

reference simulation. To obtain a scalar error measure, the norm

$$\text{NRMSE}(\mathcal{Z}) = \left\| [\text{nrsme}(\mathcal{Z}_1), \dots, \text{nrsme}(\mathcal{Z}_{n_z})]^T \right\|_2 \quad (5.2)$$

of the  $n_z$ -dimensional vector, which collects the normalized root mean square errors of the output variables, is computed. The NRMSE can be computed for all output variables or only for a subset of the output variables, e.g. variables on position level or on velocity level. Depending on the type of the reference simulation, the obtained NRSME is either a measure for the global error or for the local error. In terms of comparing the different error estimators of the macro-step size control algorithm, the local error is typically the value of interest. For large models, however, the computation of a local reference solution is very costly; therefore one has to fall back to the global error in some cases.

*Remark on the output file generation:* To compute the local error of a co-simulation, output data have to be printed at every macro-step. Result files of an arbitrary simulation contain typically data points, which are printed at a defined output step size independently of the macro-/solver-step size. To avoid that the recorded computation time of the co-simulation is distorted by an unrealistically high output density, the results for the local error computation ( $H_{out} = H$ ) and the results for time measurement ( $H_{out} \gg H$ ) are obtained by separate simulations.

### 5.1. Convergence Analysis

To examine the convergence behavior of the co-simulation approaches, a test model is simulated with a successively decreased (fixed) macro-step size and the error of the results is computed. The test model consists of 50 masses and is split into 10 subsystems of equal size. The model parameters are given in Table 5.1. One end of the oscillator chain is free ( $c_{51} = d_{51} = C_{51} = D_{51} = 0$ ); the last body  $m_{50}$  is excited by a harmonic force (2.11) with the parameters  $\Omega_{H,50} = 5.0e2 \text{ s}^{-1}$ ,  $\varphi_{H,50} = 0$ ,  $\Delta F_{H,50} = 5.0e7 \text{ N}$ . The subsystem solver tolerances are set to  $\text{rtol}_{\text{IDA}} = 10^{-10}$  and  $\text{atol}_{\text{IDA}}^{\text{pos}} = 10^{-3} \cdot \text{rtol}_{\text{IDA}}$ ,  $\text{atol}_{\text{IDA}}^{\text{vel}} = 1.0 \cdot \text{rtol}_{\text{IDA}}$ . The displacements  $x_i$  and the velocities  $v_i$  of all masses are depicted in Fig. 5.1 with an artificial offset ( $\Delta x_0 = 1.0 \text{ m}$  and  $\Delta v_0 = 330.6 \text{ m/s}$ ), to visualize the dynamics of the system. The convergence analysis is carried out for the linear test model with  $C_i = 0$ ,  $D_i = 0$ ,  $i = 1, \dots, 51$  and for the nonlinear test model with the parameters of Table 5.1. The results are shown in Fig. 5.2 and Fig. 5.3. The graphs show the resulting local and global errors (NRMSE) of the states of the coupling bodies obtained by the explicit co-simulation method (Section 2.4.1). The convergence order of the co-simulation approaches for different degrees  $\kappa$  of the approximation polynomials can be determined by the slope of the error plots:

- local error on position level:  $\mathcal{O}(H^{\kappa+3})$
- local error on velocity level:  $\mathcal{O}(H^{\kappa+2})$
- global error:  $\mathcal{O}(H^{\kappa+1})$ .

**Table 5.1:** Test model parameters.

$n_K$	50
$n_s$	10
$t_{sim}$	$2.5e-1 \text{ s}$
$m_i$	$1.0e1 \text{ kg}$
$c_i$	$1.0e6 \text{ N/m}$
$d_i$	$1.0e2 \text{ Ns/m}$
$C_i$	$1.0e12 \text{ N/m}^6$
$D_i$	$1.0e0 \text{ Ns}^3/\text{m}^3$
$e_x$	6
$e_v$	3

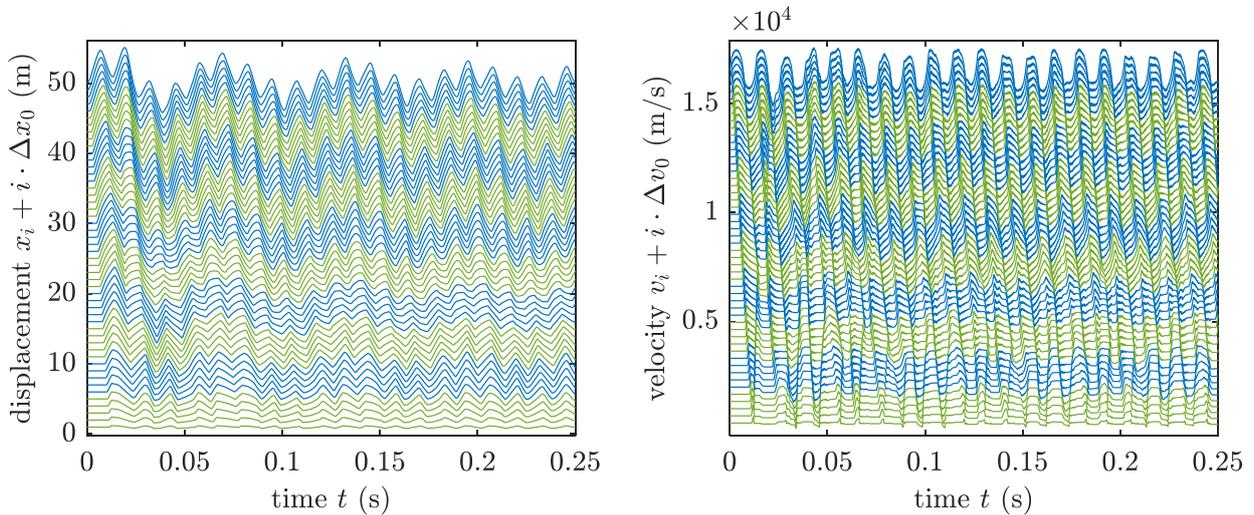
*Remark on the starting procedure of a co-simulation:* Independently of the defined polynomial degree  $\kappa$  for the approximation polynomials, the first macro-step of a co-simulation is carried out with constant approximation polynomials ( $\kappa = 0$ ), because there is only one sampling point  $T_0$  available. Then  $\kappa$  is increased after each macro-step until the defined value is reached. To minimize the effect of these reduced order approximation polynomials on the global error, the macro-step size is decreased in the first few macro-steps, by applying the following procedure. The co-simulation starts with the reduced macro-step size  $H_{red} = 10^{-5} \cdot H$ . Within each following macro-step,  $H_{red}$  is doubled until the condition  $T_N + H_{red} > 2.0 \cdot H$  is fulfilled. The procedure is given in Listing 5.1 in pseudo code to clarify the process.

```

if (TN+1 < 2.0 * H) {
  Hred = 1.0e-5 * H * pow(2.0, nMacSteps);      /* reduced macro-step size */
  TN+1 = TN + Hred;
  if (TN+1 > 2.0 * H) {
    Hred = 2.0 * H - TN;
    TN+1 = 2.0 * H;
  }
}
else {
  Hred = H;
  TN+1 = TN + H;
}

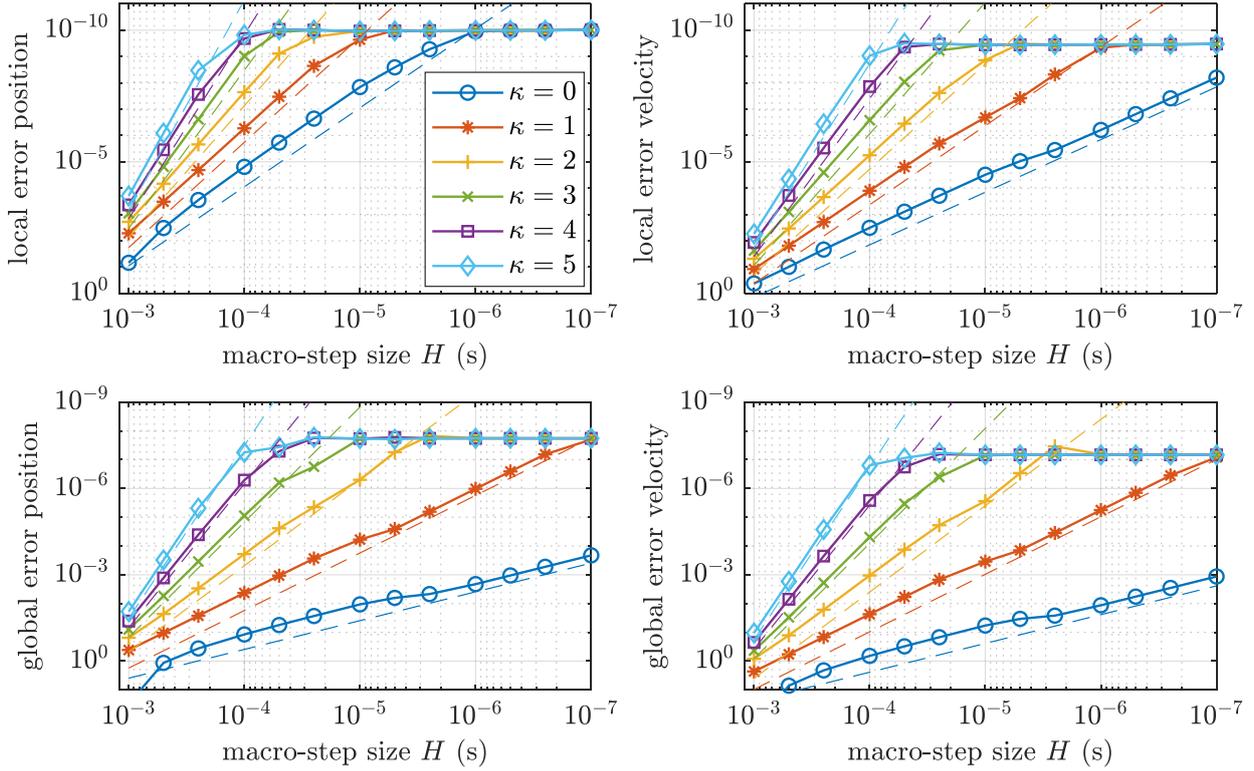
```

**Listing 5.1:** Starting procedure of a co-simulation with a fixed macro-step size: macro-step size is successively increased.

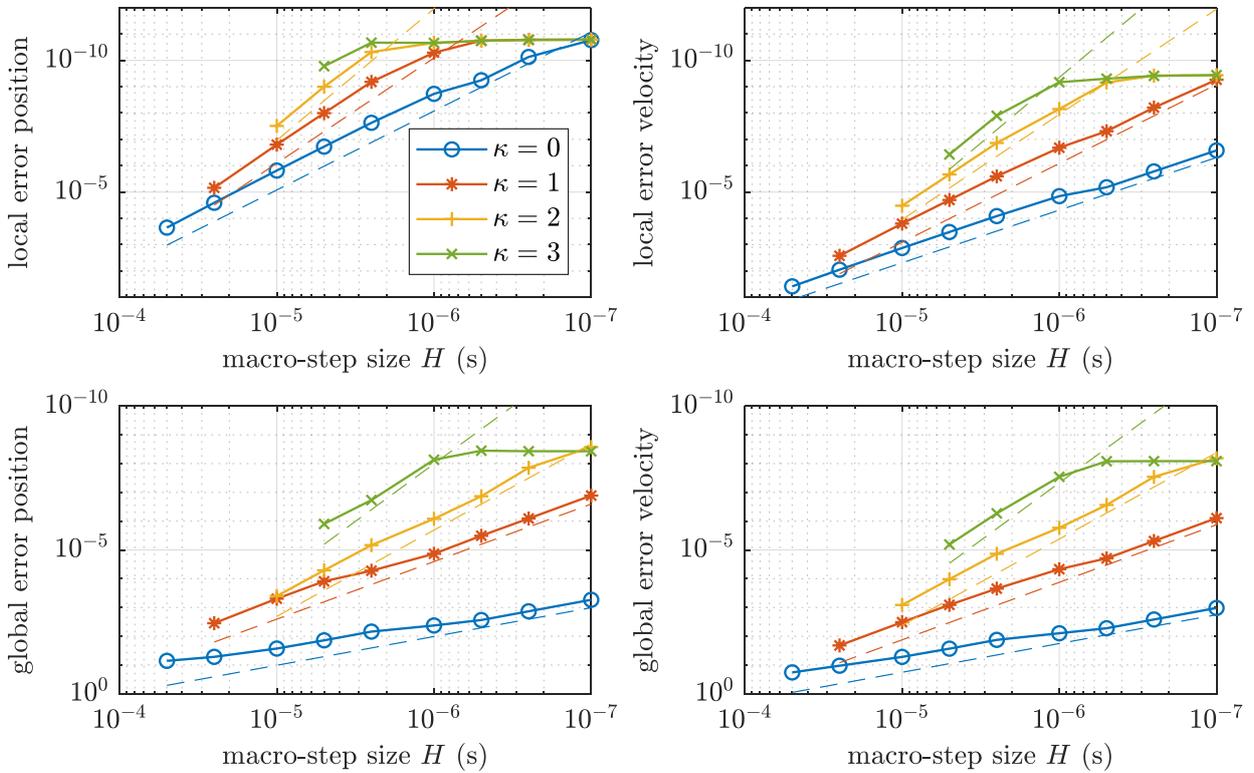


**Figure 5.1:** Nonlinear convergence analysis test model: displacement and velocity of the masses plotted with an offset of  $\Delta x_0 = 1.0$  m and  $\Delta v_0 = 330.6$  m/s (subsystems are indicated by different colors).

The requirements regarding the macro-step size  $H$  for obtaining a stable simulation increase with the polynomial degree  $\kappa$  of the approximation polynomials. In the case of constant approximation polynomials ( $\kappa = 0$ ) the macro-step size  $H = 5.0e-5$  s is sufficient to run a stable explicit co-simulation of the nonlinear model. When cubic polynomials ( $\kappa = 3$ ) are used, the macro-step size has to be reduced by the factor 10 to obtain a stable simulation. The accuracy of the results



**Figure 5.2:** Convergence analysis of the explicit co-simulation approach: linear model. Local and global errors of the coupling bodies (NRMSE) for different approximation orders. The dashed lines indicate the convergence order.

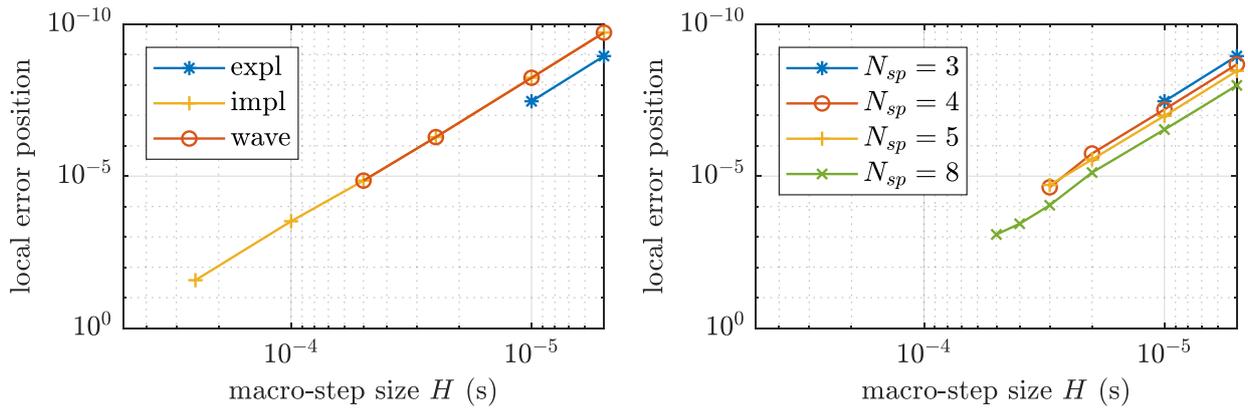


**Figure 5.3:** Convergence analysis of the explicit co-simulation approach: nonlinear model. Local and global errors of the coupling bodies (NRMSE) for different approximation orders. The dashed lines indicate the convergence order.

is limited by the error of the subsystem solver. Considering an explicit co-simulation with cubic approximation polynomials, reducing the macro-step size beyond  $H = 1.0e-6$  s does not increase the accuracy of the results because the error of the subsystem solver becomes the dominating part of the overall numerical error.

If a co-simulation with a fixed approximation order  $\kappa$  should be carried out with an increased macro-step size, larger than the stability limit of the explicit co-simulation approach, there are different options: The more obvious one is to use an alternative co-simulation approach, namely the implicit or the waveform co-simulation method. The limits of the macro-step size for the different co-simulation approaches with quadratic approximation polynomials ( $\kappa = 2$ ) are shown in Fig. 5.4 (left figure). For a stable run of the explicit co-simulation method of this particular nonlinear model, a macro-step size of  $H \leq 1.0e-5$  s is required; for the waveform approach  $H \leq 5.0e-5$  s and for the implicit approach  $H \leq 2.5e-4$  s is sufficient to obtain a stable simulation. The convergence plots for the implicit and for the waveform method are given in Fig. 5.5. The convergence order is the same as for the explicit approach:  $\mathcal{O}(H^{\kappa+3})$  for the local error on position level,  $\mathcal{O}(H^{\kappa+2})$  for the local error on velocity level and  $\mathcal{O}(H^{\kappa+1})$  for the global error (not shown here). It should be mentioned that the relaxation parameter of the waveform method has been set to  $\omega = 0.3$ .

The second option to increase the region of stability might be to use an alternative predictor for the explicit co-simulation. Within the alternative approach, the predicted coupling variables  $\mathbf{u}_{N+1}^{pre}$  are determined by a polynomial fit through the last  $N_{sp} > \kappa + 1$  macro-points. The polynomial fit is generated by using a least squares approach. It should be noted that the polynomial fit is used only to obtain the predicted values  $\mathbf{u}_{N+1}^{pre}$  of the coupling variables at  $T_{N+1}$ . The approximation polynomials, which are used for the subsystem integrations, are interpolation polynomials computed in a second step according to Eq. (2.24). As shown in Fig. 5.4 (right figure) for  $\kappa = 2$ , the alternative predictor with  $N_{sp} = 8$  allows to increase the macro-step size by a factor of 5 compared to the original explicit co-simulation approach. The numerical error increases with the number of additional sampling points  $N_{sp}$ , but the order of convergence is not decreased.



**Figure 5.4:** Local error of the coupling bodies on position level for different co-simulation approaches (left figure) and for different predictor polynomials of degree  $\kappa = 2$  (right figure).

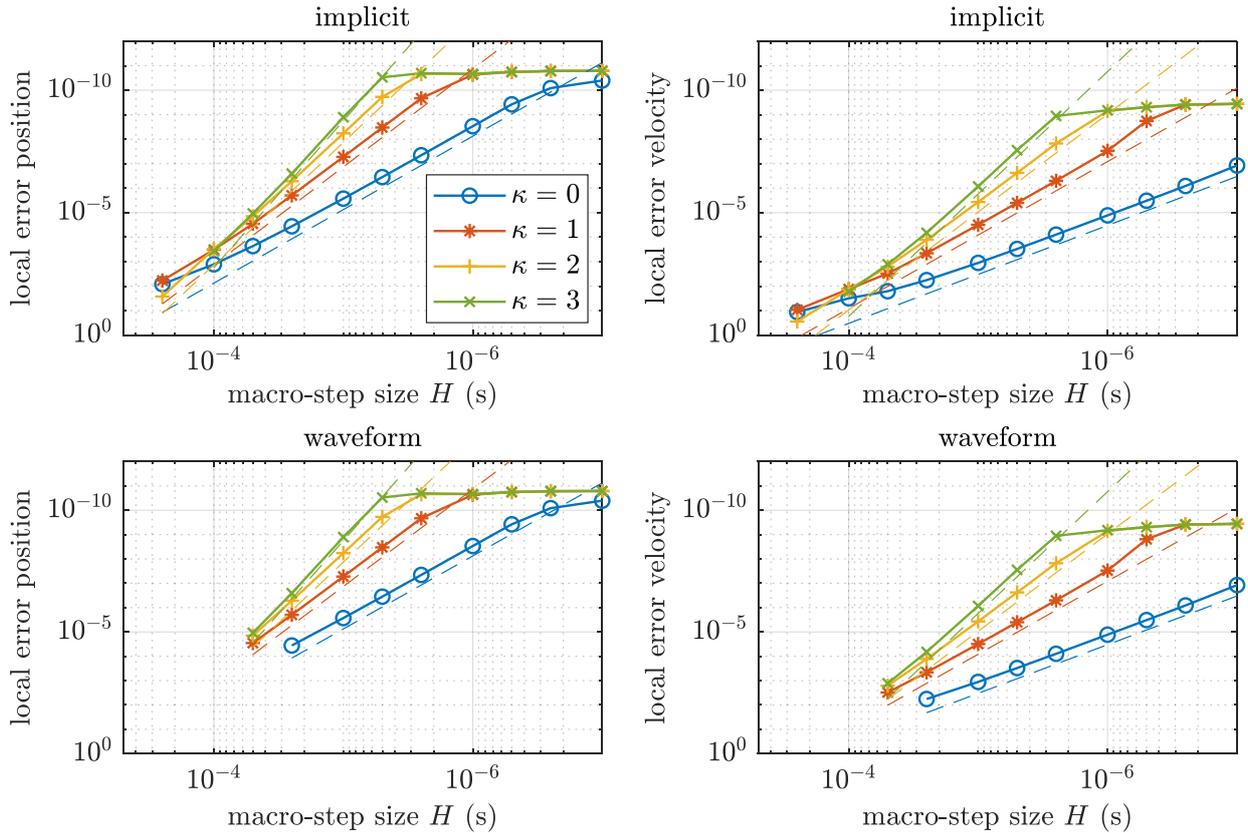
*Remark on the implicit co-simulation approach with constant macro-step size:* The implicit co-simulation method is implemented in order to be used in combination with a macro-step size controller. If it is used with a fixed macro-step size, two considerations have to be made:

- The criterion of convergence (2.31) is defined in terms of the error tolerances of the macro-

step size controller. When a macro-step size controller is not applied, the tolerances of the criterion of convergence have to be specified directly.

- If the corrector iteration does not converge, there is no obvious way of proceeding, since a macro-step size reduction is not provided when the macro-step size is fixed.

For this numerical study, the tolerances of the convergence criterion are set very accurate with  $\text{rtol} = 10^{-10}$  and a maximum number of 10 corrector steps is allowed. If the corrector iteration is not converged after 10 steps, the simulation was continued nevertheless. The same considerations are made for the waveform approach.



**Figure 5.5:** Convergence analysis of the implicit and of the waveform co-simulation approach: nonlinear model. Local error of the coupling bodies (NRMSE) for different approximation orders. The dashed lines indicate the convergence order.

## 5.2. Comparison of Different Error Estimators

Within the following subsection, the five different error estimators described in Section 3.1 are compared by using a two-mass oscillator as test model. The two-mass oscillator is examined for three different parameter sets. The parameters are collected in Table 5.2. Model  $\mathcal{M}1$  is a linear undamped two-mass oscillator,  $\mathcal{M}2$  contains nonlinear stiffness and damping terms and  $\mathcal{M}3$  contains a stiff contact at  $x_1 = 0$  according to Eq. (2.12). The two-mass oscillator is decomposed into two subsystems to obtain a co-simulation model. The subsystems are integrated with the *IDA* solver [HBG<sup>+</sup>05] (relative and absolute error tolerance  $\text{rtol}_{\text{IDA}} = 10^{-12}$  and  $\text{atol}_{\text{IDA}}^{\text{pos}} = \text{rtol}_{\text{IDA}}$ ,  $\text{atol}_{\text{IDA}}^{\text{vel}} = 10^2 \cdot \text{rtol}_{\text{IDA}}$ ). Co-simulations are carried out with quadratic approximation polynomials ( $\kappa = 2$ ) and the following macro-step size controller parameters: safety factor  $\Gamma = 2$ ,  $\text{rtol} = 10^{-5}$ ,

$\text{atol}^{pos} = 10^{-3} \cdot \text{rtol}$  and  $\text{atol}^{vel} = \text{rtol}$ ,  $[r_{min}, r_{max}] = [0.75, 1.25]$ , in case of using an error estimator for the coupling variables (*exCV* or *imCV*) the error tolerances  $\text{rtol} = 10^{-3}$  and  $\text{atol}^u = 10^3 \cdot \text{rtol}$  are used.

For each model and each co-simulation approach, the estimated error is compared to the local error. For the analysis of the estimated error, co-simulations with a fixed macro-step size and co-simulations with a variable macro-step size are carried out. Finally the results of the co-simulation approaches in combination with the different error estimators are compared with respect to the numerical errors and the total number of macro-steps.

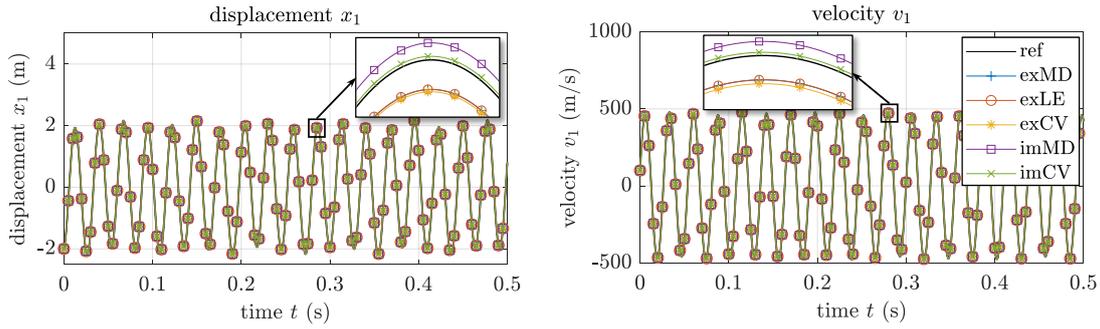
**Table 5.2:** Error estimator comparison: parameterizations of the two-mass oscillator.

	$\mathcal{M}1$ (linear)	$\mathcal{M}2$ (nonlinear)	$\mathcal{M}3$ (contact)	
$m_1$	2.0	5.0	5.0e1	kg
$m_2$	1.0	1.0	1.0	kg
$c_1$	1.0e5	1.0e3	1.0e5	N/m
$c_c$	5.0e3	1.0e2	1.0e5	N/m
$c_2$	1.0e3	1.0e4	1.0e3	N/m
$d_i$	0.0	0.0	0.0	Ns/m
$C_1$	-	1.0e4	0.0	N/m <sup>ex</sup>
$C_c$	-	1.0e5	1.0e5	N/m <sup>ex</sup>
$C_2$	-	1.0e3	0.0	N/m <sup>ex</sup>
$D_1$	-	1.0e-3	0.0	Ns <sup>3</sup> /m <sup>3</sup>
$D_c$	-	5.0e-4	1.0e-2	Ns <sup>3</sup> /m <sup>3</sup>
$D_2$	-	0.0	0.0	Ns <sup>3</sup> /m <sup>3</sup>
$e_x$	-	7	3	-
$e_v$	-	3	3	-
$A_C$	-	-	-1.0e-2	N
$B_C$	-	-	1.0e7	1/m
$x_{10}$	-2.0	-1.0	-2.0	m
$x_{20}$	0.0	0.0	0.0	m
$v_{10}$	1.0e2	1.0e2	1.0e2	m/s
$v_{20}$	-2.0e2	-2.0e2	-2.0e2	m/s
$\kappa$	2	2	2	-

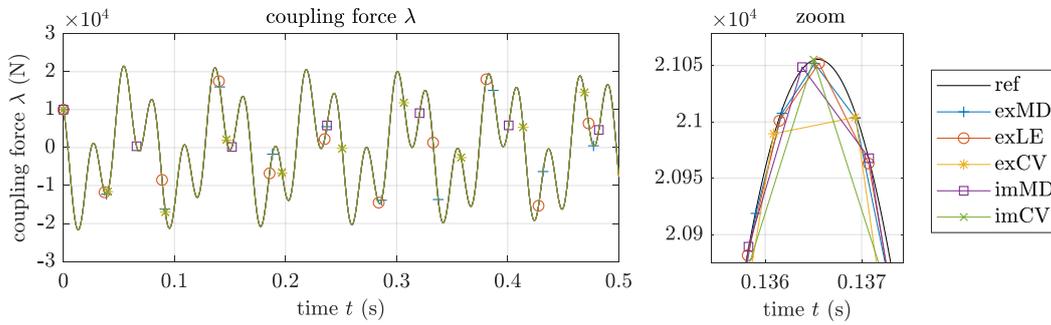
### Linear Two-Mass Oscillator

Figure 5.6 shows the resulting time response  $x_1$  of mass  $m_1$  and the corresponding velocity  $v_1$  obtained by co-simulations using the different error estimators. The coupling force  $\lambda$  is shown in Fig 5.7. As reference, the analytical solution of the test model has been used. In Figs. 5.8 – 5.12, the estimated errors of co-simulations with the fixed macro-step size  $H = 5.0e-4$  s are compared with the exact local errors. The same comparison is made for a co-simulations with a variable macro-step size, the results are shown in Figs. 5.13 – 5.17. The figures show the errors of mass  $m_1$  only, but the error plots of  $m_2$  look equivalent. As can be seen, the exact local error is predicted

with acceptable accuracy by the five error estimators.



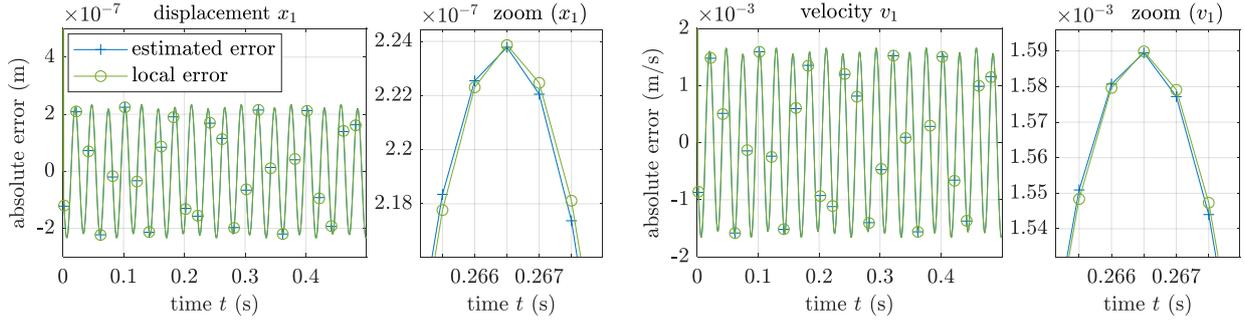
**Figure 5.6:** Linear two-mass oscillator ( $\mathcal{M}1$ ): displacement  $x_1$  and velocity  $v_1$  computed with the different co-simulation approaches.



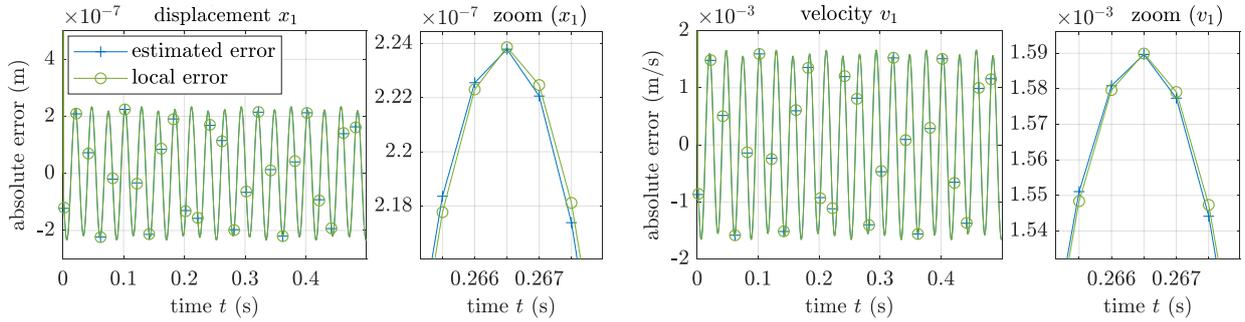
**Figure 5.7:** Linear two-mass oscillator ( $\mathcal{M}1$ ): resulting coupling force  $\lambda$  computed with the different co-simulation approaches.

A comparison of the controlled macro-step size is depicted in Fig. 5.18 for the error estimators *exLE*, *exMD* and *imMD*. In Fig. 5.19, corresponding plots for the error estimators *exCV* and *imCV* are depicted. The resulting global and local errors (NRMSE) of the state variables of the two masses are depicted in Fig. 5.20. The local error is clearly dominated by the local error on velocity level. This observation is reasonable, because the local error on position level converges with a higher order for the considered co-simulation approaches. Convergence plots are collected in Fig. 5.22, where the global and local errors on position and velocity level are plotted as a function of the relative error tolerance *rtol*. The number of macro-time steps as a function of *rtol* is depicted in Fig. 5.21. The results of the numerical studies with the linear two-mass oscillator ( $\mathcal{M}1$ ) can be summarized as follows:

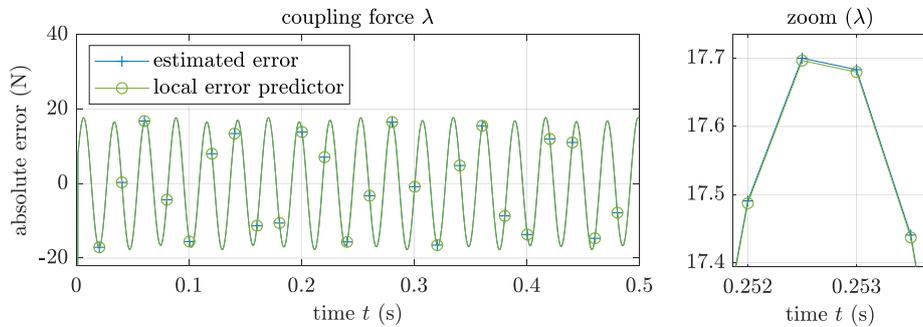
- All five error estimators predict the local errors of the considered variables with acceptable accuracy.
- The *exLE* approach and the *exMD* approach provide almost identical results with respect to the macro-step size and the resulting errors.
- The implicit *imMD* approach allows an increased macro-step size compared to the explicit *exLE* and *exMD* approaches.
- The application of the two error estimators for the coupling force *exCV* and *imCV* results in an almost identical macro-step size. This is reasonable because both estimate the local



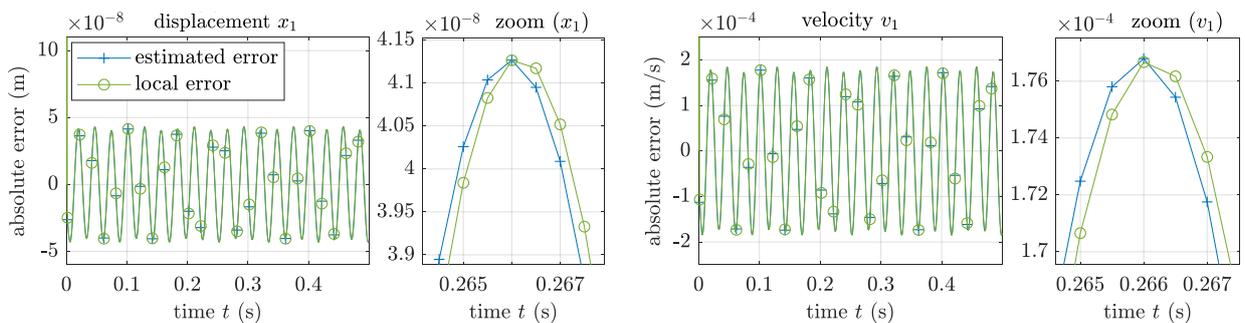
**Figure 5.8:** Explicit co-simulation (*exMD*) of  $\mathcal{M}1$  with fixed macro-step size  $H = 5.0e-4$  s: estimated and local error of the displacement  $x_1$  and the velocity  $v_1$  of mass  $m_1$ .



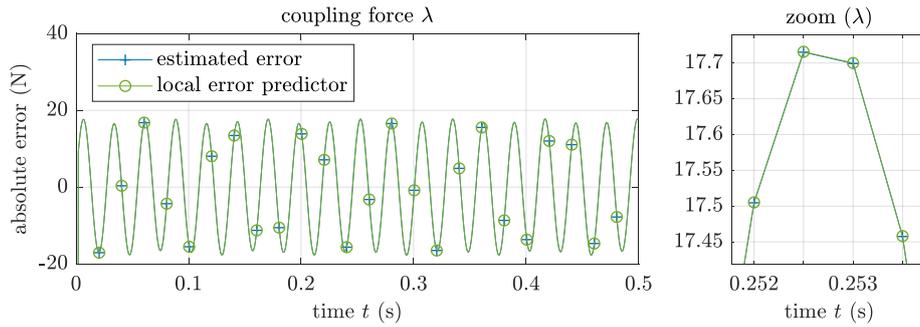
**Figure 5.9:** Explicit co-simulation (*exLE*) of  $\mathcal{M}1$  with fixed macro-step size  $H = 5.0e-4$  s: estimated and local error of the displacement  $x_1$  and the velocity  $v_1$  of mass  $m_1$ .



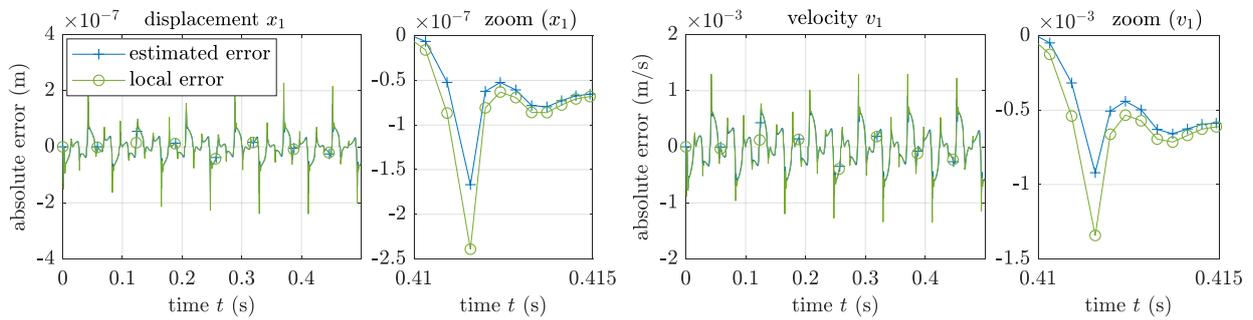
**Figure 5.10:** Explicit co-simulation (*exCV*) of  $\mathcal{M}1$  with fixed macro-step size  $H = 5.0e-4$  s: estimated and local error of the coupling force  $\lambda$ .



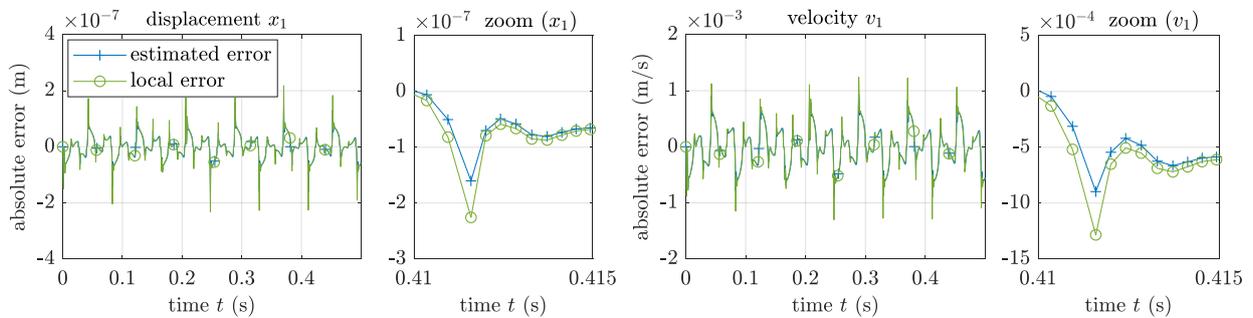
**Figure 5.11:** Implicit co-simulation (*imMD*) of  $\mathcal{M}1$  with fixed macro-step size  $H = 5.0e-4$  s: estimated and local error of the displacement  $x_1$  and the velocity  $v_1$  of mass  $m_1$ .



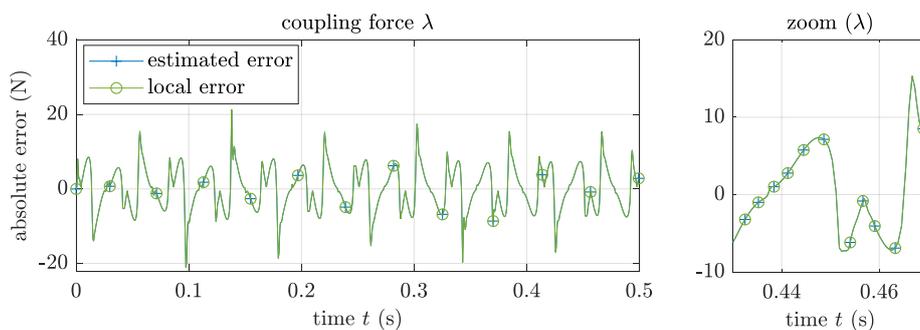
**Figure 5.12:** Implicit co-simulation (*imCV*) of  $\mathcal{M}1$  with fixed macro-step size  $H = 5.0e-4$  s: estimated and local error of the (predicted) coupling force  $\lambda$ .



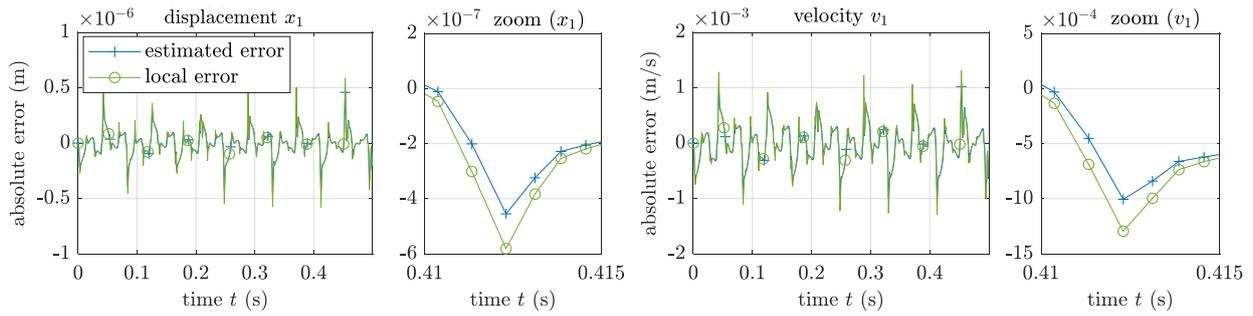
**Figure 5.13:** Explicit co-simulation (*exMD*) of  $\mathcal{M}1$  with variable macro-step size: estimated and local error of the displacement  $x_1$  and the velocity  $v_1$  of mass  $m_1$ .



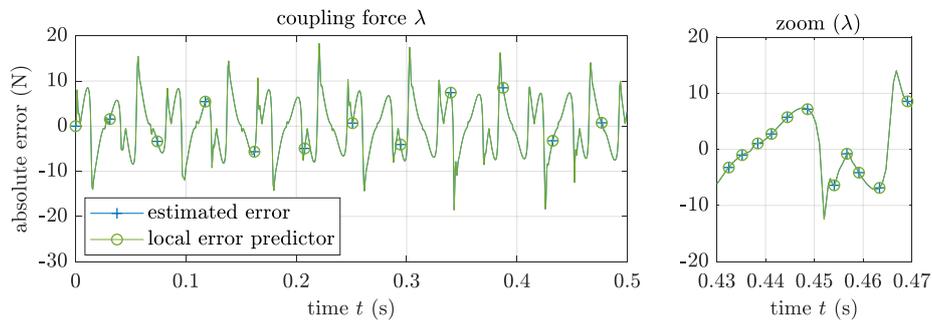
**Figure 5.14:** Explicit co-simulation (*exLE*) of  $\mathcal{M}1$  with variable macro-step size: estimated and local error of the displacement  $x_1$  and the velocity  $v_1$  of mass  $m_1$ .



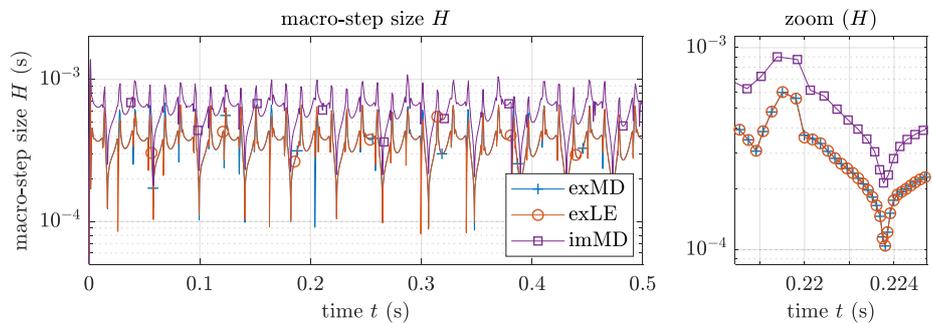
**Figure 5.15:** Explicit co-simulation (*exCV*) of  $\mathcal{M}1$  with variable macro-step size: estimated and local error of the coupling force  $\lambda$ .



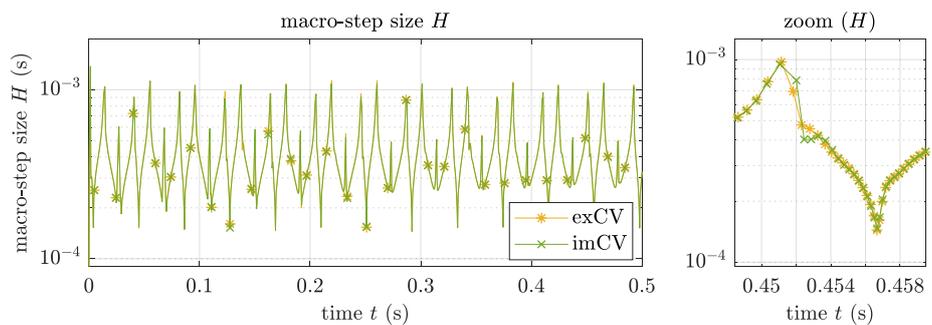
**Figure 5.16:** Implicit co-simulation (*imMD*) of  $\mathcal{M}1$  with variable macro-step size: estimated and local error of the displacement  $x_1$  and the velocity  $v_1$  of mass  $m_1$ .



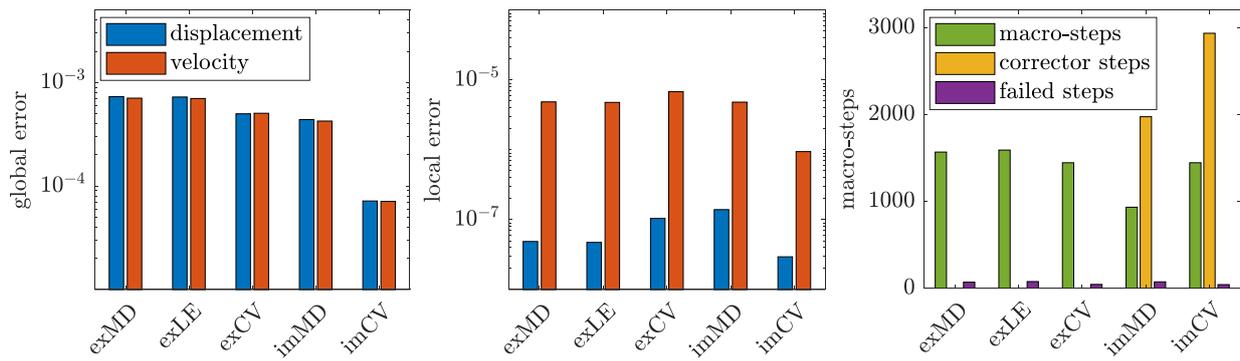
**Figure 5.17:** Implicit co-simulation (*imCV*) of  $\mathcal{M}1$  with variable macro-step size: estimated and local error of the (predicted) coupling force  $\lambda$ .



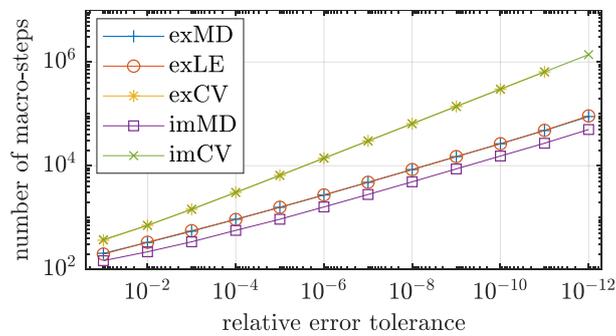
**Figure 5.18:** Linear two-mass oscillator ( $\mathcal{M}1$ ): macro-step size determined by the macro-step size controller with the error estimators *exMD*, *exLE*, *imMD*.



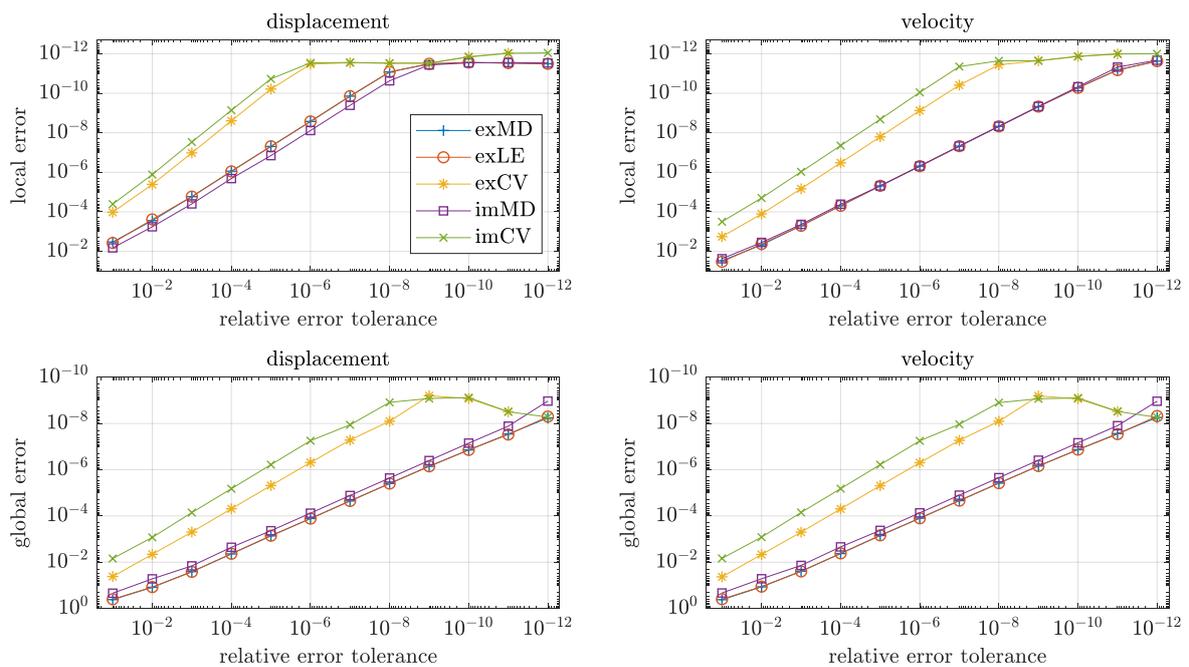
**Figure 5.19:** Linear two-mass oscillator ( $\mathcal{M}1$ ): macro-step size determined by the macro-step size controller with the error estimators *exCV*, *imCV*.



**Figure 5.20:** Linear two-mass oscillator ( $\mathcal{M}1$ ): resulting errors and number of macro-steps.



**Figure 5.21:** Linear two-mass oscillator ( $\mathcal{M}1$ ): number of macro-steps depending on the error tolerances.



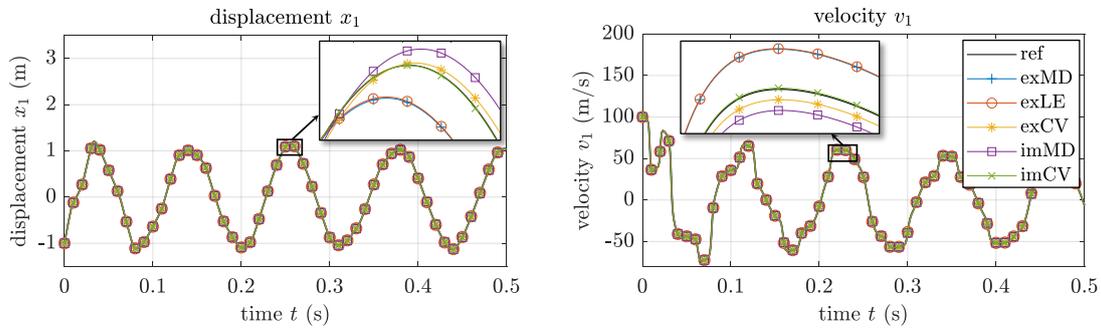
**Figure 5.22:** Linear two-mass oscillator ( $\mathcal{M}1$ ): convergence analysis.

error of the predicted coupling force. The resulting numerical errors are, however, decreased if the implicit (*imCV*) co-simulation approach is used.

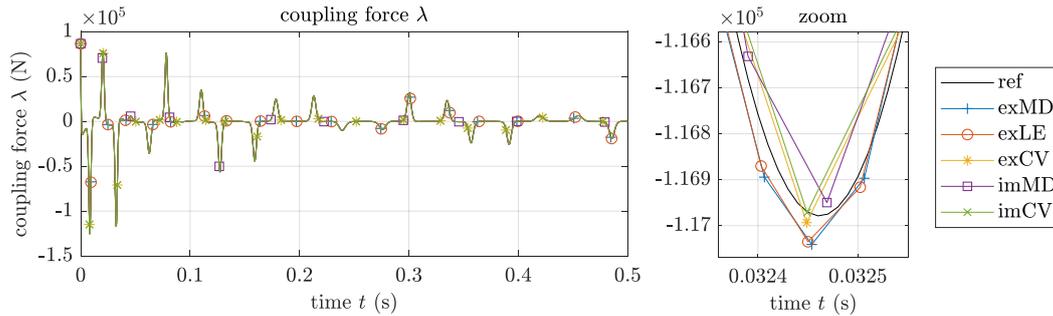
- When using the same relative error tolerance  $rtol$ , the error estimators *exCV* and *imCV* provide a reduced average macro-step size, but also smaller resulting errors compared to the error estimators *exLE*, *exMD* and *imMD*.

### Nonlinear Two-Mass Oscillator

Next, the same numerical studies are carried out for the nonlinear two-mass oscillator ( $\mathcal{M}2$ ). The parameterization of the test model is given in Table 5.2. Figure 5.23 shows the resulting displacement  $x_1$  of mass  $m_1$  and the corresponding velocity  $v_1$  obtained by co-simulations using the different error estimators. The corresponding coupling force  $\lambda$  is shown in Fig 5.24. As reference, a monolithic computation with very tight error tolerances ( $rtol_{IDA} = 10^{-12}$  and  $atol_{IDA}^{pos} = rtol_{IDA}$ ,  $atol_{IDA}^{vel} = 10^2 \cdot rtol_{IDA}$ ) of the test model has been used.



**Figure 5.23:** Nonlinear two-mass oscillator ( $\mathcal{M}2$ ): displacement  $x_1$  and velocity  $v_1$  computed with the different co-simulation approaches.

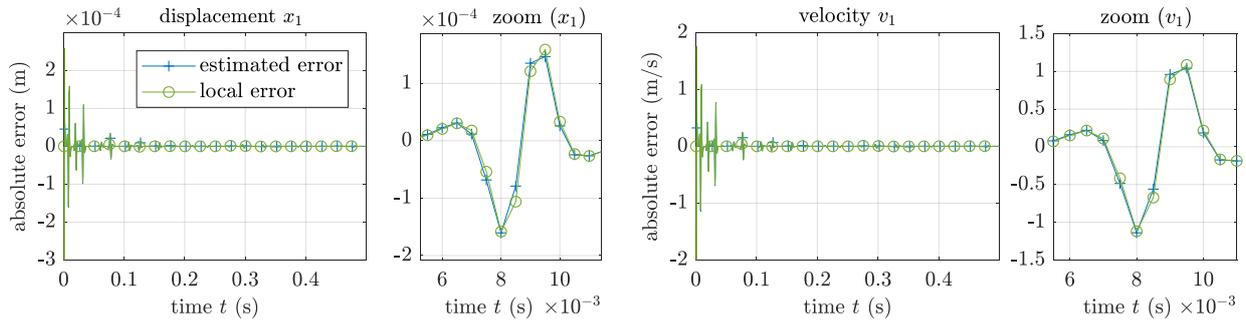


**Figure 5.24:** Nonlinear two-mass oscillator ( $\mathcal{M}2$ ): resulting coupling force  $\lambda$ .

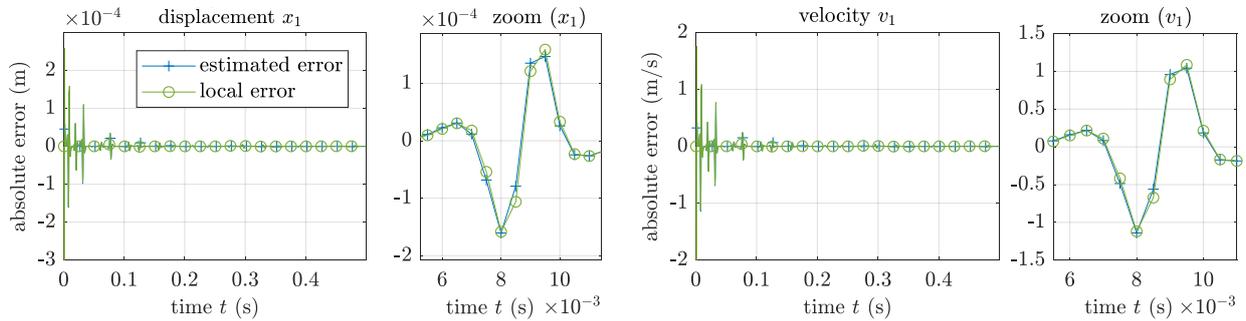
In Figs. 5.25 - 5.29, the estimated errors of a co-simulation with a fixed macro-step size  $H = 5.0e - 4$  s are compared with the exact local errors. The same comparison is made for a co-simulation with a variable macro-step size, the results are shown in Figs. 5.30 - 5.34. As for the linear two-mass oscillator, the local errors of the nonlinear model are predicted with acceptable accuracy by the five suggested error estimators. However, the zoom plots in Figs. 5.30, 5.31 and 5.33 show that certain peaks in the local error are underestimated by the estimators *exLE*, *exMD* and *imMD*. However, the error estimators *exCV* and *imCV* approximate the local error of the predicted coupling force very accurate.

A comparison of the macro-step size is depicted in Fig. 5.35 for the three error estimators  $exLE$ ,  $exMD$  and  $imMD$  and in Fig. 5.36 for the two error estimators  $exCV$  and  $imCV$ . The resulting global and local errors (NRMSE) of the state variables of the two masses are shown in Fig. 5.35. Convergence plots are collected in Fig. 5.38, where the global and local errors on position and velocity level are plotted as a function of the relative error tolerance  $rtol$ . The number of macro-time steps as a function of  $rtol$  is depicted in Fig. 5.39.

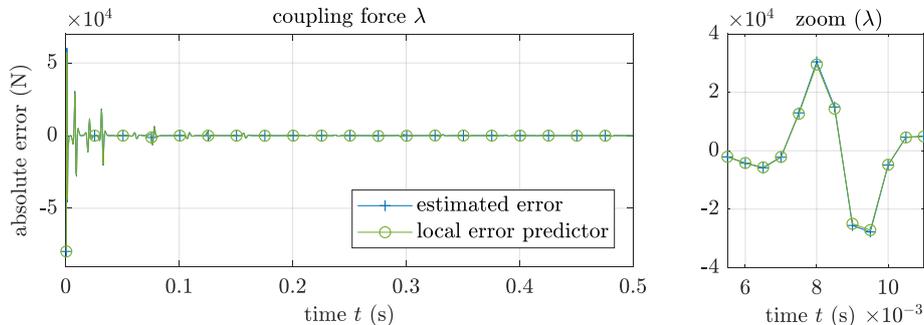
The studies with the nonlinear two-mass oscillator ( $\mathcal{M}2$ ) show similar results as the corresponding studies with the linear model. However, the error estimators  $exCV$  and  $imCV$  show a slightly increased accuracy in the predicted local errors compared to the error estimators  $exLE$ ,  $exMD$  and  $imMD$ .



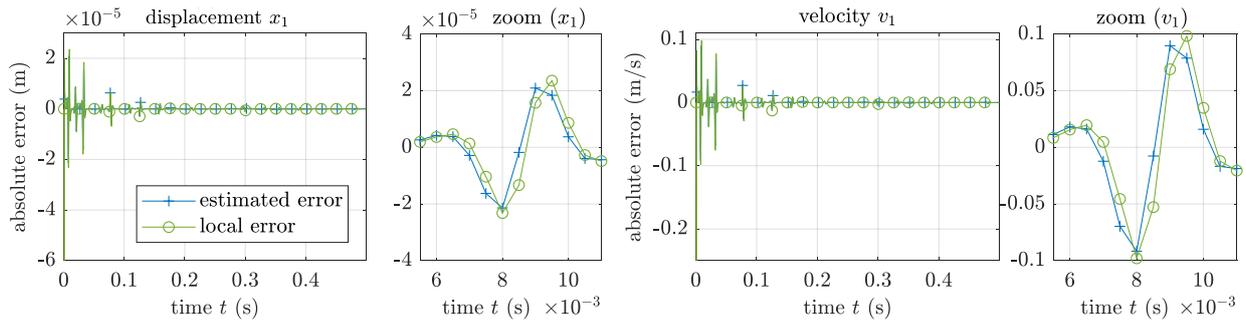
**Figure 5.25:** Explicit co-simulation ( $exMD$ ) of  $\mathcal{M}2$  with fixed macro-step size  $H = 5.0e-4$  s: estimated and local error of the displacement  $x_1$  and the velocity  $v_1$  of mass  $m_1$ .



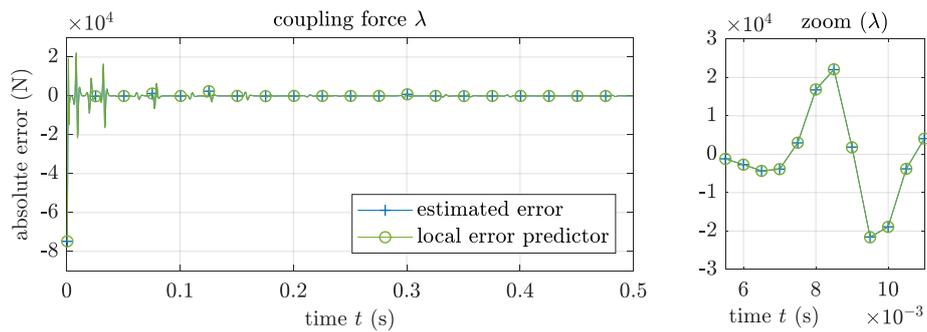
**Figure 5.26:** Explicit co-simulation ( $exLE$ ) of  $\mathcal{M}2$  with fixed macro-step size  $H = 5.0e-4$  s: estimated and local error of the displacement  $x_1$  and the velocity  $v_1$  of mass  $m_1$ .



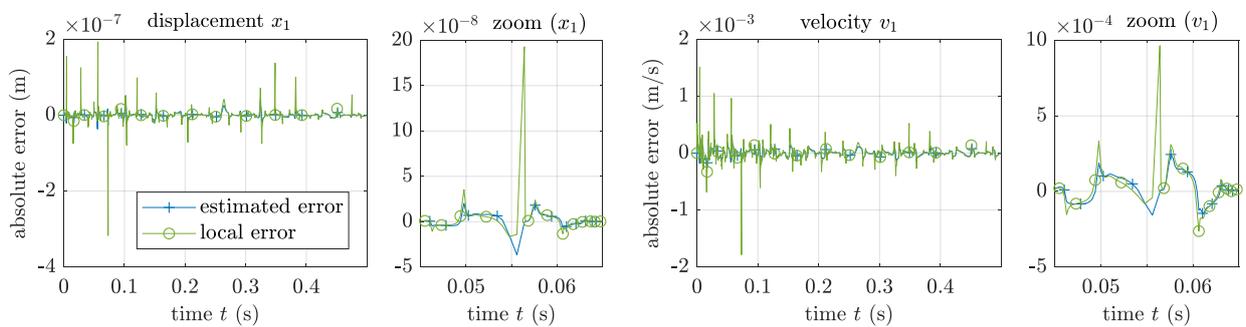
**Figure 5.27:** Explicit co-simulation ( $exCV$ ) of  $\mathcal{M}2$  with fixed macro-step size  $H = 5.0e-4$  s: estimated and local error of the coupling force  $\lambda$ .



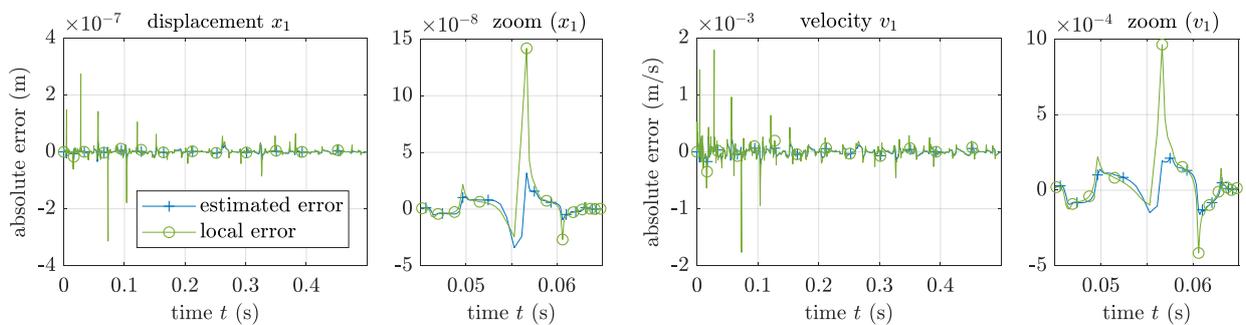
**Figure 5.28:** Implicit co-simulation (*imMD*) of  $\mathcal{M}2$  with fixed macro-step size  $H = 5.0e-4$  s: estimated and local error of the displacement  $x_1$  and the velocity  $v_1$  of mass  $m_1$ .



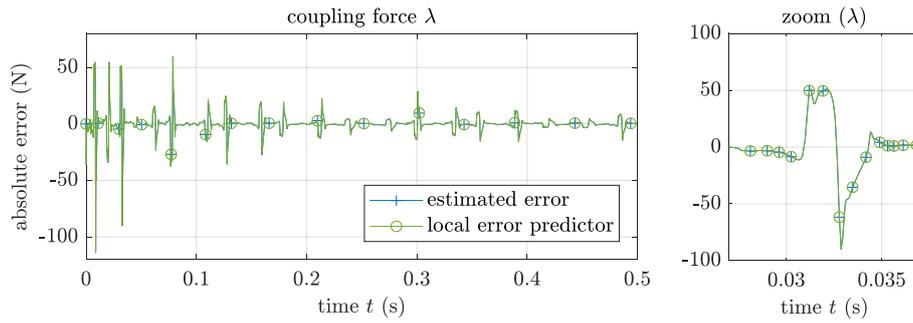
**Figure 5.29:** Implicit co-simulation (*imCV*) of  $\mathcal{M}2$  with fixed macro-step size  $H = 5.0e-4$  s: estimated and local error of the (predicted) coupling force  $\lambda$ .



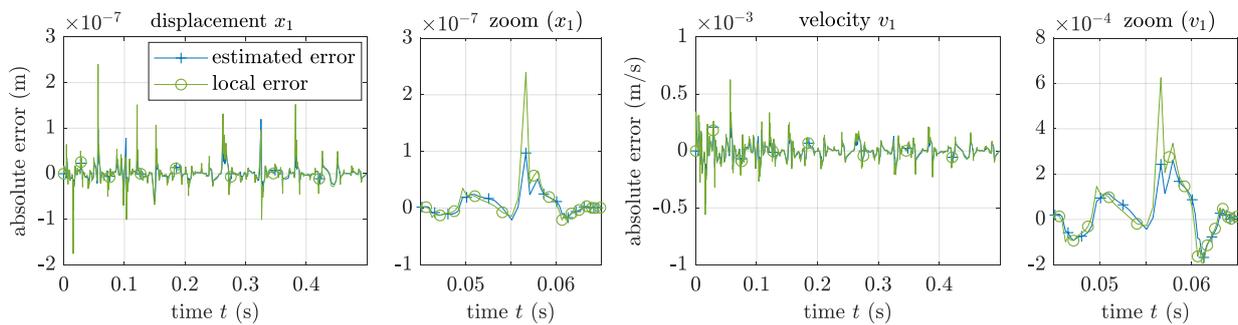
**Figure 5.30:** Explicit co-simulation (*exMD*) of  $\mathcal{M}2$  with variable macro-step size: estimated and local error of the displacement  $x_1$  and the velocity  $v_1$  of mass  $m_1$ .



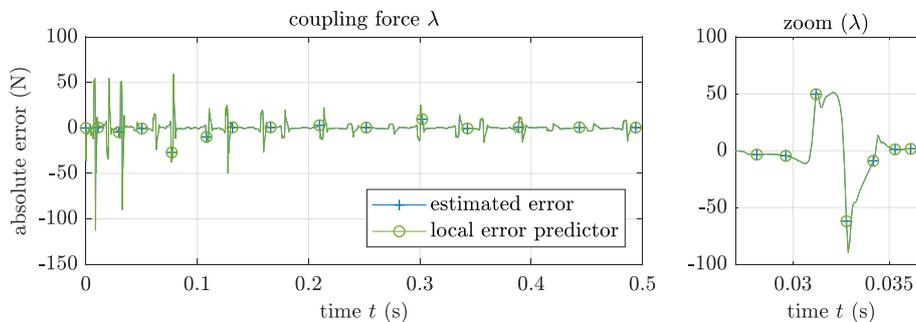
**Figure 5.31:** Explicit co-simulation (*exLE*) of  $\mathcal{M}2$  with variable macro-step size: estimated and local error of the displacement  $x_1$  and the velocity  $v_1$  of mass  $m_1$ .



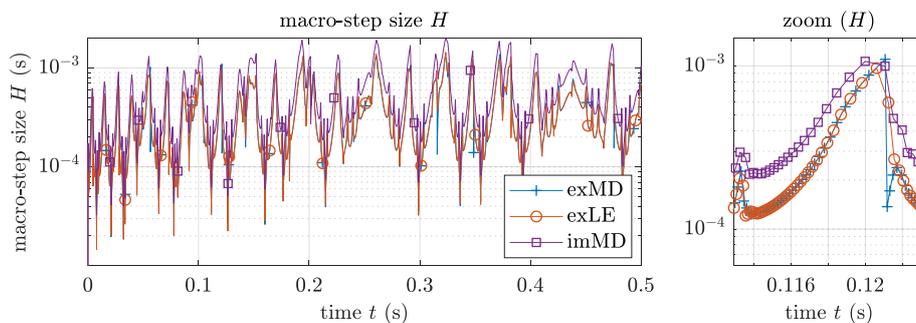
**Figure 5.32:** Explicit co-simulation (*exCV*) of  $\mathcal{M}2$  with variable macro-step size: estimated and local error of the coupling force  $\lambda$ .



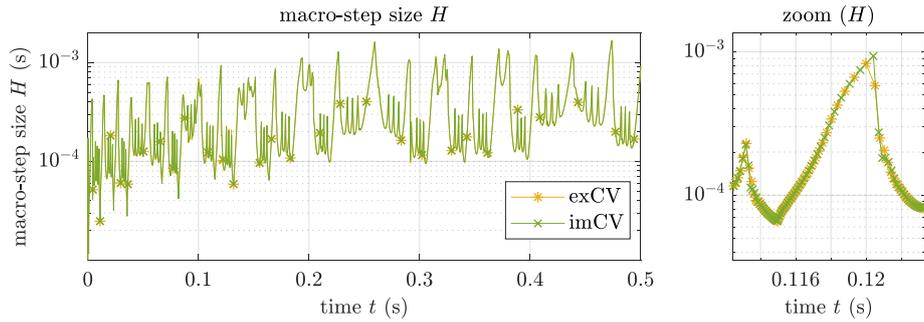
**Figure 5.33:** Implicit co-simulation (*imMD*) of  $\mathcal{M}2$  with variable macro-step size: estimated and local error of the displacement  $x_1$  and the velocity  $v_1$  of mass  $m_1$ .



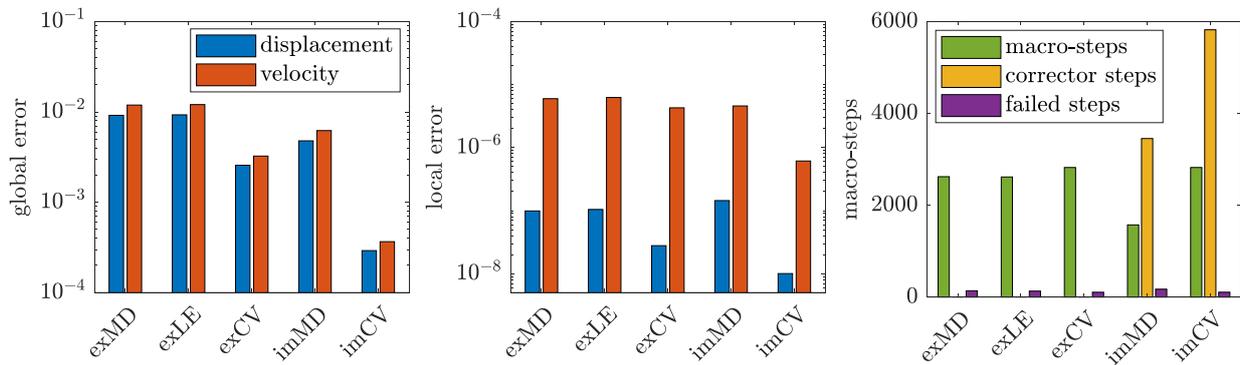
**Figure 5.34:** Implicit co-simulation (*imCV*) of  $\mathcal{M}2$  with variable macro-step size: estimated and local error of the (predicted) coupling force  $\lambda$ .



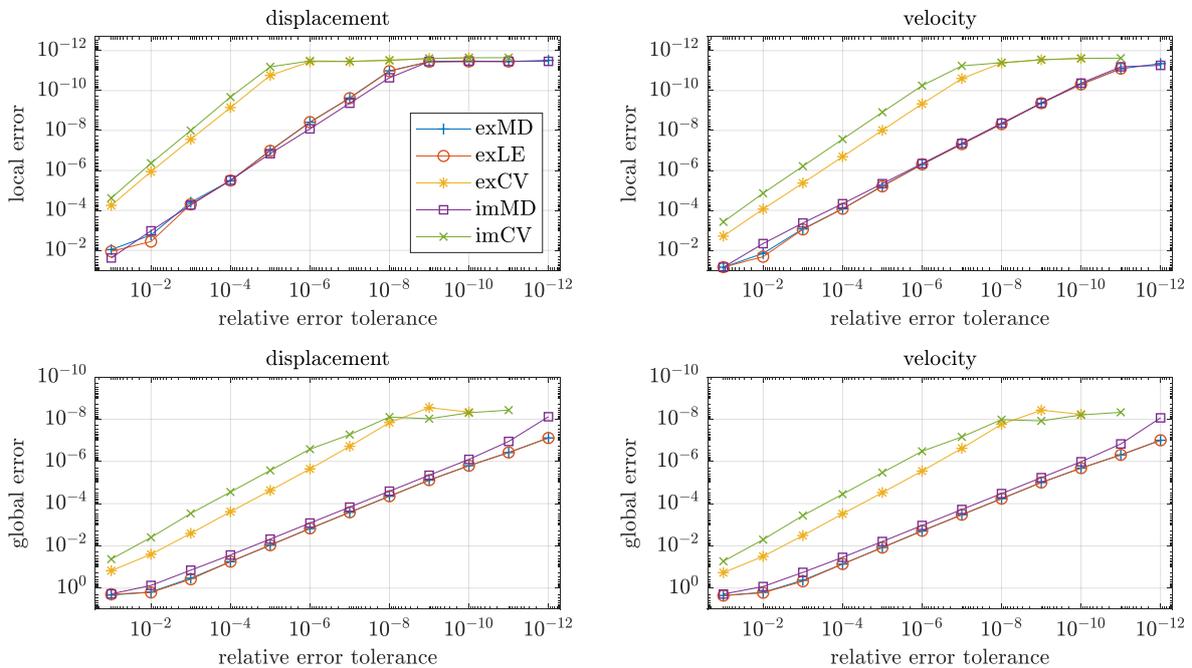
**Figure 5.35:** Nonlinear two-mass oscillator ( $\mathcal{M}2$ ): macro-step size determined by the macro-step size controller with the error estimators *exMD*, *exLE*, *imMD*.



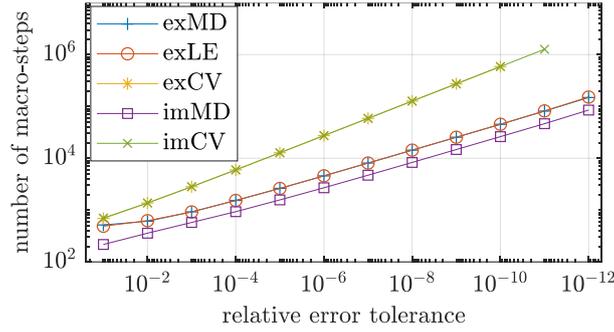
**Figure 5.36:** Nonlinear two-mass oscillator ( $\mathcal{M}2$ ): macro-step size determined by the macro-step size controller with the error estimators *exCV*, *imCV*.



**Figure 5.37:** Nonlinear two-mass oscillator ( $\mathcal{M}2$ ): resulting errors and number of macro-steps.



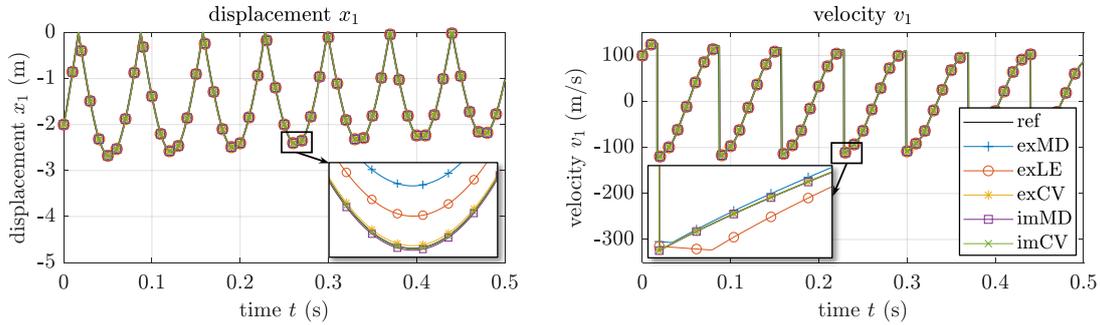
**Figure 5.38:** Nonlinear two-mass oscillator ( $\mathcal{M}2$ ): convergence analysis.



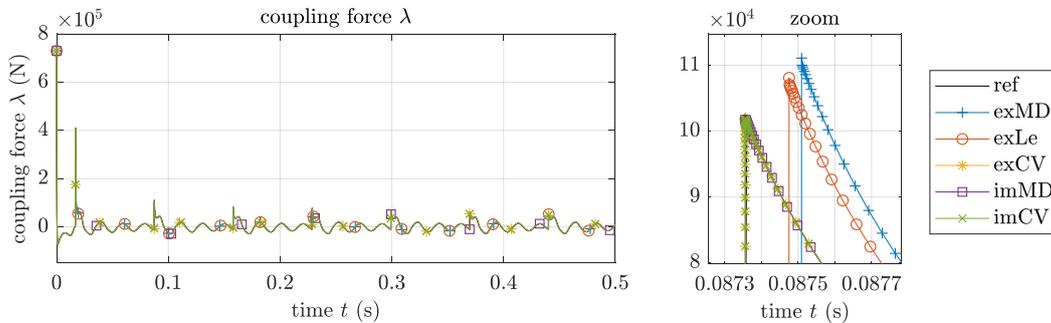
**Figure 5.39:** Nonlinear two-mass oscillator ( $\mathcal{M}2$ ): number of macro-steps depending on the error tolerances.

### Two-Mass Oscillator with Contact

Finally, the performance of the five considered error estimation approaches is compared based on a nonlinear two-mass oscillator parameterization containing a stiff contact ( $\mathcal{M}3$ ). The contact of mass  $m_1$  (occurring at the position  $x_1 = 0$ ) according to Eq. (2.12) is defined by the parameters  $A_C$  and  $B_C$  given in Table 5.2. In Fig. 5.40, the time responses  $x_1$  and  $v_1$  of the two-mass oscillator with contact obtained by simulations with different error estimators are plotted. The corresponding coupling force  $\lambda$  is shown in Fig 5.41. As reference solution, a numerical solution of the monolithic model with very tight error tolerances has been used.



**Figure 5.40:** Two-mass oscillator with contact ( $\mathcal{M}3$ ): displacement  $x_1$  and velocity  $v_1$  computed by the considered co-simulation approaches.

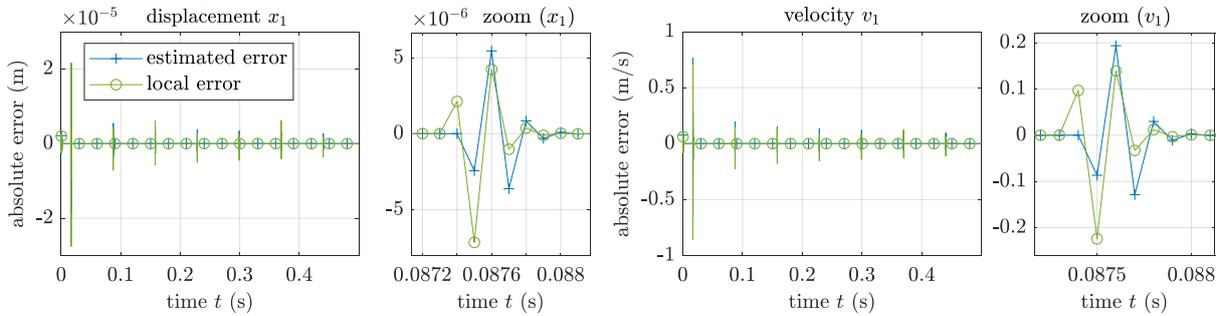


**Figure 5.41:** Two-mass oscillator with contact ( $\mathcal{M}3$ ): resulting coupling force  $\lambda$  computed by the considered co-simulation approaches.

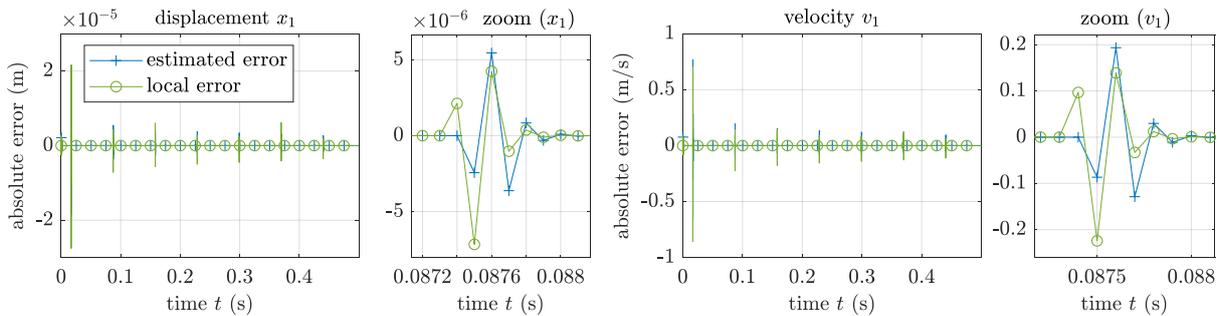
In Figs. 5.42 – 5.46, the estimated errors of co-simulations with a fixed macro-step size  $H = 1.0e - 4$  s are compared with the local errors. The estimated local errors of simulations with a

controlled macro-step size are shown in Figs. 5.47 – 5.51. The macro-step size of simulations with the five error estimators is shown in Figs. 5.52 and 5.53. It can clearly be seen that the macro-step size is reduced, when the contacts occur. The macro-step size is reduced very strongly with the error estimators *exCV* and *imCV*, since these error estimators are based on the coupling force, which has large values at the contact points. The global and local errors on position and velocity level are compared in Fig. 5.54 for the different error estimators. The figure also exhibits some statistical information concerning the number of macro-steps.

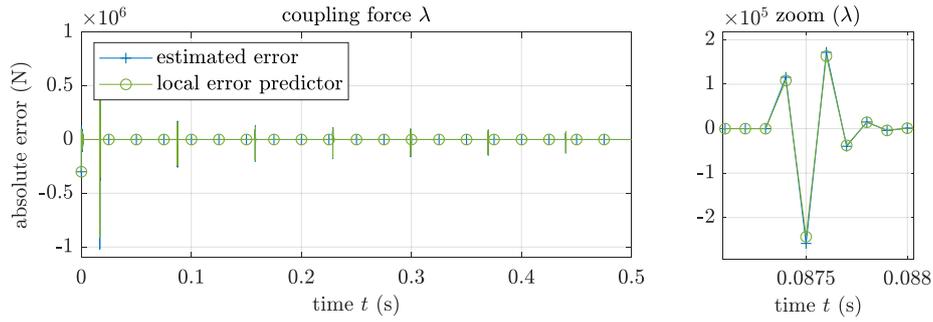
As can be noticed, the explicit and implicit co-simulations yield stable and correct results, although a very stiff and highly nonlinear model is considered. The zoom-plots of the coupling force (Fig. 5.41) and the plot depicting the macro-step size (Fig. 5.52) exhibit that the explicit co-simulation approaches (*exMD* and *exLE*) using an error estimator on state level react with a short delay (in the range of one macro-step) at the contact points. This behavior can be traced back to the considered error estimation methods. Applying the approaches *exMD* or *exLE*, the error of the (coupling) bodies is estimated by carrying out two subsystem integrations with different approximation polynomials. The error contribution of each subsystem is estimated by using information of this subsystem only; an exchange of information between the subsystems is not considered for the error estimation. The error estimation takes place before the coupling equations are evaluated and the coupling variables are updated. Therefore, the error estimators use only predicted values to estimate the errors. The *exCV* approach and the implicit approaches estimate the error after the coupling equation have been evaluated. As a result, these approaches may realize the contact before the macro-step is accepted. As a consequence, the macro-step can be repeated with a smaller macro-step size. The *exMD* and *exLE* approaches recognize those effects only after the next macro-step has been carried out and react therefore with a delay of one macro-step.



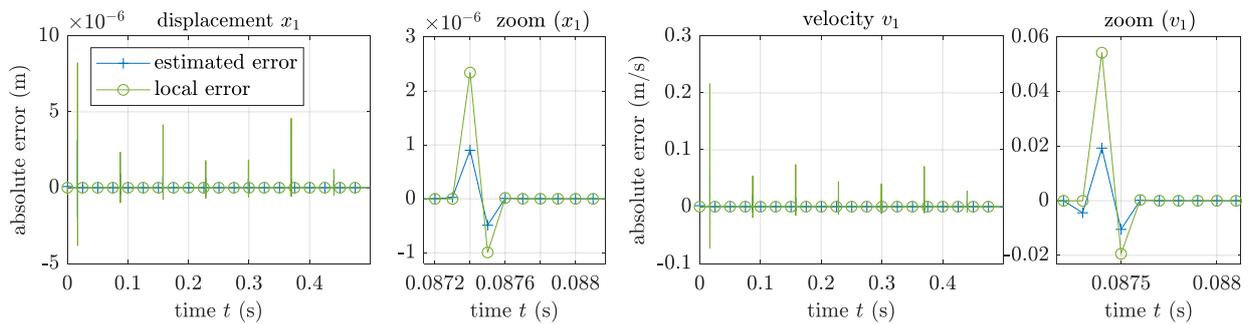
**Figure 5.42:** Explicit co-simulation (*exMD*) of  $\mathcal{M}3$  with fixed macro-step size  $H = 1.0e-4$  s: estimated and local error of the displacement  $x_1$  and the velocity  $v_1$  of mass  $m_1$ .



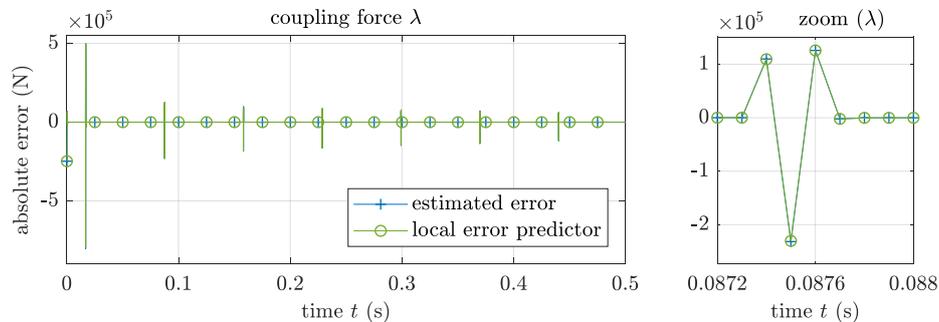
**Figure 5.43:** Explicit co-simulation (*exLE*) of  $\mathcal{M}3$  with fixed macro-step size  $H = 1.0e-4$  s: estimated and local error of the displacement  $x_1$  and the velocity  $v_1$  of mass  $m_1$ .



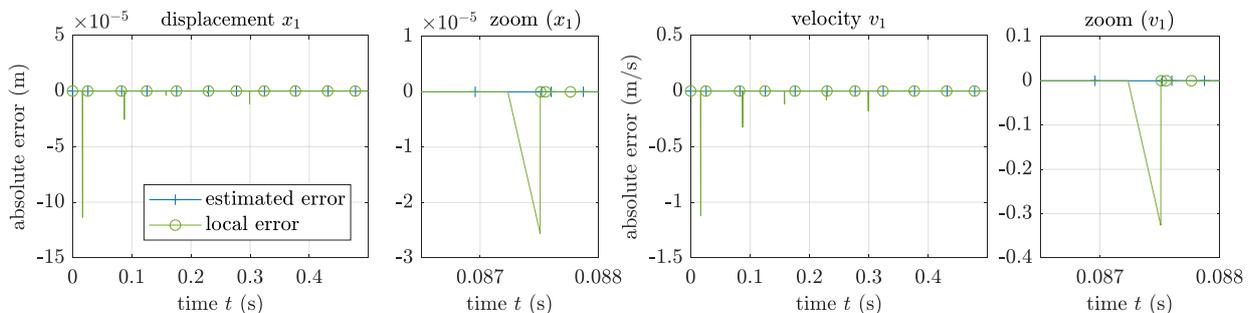
**Figure 5.44:** Explicit co-simulation (*exCV*) of  $\mathcal{M}3$  with fixed macro-step size  $H = 1.0e-4$  s: estimated and local error of the coupling force  $\lambda$ .



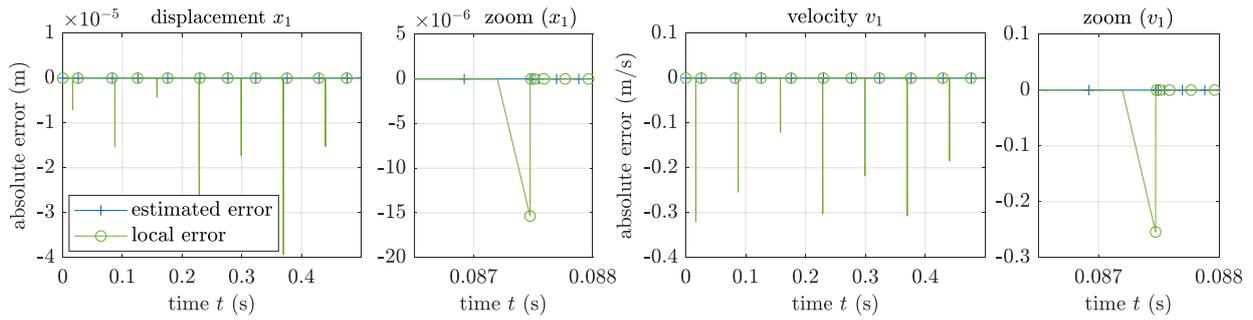
**Figure 5.45:** Implicit co-simulation (*imMD*) of  $\mathcal{M}3$  with fixed macro-step size  $H = 1.0e-4$  s: estimated and local error of the displacement  $x_1$  and the velocity  $v_1$  of mass  $m_1$ .



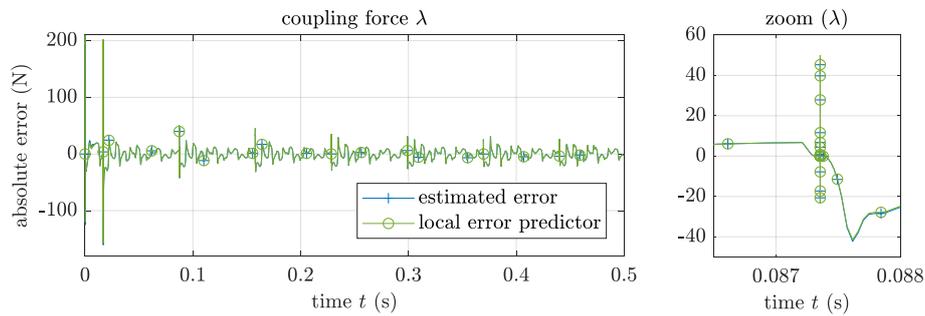
**Figure 5.46:** Implicit co-simulation (*imCV*) of  $\mathcal{M}3$  with fixed macro-step size  $H = 1.0e-4$  s: estimated and local error of the (predicted) coupling force  $\lambda$ .



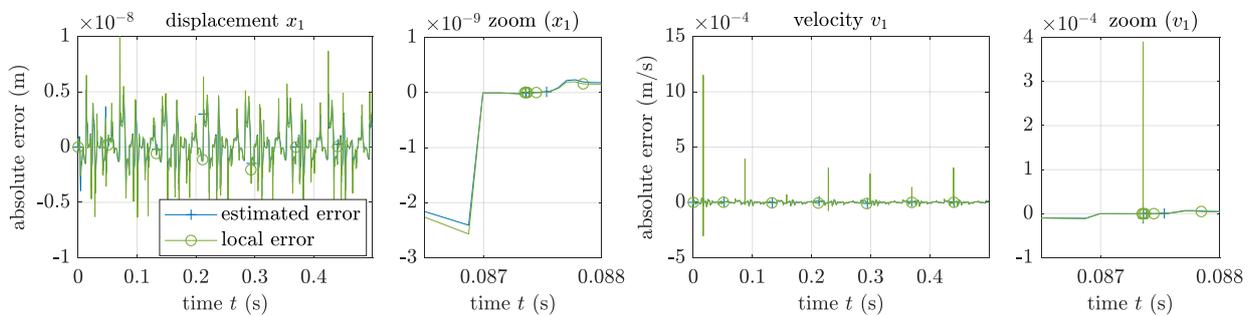
**Figure 5.47:** Explicit co-simulation (*exMD*) of  $\mathcal{M}3$  with variable macro-step size: estimated and local error of the displacement  $x_1$  and the velocity  $v_1$  of mass  $m_1$ .



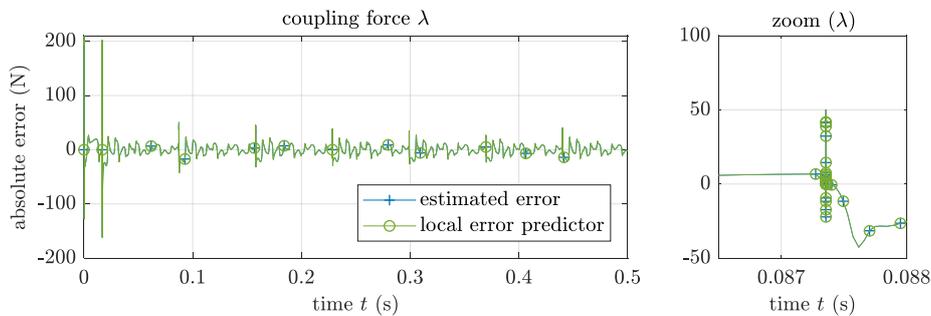
**Figure 5.48:** Explicit co-simulation (*exLE*) of  $\mathcal{M}3$  with variable macro-step size: estimated and local error of the displacement  $x_1$  and the velocity  $v_1$  of mass  $m_1$ .



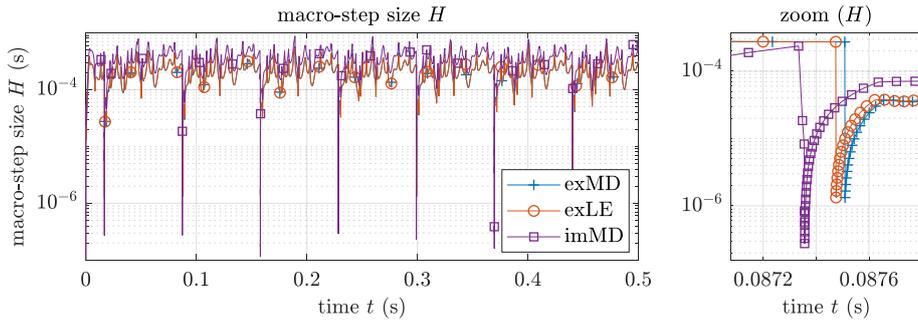
**Figure 5.49:** Explicit co-simulation (*exCV*) of  $\mathcal{M}3$  with variable macro-step size: estimated and local error of the coupling force  $\lambda$ .



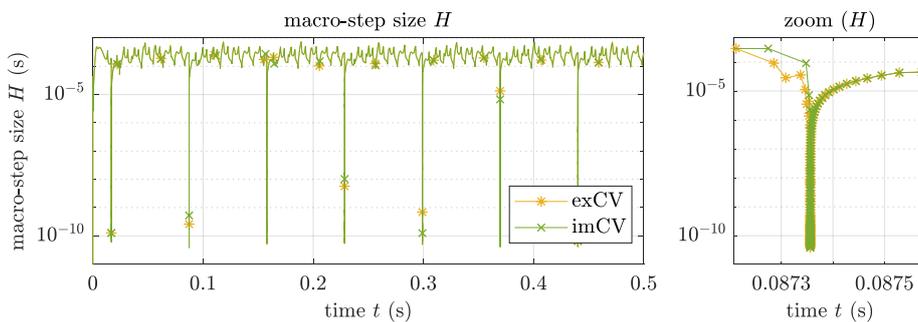
**Figure 5.50:** Implicit co-simulation (*imMD*) of  $\mathcal{M}3$  with variable macro-step size: estimated and local error of the displacement  $x_1$  and the velocity  $v_1$  of mass  $m_1$ .



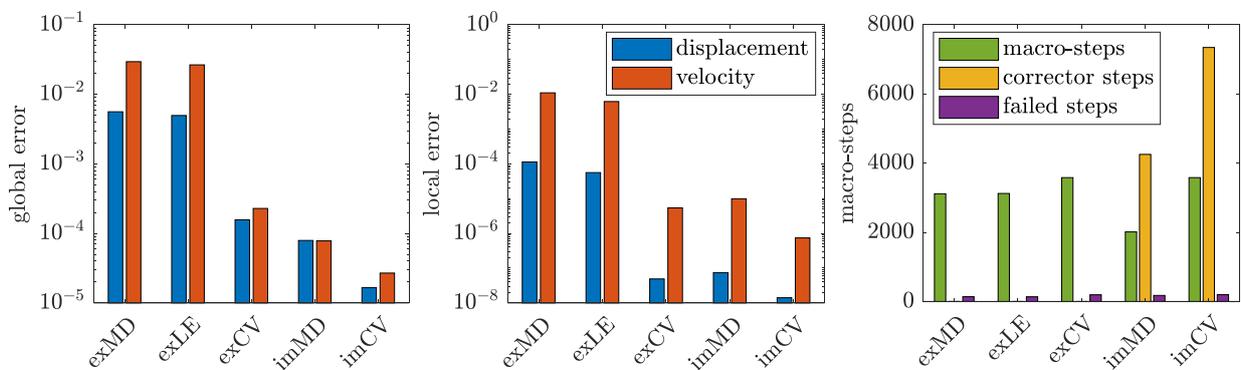
**Figure 5.51:** Implicit co-simulation (*imCV*) of  $\mathcal{M}3$  with variable macro-step size: estimated and local error of the (predicted) coupling force  $\lambda$ .



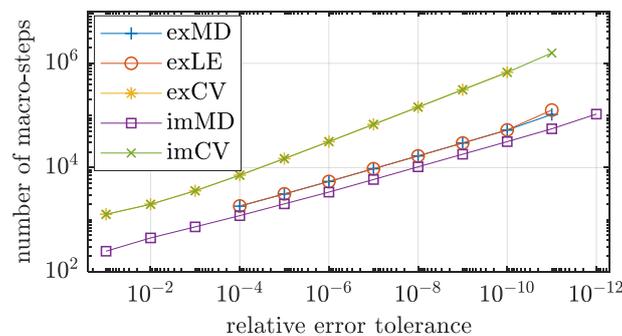
**Figure 5.52:** Two-mass oscillator with contact ( $\mathcal{M}3$ ): macro-step size determined by the macro-step size controller with the error estimators *exMD*, *exLE*, *imMD*.



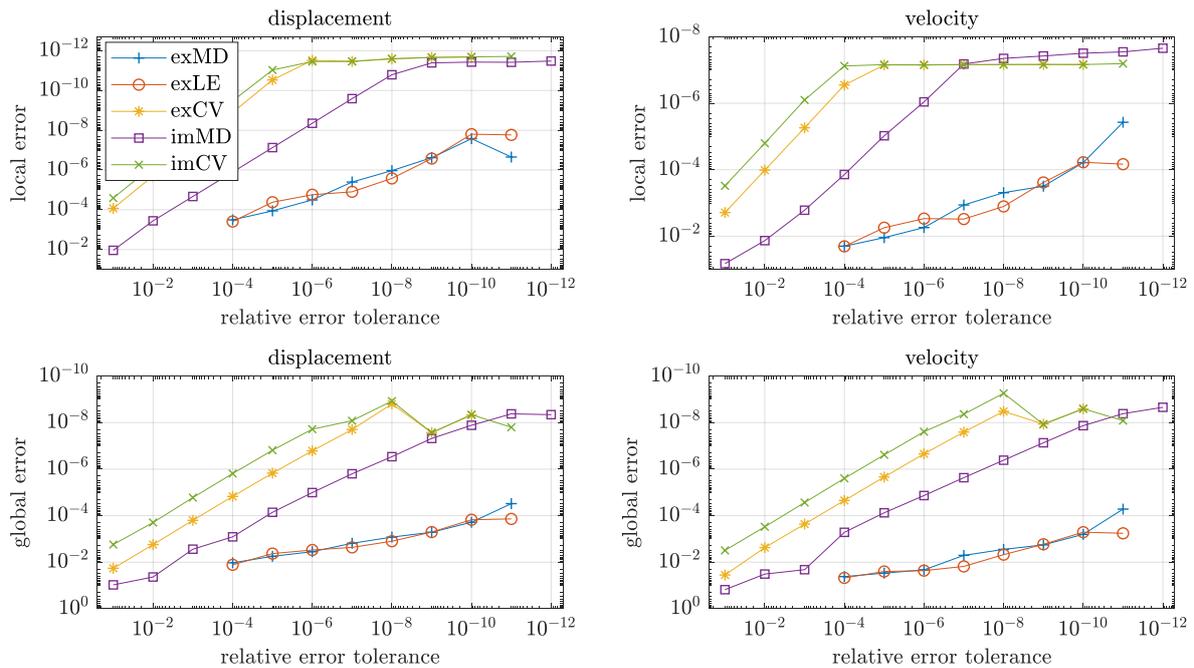
**Figure 5.53:** Two-mass oscillator with contact ( $\mathcal{M}3$ ): macro-step size determined by the macro-step size controller with the error estimators *exCV*, *imCV*.



**Figure 5.54:** Two-mass oscillator with contact ( $\mathcal{M}3$ ): resulting errors and number of macro-steps.



**Figure 5.55:** Two-mass oscillator with contact ( $\mathcal{M}3$ ): number of macro-steps depending on the error tolerances.



**Figure 5.56:** Two-mass oscillator with contact ( $\mathcal{M}3$ ): convergence analysis.

*Comment on the comparison of the five error estimators:* The results indicate that all five error estimators operate as expected and the provided local error estimates can be used successfully to control the macro-step size of a co-simulation. Therefore, the question arises which estimator should be preferred. Firstly, the explicit and the implicit approaches have to be distinguished. The choice whether an explicit or an implicit algorithm is preferable to compute a certain model depends on the properties ("stiffness") of the model and is typically not decided with regard to the error estimation method. Considering the explicit approaches, there is no significant difference between the results of the *exMD* and the *exLE* approaches with regard to the accuracy and the macro-step size identifiable. The actual question is, whether the state variables (of the coupling bodies) or the coupling variables are better suited for the error estimation.

The results of the numerical studies indicate that an estimation of the errors of the coupling variables leads to a more sensitive macro-step size controller, especially in combination with the explicit co-simulation scheme (see Figs. 5.52 and 5.53). The drawback is, that only the error of the predicted coupling variables (and not of the updated/corrected coupling variables) is estimated. Considering numerical time integration schemes, the estimation of the errors on force/acceleration level is unusual. On the other hand, the coupling variables are the essential variables of in a co-simulation. The approximation polynomials are stated in terms of the coupling variables. Also, the coupling equations within the corrector iteration (implicit scheme) are solved for the coupling variables and therefore the convergence criterion of the corrector iteration is typically defined with respect to the coupling variables. The order control algorithm, see Section 3.3, is also based on the scaled derivative norm of the approximation polynomials of the coupling variables. In addition, if the degree of the approximation polynomials is changed by the order controller, the error estimate of the current macro-step is not longer suitable to predict the local error of the following macro-step, therefore the macro-step size controller has to fall back on an alternative estimate to calculate an adequate macro-step size. The only quantity which can be computed a priori is the local truncation error of the approximation polynomials of the coupling variables.

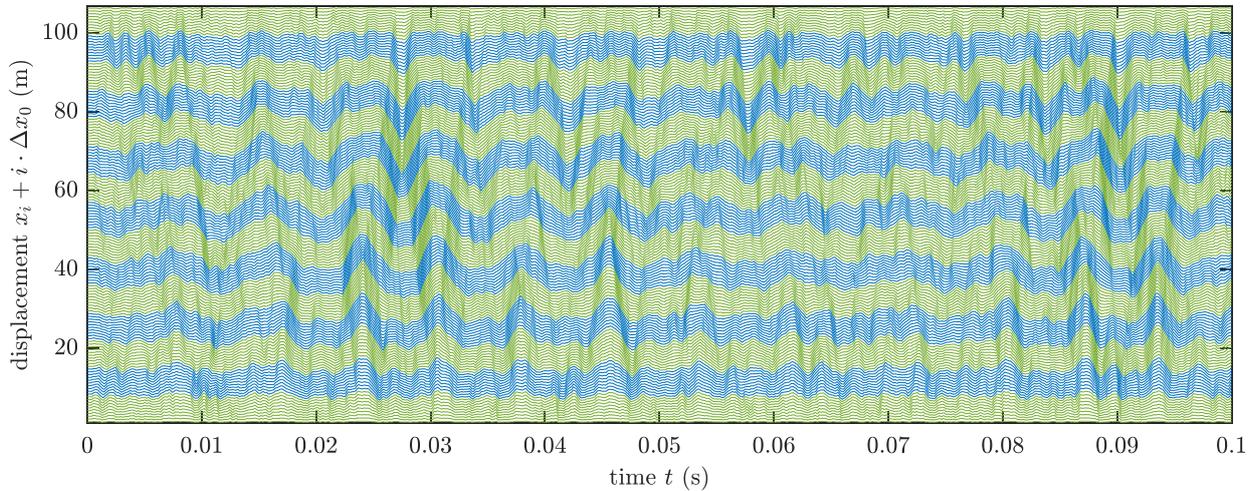
Considering all numerical studies which have been carried out in his work, both error estimation concepts show a similar performance with regard to the computation time and the resulting numerical errors. Therefore, a clear statement on the question of which error estimator should be preferred cannot be made.

### 5.3. Safety Factor

As described in Section 3.4, the safety factor  $\Gamma$  introduced in Eq. (3.27) is a parameter to tune the ratio between the number of macro-steps and the number of failed steps (local error test (Eq. (3.22)) misses). The following numerical study shows the effect of the safety factor  $\Gamma$  on the macro-step size and the number of failed steps. The model parameters are given in Table 5.3. The modified sinus force defined in Eq. (2.14) is applied to 50 % of the bodies. The parameters of the forces are chosen randomly within the ranges  $\Delta F_{S,i} = \pm [1.0e6, 1.0e7]$  N and  $\Omega_{S,i} = [5.0e2, 5.0e3]$  s<sup>-1</sup>. The phase shift is  $\varphi_{S,i} = 0$  and the exponent is set to  $A_S = 95$ . To visualize the system's dynamics, the displacements are shown in Fig. 5.57 with an artificial offset  $\Delta x_0$ . Each line in the figure is the displacement of a mass; the colors indicate the different subsystems.

**Table 5.3:** Test model parameters.

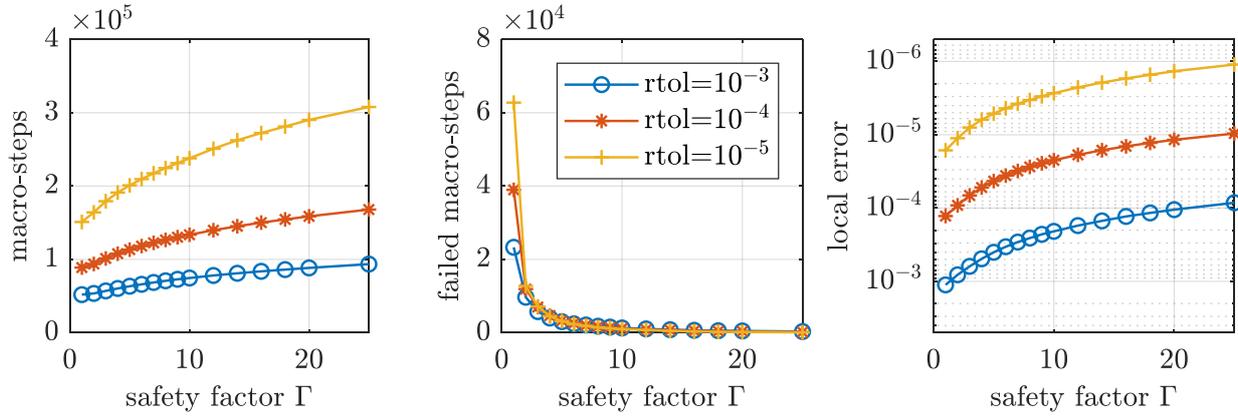
$n_K$	150
$n_s$	15
$t_{sim}$	1.0e-1 s
$m_i$	5.0e-2 kg
$c_i$	2.5e6 N/m
$d_i$	2.5e1 Ns/m
$C_i$	1.0e10 N/m <sup>3</sup>
$D_i$	1.0e-4 Ns <sup>3</sup> /m <sup>3</sup>
$\kappa$	2



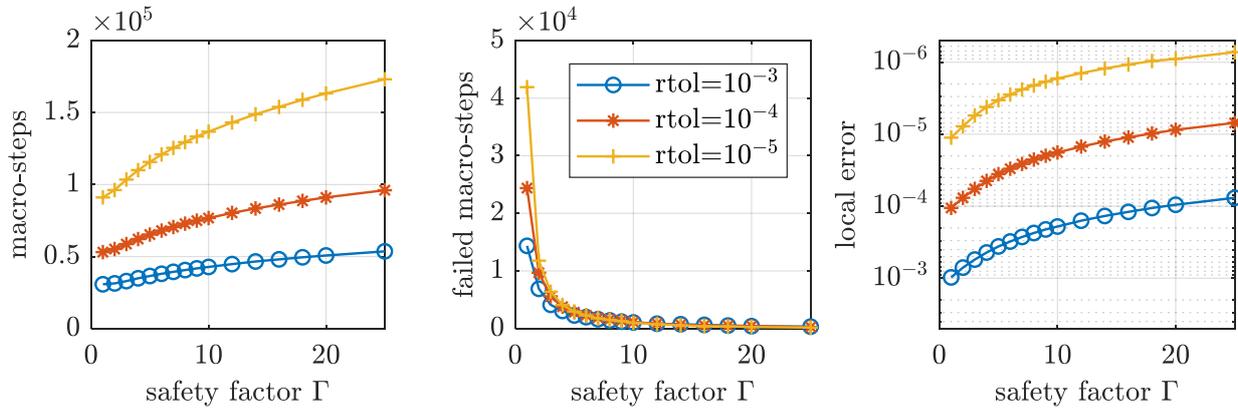
**Figure 5.57:** Visualization of the displacements of all masses with the offset  $\Delta x_0 = 0.70$  m; subsystems are indicated by two different colors.

The model is computed by an explicit and by an implicit co-simulation approach as described in Section 2.4 with quadratic approximation polynomials for the coupling variables. The macro-step size is controlled by using the error estimators *exMD* and *imMD* described in Section 3.1. The effect of the safety factor should be isolated, therefore a dead-zone (see Section 3.4) is not considered here, because it would also affect the macro-step size and the number of failed macro-steps. The limits for the scaling factor  $r$  of the macro-step size between two consecutive macro-steps is set to  $[r_{min}, r_{max}] = [0.5, 1.5]$ . Simulations are carried out with three different sets of error tolerances using  $atol^{pos} = 10^{-4} \cdot rtol$  and  $atol^{vel} = rtol$  for the absolute error tolerance for displacement and velocity variables. The simulation results, namely the number of macro-steps,

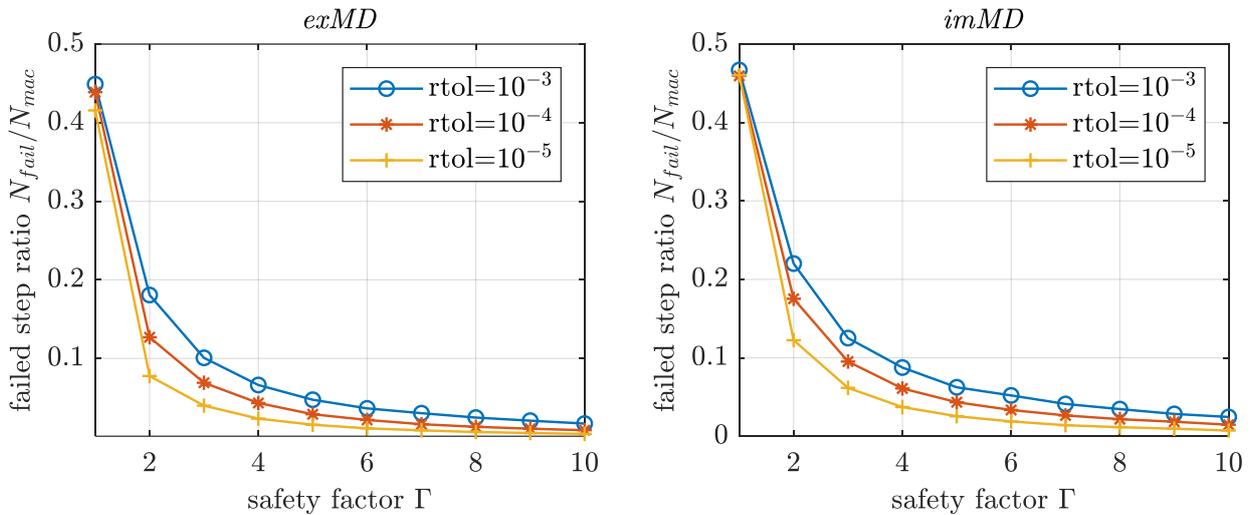
the number of failed macro-steps and the local error (NRMSE) of the coupling bodies, are shown in Fig. 5.58 and Fig. 5.59.



**Figure 5.58:** Total number of macro-steps, total number of failed macro-steps and local error of the explicit co-simulation (*exMD*) for the parameter set given in Table 5.3.



**Figure 5.59:** Total number of macro-steps, total number of failed macro-steps and local error of the implicit co-simulation (*imMD*) for the parameter set given in Table 5.3.

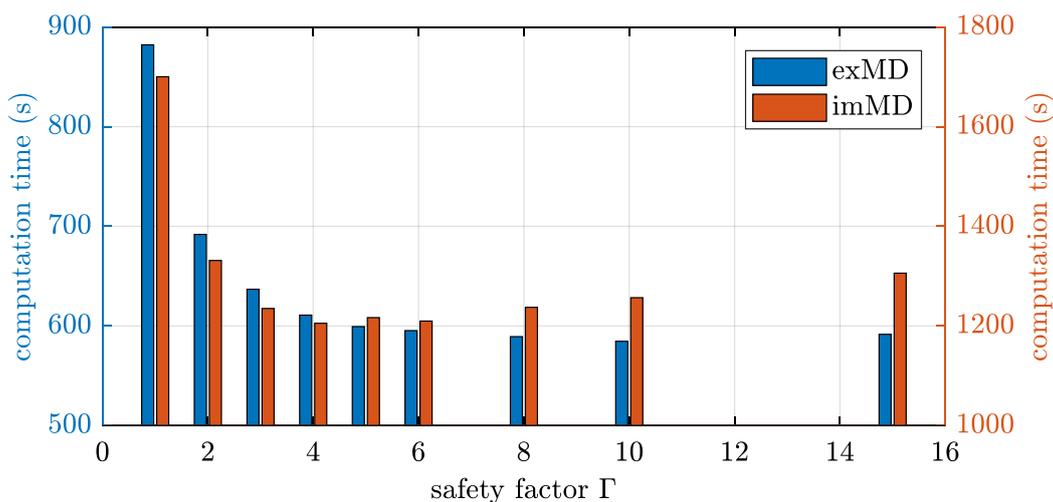


**Figure 5.60:** Failed step ratio  $N_{fail}/N_{mac}$  of the explicit and the implicit co-simulation approach for the parameter set given in Table 5.3.

The number of macro-steps of the explicit approach is, as expected, larger compared to the implicit approach, but apart from that, the results look very similar. The number of macro-steps increases and the error decreases as the safety factor  $\Gamma$  increases. The number of failed macro-steps decreases rapidly for increasing values of  $\Gamma$  within the range  $\Gamma \in [2, 6]$ .

The ratio of the number of local error test misses divided by the number of macro-steps (failed step ratio) is shown in Fig. 5.60. It can be seen that without using a safety factor ( $\Gamma = 1$ ), the failed step ratio is about 45% for the two co-simulation approaches. Hence, if the macro-step size controller is applied without using a safety factor, almost every other macro-step is repeated. When the safety factor is increased to  $\Gamma = 6$ , the failed step ratio is 1% - 5%, depending on the error tolerances of the macro-step size controller.

To examine the effect of the safety factor on the overall computation time of the co-simulation, the model is scaled up to  $n_K = 100\,000$  masses and decomposed into  $n_s = 100$  subsystems of equal size. The other parameters given in Table 5.3 are left unchanged. The relative error tolerance of the co-simulation is set to  $\text{rtol} = 10^{-4}$ . For this particular parameterization of the test model, the explicit co-simulation approach is more efficient than the implicit approach. As shown in Fig. 5.61, the *imMD* method (right y-axis, red bars) takes about twice as long as the *exMD* method (left y-axis, blue bars) to compute the model.



**Figure 5.61:** Computation time of the explicit (blue) and the implicit (red) co-simulation approaches depending on the safety factor  $\Gamma$ .

The computation time of both co-simulation approaches can be reduced significantly by selecting an adequate value for the safety factor  $\Gamma$ . For all applied co-simulation schemes, the computation time can be decreased by about 30% if a safety factor in the range  $\Gamma \in [4, 6]$  is used instead of  $\Gamma = 1$ . The plot shows that if  $\Gamma$  is varied in the range  $\Gamma \in [3, 8]$  the computation time of both approaches is only slightly influenced. This is related to the fact that an increased number of macro-steps does not necessarily imply an increase in the computation time, as described in Section 4.4. When  $\Gamma$  is increased further, it has a negative effect on the computation time of the implicit approach. It should be stressed that these results are model dependent. Further studies, not presented here, indicate that a safety factor in the range  $\Gamma \in [3, 6]$  might in general be an adequate choice.

## 5.4. Error Tolerances

The relation between macro-step size and computation time of a co-simulation model has been discussed in detail in Section 4.4. It has been explained that a reduced macro-step size may result in a decreased computation time and vice versa. Hence, also tighter error tolerances which lead to a decreased average macro-step size, may result in a more efficient computation. This statement holds only if the subsystem integration processes are the dominating part of the computation time and if the average macro-step size is not significantly smaller than the (unrestricted) average subsystem solver step size. Numerical studies are carried out with two different co-simulation models to analyze the relation between the error tolerances of the macro-step size controller and the computation time. The following two models are simulated:

- 1) A linear oscillator chain with  $n_K = 1000$  masses is simulated with two different model decompositions, namely a decomposition into  $n_s = 5$  subsystems of equal size and a decomposition into  $n_s = 100$  subsystems of equal size.
- 2) Two identically parametrized nonlinear models with  $n_K = 290$  and  $n_K = 29\,000$  masses are split into  $n_s = 11$  subsystem.

For both models, the ratio of the subsystem size to the number coupling variables is varied, in order to expose different parts of the overall computation time.

**1) Fixed model size – varying number of subsystems.** A linear oscillator chain with a free end is used as test model. The chain is uniformly elongated through the initial conditions. The stiffness coefficients are defined by  $c_i = 2.0e11 \frac{N}{m} \cdot [1.0 + 9.0 \cdot \sin(0.3142 \cdot ((i - 1) \bmod 10))]$  for  $i = 1, \dots, 1000$  so that the stiffness distribution is half-sine-shaped with a period length of 10 elements as shown in Fig. 5.62 (right figure). The parameterization of the model is given in Table 5.4. The displacements of the masses are visualized in Fig. 5.62 (left figure).

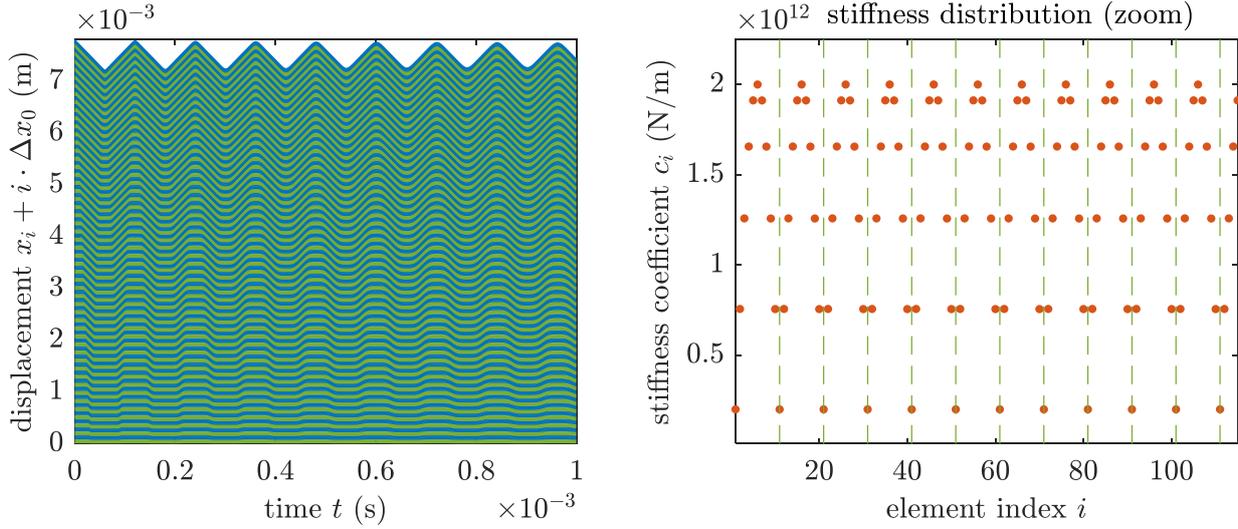
Simulations are carried out for two different model decompositions: the system is split into  $n_s = 5$  and into  $n_s = 100$  subsystems. In each case, the subsystems are of equal size and the coupling elements have the lowest values of the system's stiffness coefficients ( $c_c = 2.0e11 \frac{N}{m}$ ).

Cutting the model at elements with low stiffness properties is advantageously with respect to the overall computation time. The two model decompositions differ significantly with regard to the resulting co-simulation attributes. In the first case, the number of subsystems  $n_s = 5$  and the number of coupling variables  $n_c = 4$  is small and the subsystem size is moderate with 200 degrees of freedom per subsystem. In the second case, the subsystem size is very small with 10 degrees of freedom and the number of subsystems  $n_s = 100$  is rather large, at least in relation to the model size.

The purpose of the numerical studies with the two described model decompositions is to examine the influence of the error tolerances of the macro-step size and order control algorithm on different parts of the computation time that are discussed in Section 4.4. The value of the relative error tolerance is varied within the range  $\text{rtol} = [10^{-2}, \dots, 10^{-6}]$ ; the absolute tolerances are selected depending on the relative error tolerance according to  $\text{atol}^{\text{pos}} = 10^{-3} \cdot \text{rtol}$ ,  $\text{atol}^{\text{vel}} = \text{rtol}$  and  $\text{atol}^u = 10^4 \cdot \text{rtol}$ .

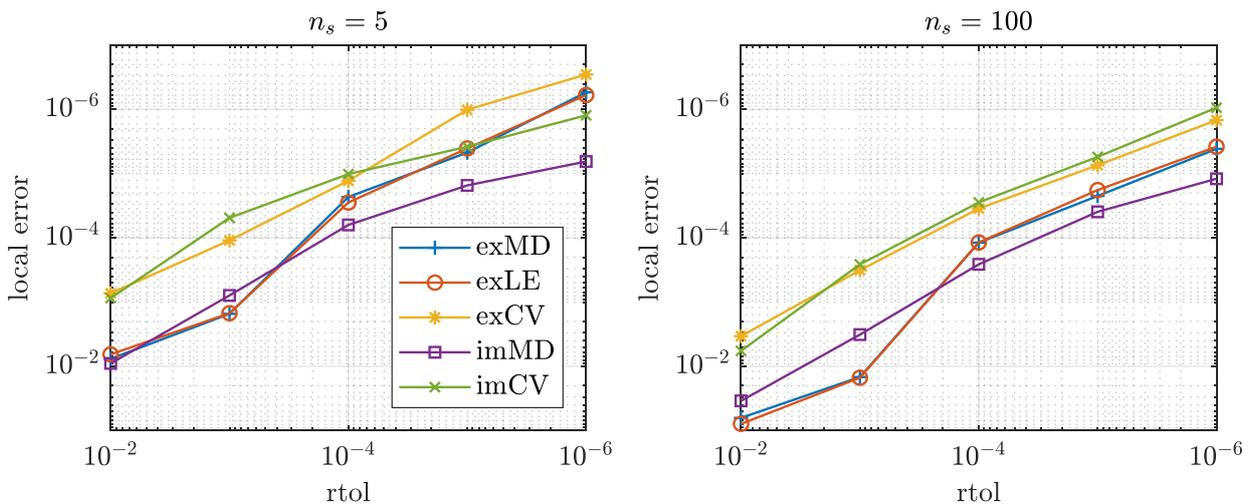
**Table 5.4:** Linear test model parameters.

$n_K$	1000
$n_s$	5 / 100
$t_{sim}$	$1.0e-3$ s
$m_i$	$7.5e-4$ kg
$d_i$	$2.5e4$ Ns/m
$x_{i,0}$	$i \cdot 3.0e-7$ m
$v_{i,0}$	0.0 m/s



**Figure 5.62:** Left figure: Visualization of the displacements with the offset  $\Delta x_0 = 7.5e-6$  m, subsystems are indicated by two different colors. Right figure: half-sine-shaped stiffness distribution, dashed green lines indicate the decomposition into 100 subsystems.

The resulting errors, namely the local error of all state variables (NRMSE), obtained by co-simulations using the different error estimators introduced in Section 3.1 are shown in Fig. 5.63. The macro-step size and order control algorithm works as expected and the resulting errors match quite well with the defined error tolerances. It can be seen that the resulting errors obtained by simulations using the error estimators *exCV* and *imCV* are slightly smaller than the errors obtained by simulations with the error estimators *exLE*, *exMD* and *imMD*. Also the resulting errors of the co-simulations with  $n_s = 100$  subsystems are slightly increased compared to the results obtained by the decomposition in  $n_s = 5$  subsystems. A detailed study of the resulting errors of co-simulations incorporating the macro-step controller using the different error estimators can be found in Section 5.2.



**Figure 5.63:** Local error (NRMSE) of all state variables depending on the error tolerances for  $n_s = 5$  and  $n_s = 100$ .

The bar plots in Fig. 5.64 and Fig. 5.65 show the computation time of the explicit and implicit co-simulation approaches depending on the error tolerances of the macro-step size and order

control algorithm. The results of the model with  $n_s = 5$  subsystems and the model with  $n_s = 100$  subsystems are depicted separately. The colors indicate the time spent on the main processes of the computation (see Section 4.4):

- data exchange between the co-simulation interface and the subsystems (*mpi*)
- co-simulation method specific computations of the co-simulation interface (*co-sim*)
- save/reload of subsystem solver workspace in connection with macro-step repetitions (*sws dump*)
- subsystem integration, subdivided into  $T^{\text{solv}}$  (*solver*) and  $T^{\text{idle}}$  (*solver (idle)*) as defined in Eq. (4.5)
- generation of output files (*output*).

The time spent on writing output files is negligible and only shown for the sake of completeness.

The interpretation of the results of the co-simulation model with 100 subsystems is straightforward. The subsystems are very small with only 10 masses each. Therefore, the subsystem computation time and also the time for the solver workspace dump are negligible. The overall computation time is dominated by the data exchange and by the computations of the co-simulation interface. Both of these tasks have a clear relation to the macro-step size: the computational effort increases linearly with the number of macro-steps. For tighter error tolerances, the average macro-step size is decreased. Therefore, the overall computation time behaves in the same way as it would be expected for classical ODE solvers: more accurate results require a higher computational effort.

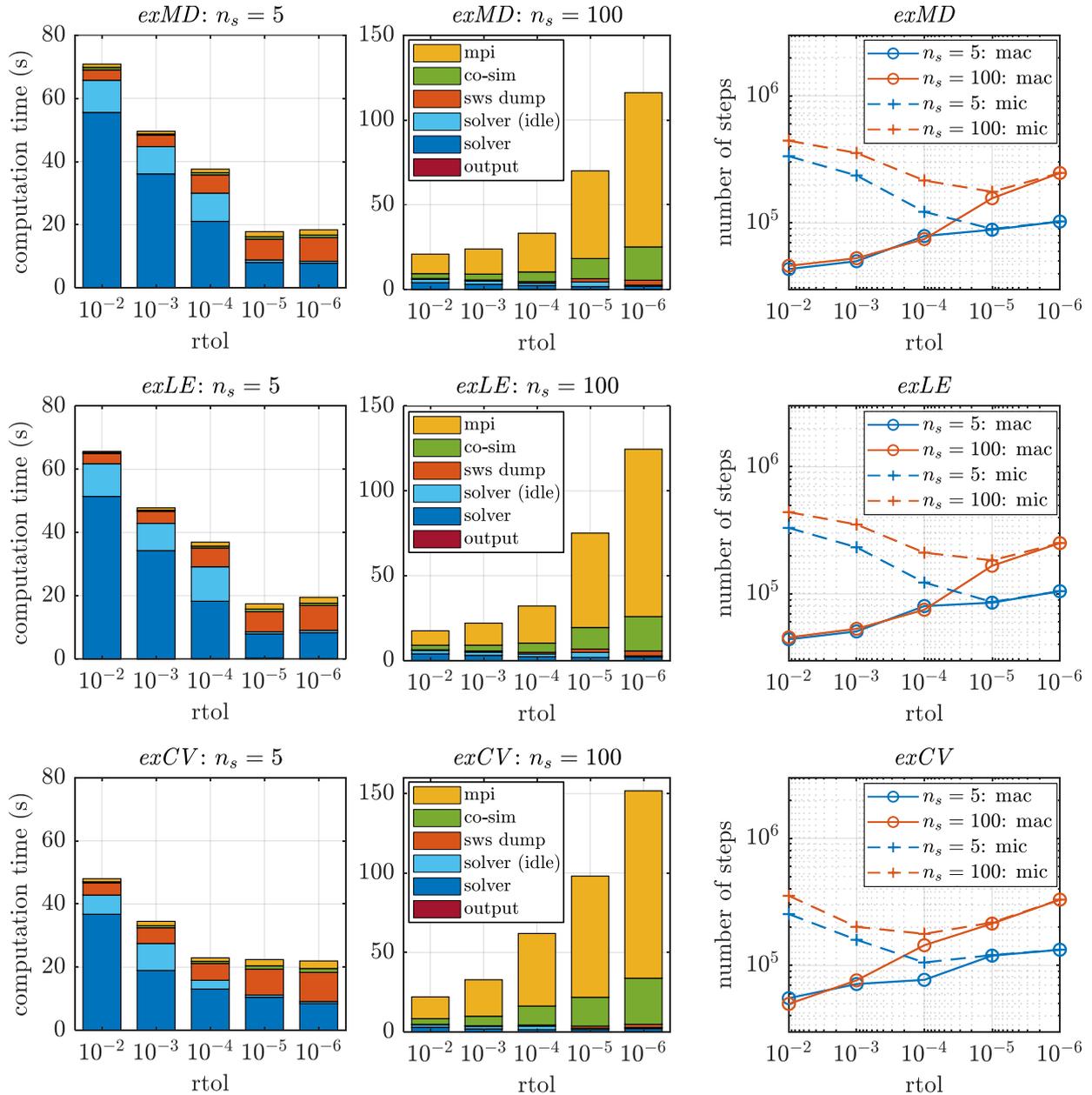
Following the same argumentation, the implicit co-simulation approach is more efficient than the explicit method for the model with  $n_s = 100$  subsystems. Regarding the simulations with  $\text{rtol} = 10^{-6}$  as an example, about  $\approx 2.5e5$  macro-steps are accomplished by the *exMD* approach. The *imMD* approach requires only about  $\approx 1.6e4$  macro-steps (predictor steps) plus  $\approx 3.6e4$  corrector iterations and is therefore about 5 times faster. It should be mentioned that a predictor/corrector step of the implicit co-simulation approach is, from the computational point of view, almost the same as a macro-step of the explicit approach. The number of failed steps is very small ( $\approx 2.0e2$  for *exMD* and  $\approx 5.0e2$  for *imMD*), hence the effect on the computation time can be neglected.

The segmentation of the computation time of the co-simulations with  $n_s = 5$  subsystems is completely different. The time spent on data exchange and on the computations carried out by the co-simulation interface can be neglected. The major parts are, independent of the applied co-simulation method, the subsystem solving process and the solver workspace dump. The solver workspace copy process is carried out once in each macro-step (in each corrector iteration step), therefore the computational effort increases linearly with the number of macro-steps (corrector iteration steps).

Considering the explicit co-simulation approaches, the subsystem solver computation time decreases for tighter error tolerances. As can be seen in the right plots of Fig. 5.64, the average micro-step size (subsystem solver step size) increases as the average macro-step size decreases, until the micro-step size gets restricted by the macro-step size (at  $\text{rtol} \approx 1.0e-5$ ). This phenomenon

is described in Section 4.4 and can be explained by the jumps in the approximation polynomials of the coupling variables, which occur at the macro-time points.

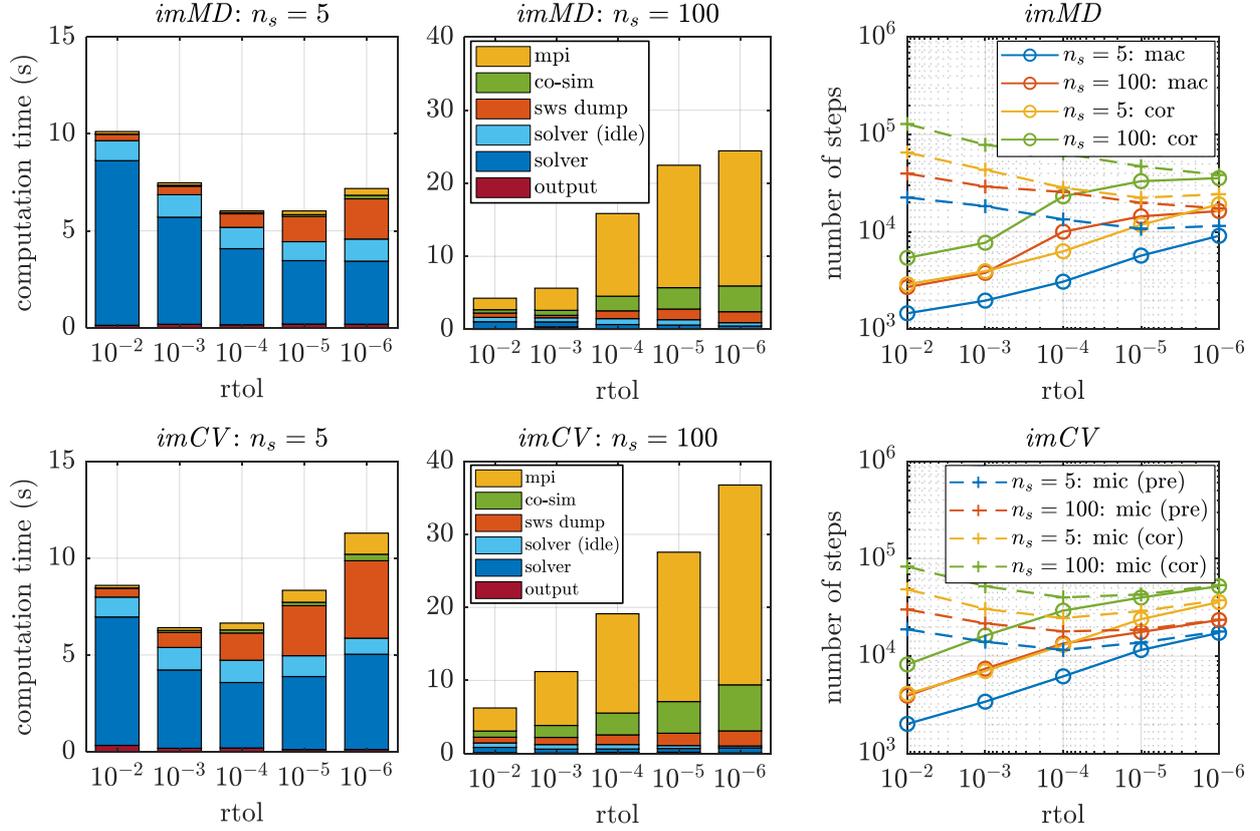
Although the computational effort of the solver workspace dump increases with tighter error tolerances, the overall computation time of the explicit methods decreases because the dominating part, namely the subsystem solver time, is reduced. When the relative error tolerance is set to  $\text{rtol} = 1.0\text{e-}6$  both processes require almost the same computation time.



**Figure 5.64:** Explicit co-simulation approaches: computation time depending on the error tolerances for  $n_s = 5$  and  $n_s = 100$ . Right figures: number of macro-steps and average number of subsystem solver steps (micro-steps).

The computation time of the implicit co-simulation approaches shows a similar behavior for the model with  $n_s = 5$  subsystems, as shown in Fig. 5.65. However, the approximation polynomials are  $C^0$ -continuous at the macro-time points (for  $\kappa > 0$ ) because of the corrector iteration; jumps only occur in the derivatives. Therefore, the effect of the reduced macro-step size on the average micro-

step size is not as strongly observable as for the explicit co-simulation approaches. For very tight error tolerances of the macro-step size controller, the overall computation time increases because of the increased effort for the solver workspace dump. The implicit co-simulation methods show a better computational efficiency than the explicit approaches for the considered model, although the method specific computational overhead is generally higher for implicit approaches.



**Figure 5.65:** Implicit co-simulation approaches: computation time depending on the error tolerances for  $n_s = 5$  and  $n_s = 100$  subsystems. Right figures: number of macro-steps, corrector iterations and average number of subsystem solver steps (micro-steps).

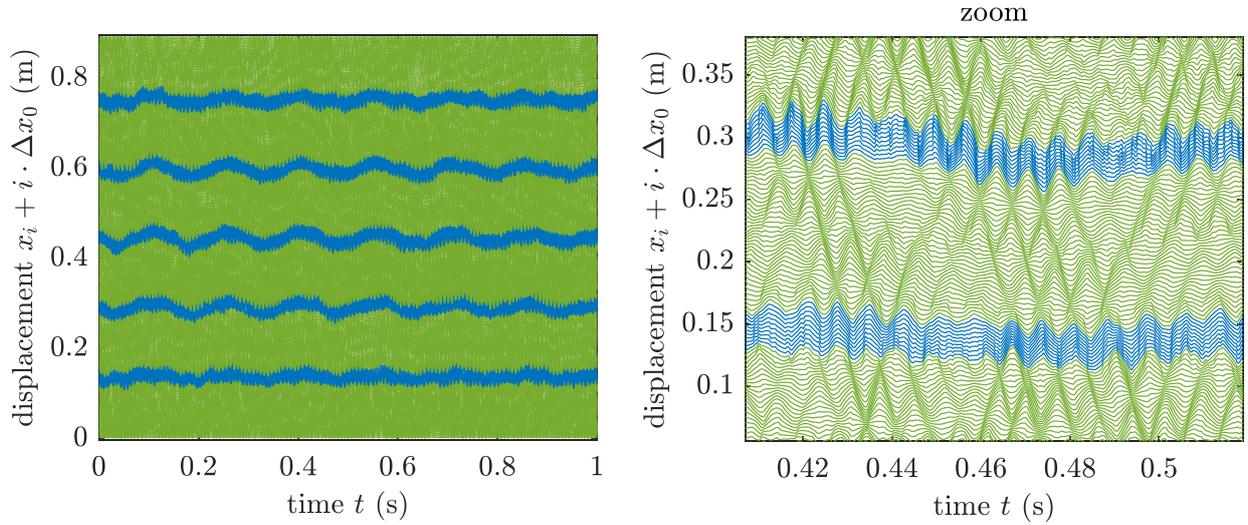
**2) Varying model size – fixed number of subsystems.** A second study to investigate the influence of the error tolerances of the macro-step size controller on the computation time of the co-simulation approaches is carried out with a nonlinear parameterization of the test model, as given in Table 5.5. Simulations are carried out for two model sizes, namely a small model with  $n_K = 290$  masses and a large model with  $n_K = 29\,000$  masses. Each model is decomposed into  $n_s = 11$  subsystems, in a way that subsystems with odd indices contain 40 (4000) masses with  $m_i = 1.0e1$  kg and subsystems with even indices contain 10 (1000) masses with  $m_i = 1.0e-1$  kg.

The initial conditions for each mass are selected randomly within the intervals  $[-1.0e-4, 1.0e-4]$  m and  $[-1.0e-2, 1.0e-2]$  m/s. Randomly picked 20 % of the masses are excited by a modified sinus force according to Eq. (2.14). The force amplitude  $\Delta F_{S,i}$  is chosen randomly within the interval  $\pm[1.0e5, 1.0e6]$  N;  $\Omega_{S,i}$  is selected randomly

**Table 5.5:** Nonlinear test model parameters.

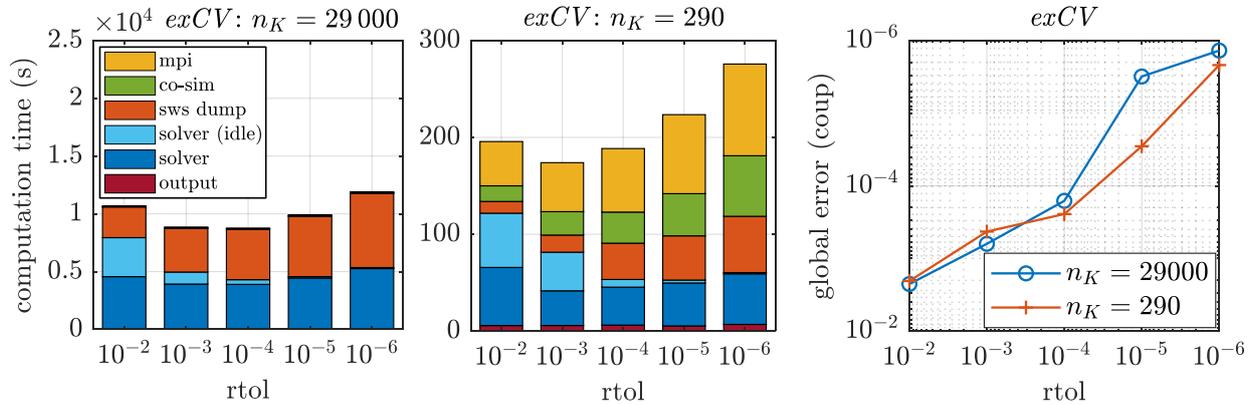
$n_K$	290 / 29 000
$n_s$	11
$t_{sim}$	1.0 s
$c_i$	$1.0e7$ N/m
$d_i$	$1.0e3$ Ns/m
$C_i$	$1.0e14$ N/m <sup>3</sup>
$D_i$	$1.0e4$ Ns <sup>3</sup> /m <sup>3</sup>

within the interval  $[7.5e2, 1.0e3] s^{-1}$ , the exponent is set to  $A_S = 95$ . The displacements of the small system ( $n_K = 290$ ) are visualized in Fig. 5.66.



**Figure 5.66:** Visualization of the displacements with the offset  $\Delta x_0 = 3.1e-3$  m of the model with  $n_K = 290$  masses; subsystems are indicated by two different colors.

Simulations are carried out using the explicit (*exCV*) and the implicit (*imCV*) co-simulation approach. The value of the relative error tolerance is varied within the range  $rtol = [10^{-2}, \dots, 10^{-6}]$ . The absolute tolerances are selected depending on the relative error tolerance according to  $atol^u = 10^4 \cdot rtol$ . The resulting computation times and global errors of the coupling bodies are shown in Fig. 5.67 and Fig. 5.68.

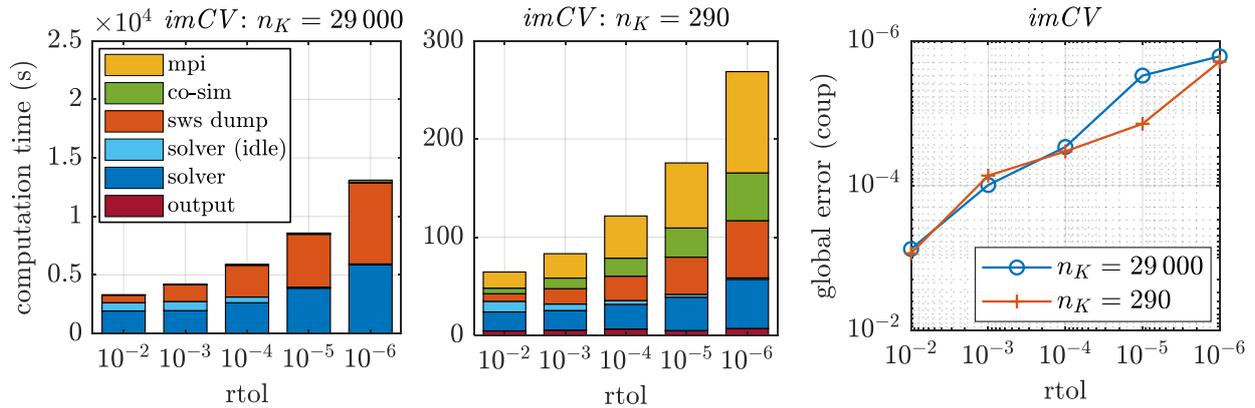


**Figure 5.67:** Explicit co-simulation approach (*exCV*): computation time depending on the error tolerances for  $n_K = 29\,000$  and  $n_K = 290$ . Right figure: global error (NRMSE) of the state variables of the coupling bodies.

It can be seen that the computation time of the large model with  $n_K = 29\,000$  masses is defined by two main parts: the time spent on the subsystem integrations and the time required for the solver workspace dump. For tighter error tolerances of the macro-step size controller, the idle-time  $T^{\text{idle}}$  (see Eq. (4.5)) of the subsystems decreases and the subsystem integration time remains almost constant for the explicit co-simulation approach. When the implicit co-simulation approach is applied, the computation time of the subsystem solver increases for tighter error tolerances. The time for the solver workspace dump increases as the number of macro-steps increases and

becomes the major part of the overall computation time for very tight tolerances.

The simulations of the smaller model ( $n_K = 290$ ) show a more diverse segmentation of the overall computation time. For both co-simulation approaches, the computational effort of the co-simulation interface, the data transfer and the time spent on the solver workspace dump increase for tighter error tolerances. As for the large model, the idle-time vanishes for tight error tolerances. Due to the contrasting development of the different parts of the computation time with respect to the error tolerances, the explicit co-simulation approach shows the interesting behavior that the resulting computation time remains almost constant for a wide range of the error tolerances. Applying the implicit approach, the idle-time is insignificant. Therefore, the overall computation time increases continuously for tighter error tolerances.



**Figure 5.68:** Implicit co-simulation (*imCV*): computation time depending on the error tolerances for  $n_K = 29000$  and  $n_K = 290$ . Right figure: global error (NRMSE) of the state variables of the coupling bodies.

*Remark on the model decomposition:* The considered co-simulation model is an example of exploiting the multirate attribute of the co-simulation methods. Subsystems with small masses  $m_i = 1.0e-1$  kg and higher frequencies and therefore a smaller subsystem solver step size have a lower degree of freedom than subsystems with  $m_i = 1.0e1$  kg and a therefore larger solver step size. Considering the implicit (*imCV*) simulation of the larger model ( $n_K = 29000$ ) with  $\text{rtol} = 1.0e-3$  as an example, the average micro-step size of the small subsystems is  $\bar{h}_{mic}^S = 6.31e-7$  s but the average micro-step size of the large subsystems is  $\bar{h}_{mic}^L = 2.59e-6$  s. The solver of the small subsystems needs  $\bar{h}^L/\bar{h}^S \approx 4$  times more micro-steps than the solver of the large subsystems. The ratio of the degrees of freedom between the large and the small subsystems is also  $4000/1000 = 4$ . Under the assumption of a linear scaling of the computation time of a subsystem with respect to the degree of freedom, which is given in this case, this is the ideal partitioning to obtain almost equal computation times for all subsystems.

An even more efficient computation could be achieved, if the partitioning is modified in a way, that one mass with  $m_i = 1.0e1$  kg is shifted from each larger subsystem to each neighboring smaller subsystem. The consequence of this modification would be that the system is cut through spring/damper-elements connecting two masses with  $m_i = m_{i+1} = 1.0e1$  kg, instead of a mass with  $m_i = 1.0e1$  kg and the other with  $m_{i+1} = 1.0e-1$  kg. Then the mass ratio of the coupling bodies is  $m_r = 1$  instead of  $m_r = 100$ , which allows a larger macro-step size especially if an explicit co-simulation scheme is applied. Due to the described restructuring, the total number of macro-steps of the implicit (*imCV*) simulation of the smaller model ( $n_K = 290$ ) with  $\text{rtol} = 1.0e-5$  is reduced

from 708 876 to 319 041. The resulting computation time is reduced from 176 s to 91 s. Considering the explicit approach, the effect of the restructuring is even stronger. The number of macro-steps of the explicit (*exCV*) simulation of the smaller model ( $n_K = 290$ ) with  $\text{rtol} = 1.0e-5$  is reduced from 2 941 675 to 334 829 and the resulting computation time is reduced from 224 s to 36 s. In other words, a speed-up factor of  $> 6$  is achieved by choosing a more sophisticated decomposition of the model.

## 5.5. Asymmetric Coupling Properties

The objective of the following study is to examine the responsiveness of the different co-simulation approaches to asymmetric coupling properties in terms of the mass ratio  $m_r = \frac{m_2}{m_1}$  of the coupling bodies. The situation of large mass ratios may occur, if a discretized flexible body is treated as a subsystem of a multibody system, for example. In this case an element (small mass) of the flexible body is coupled to a rigid body, which may have very different mass properties.

A linear two-mass oscillator is used as test model. The parameterization is given in Table 5.6. Mass  $m_2 = 1.0$  kg is kept constant, mass  $m_1$  is varied within the range  $[1.0e-6, \dots, 1.0]$  kg. In addition, the coupling damping  $d_c$  is varied within the interval  $[0.0, \dots, 1.0e6]$  Ns/m. Mass  $m_2$  is excited by a harmonic force (2.11) with the parameters  $\Delta F_{H,2} = 1.0e5$  N and  $\Omega_{H,2} = 5.0e1$  s $^{-1}$ .

Simulations are carried out with the explicit (*exMD*), the implicit (*imMD*) and the waveform relaxation (*waCV*) approach. The Milne Device error estimator is used with the error tolerances  $\text{rtol} = 10^{-3}$  and  $\text{atol}^{\text{pos}} = 10^{-3} \cdot \text{rtol}$ ,  $\text{atol}^{\text{vel}} = 1.0 \cdot \text{rtol}$  for the macro-step size controller. The error estimator based on the coupling variables of Section 3.1.4 is used for the waveform relaxation approach with the tolerances  $\text{rtol} = 10^{-3}$  and  $\text{atol}^u = 10 \cdot \text{rtol}$ . The relaxation parameter according to Eq. (2.35) is set to  $\omega = 0.3$ . The subsystem solver tolerances are set to  $\text{rtol}_{\text{IDA}} = 10^{-6}$  and  $\text{atol}_{\text{IDA}}^{\text{pos}} = 10^{-3} \cdot \text{rtol}$ ,  $\text{atol}_{\text{IDA}}^{\text{vel}} = \text{rtol}$ .

The resulting computation time and the total number of macro-steps are shown in Fig 5.69 for the different parameter combinations. The global errors are depicted in Fig. 5.70. It can clearly be seen that the explicit and the waveform relaxation approach become inefficient for large mass ratios in combination with high damping values.

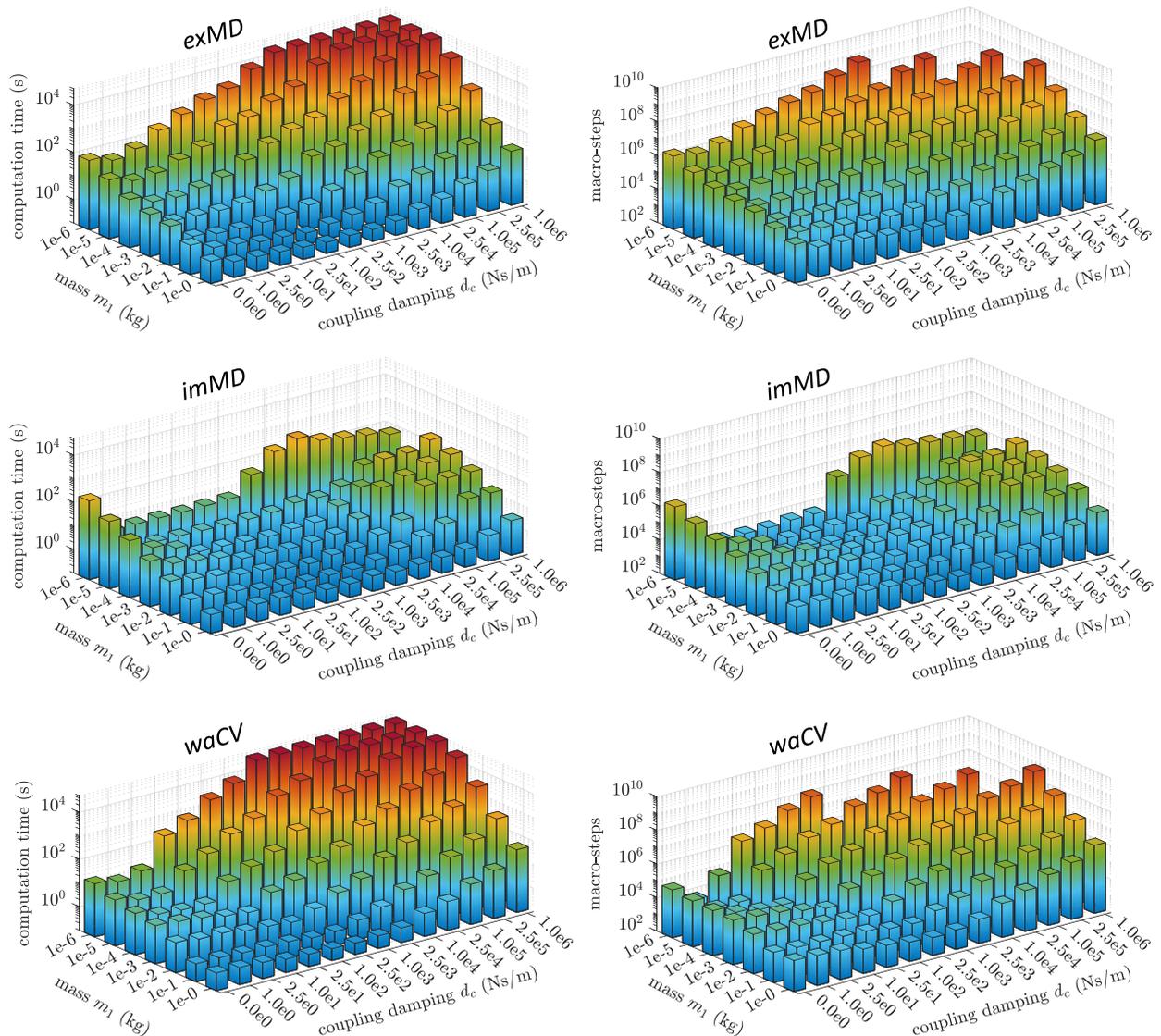
*Remark on the order control algorithm for the undamped case:* Using the order control strategy described in Listing 3.2 in combination with the explicit co-simulation approach, the order control algorithm will increase the degree of the approximation polynomials to  $\kappa = 5$ . This results in numerical instabilities. If the order is limited to  $\kappa \leq 2$  the simulation becomes stable. The more conservative order control strategy of Listing 3.1 is able to detect the instabilities and lowers the order to enable a stable simulation even for strongly asymmetric systems ( $m_r = \frac{m_2}{m_1} = 10^6$ ). The explicit co-simulations of the undamped system  $d_c = 0$  have been carried out with the order control strategy described in Listing 3.1, all others with the scheme described in Listing 3.2.

The rather large values of the global error for the undamped model can be explained by the

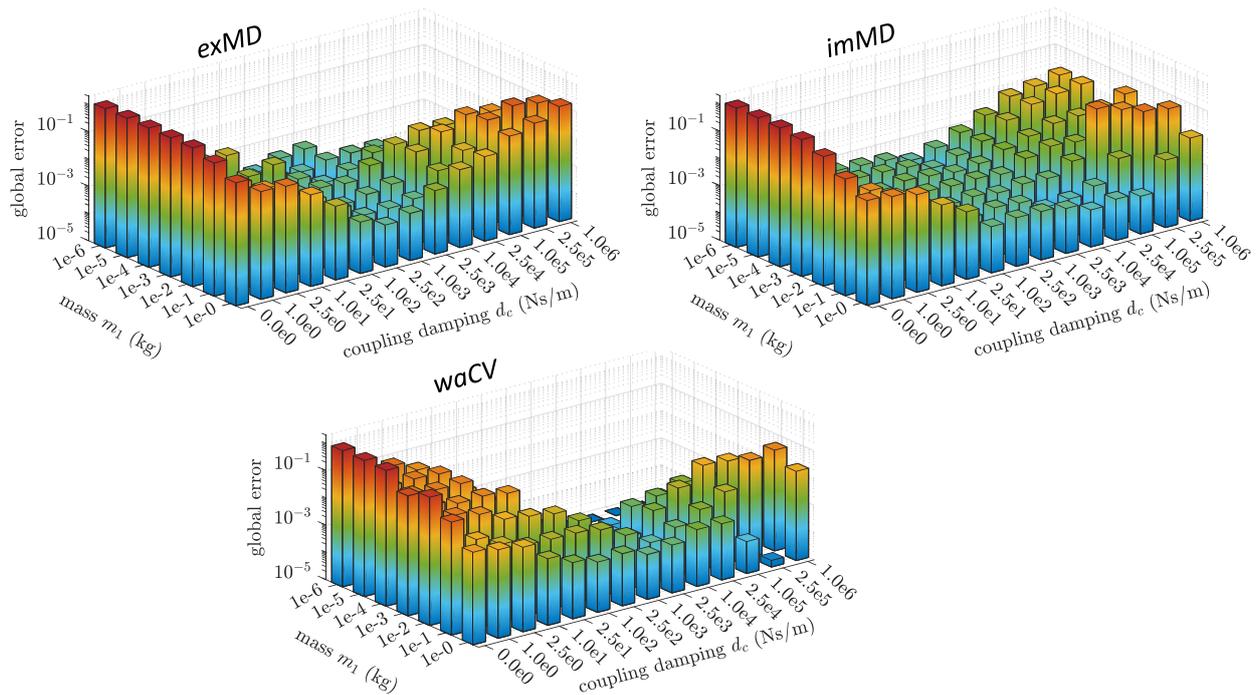
**Table 5.6:** Test model parameters.

$n_K$	2
$n_s$	2
$t_{sim}$	1.0 s
$m_2$	1.0 kg
$c_i$	1.0e6 N/m
$d_i$	0.0 Ns/m
$\mathbf{x}_0$	$[5.0e-2, 0.0]$ m
$\mathbf{v}_0$	$[0.0, 0.0]$ m/s

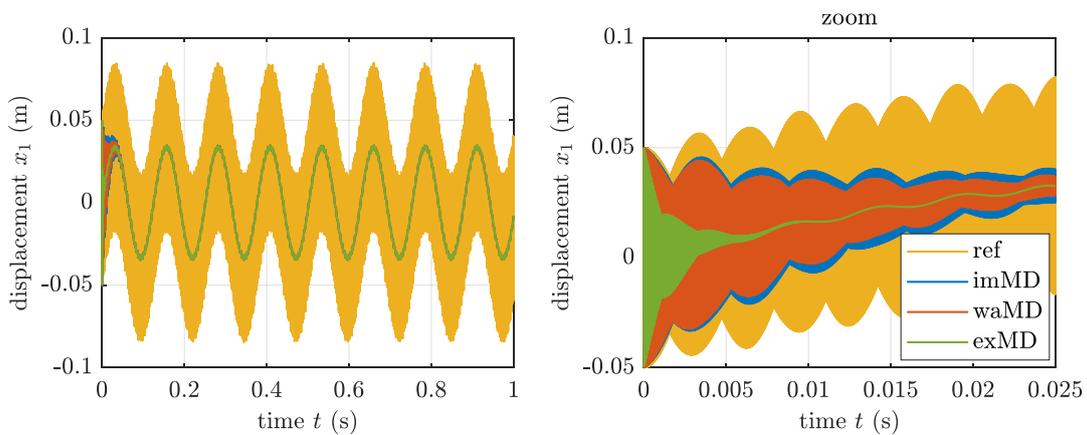
numerical damping of the co-simulation approaches and the numerical damping induced by the subsystem solver (BDF method). The high frequent oscillations of mass  $m_1$  are damped out as shown in Fig. 5.71. To avoid this effect and to fully resolve the high frequent oscillations, the error tolerances of the co-simulation have to be reduced as shown in Fig. 5.72 for the implicit approach.



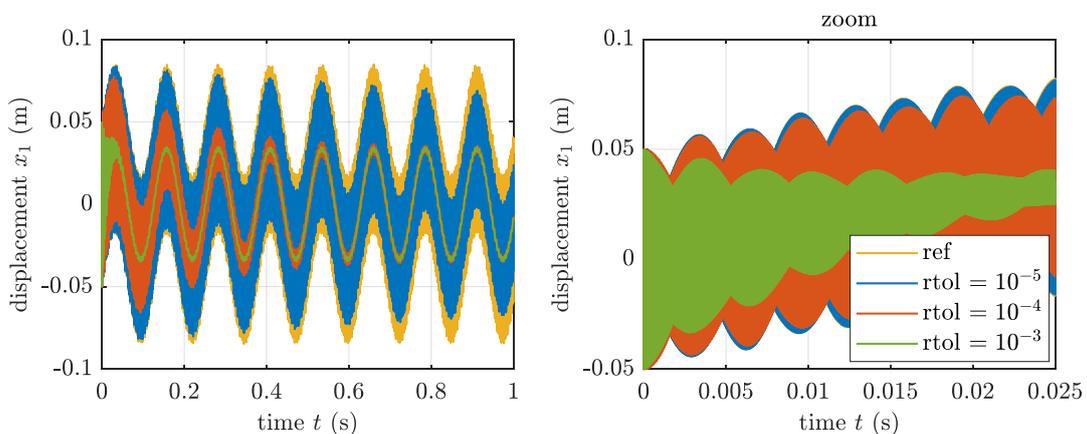
**Figure 5.69:** Computation time and number of macro-steps of the explicit (*exMD*), the implicit (*imMD*) and the waveform (*waCV*) co-simulation approach. The time limit for the computation is set to 8:30 h. Truncated (left figures) or missing (right figures) bars indicate an excess of the time limit.



**Figure 5.70:** Global error (NRMSE) of the explicit (*exMD*), the implicit (*imMD*) and the waveform (*waCV*) co-simulation. Missing bars indicate an excess of the computation time limit of 8:30 h.



**Figure 5.71:** Undamped model: displacement  $x_1$  of mass  $m_1 = 1.0e-6$  kg for  $d_c = 0$ ,  $rtol = 10^{-3}$ .

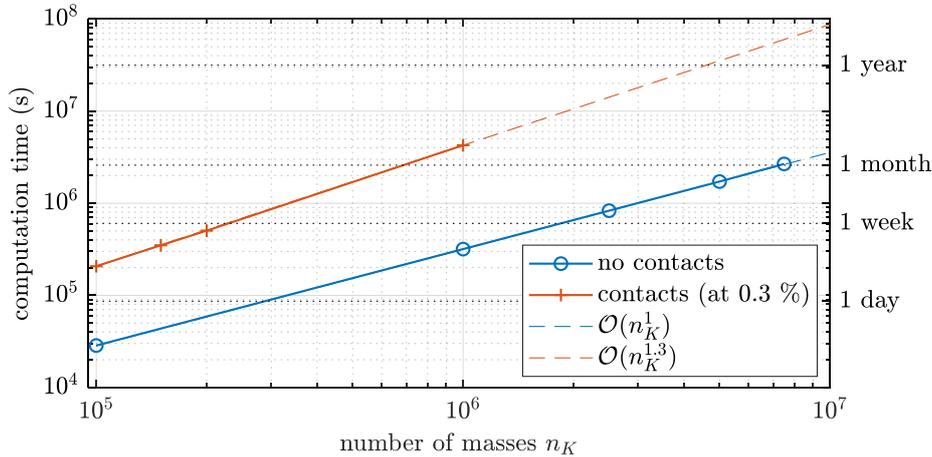


**Figure 5.72:** Implicit co-simulation (*imMD*): displacement  $x_1$  of mass  $m_1 = 1.0e-6$  kg for  $d_c = 0$ .

## 5.6. Number of Subsystems

The following numerical study deals with the subject of the co-simulation of large-scale multibody systems. The parameterization of the model is given in Table 5.7. The brackets-notation  $[a, b]$  used in the table denotes that the corresponding parameter is selected randomly within the given interval. The masses are excited by harmonic forces (2.11) defined by the parameters  $\Delta F_{H,i}$  and  $\Omega_{H,i}$  as given in Table 5.7. Simulations are carried out for three different model sizes, namely for the case  $n_K = 10^5$ ,  $n_K = 10^6$  and  $n_K = 10^7$  masses. In a second study, the simulations are carried out again for models with the same parameterization but including additional contacts within the subsystems. Contact forces according to Eq. (2.12) defined by the parameters  $A_C = -1.0e-2$  N and  $B_C = 2.5e4$  m<sup>-1</sup> are applied to randomly selected 0.3% of all masses. For the models without contacts, the average micro-step size  $\bar{h}_{mic}$  (subsystem solver step size) and the average macro-step size  $\bar{H}$  are of similar size. The models including contacts within the subsystems show a smaller average micro-step size  $\bar{h}_{mic}$ .

Firstly, monolithic computations of the models are carried out to obtain reference values for the computation time. The error tolerances used for the monolithic computation  $rtol_{IDA} = 10^{-6}$ ,  $atol_{IDA}^{pos} = 10^{-3} \cdot rtol_{IDA}$ ,  $atol_{IDA}^{vel} = rtol_{IDA}$  are the same as for the subsystem solvers of the co-simulation.

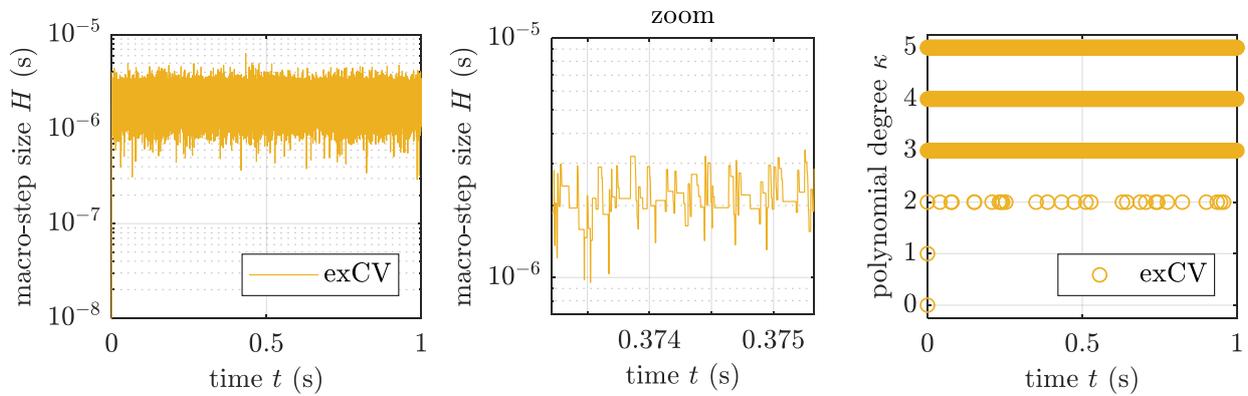


**Figure 5.73:** Computation time of the monolithic simulation.

The computation time of the models without contacts scales about linearly with the total number of masses (scaling order  $P \approx 1$ ). If contact forces are applied, the scaling order is increased to  $P \approx 1.3$ . The increased scaling order  $P$  can be explained by the model properties. The number of contacts increases with the number of masses according to  $3.0e-3 \cdot n_K$ . The occurrence of contacts causes an increased number of solver steps. Therefore, the computation time of a larger model is not only increased because of the extended number of masses, but also because of the increased number of solver steps. The resulting computation time of the monolithic simulations is

depicted in Fig. 5.73.

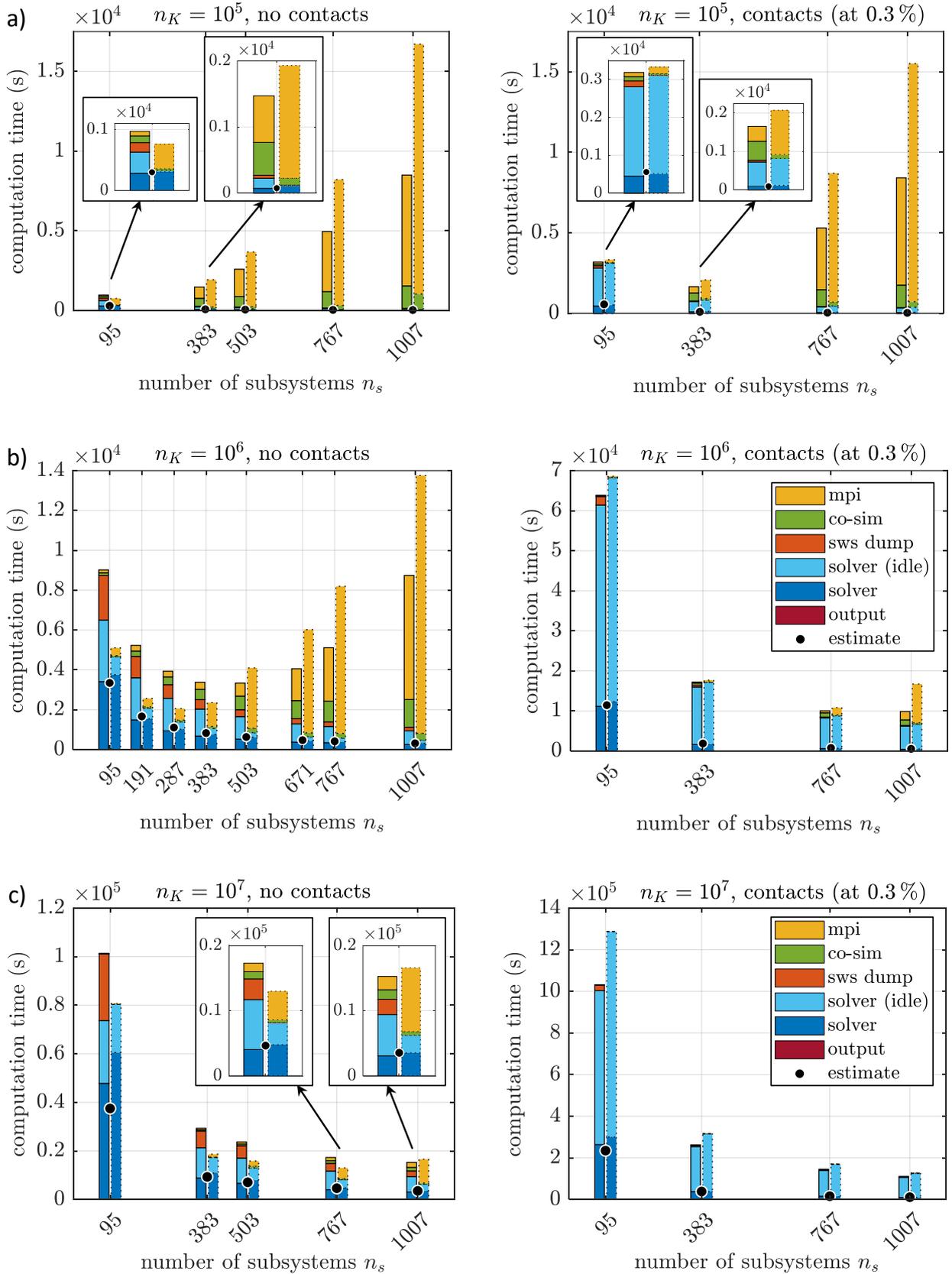
Next, the considered models are decomposed into  $n_s$  subsystems of equal size and computed by using the explicit co-simulation approach. Simulations are carried out with the macro-step size and order control algorithm (*exCV*,  $\text{rtol} = 10^{-4}$ ,  $\text{atol}^u = 10^4 \cdot \text{rtol}$ ). Also simulations with a fixed polynomial degree  $\kappa = 2$  and a fixed macro-step size  $H = 1.0e-6$  s (*ex2H*) are carried out. The controlled macro-step size and polynomial degree of the *exCV* approach is given in Fig. 5.74 exemplarily for the  $n_s = 383$  subsystem decomposition of the  $n_K = 10^6$  model without contacts. Additional simulations, not presented here, show that both quantities are only slightly affected by the defined number of subsystems, by the model size and by the occurrence of contacts within the subsystems. The average macro-step size of this particular configuration is  $\bar{H} = 2.14e-6$  s; only minor fluctuations of the macro-step size are observed. Therefore,  $H = 1.0e-6$  s seems to be an adequate choice for a co-simulation with a fixed macro-step size.



**Figure 5.74:** Macro-step size  $H$  and polynomial degree  $\kappa$  of the fully controlled *exCV* approach ( $\text{rtol} = 10^{-4}$ ) of the  $n_s = 383$  subsystem decomposition of the  $n_K = 10^6$  body model without contacts.

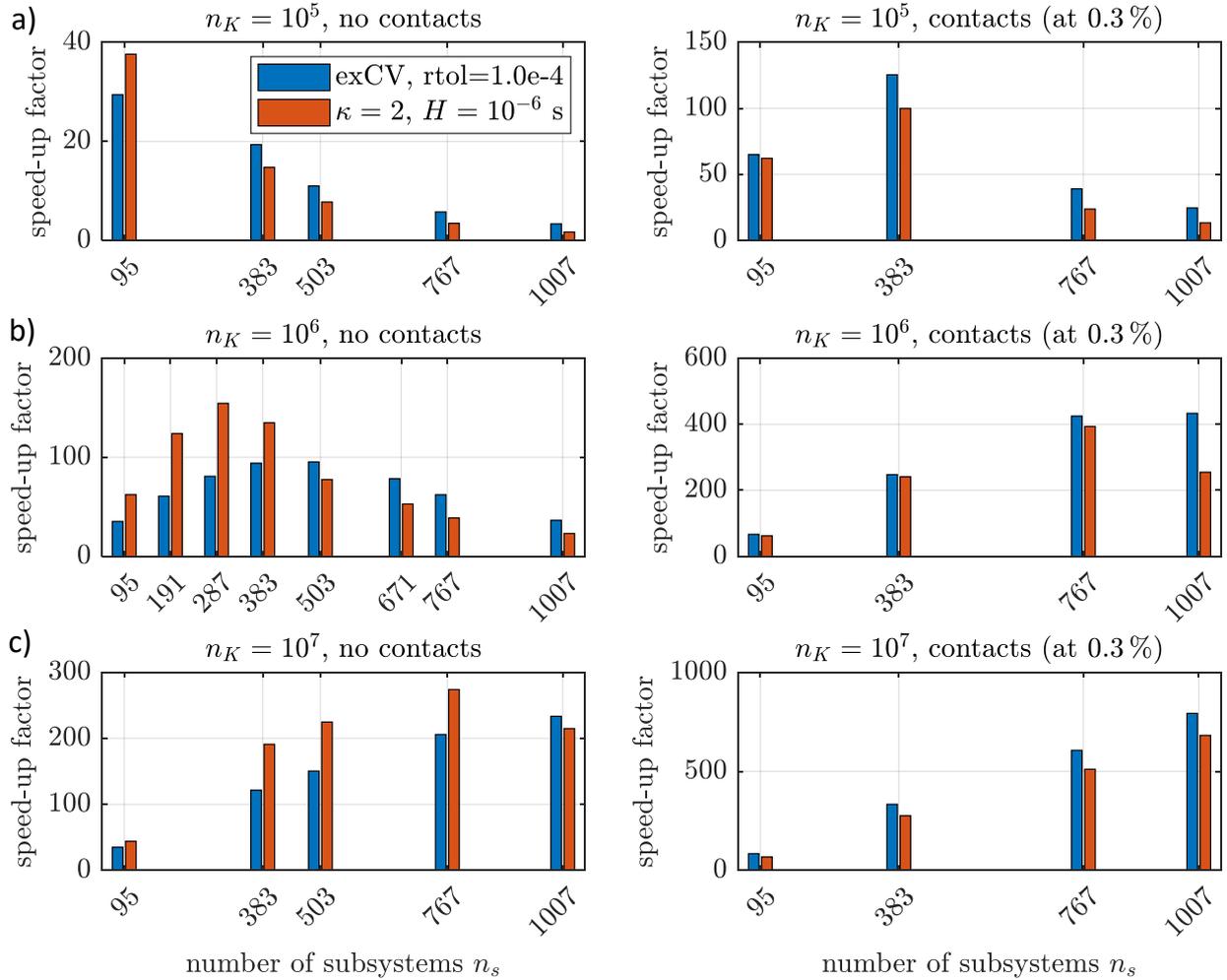
The average subsystem solver step size of the above considered model is  $\bar{h}_{mic} = 1.61e-6$  s, when the *exCV* approach is applied, and  $\bar{h}_{mic} = 1.00e-6$  s for the co-simulation with a fixed macro-step size of  $\bar{H} = 1.0e-6$  s. In the case of the fixed macro-step size, the computationally advantageous  $h_{mic} = H$  condition is achieved without restricting the solver step size very much. Therefore, the explicit co-simulation approach with  $H = 1.0e-6$  s is expected to allow an efficient computation of this model. For comparison, the average solver step size of the monolithic computation of the  $n_K = 10^6$  model is  $\bar{h} = 1.34e-6$  s.

The resulting computation times of the explicit co-simulation approach with a controlled macro-step size and order (solid lines) and with a fixed macro-step size and order (dotted lines) are shown in Fig. 5.75. Results are presented for  $n_K = 10^5$  in Fig. 5.75a), for  $n_K = 10^6$  in Fig. 5.75b) and for  $n_K = 10^7$  in Fig. 5.75c). The overall computation time depending on the number of subsystem  $n_s$  is depicted for the models without contacts (left figures) and for the models with contacts (right figures). The different parts of the overall computation time, which are described in Section 4 are indicated by different colors. The black dots mark the estimated computation time according to Eq. (4.2). The estimation neglects all sources of computational overhead and therefore approximates only the pure subsystem solver time  $T^{\text{solv}}$ . The solver time is, however, estimated very well.



**Figure 5.75:** Computation time of the fully controlled *exCV* co-simulation approach (solid lines) and the explicit co-simulation approach with fixed macro-step size  $H = 1.0e-6$  s and  $\kappa = 2$  (dotted lines), a)  $n_K = 10^5$ , b)  $n_K = 10^6$ , c)  $n_K = 10^7$ , without contacts (left figures) and with contacts at 0.3% of the bodies (right figures).

The choice of the optimal number of subsystems is a trade-off between reducing the subsystem integration time and increasing the overhead generated by the co-simulation interface and the MPI data transfer. The optimal choice depends on the model size, on the parameterization of the model (e.g. stiffness coefficients, damping coefficients, contacts, ...) and also on the amount of data that is transferred between the subsystems and the co-simulation interface. A general rule on the optimal number of subsystems cannot be given.



**Figure 5.76:** Speed-up factors ( $T_{\text{monolith}}/T_{\text{cosim}}$ ) of the co-simulation approaches with respect to the computation time of the monolithic simulation: a)  $n_K = 10^5$ , b)  $n_K = 10^6$ , c)  $n_K = 10^7$ . Left figures: models without contacts, right figures: models with contacts at 0.3% of the bodies.

Comparing the approach with a controlled macro-step size (*exCV*) to the co-simulation with fixed macro-step size (*ex2H*), the following observations can be made. The *exCV* uses a slightly increased average macro-step size and allows therefore a slightly increased average micro-step size. The consequence is a slightly reduced overall subsystem integration time. Due to the minor changes in the macro-step size during the simulation process, these effects influence the computation time only slightly. Considering simulations with a large number of subsystems, the decreased number of macro-steps reduces the overhead caused by MPI data traffic. The *ex2H* approach on the other hand, does not perform macro-step repetitions and therefore a subsystem solver workspace dump (sws dump) is not required. In addition, the computations of the co-simulation interface are very limited. As already mentioned, the *ex2H* approach shows the

behavior  $H = h_{mic}$ , i.e. the subsystem solver makes one solver step in (almost) each macro-step. This results in a reduced idle-time  $T^{idle}$  of the subsystems, as explained in Section 4.5.

Considering the models with contacts, the overall computation time is dominated by the idle-time  $T^{idle}$ , which can be interpreted in a way that the subsystem integration time within each macro-step differs significantly between the subsystems. The *exCV* approach shows a slightly better overall performance compared to the *ex2H* approach for the models with contacts.

Since all subsystems are very similar, a significant reduction of the computation time caused by multirate effects does not appear for the considered models. Therefore, the speed-up factors achieved by the co-simulation approaches are solely explainable by the parallelization. The resulting speed-up factors ( $T_{monolith}/T_{cosim}$ ) are presented in Fig. 5.76 for the three models without contacts and with contacts. The highest measured speed-up factor is 793 for the  $n_K = 10^7$  body model with contacts. In other words, a co-simulation with  $n_s = 1007$  subsystems is almost 800 times faster than the monolithic computation of the same model.

It should be mentioned that an analysis of the numerical errors has not been carried out. Because of the large model size, a reference simulation with very tight error tolerances could not be completed in a realistic time slot. The computation time of the monolithic simulation of the  $n_K = 10^7$  model has only been estimated for the same reason. However, other numerical studies which have been carried out within the scope of this work suggest, that the selected error tolerances of the co-simulation (*exCV*,  $rtol = 10^{-4}$ ,  $atol^u = 10^4 \cdot rtol$ ) and the subsystem solver ( $rtol_{IDA} = 10^{-6}$ ,  $atol_{IDA}^{pos} = 10^{-3} \cdot rtol_{IDA}$ ,  $atol_{IDA}^{vel} = rtol_{IDA}$ ) yield results of comparable accuracy as the monolithic simulation.

## 5.7. Scaling Order of the Computation Time

All numerical studies in this work, except for the following, are carried out by using a sparse matrix solver (*KLU* [DPN10]) to solve the linear system within the Newton iterations of the BDF solver (*IDA* [HBG<sup>+</sup>05]), which is applied for the subsystem integrations and monolithic integrations. As a consequence, the computation time of the subsystem/monolithic integration scales about linearly ( $P \approx 1$ ) with respect to the degree of freedom of the considered multibody system. In Section 4, it is explained that the possible computation time reduction achieved by applying a co-simulation method increases for higher scaling orders ( $P > 1$ ). To obtain a simulation with a cubic ( $P = 3$ ) scaling, a dense linear solver (fully populated Jacobian) is used instead of the sparse matrix solver (only non-zero elements of the Jacobian are considered). The parameterization of the model is given in Table 5.8.

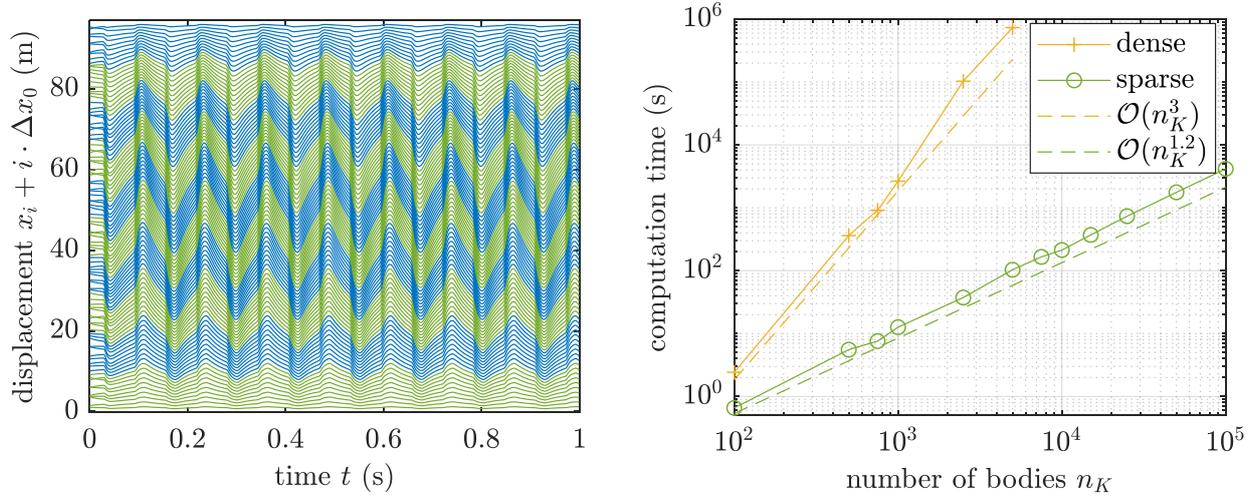
The degree of freedom  $n_K$  is increased successively by adding further masses to the oscillator chain. The initial conditions for each mass are selected randomly within the intervals  $[-1.0, 1.0]$  m and  $[-1.0e2, 1.0e2]$  m/s. Randomly picked 10 % of the masses are excited by a modified sinus force according to Eq. (2.14). The force amplitudes  $\Delta F_{S,i}$  are chosen randomly within the interval  $\pm[1.0e7, 1.0e8]$  N, the angular frequency is set to  $\Omega_{S,i} = 5.0e1 \text{ s}^{-1}$  and the exponent is set to  $A_S = 95$ . The displacements of the  $n_K = 100$  model are visualized in Fig. 5.77 (left figure).

The computation time of the monolithic simulations as a function on the number of bodies  $n_K$

**Table 5.8:** Test model parameters.

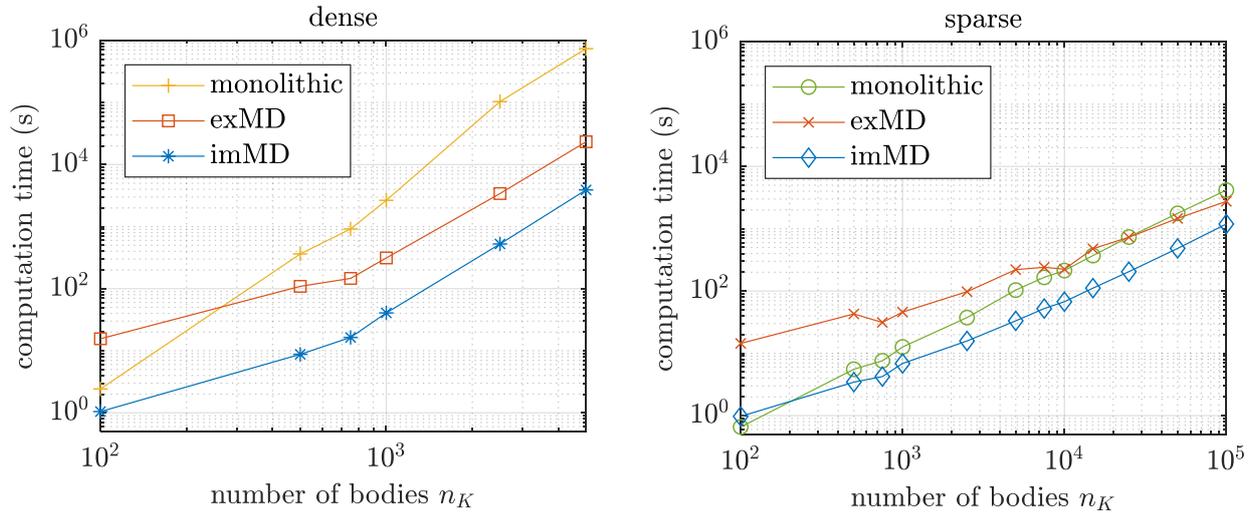
$n_s$	10
$t_{sim}$	1.0 s
$m_i$	1.0e1 kg
$c_i$	1.0e5 N/m
$d_i$	1.0 Ns/m
$C_i$	1.0e9 N/m <sup>3</sup>
$D_i$	1.0e4 Ns <sup>3</sup> /m <sup>3</sup>

(= degree of freedom) is shown in Fig. 5.77 (right figure). As expected, the scaling order of the computation changes from  $P = 1.2$  to  $P = 3$  depending on the type of the linear system solver.



**Figure 5.77:** Left figure: Visualization of the displacements of the  $n_K = 100$  model with the offset  $\Delta x_0 = 0.96$  m, subsystems are indicated by two different colors. Right figure: scaling of the computation time of the monolithic simulation using a dense or a sparse representation of the Jacobian.

The overall computation time of the explicit (*exMD*) and the implicit (*imMD*) co-simulation approaches depending on the number of bodies  $n_K$  is shown in Fig. 5.78. For the considered parameterization of the model, the implicit co-simulation approach is more efficient than the explicit approach. The average macro-step size of the  $n_K = 5000$  model is  $\bar{H} = 1.8e-4$  s, if the implicit method is applied and  $\bar{H} = 3.1e-6$  s, if the explicit approach is used.



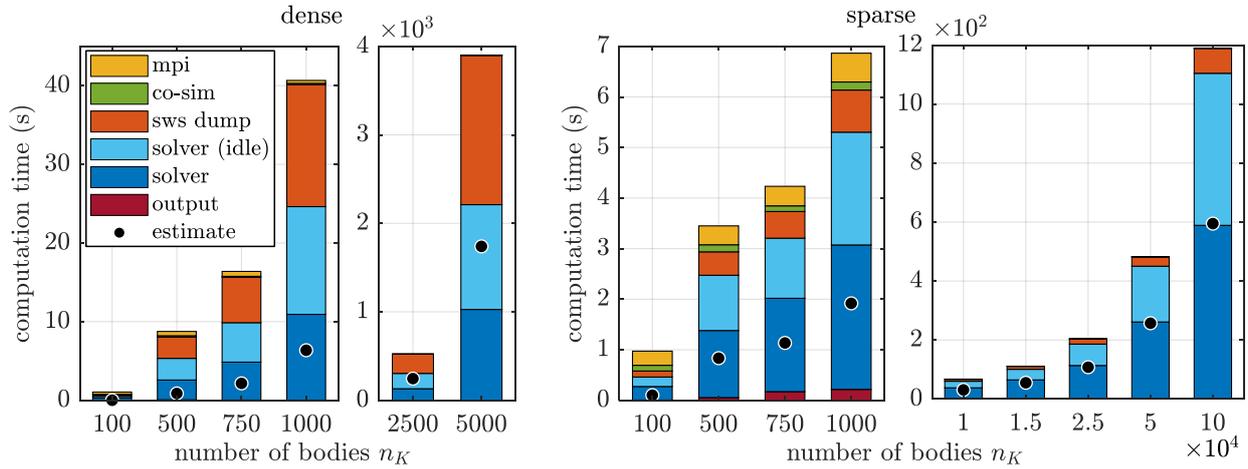
**Figure 5.78:** Computation time of the monolithic simulation and the explicit (*exMD*) and the implicit (*imMD*) co-simulation approaches depending on the number of bodies  $n_K$ . Monolithic/subsystem integrations are carried out by using a BDF solver in combination with either a dense (left figure) or a sparse (right figure) linear solver.

The plots in Fig. 5.78 give an idea of the significant influence of the scaling order on the savings in computational effort, which can be achieved by a parallelized co-simulation approach. Considering the  $n_K = 5000$  model, the speed-up factor obtained by a co-simulation with  $n_s = 10$

subsystems compared to the monolithic simulation is about three, if the sparsity of the matrices is exploited (scaling order  $P = 1.2$ ). But if the computation time shows a cubic dependency on the model size ( $P = 3$ ), the co-simulation of the same model is about 189 times faster than the monolithic simulation.

The detailed segmentation of the computation time of the implicit co-simulation approach is shown in Fig. 5.79. The time spent on copying and reloading the subsystem solver workspace (sws dump), which is required to provide macro-step repetitions increases strongly, if the dense representation of the subsystem Jacobian is used instead of the sparse representation. The solver workspace includes all data of the linear solver; therefore, also the Jacobians are dumped.

The black dots in Fig. 5.79 mark the estimated computation time according to Eq. (4.4). The average number of corrector iterations per macro-step is  $\bar{n}_{cor} \approx 2.3$  for all simulations. In theory, the estimation would match with the sum of the solver time (blue) and the time spent on writing output files (dark red) because these two processes are also accomplished by the monolithic computation. Possible reasons for the deviation of the solver time from the theoretical value are explained in Section 4.



**Figure 5.79:** Computation time of the implicit (*imMD*) co-simulation approach using a dense (left figure) or a sparse (right figure) linear solver. The black dots mark the estimated computation time according to Eq. (4.4).

## 5.8. Co-Simulation Model with Additional Forces at the Coupling Elements

To demonstrate the advantages of the macro-step size controller, the chain-structured multibody system introduced in Section 2.2 is parametrized in a way to clearly illustrate the benefit of a macro-step size and order control algorithm. The oscillator chain with  $n_K = 100\,000$  masses is split into  $n_s = 100$  subsystems of equal size. The model is simulated over  $t_{sim} = 1.0$  s; the initial conditions of each mass are chosen randomly within the interval  $[-1.0e-2, 1.0e-2]$  mm for the displacement variables and in the interval  $[-1.0e2, 1.0e2]$  mm/s for the velocity variables.

The model parameters are given in Table 5.9. The coupling bodies are directly affected by impulse shaped forces according to Eq. (2.13) with high amplitudes at certain time points  $t_{I,i} = [1.0e-4\text{s}, 0.1\text{s}, 0.2\text{s}, \dots, 0.9\text{s}]$ . The impulse shaped forces are defined by the parameters  $\Delta F_{I,i} = (-1)^{i+1} \cdot 1.0e6$  N,  $\Delta t_{I,i} = 1.0e-3$  s and  $\delta_I = 1.0e-6$  s, the index  $i$  runs over all coupling bodies. Because of the alternating sign of  $\Delta F_{I,i}$ , the impulse shaped forces act pairwise in oppo-

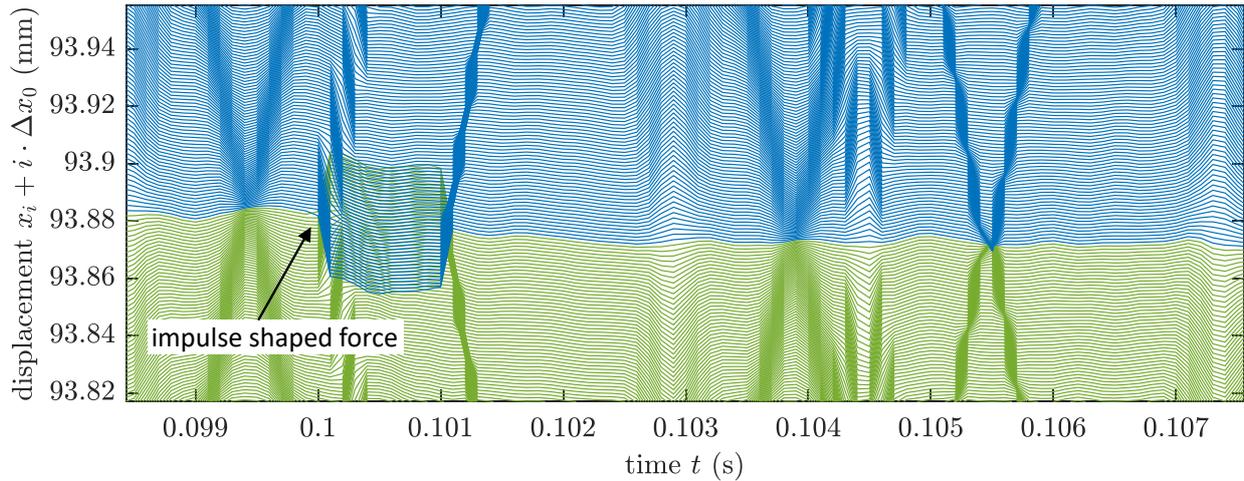
site direction so that each coupling elements gets compressed by the forces. Figure 5.80 shows the displacements of the bodies at the coupling between subsystems 80 and 81 to give an idea of the dynamics of the system.

The subsystem solver tolerances are set to  $\text{rtol}_{\text{IDA}} = 10^{-6}$  and  $\text{atol}_{\text{IDA}}^{\text{pos}} = 10^{-3} \cdot \text{rtol}_{\text{IDA}}$ ,  $\text{atol}_{\text{IDA}}^{\text{vel}} = \text{rtol}_{\text{IDA}}$ . The absolute error tolerances of the macro-step size controller are set to  $\text{atol}^{\text{pos}} = 10^{-3} \cdot \text{rtol}$ ,  $\text{atol}^{\text{vel}} = \text{rtol}$  and  $\text{atol}^u = 10^4 \cdot \text{rtol}$ .

The model is computed using the explicit and the implicit co-simulation approaches with the following three options: fixed order ( $\kappa = \text{const.}$ ) and fixed macro-step size ( $H = \text{const.}$ ), fixed order ( $\kappa = \text{const.}$ ) and variable macro-step size ( $H = \text{var.}$ ), variable order ( $\kappa = \text{var.}$ ) and variable macro-step size ( $H = \text{var.}$ ). The simulation results are compared with the results of a reference simulation and the global error of the states of the coupling bodies is computed. For a reasonable comparison, the accuracy condition  $\text{ERR} := \text{NRMSE}(\mathbf{Z}^{\text{coup}}) \approx 1.0\text{e-}3$  must be satisfied by the results of the co-simulations. A monolithic computation of the model takes  $2.69\text{e}4\text{s}$  to achieve a resulting error of  $\text{ERR}_{\text{mono}} = 2.34\text{e-}3$ . The average solver-step size is  $\bar{h} = 1.34\text{e-}6\text{s}$ .

**Table 5.9:** Test model parameters.

$n_K$	100 000
$n_s$	100
$t_{\text{sim}}$	1.0 s
$m_i$	$5.0\text{e-}5 \text{Ns}^2/\text{mm}$
$c_i$	$2.5\text{e}6 \text{N}/\text{mm}$
$d_i$	$2.5\text{e-}1 \text{Ns}/\text{mm}$
$C_i$	$1.0\text{e}10 \text{N}/\text{mm}^3$
$D_i$	$5.0\text{e-}5 \text{Ns}^3/\text{mm}^3$



**Figure 5.80:** Visualization of the displacements of the bodies around the coupling between subsystem 80 and 81 with the offset  $\Delta x_0 = 1.16\text{e-}3 \text{mm}$ , output resolution  $H_{\text{out}} = 1.0\text{e-}4 \text{s}$  (zoom).

To obtain results of the required accuracy with an implicit co-simulation approach using a fixed order and a fixed macro-step size, the parameters  $\kappa = 2$  and  $H = 1.0\text{e-}7 \text{s}$  show the best efficiency. With these parameters, the implicit co-simulation takes  $2.0\text{e}4 \text{s}$  to achieve a resulting error of  $\text{ERR}_{\text{impl}} = 1.07\text{e-}3$ . Using the explicit co-simulation approach, best results are achieved with the parameters  $\kappa = 1$  and  $H = 2.5\text{e-}8 \text{s}$ . The computation time of the explicit co-simulation is  $1.89\text{e}4 \text{s}$  and the resulting error  $\text{ERR}_{\text{expl}} = 7.94\text{e-}4$ .

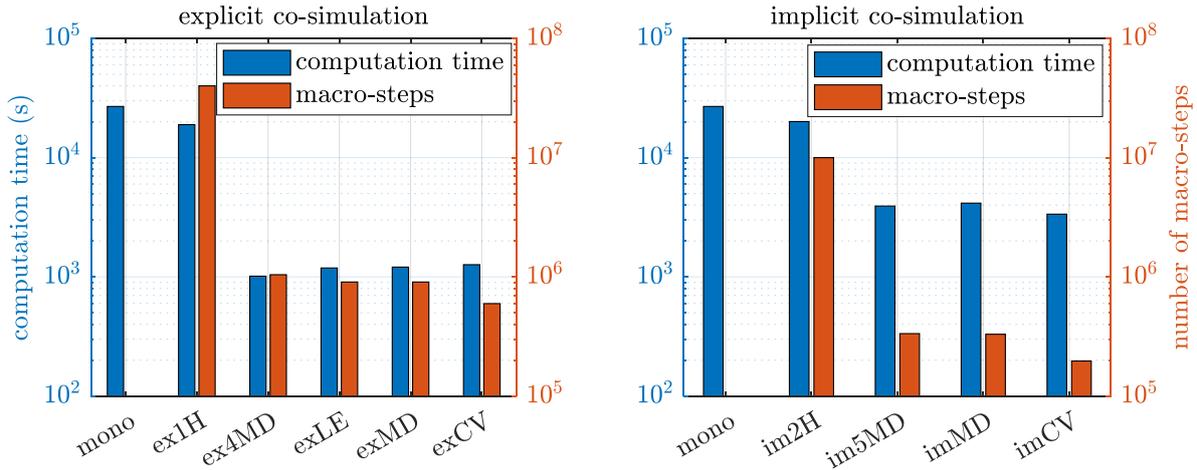
When the macro-step size is variable and only the polynomial degree is fixed, best results have been achieved with  $\kappa = 5$  and  $\text{rtol} = 1.0\text{e-}4$  for the implicit approach. The computation time is  $3925 \text{s}$ , the average macro-step size  $\bar{H} = 2.99\text{e-}6 \text{s}$  and the resulting error  $\text{ERR}_{\text{impl}} = 1.86\text{e-}3$ . The explicit approach performs best for  $\kappa = 4$  and  $\text{rtol} = 1.0\text{e-}6$ . The computation time of the explicit approach is  $1014 \text{s}$ , the average macro-step size  $\bar{H} = 9.59\text{e-}7 \text{s}$  and the resulting error

$$ERR_{expl} = 3.95e-5.$$

The fully controlled implicit co-simulation (*imMD*) with  $rtol = 1.0e-4$  takes 4154 s to obtain a resulting error of  $ERR_{impl} = 9.00e-4$ . The average macro-step size is  $\bar{H} = 3.03e-6$  s. Applying the error estimator based on the coupling variables in combination with implicit co-simulation (*imCV*) with  $rtol = 1.0e-4$ , the computation takes 3356 s to obtain a resulting error of  $ERR_{impl} = 9.36e-4$ . The average macro-step size is  $\bar{H} = 5.08e-6$  s.

The explicit co-simulation (*exMD*) shows the in Section 5.4 described behavior, that the computation time decreases as the macro-step size decreases, as shown in Fig. 5.83. Best results with regard to the computation time are achieved with very tight error tolerances  $rtol = 1.0e-6$ . The fully controlled explicit co-simulation (*exMD*) takes 1187 s to obtain a resulting error of  $ERR_{expl} = 6.81e-5$ . The average macro-step size is  $\bar{H} = 1.10e-6$  s. The results obtained by using the *exLE* approach with the same error tolerances are very similar: computation time 1207 s, global error  $ERR_{expl} = 7.49e-5$ , and average macro-step size  $\bar{H} = 1.10e-6$  s.

The fully controlled explicit co-simulation in combination with the error estimator based on the coupling variables (*exCV*) with  $rtol = 1.0e-6$  takes 1265 s to obtain a resulting error of  $ERR_{expl} = 3.43e-4$ . The average macro-step size is  $\bar{H} = 1.68e-6$  s. The resulting computation time of the different approaches are collected in Fig. 5.81.



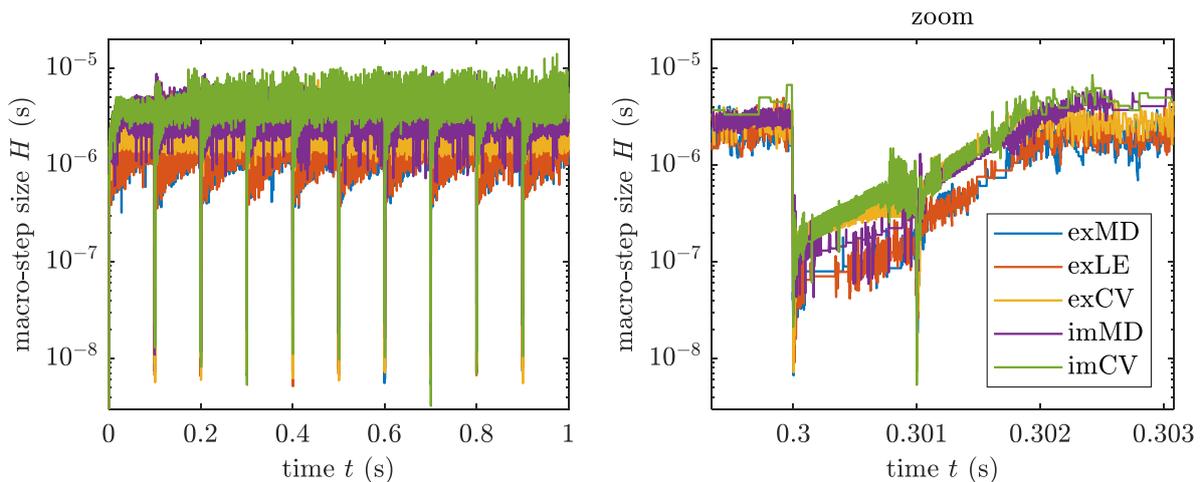
**Figure 5.81:** Comparison of the different co-simulation approaches: best computation time under the accuracy condition  $ERR := \text{NRMSE}(\mathbf{Z}^{coup}) \approx 1.0e-3$ .

Based on the results, following conclusions can be drawn:

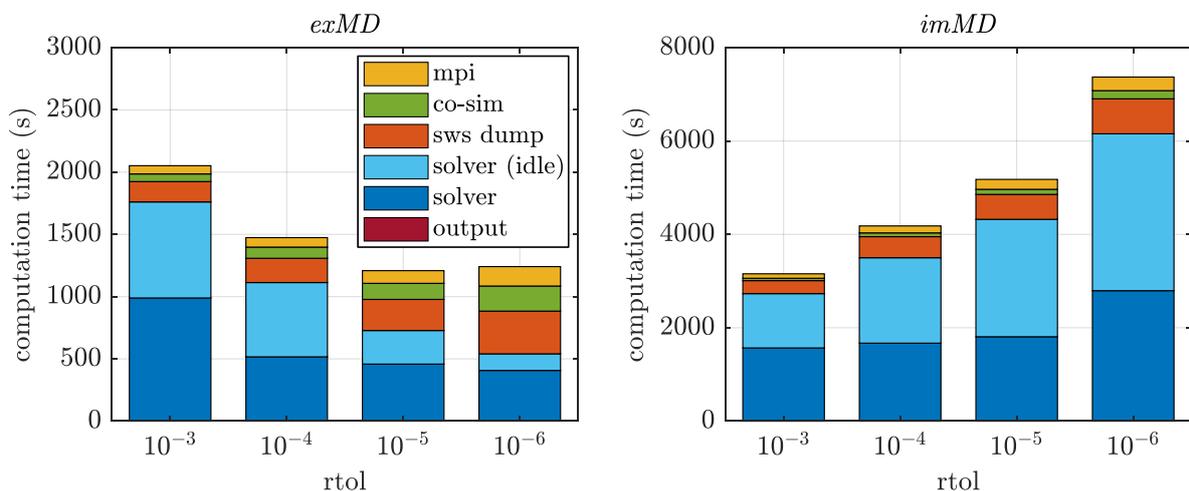
- Using the macro-step size controller, the macro-step size is decreased significantly when the impulse shaped forces occur, as shown in Fig. 5.82.
- A co-simulation with a fixed macro-step size is inefficient for the considered model. The computation time is reduced by only 30% compared to the monolithic simulation.
- When the macro-step size controller is applied, the efficiency of all considered co-simulation approaches is significantly increased. A speed-up factor of 26.5 compared to the monolithic computation can be achieved.
- For the considered model, the order control algorithm does not significantly reduce the computation time, although it allows a slightly increased average macro-step size. Except for the

time intervals, when the coupling elements are compressed by the impulse shaped forces (which are very short), the optimal degree of the approximation polynomials is in the range  $\kappa = 3, 4, 5$  (see Fig. 5.87). Therefore, the benefit of allowing a variable order is rather low.

- The three error estimators for the explicit co-simulation approach show a similar efficiency and give comparable results.
- The two error estimators for the implicit co-simulation approach show a similar efficiency and give comparable results.

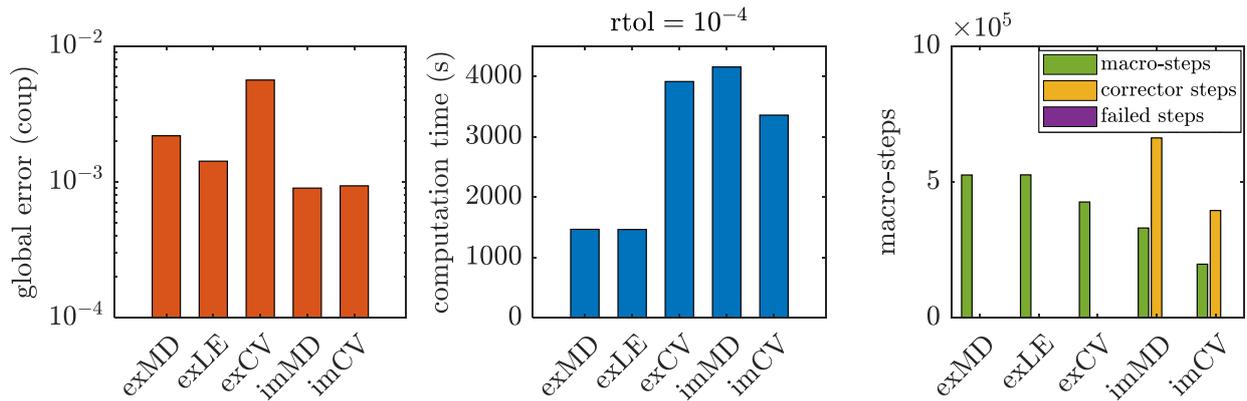


**Figure 5.82:** Macro-step size of the co-simulation approaches in combination with different error estimators (relative error tolerance  $\text{rtol} = 10^{-5}$ ).

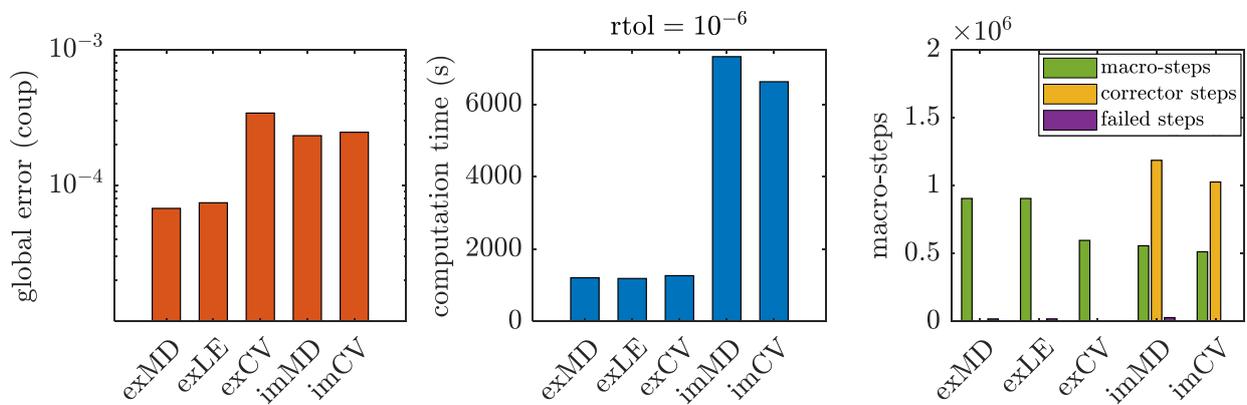


**Figure 5.83:** Computation time of the explicit (*exMD*) and the implicit (*imMD*) co-simulation approach depending on the error tolerances.

A more detailed comparison of the different error estimators is shown in Figs. 5.84 and 5.85 for two different sets of error tolerances. It can be seen, that the resulting global error of the states of the coupling bodies is on a comparable level for all estimators. The number of local error test misses (failed macro-steps) is very low for all considered estimators.



**Figure 5.84:** Comparison of the different error estimators (relative error tolerance  $\text{rtol} = 10^{-4}$ ): global error (NRMSE) of the state variables of the coupling bodies, computation time and number of macro-steps.



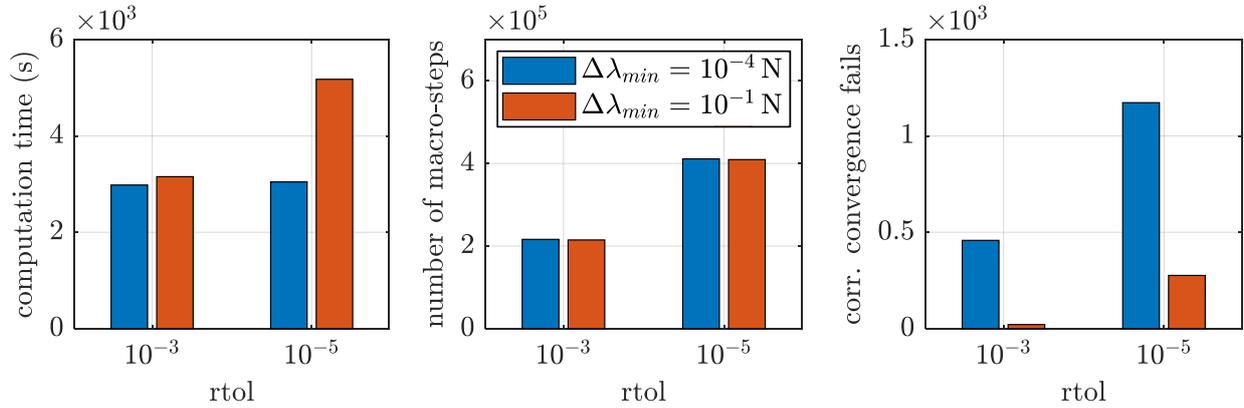
**Figure 5.85:** Comparison of the different error estimators (relative error tolerance  $\text{rtol} = 10^{-6}$ ).

However, for this particular model the explicit co-simulation approach is more efficient than the implicit approach, especially for small error tolerances. The two error estimators for the explicit co-simulation method *exMD* and *exLE* give almost identical results with regard to the errors, computation time and the number of macro-steps. The error estimator *exCV* allows a slightly increased macro-step size, but therefore the resulting errors are also slightly increased.

Considering the implicit co-simulation approach, both error estimators (*imMD* and *imCV*) produce almost identical results regarding the global error. When the coupling variables are used for the error estimation (*imCV*) the number of macro-steps is reduced; also, the resulting computation time is slightly decreased.

It should be noted that the subsystem integrations with perturbed approximation polynomials (for the numerical computation of the interface Jacobian) require a longer computation time than the subsystem integrations with predicted/corrected approximation polynomials. Considering the *imMD* approach with  $\text{rtol} = 1.0 \times 10^{-5}$  as an example, the average (total) subsystem integration time with predicted/corrected approximation polynomials is 1040 s, while the average subsystem integration time with perturbed approximation polynomials is 1537 s. This effect appears especially for small error tolerances. The computational efficiency could be increased by reducing the magnitude of the perturbations. If the lower limit  $\Delta\lambda_{\min}$  of the perturbations (see remark in Section 2.4.2) is decreased from  $\Delta\lambda_{\min} = 1.0 \times 10^{-1} \text{ N}$  to  $\Delta\lambda_{\min} = 1.0 \times 10^{-4} \text{ N}$ , the average subsystem integration time with perturbed approximation polynomials is reduced to 1051 s as shown in Fig. 5.86. However, if

the perturbations are selected too small, the accuracy of the interface Jacobian will be reduced. This is indicated by the increased number of failed corrector iterations. In the Appendix A.3 (Example 4), the implicit co-simulation of the model is carried out with an alternative approximation method for the interface Jacobian. The alternative approach does not require subsystem integrations with perturbed approximation polynomials. Using the alternative approach, the computation time is only slightly affected by the error tolerances, as shown in Fig. A.29.



**Figure 5.86:** Computation time and total number of steps depending on the lower limit  $\Delta\lambda_{min}$  of the perturbations of the approximation polynomials for computing the interface Jacobian by finite differences.

Figure 5.87 shows, how the degree  $\kappa$  of the approximation polynomials changes with the error tolerances. The pie charts illustrate the portions of the simulation time  $t_{sim}$ , which are carried out by using a particular degree  $\kappa$ . For example, if the *exMD* approach is applied with a relative error tolerance  $rtol = 1.0e-3$ , within 32% of  $t_{sim}$  approximation polynomials of degree  $\kappa = 2$  are used. It is clearly visible that higher polynomial degrees are used if  $rtol$  is decreased. This is reasonable, because the macro-step size decreases for smaller error tolerances and the convergence order of the co-simulation methods increases if  $\kappa$  is increased. Co-simulation approaches using the error estimators *exCV*, *imCV* use a tendentially lower degree  $\kappa$  compared to the Milne Device approaches *exMD*, *imMD*. Especially for the explicit approach, a significant difference can be observed.

As shown exemplarily for the *imMD* approach in Fig. 5.88, the polynomial degree is reduced to  $\kappa = 0$  when the impulse shaped forces are applied. In the absence of the external forces, the simulation is carried out with approximation polynomials of degree  $\kappa \geq 1$  or  $\kappa \geq 2$ , depending on the selected error tolerances.

As already mentioned, the explicit co-simulation approach is obviously more efficient for the considered parameterization of the model. To point out the benefit of the implicit co-simulation approach, the nonlinear damping coefficient  $D_c$  of all coupling elements is successively increased. The average macro-step size of the explicit co-simulation approach (*exMD*) is strongly decreased by the macro-step size controller to obtain a stable simulation for highly damped coupling elements, as shown in Fig. 5.89. As a result, the computation time increases and the resulting global error also increases. The performance of the implicit approach (*imMD*) is not influenced significantly by the increased nonlinear damping coefficients of the coupling elements, at least within the considered parameter range.

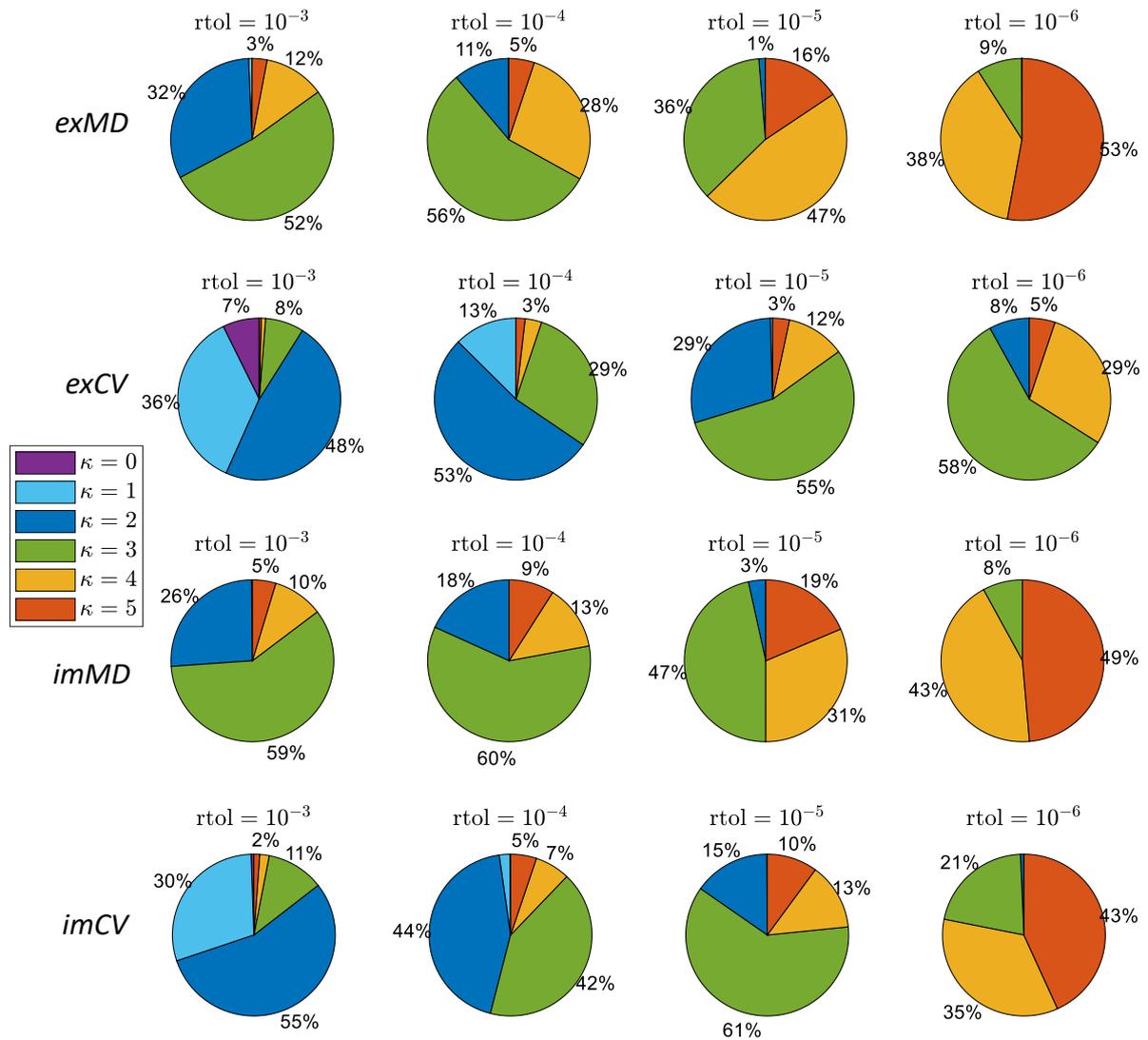


Figure 5.87: Order control: portions of the simulation time  $t_{sim}$ , carried out by using a particular degree  $\kappa$  for different co-simulation approaches and error tolerances.

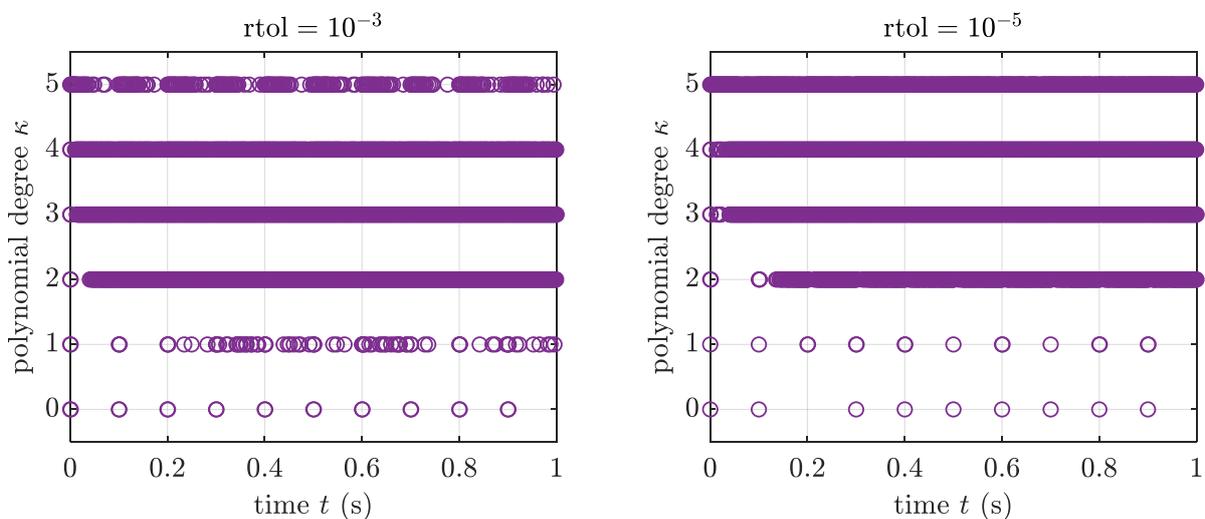
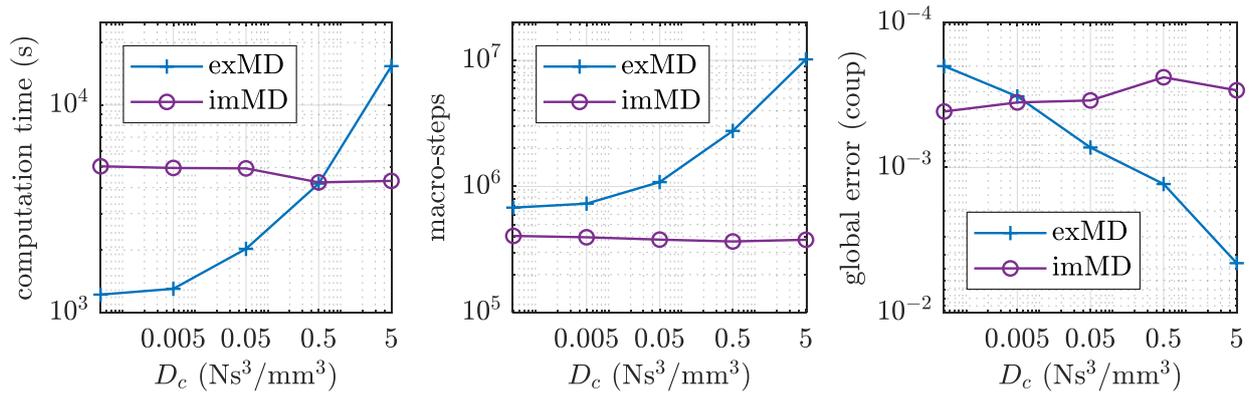
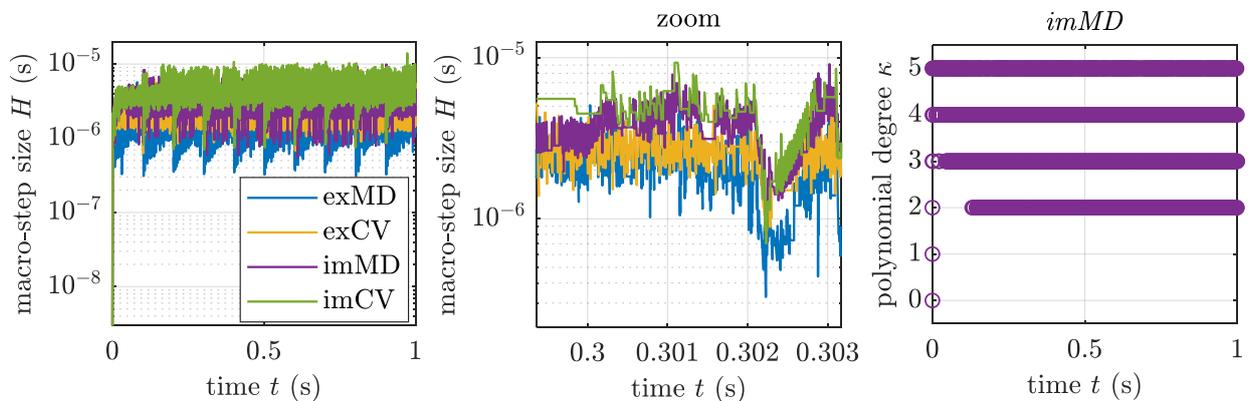


Figure 5.88: Implicit co-simulation approach (*imMD*): degree  $\kappa$  of the approximation polynomials depending on the error tolerances.

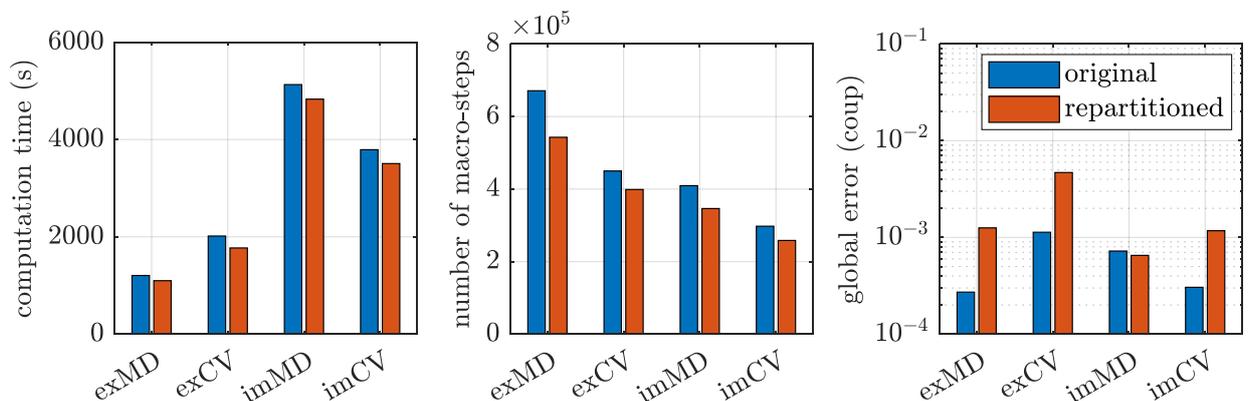


**Figure 5.89:** Increased nonlinear coupling damping coefficient  $D_c$ : comparison between the explicit (*exMD*) and the implicit (*imMD*) co-simulation approach.

The second example in Section 5.4 already demonstrated that a well-considered model decomposition may increase the computational efficiency of a co-simulation approach. The decomposition used above is not ideal, because the coupling elements are directly affected by strong external forces. To reduce the computation time, the model is repartitioned into 101 subsystems in a way that the first and the last subsystems contain 500 masses each and the remaining 99 subsystems are of equal size with 1000 masses each. The effect of the increased number of subsystems (101 instead of 100) is negligible, the essential point is that the impulse shaped forces do not longer affect the coupling bodies directly but act on bodies in the center of the subsystems 2-100.



**Figure 5.90:** Repartitioned co-simulation model: macro-step size  $H$  and polynomial order  $\kappa$  of the different co-simulation approaches with  $\text{rtol} = 10^{-5}$ .



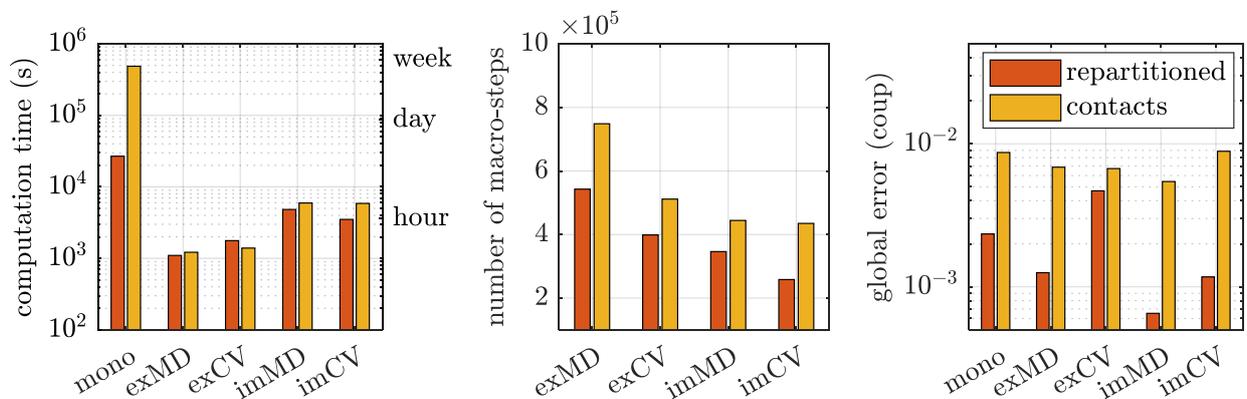
**Figure 5.91:** Comparison of the performance of the repartitioned co-simulation model and the original decomposition for relative error tolerance  $\text{rtol} = 10^{-5}$ .

It can be seen in Fig. 5.90 that the macro-step size of the restructured co-simulation model is not as strongly reduced as before when the impulse shaped forces occur. The degree of the approximation polynomials does not have to be reduced to  $\kappa = 0$  but remains  $\kappa \geq 2$  throughout the simulation. Because of the small number of occurrences of the external forces and their short duration, the computation time reduction that can be achieved by the repartitioning is rather low as shown in Fig. 5.91. The most significant changes in the results are obtained for the *exCV* approach with a reduction of the computation time by 14 %.

Finally, the parameterization of the oscillator chain is modified to demonstrate the the multirate effect in connection with co-simulation. Small subsystems with 25 masses each are added to both ends of the chain. Within the small subsystems (subsystem index 1 and 103), contact forces according to Eq. (2.12) are considered so that the subsystems become very stiff. The contact forces act on three masses ( ${}^1m_3, {}^1m_5, {}^1m_7$ ) of subsystem 1 and on three masses ( ${}^{103}m_{19}, {}^{103}m_{21}, {}^{103}m_{24}$ ) of subsystem 103 and are defined by the parameters  $A_C = -1.0e-2$  N and  $B_C = 5.0e5$  mm $^{-1}$ . Subsystems 2-102 contain the repartitioned model as described above.

Considering a monolithic simulation, the stiff contacts affect the overall simulation and will reduce the solver step size of the overall model. Applying a co-simulation approach in contrast, each subsystem solver selects its solver step size individually. Therefore, the solver step size of subsystems 1 and 103 will be reduced due to the contacts. The remaining subsystems 2-102 will be integrated with a solver step size that is not (significantly) influenced by the contacts. In addition, subsystems 1 and 103 with the contacts will not increase the overall computation time significantly because of the small size of these subsystems.

A comparison of the results of the repartitioned model without contacts and the model with contacts is shown in Fig. 5.92. As expected, the computation time of the monolithic simulation is significantly increased, while the computation time of the co-simulation approaches does not change significantly. The number of macro-steps and the resulting global error are, however, slightly increased. The speed-up factor of the explicit co-simulation (*exMD*) compared to the monolithic simulation is 400 for the model with contacts. In other words, the monolithic simulation runs five and a half days while the co-simulation is finished in less than 30 minutes. This very high speed-up factor is achieved by a combination of the parallelized computation and exploiting the multirate effect.



**Figure 5.92:** Comparison of the monolithic simulation and the co-simulation approaches of the repartitioned model and the model with stiff contacts ( $rtol = 10^{-5}$ ).

## 6. Conclusions

In this work, the application of explicit and implicit co-simulation methods with the objective of the parallelization of multibody systems is investigated. Due to the here applied weak coupling approaches, all subsystems are integrated independently of each other between the macro-time points. Therefore, co-simulation models are predestined for a parallelized computation.

For the considered co-simulation approaches the constitutive equations of the coupling elements and the state variables of the coupling bodies must be available in the co-simulation interface. Only coupling approaches have been considered, where the subsystems are connected by constitutive laws; coupling by algebraic constraint equations has not been analyzed. The subsystems must provide the functionality to repeat macro-steps for all considered co-simulation schemes, except for the explicit co-simulation approach with an equidistant macro-time grid. Within this work, macro-step repetitions are accomplished by dumping the subsystem solver workspace at the beginning of each macro-step. This allows the macro-step to be repeated without any initialization or additional start processes of the subsystem solver.

Considering classical numerical time integration schemes, implementations with constant integration step sizes are often not very time efficient. For special solvers, e.g. BDF schemes, also implementations with variable integration order are used to speed up and optimize the implementation. Here, the idea of step size and order control is applied in the framework of co-simulation. Co-simulation is frequently applied in connection with equidistant communication-time grids. Then, the user has to define the macro-step size appropriately, depending on the subsystem parameters and especially depending on the coupling parameters. Choosing the macro-step size too large, the co-simulation may become unstable, especially in connection with explicit co-simulation schemes. If the macro-step size is chosen too small, the co-simulation may become inefficient. Especially for highly nonlinear models, co-simulations with constant macro-step sizes may be problematic.

In the current work, a strategy for controlling the macro-step size for co-simulation models has been presented and investigated. Basis of the implementation of a variable communication-time grid is an error estimator for the macro-step. Different error estimators for controlling the macro-step size of explicit and implicit co-simulations schemes have been developed and tested. The presented error estimators can be used to evaluate the numerical error produced by the co-simulation approach, i.e. the error generated by the approximation of the coupling variables within a macro-step. The derived estimators are valid for the case that the error resulting from the numerical time integration of the subsystems is significantly smaller than the error generated by the co-simulation approach. This requirement is often fulfilled in practical co-simulations and may simply be enforced by an appropriate choice of the subsystem solver parameters.

In addition, an algorithm for controlling the order of the approximation polynomials, which are used for approximating the coupling variables, has been developed and analyzed. Numerical investigations have clearly shown the benefit of the order and the macro-step size controller for explicit as well as for implicit co-simulation algorithms.

Especially for nonlinear systems and models with strongly varying properties, an order and macro-step size controller may considerably increase the computational efficiency of a co-simulation approach. In dynamical systems with contacts for instance, variables often change very rapidly within very small time periods so that a reduction of the order of the approximation

polynomials and a reduction of the macro-step size will be necessary in order to get stable and accurate results within moderate simulation times. Here, a co-simulation approach with a constant macro-step size and a fixed approximation order for the coupling variables may become rather inefficient.

The considered co-simulation approaches and the macro-step size and order control algorithm have been implemented in C-code and numerical studies have been carried out to verify the accuracy of the simulation results. The computational efficiency of the co-simulation methods has been analyzed based on numerous simulations, which were conducted on the *Lichtenberg High Performance Computer* of the TU Darmstadt. Simulations of differently parameterized models (up to 10 million degrees of freedom), have been accomplished successfully by decomposing the considered multibody system into different numbers (up to 1000) of coupled subsystems. A chain-structured multibody system has been used as test model in this work. The presented co-simulation approaches and also the suggested macro-step-size and order control algorithm can, however, directly be used in general dynamical systems with arbitrary topology.

The overall computation time of the considered co-simulation approaches can basically be subdivided into five parts:

- the time spent on exchanging data between the co-simulation interface and the subsystems,
- the computation time of the co-simulation interface,
- the co-simulation method specific additional subsystem computation time, consisting primarily of the time for creating the solver workspace dump in connection with macro-step repetitions,
- the subsystem integration time,
- the complementary idle-time, which originates from the differences in the integration times between the subsystems within each co-simulation step.

The choice of the number of subsystems in which a model should be decomposed is always a trade-off between reducing the solver time per subsystem and increasing the time required for the data traffic between the subsystems and the co-simulation interface.

However, it has been shown that the considered co-simulation approaches have the potential to reduce the computation time of large-scale multibody systems drastically compared to a monolithic simulation. The computation time reduction is obtained not only through the parallelization, but also by the exploitation of multirate effects.

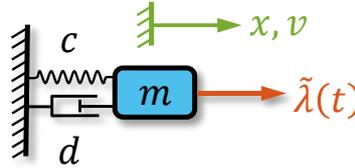
# Appendix

## A. Approximation of the Interface Jacobian

The partial derivatives  $\frac{\partial y}{\partial u}$  (interface Jacobian) of the state variables of the coupling bodies with respect to the coupling variables, which are required for the implicit co-simulation method, are approximated by finite differences within this work, as described in Section 2.4.2. The main drawback of this approach is that subsystem integrations with perturbed coupling variables have to be carried out to compute the finite differences. Assuming the computation should be fully parallelized, the number of required cores increases with the number of coupling variables. For problems with a large number of coupling variables, for instance when the coupling interface consists of a meshed surface, the finite differences approximation may therefore not be practical. A second drawback is the heuristically selection of the magnitude of the perturbations. Within the following subsections alternative approaches for the approximation of the interface Jacobian are analyzed.

### A.1. Linear Systems

A linear one-mass oscillator, as shown in Fig. A.1, is used for the illustration of the problem. The oscillator represents an arbitrary subsystem.



**Figure A.1:** Linear one mass oscillator.

The excitation force  $\tilde{\lambda}(t) = P_\kappa(\mathbf{T}, \boldsymbol{\lambda}, t)$  represents an approximation polynomial of degree  $\kappa$  of the coupling force. The supporting points are the macro-time points  $\mathbf{T} = [T_{N+1-\kappa}, \dots, T_{N+1}]^T$  and the corresponding values of the coupling force  $\boldsymbol{\lambda} = [\lambda_{N+1-\kappa}, \dots, \lambda]^T$ . Note that the supporting point at  $T_{N+1}$  (predictor point), which is obtained by extrapolation, is labeled  $\lambda$  instead of  $\lambda_{N+1}$  for the sake of simple representation. The time interval of interest is an arbitrary macro-time step from  $T_N$  to  $T_{N+1}$ . The equations of motion of the system for the case  $\kappa = 1$  read

$$\begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v \\ f(x, v) + g(\lambda) \end{bmatrix} = \begin{bmatrix} v \\ -\frac{1}{m}(cx + dv) + \frac{1}{m} \left( \lambda_N + \frac{(\lambda - \lambda_N)(t - T_N)}{T_{N+1} - T_N} \right) \end{bmatrix}. \quad (\text{A.1})$$

A differentiation with respect to  $\lambda$  yields

$$\begin{bmatrix} \dot{x}_\lambda \\ \dot{v}_\lambda \end{bmatrix} = \begin{bmatrix} v_\lambda \\ f(x_\lambda, v_\lambda) + g_\lambda(\lambda) \end{bmatrix} = \begin{bmatrix} v_\lambda \\ -\frac{1}{m}(cx_\lambda + dv_\lambda) + \frac{1}{m} \left( \frac{t - T_N}{T_{N+1} - T_N} \right) \end{bmatrix}. \quad (\text{A.2})$$

The index  $\lambda$  is an abbreviation for  $\frac{\partial}{\partial \lambda}$ , the partial derivative with respect to the supporting point  $\lambda$  at  $T_{N+1}$ . Equation (A.2) is a linear ODE-system for the partial derivatives  $x_\lambda$  and  $v_\lambda$  (interface

Jacobian). The system can be solved analytically with the matrix-exponential according to

$$\begin{bmatrix} x_\lambda \\ v_\lambda \end{bmatrix} = \exp(\mathbf{A}t) \left[ \mathbf{K} + \int_{T_N}^t \exp(-\mathbf{A}\tau) \mathbf{b}(\tau) d\tau \right] \quad (\text{A.3})$$

with

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -c/m & -d/m \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 \\ g_\lambda/m \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} K_1 \\ K_2 \end{bmatrix}. \quad (\text{A.4})$$

The integration constants  $K_1$  and  $K_2$  are computed for the known initial conditions at  $T_N$ . The evaluation of the matrix-exponential is computationally costly, therefore it may be more efficient to approximate the solution of Eq. (A.3) at  $T_{N+1}$  by a series expansion according to

$$\begin{aligned} \begin{bmatrix} x_\lambda(T_{N+1}) \\ v_\lambda(T_{N+1}) \end{bmatrix} &= \begin{bmatrix} \frac{1}{2} \frac{H^2}{m} - \frac{1}{6} \frac{dH^3}{m^2} + \frac{1}{24} \frac{(d^2-cm)H^4}{m^3} + \mathcal{O}(H^5) \\ \frac{H}{m} - \frac{1}{2} \frac{dH^2}{m^2} + \frac{1}{6} \frac{(d^2-cm)H^3}{m^3} - \frac{1}{24} \frac{(d^3-2cdm)H^4}{m^4} + \mathcal{O}(H^5) \end{bmatrix} \quad (\kappa = 0) \\ \begin{bmatrix} x_\lambda(T_{N+1}) \\ v_\lambda(T_{N+1}) \end{bmatrix} &= \begin{bmatrix} \frac{1}{6} \frac{H^2}{m} - \frac{1}{24} \frac{dH^3}{m^2} + \frac{1}{120} \frac{(d^2-cm)H^4}{m^3} + \mathcal{O}(H^5) \\ \frac{1}{2} \frac{H}{m} - \frac{1}{6} \frac{dH^2}{m^2} + \frac{1}{24} \frac{(d^2-cm)H^3}{m^3} - \frac{1}{120} \frac{(d^3-2cdm)H^4}{m^4} + \mathcal{O}(H^5) \end{bmatrix} \quad (\kappa = 1) \quad (\text{A.5}) \\ \begin{bmatrix} x_\lambda(T_{N+1}) \\ v_\lambda(T_{N+1}) \end{bmatrix} &= \begin{bmatrix} \frac{1}{8} \frac{H^2}{m} - \frac{7}{240} \frac{dH^3}{m^2} + \frac{1}{180} \frac{(d^2-cm)H^4}{m^3} + \mathcal{O}(H^5) \\ \frac{5}{12} \frac{H}{m} - \frac{1}{8} \frac{dH^2}{m^2} + \frac{7}{240} \frac{(d^2-cm)H^3}{m^3} - \frac{1}{180} \frac{(d^3-2cdm)H^4}{m^4} + \mathcal{O}(H^5) \end{bmatrix} \quad (\kappa = 2) \end{aligned}$$

with the (constant) macro-step size  $H = T_{N+1} - T_N$ . The first terms of the series expansion for a higher polynomial degree  $\kappa$  are given in Table A.1.

**Table A.1:** Coefficients of the first term of the series expansion of the solution of the interface Jacobian ODE-system (A.2) at  $T_{N+1}$ .

$\kappa$	0	1	2	3	4	5	
$x_\lambda(T_{N+1})$	1/2	1/6	1/8	19/180	3/32	863/10080	$H^2/m$
$v_\lambda(T_{N+1})$	1	1/2	5/12	3/8	251/720	95/288	$H/m$

Note that most results within this section are shown for the cases  $\kappa = 0$  and  $\kappa = 1$  but the results for a higher degree  $\kappa$  are qualitatively equivalent.

Considering a variable macro-step size, i.e. non-equidistant sampling points of the approximation polynomials (for  $\kappa \geq 2$ ), the coefficients depend on the ratio  $a_i = \frac{T_{N+1} - T_{N+1-i}}{T_{N+1} - T_N}$  of the previous macro-step sizes. The first term of the series expansion, depending on the polynomial degree  $\kappa$ , is determined according to

$$\begin{aligned} \begin{bmatrix} x_\lambda(T_{N+1}) \\ v_\lambda(T_{N+1}) \end{bmatrix} &\approx \begin{bmatrix} \left( \frac{1}{6} - \frac{1}{12a_2} \right) \frac{H^2}{m} \\ \left( \frac{1}{2} - \frac{1}{6a_2} \right) \frac{H}{m} \end{bmatrix} \quad (\kappa = 2) \\ \begin{bmatrix} x_\lambda(T_{N+1}) \\ v_\lambda(T_{N+1}) \end{bmatrix} &\approx \begin{bmatrix} \left( \frac{1}{6} - \frac{1}{12} \left( \frac{1}{a_3} + \frac{1}{a_2} \right) + \frac{1}{20a_2a_3} \right) \frac{H^2}{m} \\ \left( \frac{1}{2} - \frac{1}{6} \left( \frac{1}{a_3} + \frac{1}{a_2} \right) + \frac{1}{12a_2a_3} \right) \frac{H}{m} \end{bmatrix} \quad (\kappa = 3) \end{aligned}$$

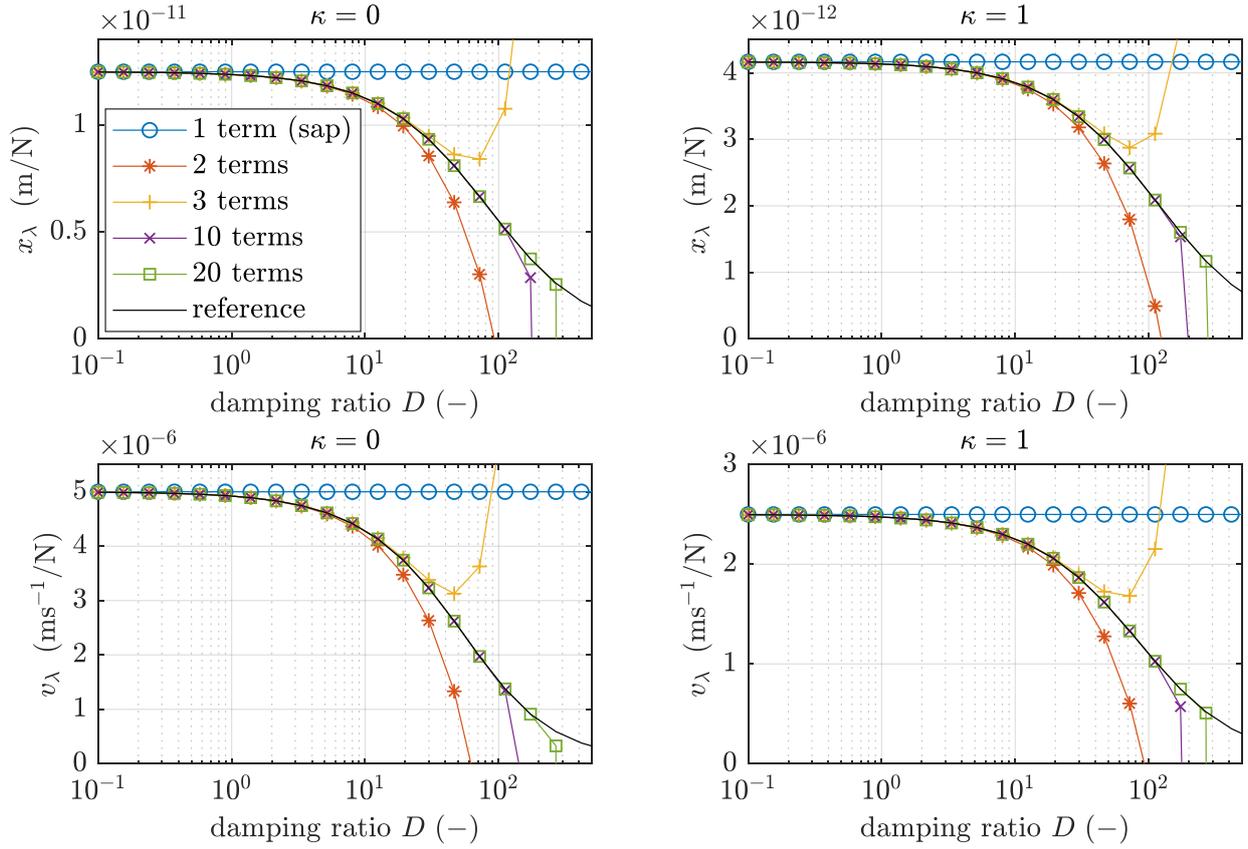
$$\begin{aligned} \begin{bmatrix} x_\lambda(T_{N+1}) \\ v_\lambda(T_{N+1}) \end{bmatrix} &\approx \begin{bmatrix} \left( \frac{1}{6} - \frac{1}{12} \left( \frac{1}{a_4} + \frac{1}{a_3} + \frac{1}{a_2} \right) + \frac{1}{20} \left( \frac{1}{a_4 a_3} + \frac{1}{a_4 a_2} + \frac{1}{a_3 a_2} \right) - \frac{1}{30 a_4 a_3 a_2} \right) \frac{H^2}{m} \\ \left( \frac{1}{2} - \frac{1}{6} \left( \frac{1}{a_4} + \frac{1}{a_3} + \frac{1}{a_2} \right) + \frac{1}{12} \left( \frac{1}{a_4 a_3} + \frac{1}{a_4 a_2} + \frac{1}{a_3 a_2} \right) - \frac{1}{20 a_4 a_3 a_2} \right) \frac{H}{m} \end{bmatrix} \quad (\kappa = 4) \\ \begin{bmatrix} x_\lambda(T_{N+1}) \\ v_\lambda(T_{N+1}) \end{bmatrix} &\approx \begin{bmatrix} \left( \frac{1}{6} - \frac{1}{12} \left( \sum_{i=2}^5 \frac{1}{a_i} \right) + \frac{1}{20} \left( \sum_{i=2}^4 \sum_{j=i+1}^5 \frac{1}{a_i a_j} \right) - \frac{1}{30} \left( \sum_{i=2}^3 \sum_{j=i+1}^4 \sum_{k=j+1}^5 \frac{1}{a_i a_j a_k} \right) + \frac{1}{42 a_5 a_4 a_3 a_2} \right) \frac{H^2}{m} \\ \left( \frac{1}{2} - \frac{1}{6} \left( \sum_{i=2}^5 \frac{1}{a_i} \right) + \frac{1}{12} \left( \sum_{i=2}^4 \sum_{j=i+1}^5 \frac{1}{a_i a_j} \right) - \frac{1}{20} \left( \sum_{i=2}^3 \sum_{j=i+1}^4 \sum_{k=j+1}^5 \frac{1}{a_i a_j a_k} \right) + \frac{1}{30 a_5 a_4 a_3 a_2} \right) \frac{H}{m} \end{bmatrix} \quad (\kappa = 5). \end{aligned}$$

The quality of the series approximation with a limited number of terms is mainly influenced by the damping of the system. Figure A.2 shows the approximation of the interface Jacobian  $[x_\lambda, v_\lambda]^T$  with a different number of series terms with respect to the damping ratio  $D = \frac{d}{2\sqrt{cm}}$  of the system. The values of the remaining parameters are set to  $c = 1.0e7 \text{ N/m}$ ,  $m = 1.0 \text{ kg}$  and  $H = 5.0e-6 \text{ s}$ . Constant ( $\kappa = 0$ ) and linear ( $\kappa = 1$ ) polynomials for  $\tilde{\lambda}(t)$  are considered.

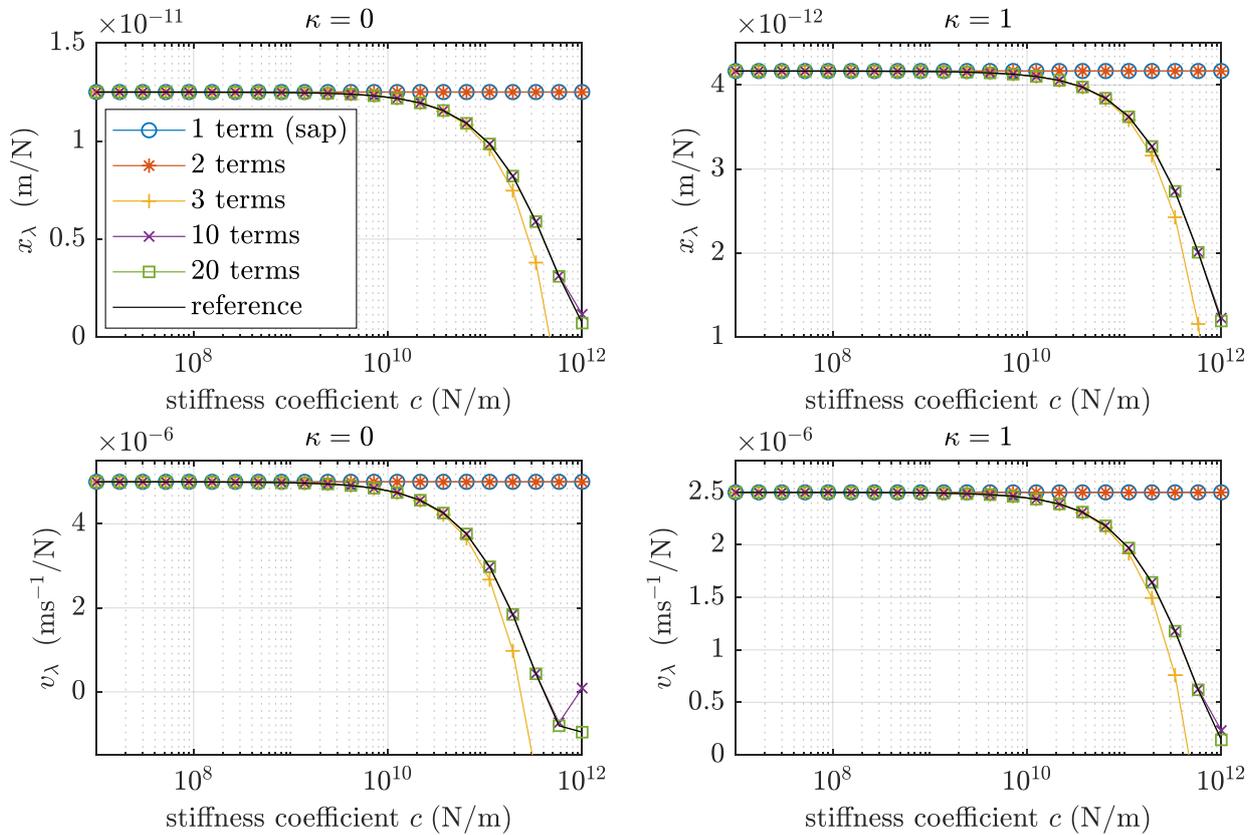
It can be seen that an approximation of the interface Jacobian with only the first term of the series expansion (*simple approximation, sap*) is sufficient for the linear oscillator with low damping ( $D < 1$ ). For higher damping ratios, the number of series terms which are required to obtain a useful approximation increases rapidly. The stiffness coefficient  $c$  does not appear before the third term of the series, therefore its influence on the interface Jacobian is lower. However, for high stiffness values the approximation with a limited number of series terms fails, as shown in Fig. A.3. The values of the remaining parameters are set to  $d = 1.0e2 \text{ Ns/m}$  (low damping),  $m = 1.0 \text{ kg}$  and  $H = 5.0e-6 \text{ s}$ .

An alternative approach for the approximation of the interface Jacobian is to solve the ODE-system (A.2) numerically. Since the approximation has to be computationally efficient, the solution should be obtained by a single integrator step. In addition, the convergence order of the integration method must be sufficient, namely  $\kappa + 1$  for  $v_\lambda$  and  $\kappa + 2$  for  $x_\lambda$ . Considering a co-simulation approach, the possibility to use at least quadratic ( $\kappa = 2$ ) approximation polynomials should be given, therefore a 4<sup>th</sup>-order method to solve the ODE-system for the interface Jacobian is required. As can be seen in Fig. A.4 and Fig. A.5 the implicit Runge-Kutta methods (Gauss-Legendre6 and Radau5) show very good results for the considered range of damping and stiffness parameters. The drawback of these methods is the rather high computational effort. The explicit Runge-Kutta methods of orders two and four (rk2 and rk4) fail for systems with high damping or stiffness values. The Newmark method [N<sup>+</sup>59] provides also inaccurate results. In addition, the convergence orders of the second order Runge-Kutta scheme and the Newmark method are not sufficient. Note that the numbers in brackets in the legends of Figs. A.4 and A.5 denote the values of the parameters  $(\beta, \gamma)$  of the Newmark scheme.

Figure A.6 shows the results of the interface Jacobian approximations for the case of a quadratic polynomial  $\tilde{\lambda}(t)$ . The offset of the curve obtained by the Radau3 method shows, that the convergence order of this method is not sufficient on position level ( $3 < \kappa + 2$ ). However, for the approximation on velocity level ( $v_\lambda$ ) the convergence order is sufficient ( $3 = \kappa + 1$ ).



**Figure A.2:** Series expansion of the interface Jacobian for different damping ratios.



**Figure A.3:** Series expansion of the interface Jacobian for different stiffness values.

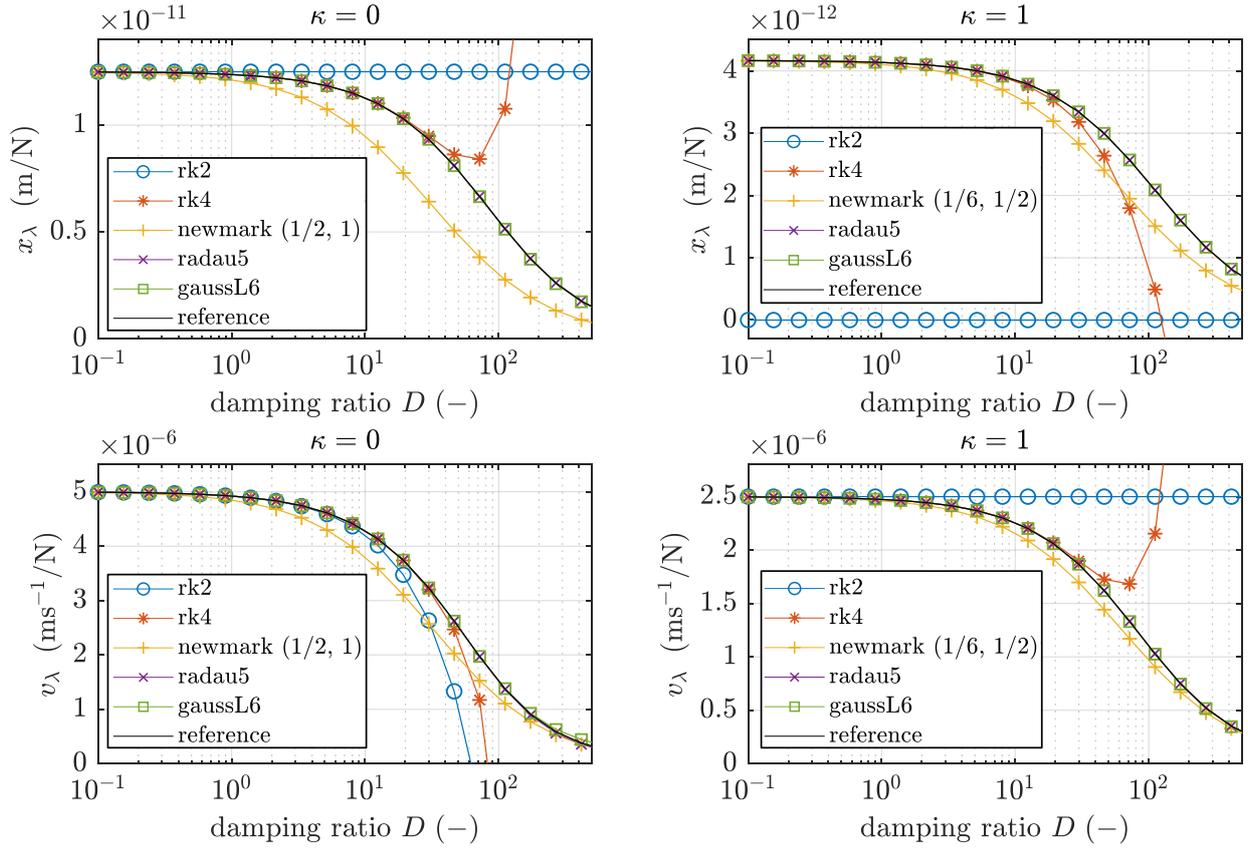


Figure A.4: Numerical integration of the interface Jacobian system for different damping ratios.

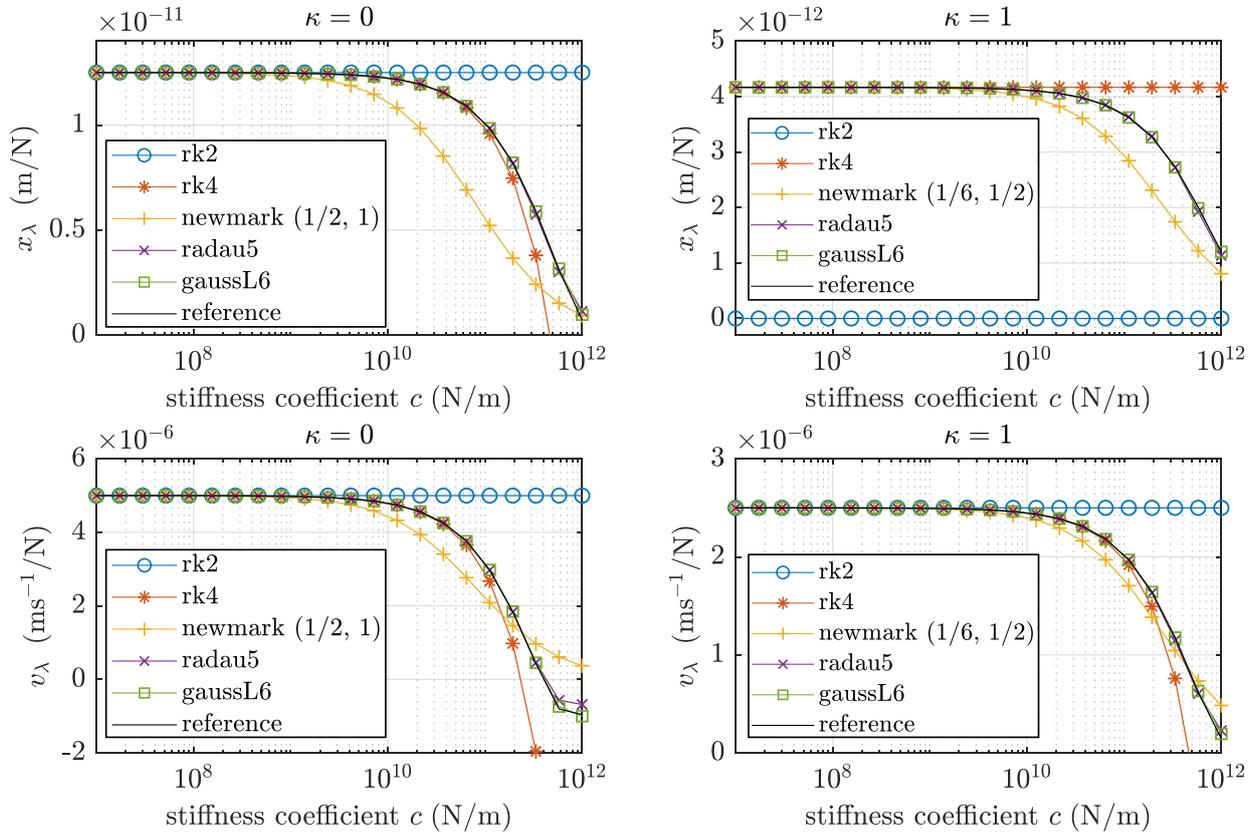
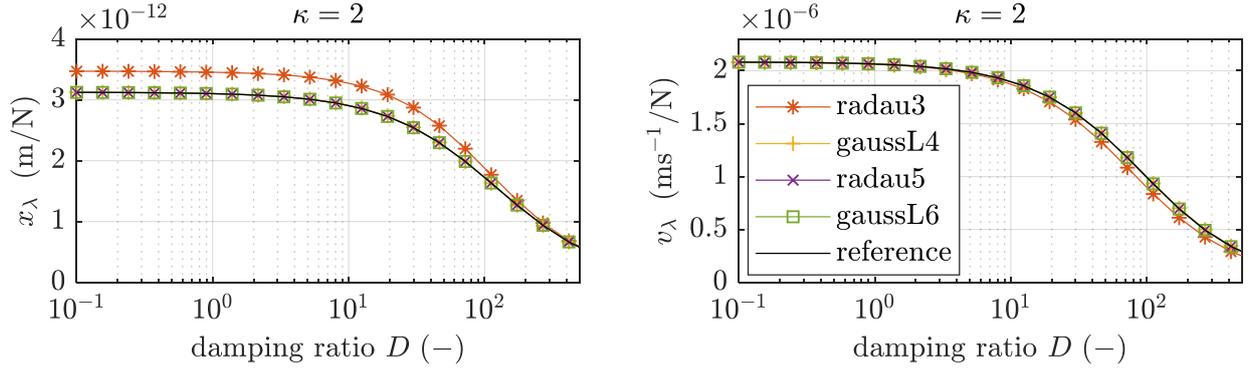


Figure A.5: Numerical integration of the interface Jacobian system for different stiffness values.



**Figure A.6:** Numerical integration of the interface Jacobian ODE-system with quadratic force polynomial ( $\kappa = 2$ ).

The results of the studies with the linear test model suggest the following conclusions:

- For linear systems with low or moderate damping a simple approximation (*sap*) of the interface Jacobian by considering only the first term of the series expansion of the solution of Eq. (A.2) seems to be sufficient.
- Highly damped or very stiff systems require a more sophisticated approach to obtain a suitable approximation of the interface Jacobian. Assuming that the approximation should be obtained in a single integrator step, the ODE-system Eq. (A.2) has to be solved by an implicit Runge-Kutta scheme of a sufficient convergence order ( $\geq \kappa + 2$ ). However, the computational effort of an implicit Runge-Kutta scheme is likely to be too high for using it to compute the interface Jacobian within a co-simulation implementation.
- It should be noted, that the studies have been carried out under the assumption of a constant macro-step size. If the step size is chosen sufficiently small, the simple approximation approach delivers accurate results for the interface Jacobian. However, if the macro-step size of a co-simulation model has to be reduced only for the reason to obtain an estimation of the interface Jacobian, the approximation method is not suitable.

## A.2. Nonlinear Systems

To analyze the approximation methods of the interface Jacobian of a nonlinear subsystem, third-order terms are added to the force law of the spring/damper-element of the test model shown in Fig. A.1 according to

$$F_{SD} = cx + dv + Cx^3 + Dv^3. \quad (\text{A.6})$$

Note that  $D$  denotes the nonlinear damping coefficient and not the damping ratio within the current subsection. The equations of motion of the nonlinear system for the case  $\kappa = 1$  read

$$\begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v \\ F(x, v) + g(\lambda) \end{bmatrix} = \begin{bmatrix} v \\ -\frac{1}{m} (cx + dv + Cx^3 + Dv^3) + \frac{1}{m} \left( \lambda_N + \frac{(\lambda - \lambda_N)(t - T_N)}{T_{N+1} - T_N} \right) \end{bmatrix}. \quad (\text{A.7})$$

A differentiation with respect to  $\lambda$  yields

$$\begin{bmatrix} \dot{x}_\lambda \\ \dot{v}_\lambda \end{bmatrix} = \begin{bmatrix} v_\lambda \\ F_\lambda(x, v, x_\lambda, v_\lambda) + g_\lambda(\lambda) \end{bmatrix} = \begin{bmatrix} v_\lambda \\ -\frac{1}{m} (cx_\lambda + dv_\lambda + 3Cx^2x_\lambda + 3Dv^2v_\lambda) + \frac{1}{m} \left( \frac{t-T_N}{T_{N+1}-T_N} \right) \end{bmatrix}. \quad (\text{A.8})$$

As can be seen in Eq. (A.8), not only  $x_\lambda$  and  $v_\lambda$  appear as in the linear case, but also the state variables  $x$  and  $v$ . Therefore, the coefficient matrix

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -c/m - 3Cx^2/m & -d/m - 3Dv^2/m \end{bmatrix} \quad (\text{A.9})$$

is time-dependent. Considering a parallelized co-simulation, the interface Jacobians are computed in parallel to the predictor step. The values of the state variables  $x(t)$  and  $v(t)$  are therefore unknown within the macro-time interval  $T_N \rightarrow T_{N+1}$ . To approximate the time-dependent coefficient matrix  $\mathbf{A}$ ,  $x(t)$  and  $v(t)$  are approximated by extrapolation polynomials using the values at the previous macro-time points as supporting points. The parameters for the following numerical examples are given in Tab. A.2.

Figure A.7 shows the approximated interface Jacobian ( $x_\lambda(T_{N+1})$  and  $v_\lambda(T_{N+1})$ ) for the nonlinear subsystem, obtained with different extrapolation orders of  $x(t)$  and  $v(t)$ . It can be seen that when the nonlinear damping coefficient is increased over  $1.0e2 \text{ N s}^3/\text{m}^3$ , at least quadratic polynomials for  $x(t)$  and  $v(t)$  have to be used to obtain a suitable approximation of the interface Jacobian. The ODE-system Eq. (A.8) is solved with the Runge-Kutta-Fehlberg (rkf45) method [Feh68] with an error tolerance of  $1.0e-12$  to minimize the error of the solver and to study the influence of the polynomial degree on the results.

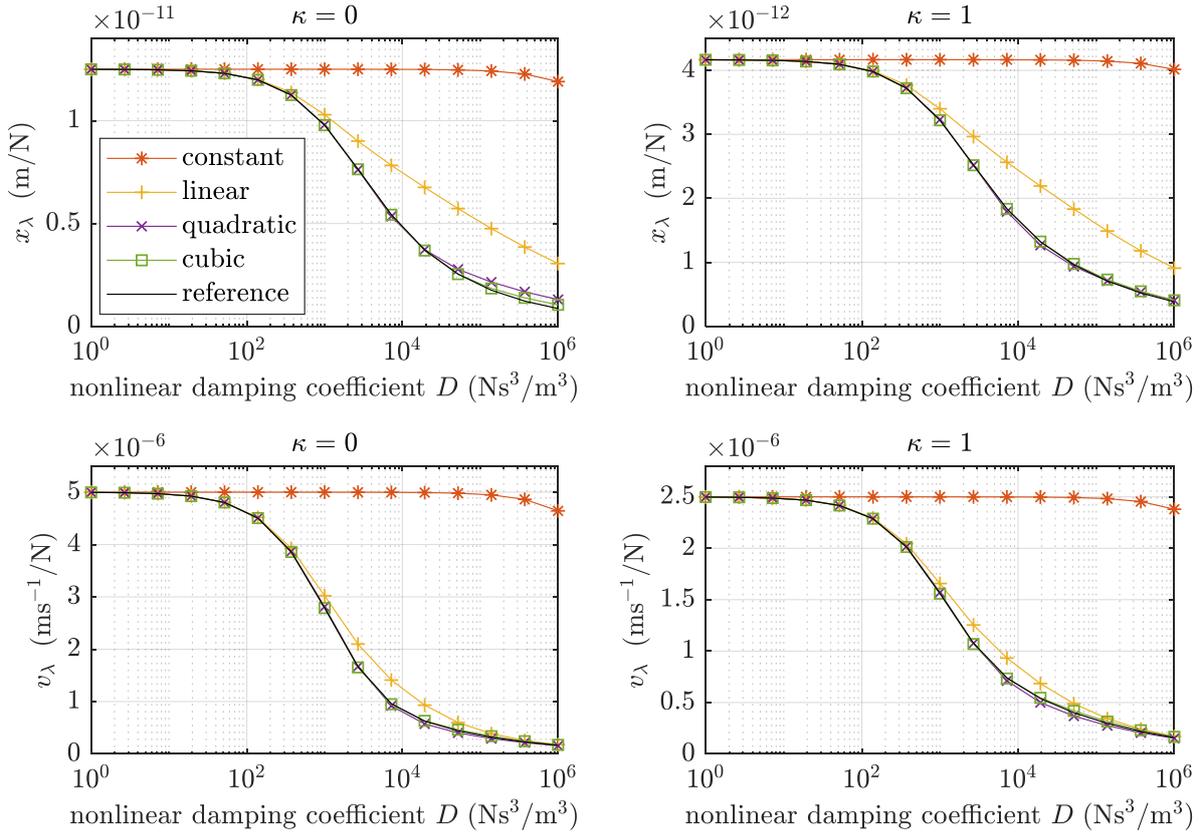
**Table A.2:** Nonlinear subsystem parameters.

$m$	1.0 kg
$c$	$1.0e7 \text{ N/m}$
$d$	$1.0 \text{ N s/m}$
$C$	$1.0e9 \text{ N/m}^3$
$D$	$1.0 \text{ N s}^3/\text{m}^3$
$H$	$5.0e-6 \text{ s}$

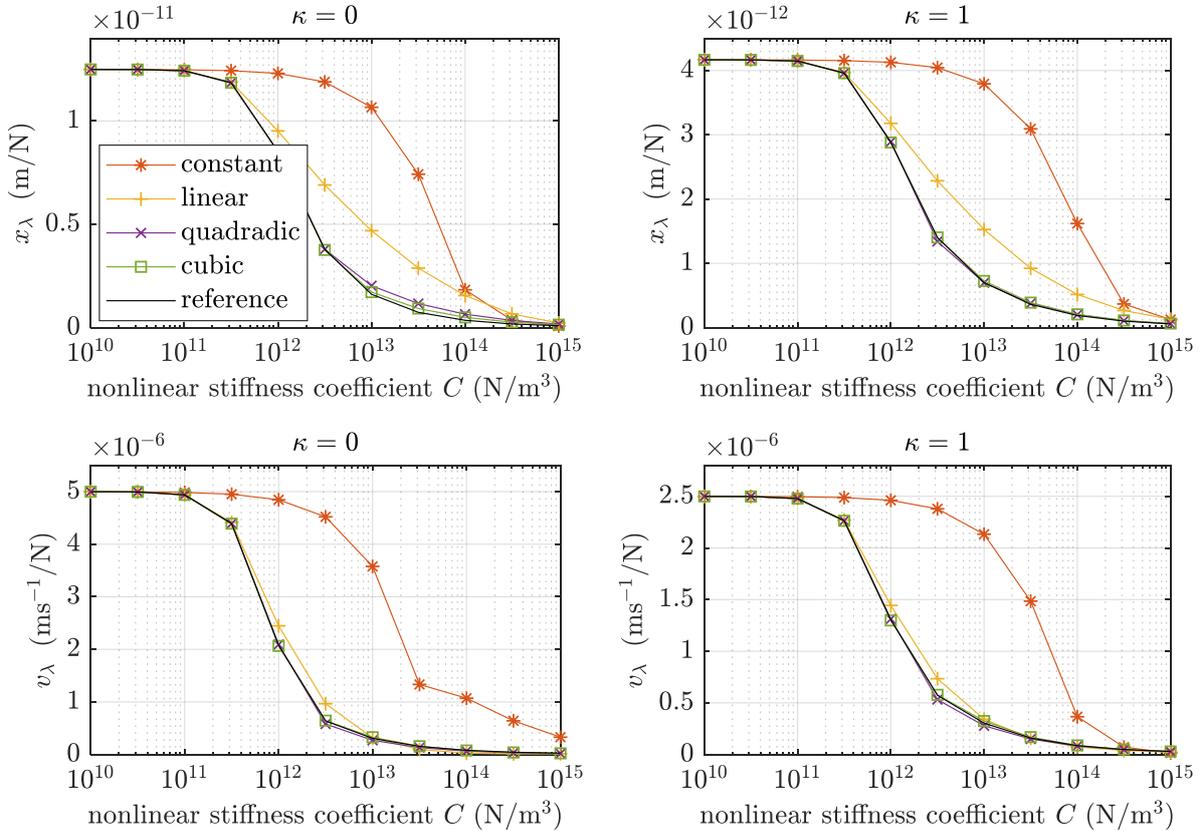
When the nonlinear stiffness coefficient is varied instead of the nonlinear damping coefficient the results are qualitatively equivalent, as shown in Fig. A.8. For higher values of the nonlinear stiffness coefficient, the approximation polynomials for  $x(t)$  and  $v(t)$  have to be of at least quadratic order to obtain useful results for the interface Jacobian.

Next, the same ODE-system Eq. (A.8) is solved with different numerical integration methods. It should be stressed that the system is solved with only one integrator step (except for rkf45); the states  $x(t)$  and  $v(t)$  are approximated by quadratic polynomials. The results ( $x_\lambda(T_{N+1})$  and  $v_\lambda(T_{N+1})$ ) are shown in Fig. A.9 and Fig. A.10. As expected the explicit Runge-Kutta (rk4) method fails. The implicit Runge-Kutta schemes, especially the Gauss-Legendre6 method, show good results for the case of linear coupling force polynomials ( $\kappa = 1$ ) and also partly for  $\kappa = 0$ .

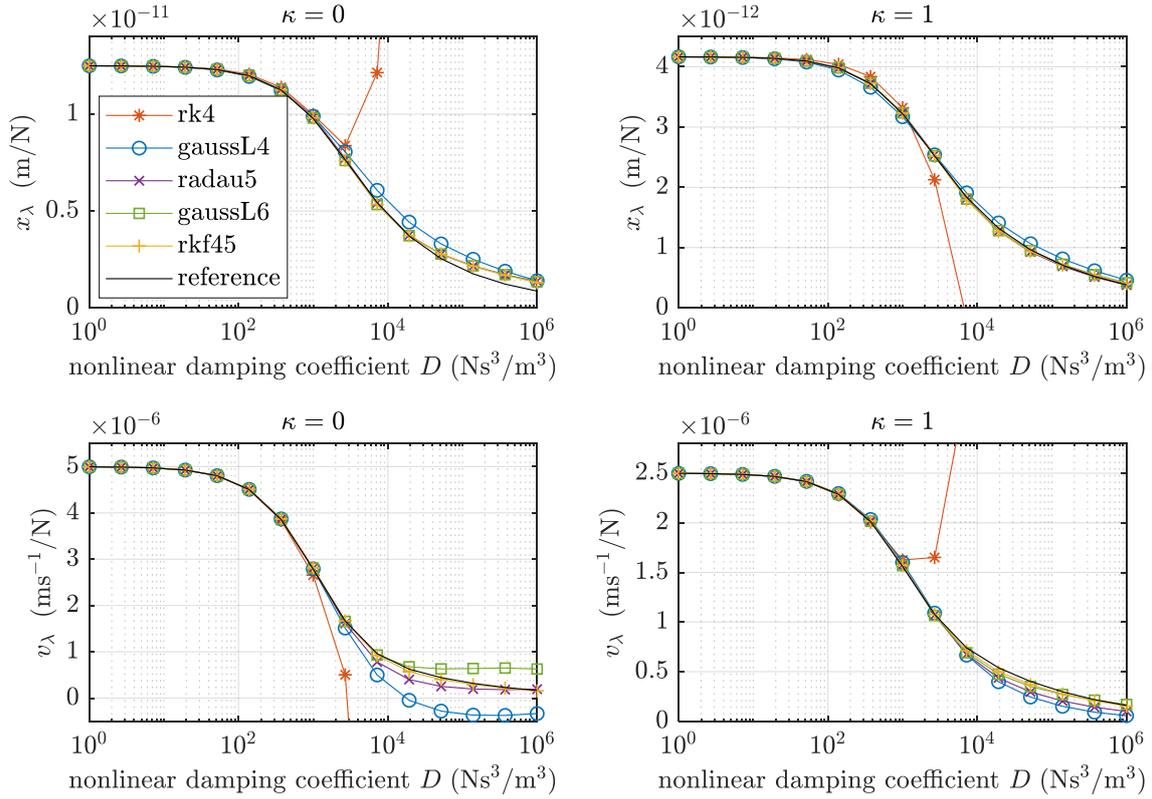
To obtain an approximation of the interface Jacobian of a nonlinear system, the state variables have to be estimated for the considered macro-step before the ODE-system Eq. (A.8) can be solved (numerically). The studies with the nonlinear test system indicate, that a constant approximation ( $x(t) \approx x(T_N)$ ,  $v(t) \approx v(T_N)$ ) of the state variables is not sufficient and that approximation polynomials of at least quadratic order have to be employed to obtain suitable results for higher nonlinear stiffness and damping coefficients. The ODE-system for the interface Jacobian can then be solved by an implicit Runge-Kutta scheme of a sufficient convergence order. This process of



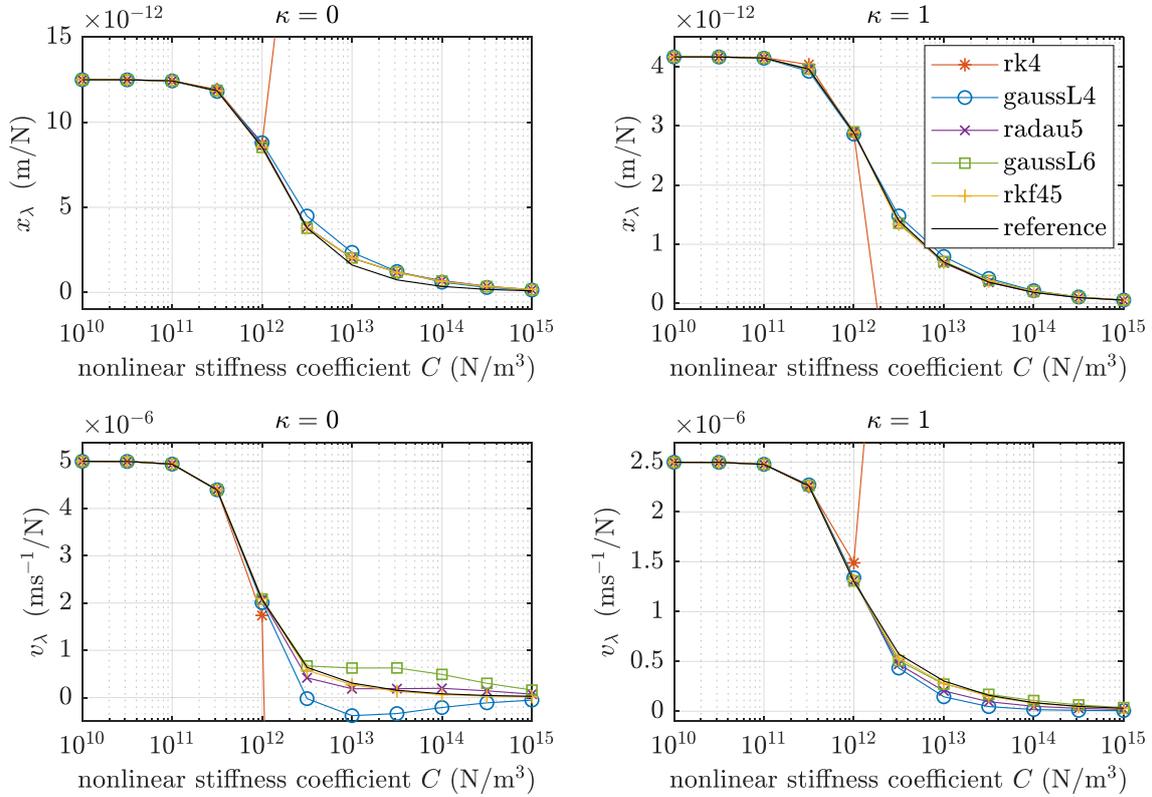
**Figure A.7:** Nonlinear system: numerical integration of the interface Jacobian ODE-system with rkf45 for different approximation orders of  $x(t)$  and  $v(t)$  and varying nonlinear damping.



**Figure A.8:** Nonlinear system: numerical integration of the interface Jacobian ODE-system with rkf45 for different approximation orders of  $x(t)$  and  $v(t)$  and varying nonlinear stiffness.



**Figure A.9:** Nonlinear system: numerical integration of the interface Jacobian ODE-system with different methods for varying nonlinear damping values, using quadratic approximation polynomials for  $x(t)$  and  $v(t)$ .



**Figure A.10:** Nonlinear system: numerical integration of the interface Jacobian ODE-system with different methods for varying nonlinear stiffness values, using quadratic approximation polynomials for  $x(t)$  and  $v(t)$ .

approximating the state variables by extrapolation polynomials and solving the ODE-system by an implicit Runge-Kutta method is likely to be too expensive to be implemented in a co-simulation scheme. However, if the system has only moderate nonlinearities, a simple approximation of the interface Jacobian by using constant values of the state variables may be sufficient.

### A.3. Numerical Examples

#### Example 1: Lumped Mass Model of a Flexible Steel Rod

Within the following subsection, a flexible steel rod is discretized as a lumped mass model, as shown in Fig. A.11. The rod is excited by an external impulse shaped force  $F_{IMP,n_K}$  according to Eq. (2.13) along the longitudinal axis only at  $t_{I,n_K} = 5.0e-7$  s. The physical parameters of the rod are given in Table A.3, the external force is defined by the parameters  $\Delta F_{I,n_K} = -5.0e5$  N,  $\Delta t_I = 5.0e-5$  s and  $\delta_I = 1.0e-7$  s. The discretized model is split into 20 equal-sized subsystems and is computed using the implicit co-simulation method with the macro-step size and order control algorithm (*imMD*,  $rtol = 10^{-5}$ ,  $atol^{pos} = 10^{-3} \cdot rtol$ ,  $atol^{vel} = rtol$ ). Simulations are carried out with different approximation methods for the interface Jacobian to study the effect on the number of macro-steps and especially on the number of corrector iterations. An increased number of corrector iterations is an indicator for the low quality of the interface Jacobian. The number of corrector iterations is limited by  $n_{cor}^{max} = 10$  corrector iterations per macro-step. If the corrector is not converged after 10 iterations, the macro-step is repeated with a decreased macro-steps size  $H_{new} = 0.25 \cdot H$ .

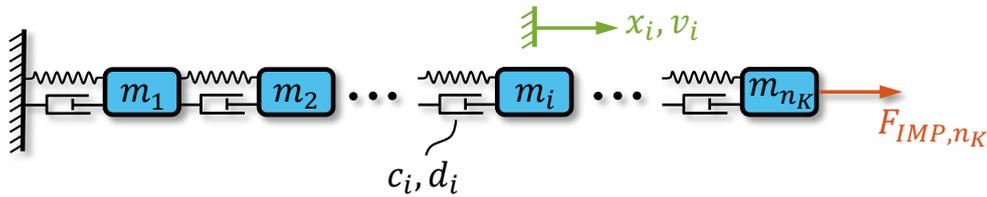


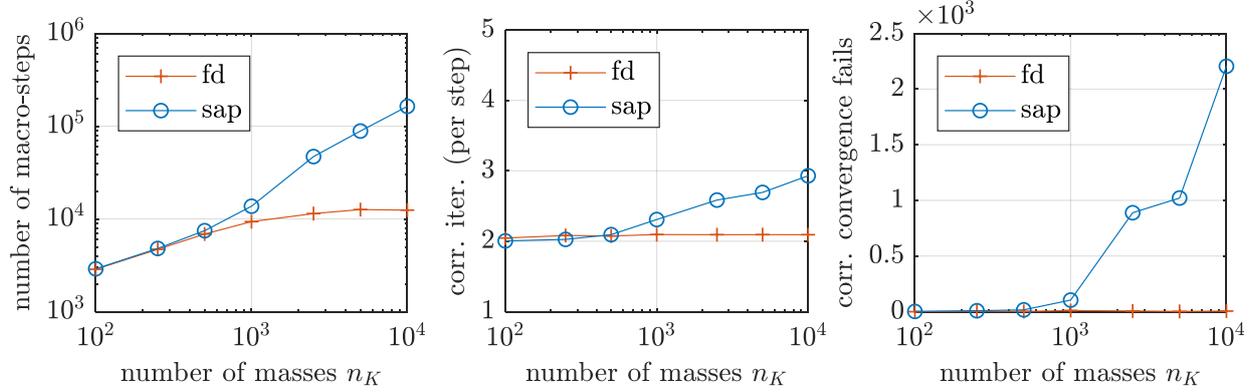
Figure A.11: Lumped mass model of a flexible steel rod.

Table A.3: Steel rod parameters.

Parameter	Value
mass	$4.93e-1$ kg
length	0.20 m
radius	0.01 m
density	$7.85e3$ kg/m <sup>3</sup>
Young's modulus	$2.10e11$ N/m <sup>2</sup>
damping ratio	$1.00e-3$

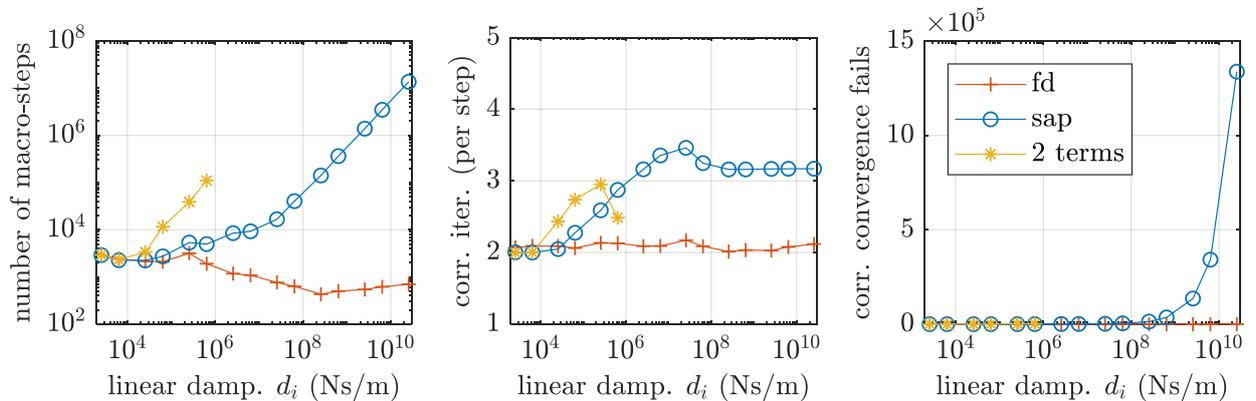
Figure A.12 shows the number of macro-steps of co-simulations with different discretizations. As expected, the number of macro-steps increases with the number of masses because of the increasing element-stiffness. The interface Jacobian is obtained by two different methods: finite differences (*fd*) and by computing the first term of the series expansion of the solution of the ODE-system for the interface Jacobian (*sap*) (see Eq. (A.5)). It should be noted that the *sap* approach depends only on the masses of the coupling bodies and on the macro-step size, since there are no

constraints on the coupling bodies within the subsystems. It can be seen, that for discretizations with  $n_K < 1000$  elements, both methods show a similar performance. If the discretization is refined further, the *sap* approach requires a significantly smaller macro-step size and also an increased number of corrector iterations.



**Figure A.12:** Lumped mass model: number of macro-steps, number of corrector iterations per macro-step and number of corrector convergence fails depending on the discretization.

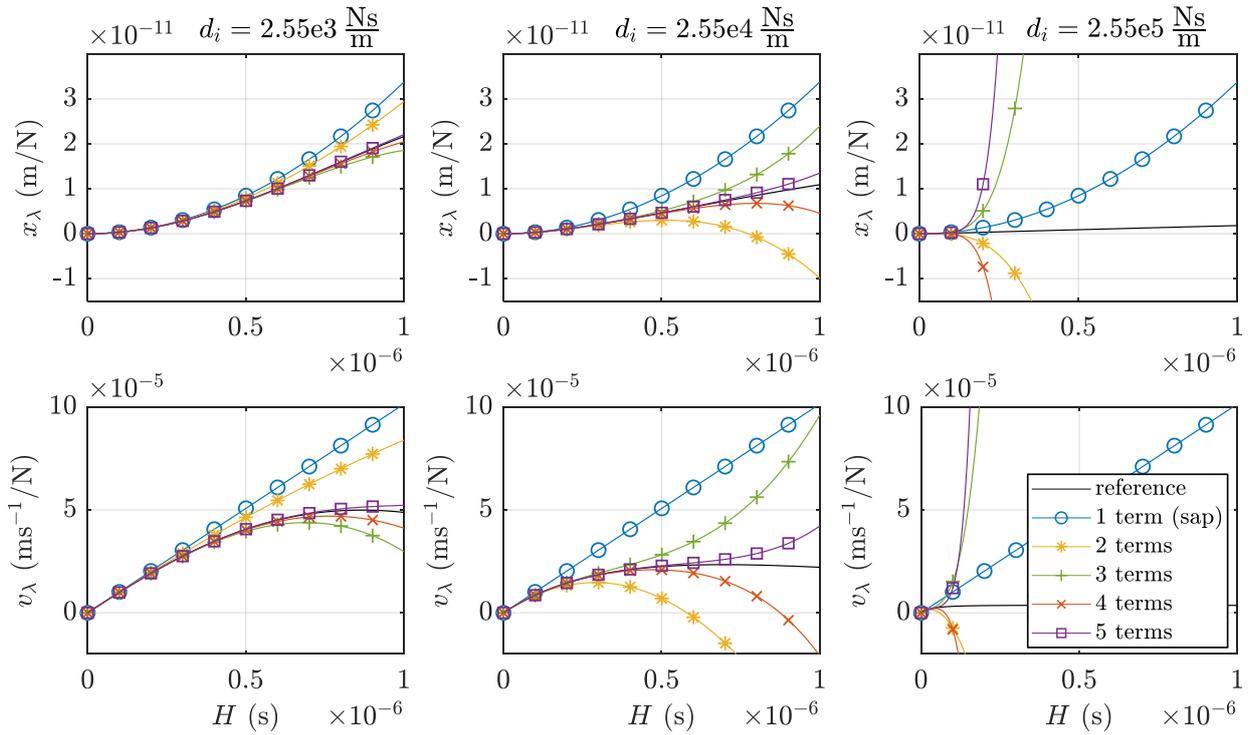
The lower performance of the *sap* approach for very fine discretizations is mainly caused by the increased damping ratio of the elements. Fig. A.13 shows the dependency of the different interface Jacobian approximations on the damping coefficient of the elements. The  $n_K = 100$  masses  $m_i = 4.93e-3$  kg and the stiffness coefficients  $c_i = 3.30e10$  N/m are constant. Besides the *fd* and the *sap* approach, also an approximation by computing the first and the second terms of the series expansion (see Eq. (A.5)) of the interface Jacobian ODE-system is applied. Figure A.13 shows the results of the simulations with a 100-element discretization of the rod depending on the damping coefficients. It should be mentioned, that the increased damping ratio is used for the numerical studies only and does not represent the physical properties of a steel rod. It can be seen that the approximation by the first term of the series expansion (*sap*) provides good results for  $D = \frac{d_i}{2\sqrt{c_i m_i}} < 1$ . For higher damping ratios, the finite differences approximation is clearly superior.



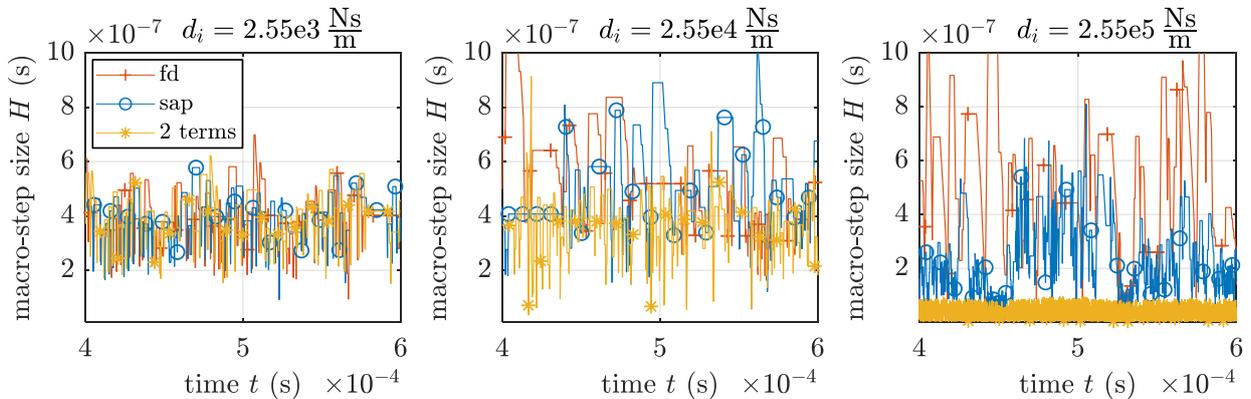
**Figure A.13:** Lumped mass model: number of macro-steps, number of corrector iterations per macro-step and number of corrector convergence fails depending on the damping coefficients  $d_i$  (100 element discretization).

An interesting observation is that an approximation of the interface Jacobian by two series terms

instead of using only the first term, does not improve the performance of the implicit co-simulation. A stable co-simulation of the considered system with damping coefficients  $d_i > 6.38e5 \text{ Ns/m}$  could not be achieved, if the interface Jacobian is approximated by using two terms of the series expansion. This can be explained by the convergence behavior of the series expansion. Figure A.14 shows the gradients of an arbitrary coupling body with respect to the corresponding coupling variable, obtained by approximations with the given number of series terms, depending on the macro-step size. The gradients are plotted for three different values of the damping coefficients and linear approximation polynomials ( $\kappa = 1$ ). However, the same observations can be made for different values of  $\kappa$ . It can be seen, that the higher order approximations diverge from the analytical solution very rapidly if the macro-step size is increased.



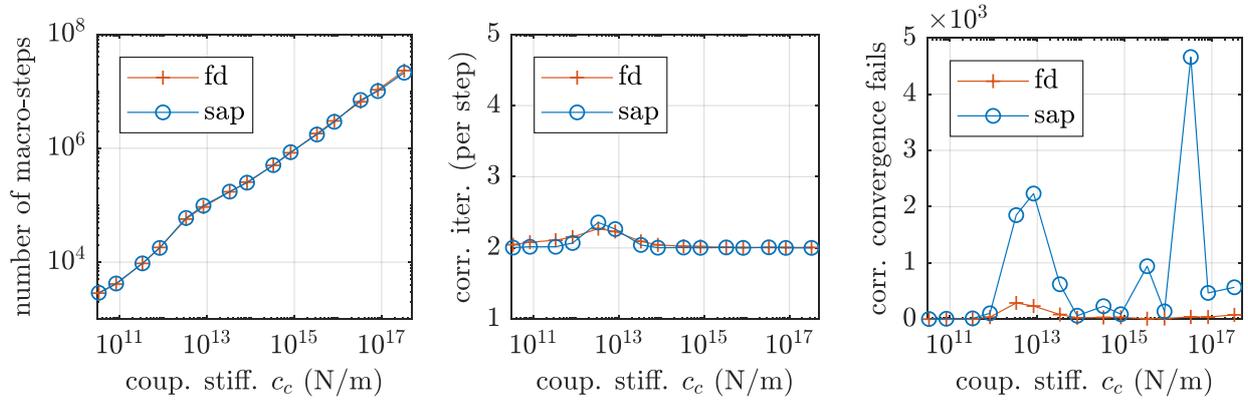
**Figure A.14:** Series expansion of the interface Jacobian depending on the macro-step size  $H$  for three different values of the damping coefficients  $d_i$  ( $\kappa = 1$ ).



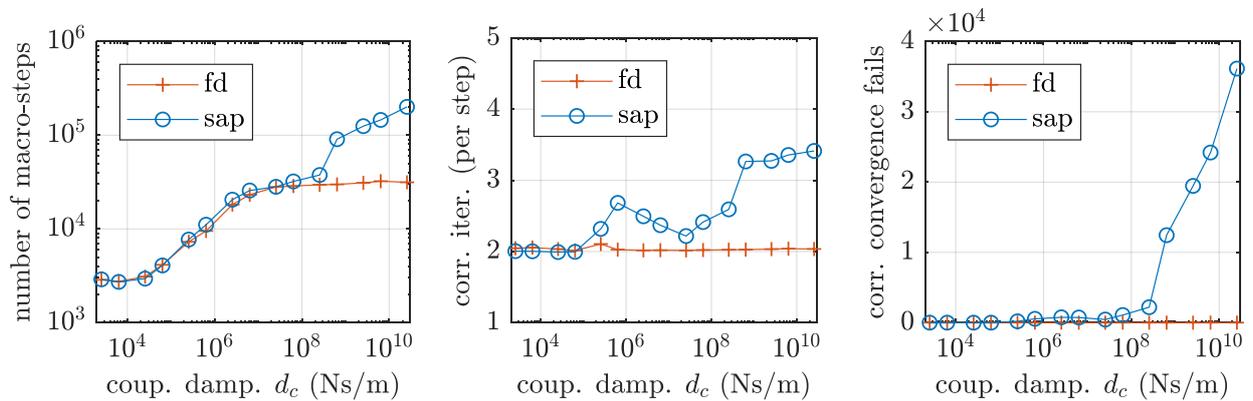
**Figure A.15:** Lumped mass model: macro-step size  $H$  (zoom) of simulations using different approximation methods of the interface Jacobians for different values of the damping coefficients  $d_i$ .

Numerical tests indicate that the macro-step size of the co-simulation, which is computed by the macro-step size controller, is often larger than the step size that would be required to obtain an accurate approximation of the interface Jacobian by a series expansion with a limited number of terms, as shown in Fig. A.15. Therefore, an approximation of the gradients by using only the first term of the series expansion (*sap*) allows a more robust and efficient co-simulation than using more terms, because even if the approximation is not very accurate, it is typically at least of the correct order of magnitude.

Next, the dependency of the approximated interface Jacobian on the coupling properties is studied. Therefore, simulations are carried out with increased coupling stiffness and damping coefficients. The results shown in Figs. A.16 and A.17 indicate that the stiffness coefficient of the coupling element does not affect the quality of approximated interface Jacobian significantly. An increased coupling stiffness entails a reduced macro-step size, which has a positive effect on the *sap* approach. For strongly damped coupling elements, the performance of the *fd* approximation is clearly superior.



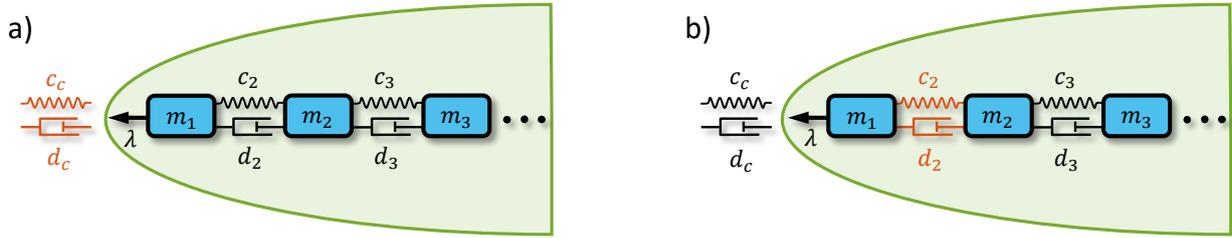
**Figure A.16:** Lumped mass model: number of macro-steps, number of corrector iterations per macro-step and number of corrector convergence fails depending on the coupling stiffness  $c_c$  (100 element discretization).



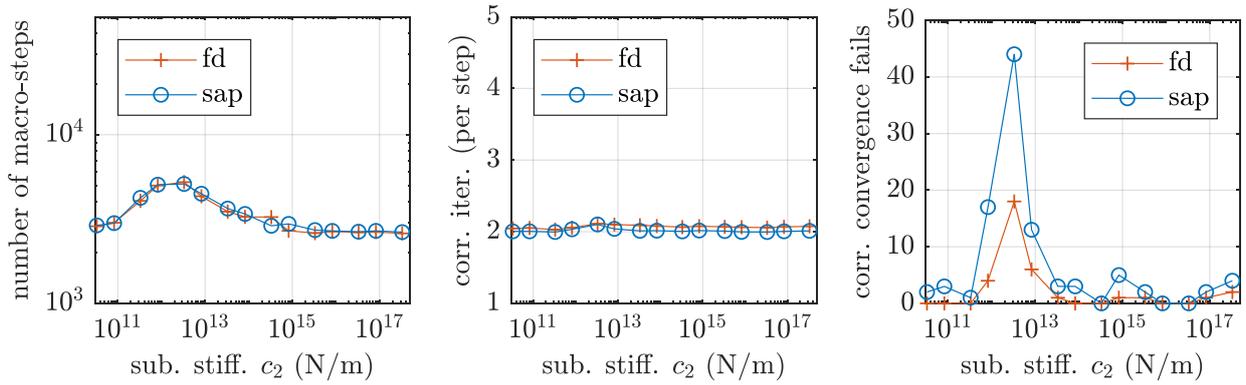
**Figure A.17:** Lumped mass model: number of macro-steps, number of corrector iterations per macro-step and number of corrector convergence fails depending on the coupling damping  $d_c$  (100 element discretization).

Increasing the stiffness and damping parameters of the element next to the coupling element (see Fig. A.18b) instead of the coupling element, does not reveal a significant difference in the

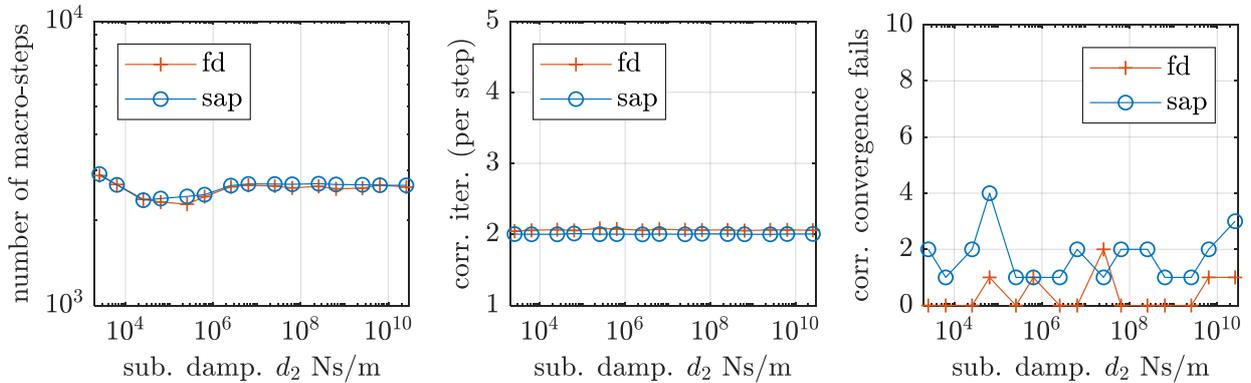
performance of the two approximation methods of the interface Jacobian, as shown in Figs. A.19 and A.20.



**Figure A.18:** Arbitrary subsystem: a) coupling element, b) element next to the coupling element.



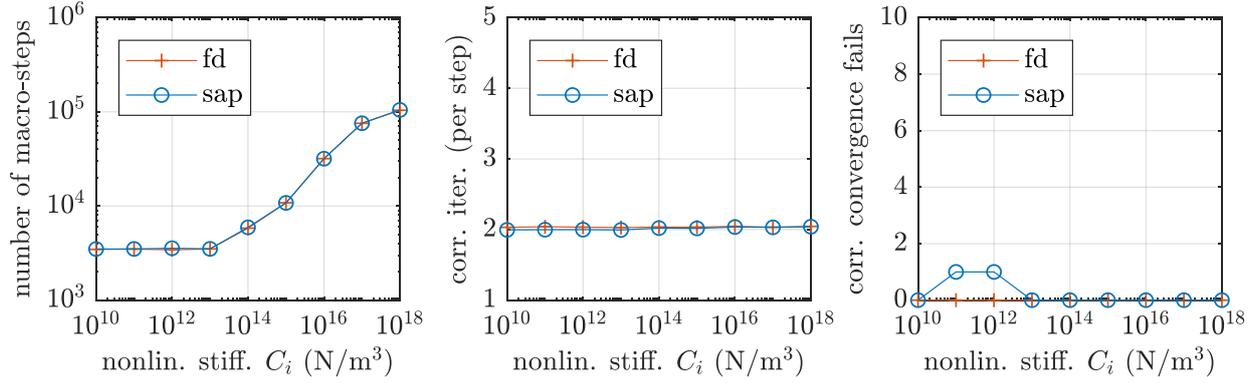
**Figure A.19:** Lumped mass model: number of macro-steps, number of corrector iterations per macro-step and number of corrector convergence fails depending on the subsystem stiffness (100 element discretization).



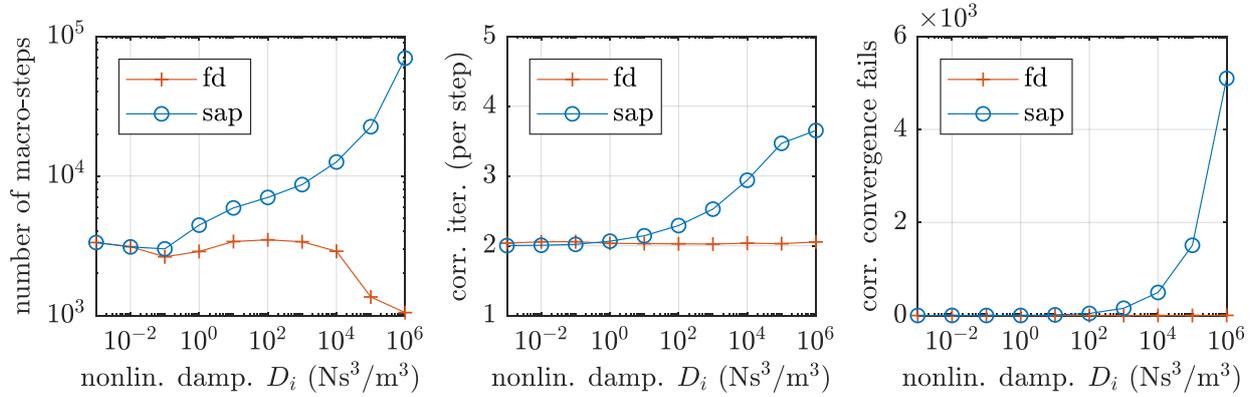
**Figure A.20:** Lumped mass model: number of macro-steps, number of corrector iterations per macro-step and number of corrector convergence fails depending on the subsystem damping (100 element discretization).

Finally, the influence of nonlinearities on the quality of the interface Jacobian approximations is studied. Therefore, the constitutive law of the spring/damper-elements is extended by adding third order terms according to Eq. (A.6). Again, the finite differences approximation (*fd*) is compared to the simple approximation method (*sap*), which computes only the first term of the series expansion of the solution of the (linear) interface Jacobian ODE-system. The resulting number of macro-steps and corrector iterations shown in Figs. A.21 and A.22 suggest, that the simple approximation method is sufficient if only nonlinear stiffness terms are added to the system. When nonlinear

damping occurs, the finite differences approximation is clearly superior.



**Figure A.21:** Nonlinear lumped mass model: number of macro-steps, number of corrector iterations per macro-step and number of corrector convergence fails depending on the nonlinear stiffness coefficients  $C_i$  (100 element discretization).



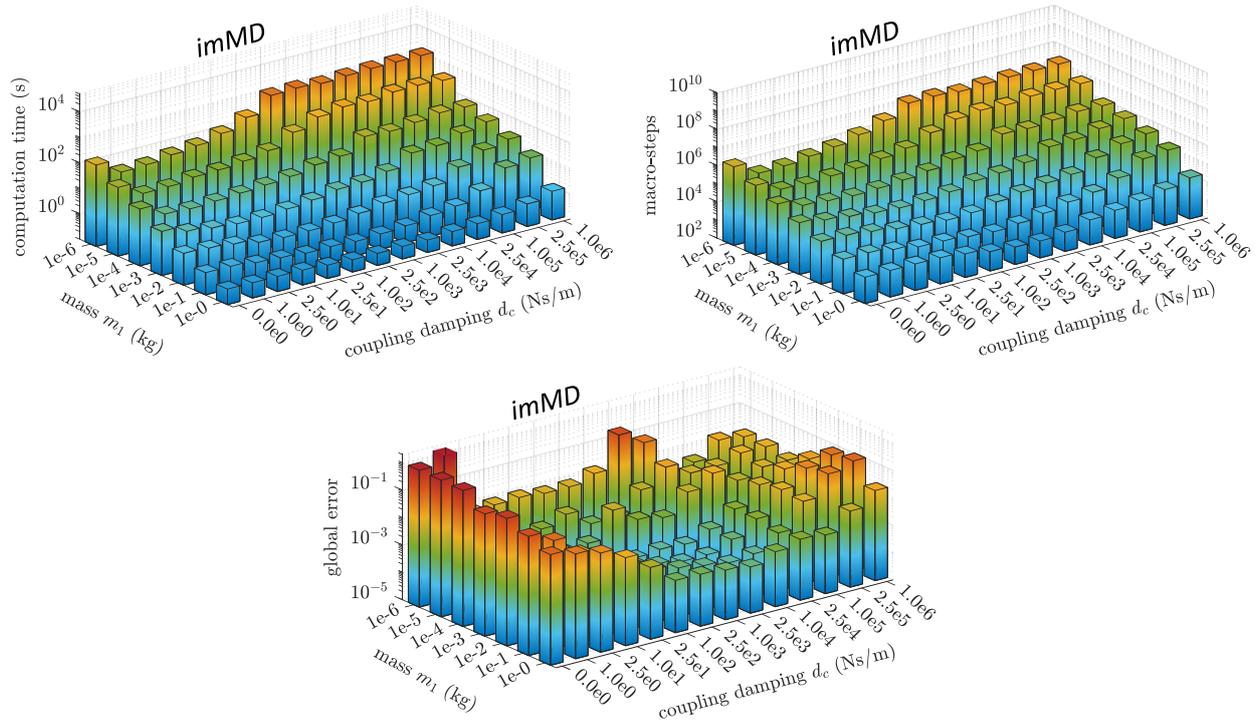
**Figure A.22:** Nonlinear lumped mass model: number of macro-steps, number of corrector iterations per macro-step and number of corrector convergence fails depending on the nonlinear damping coefficients  $D_i$  (100 element discretization).

### Example 2: Asymmetric Linear Two-Mass Oscillator

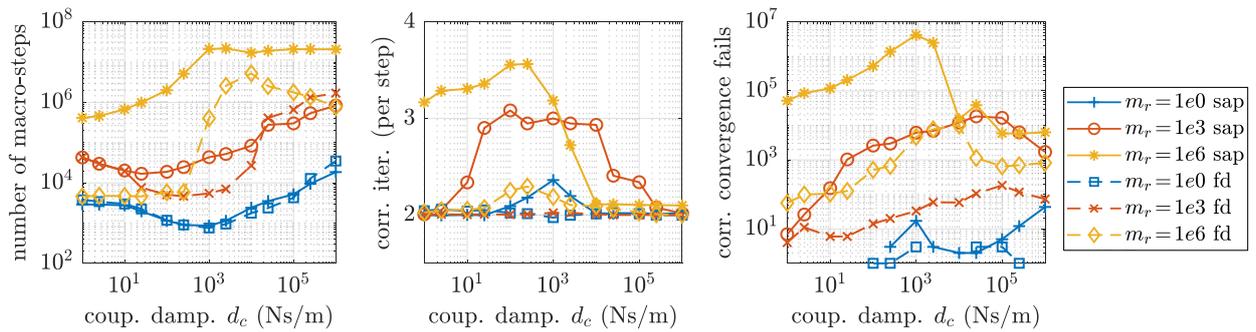
The linear two-mass oscillator of Section 5.5 is computed again using the implicit co-simulation method (*imMD*) with the macro-step size and order control algorithm. The simulations are carried out by approximating the interface Jacobian with the first series term (*sap*) of Eq. (A.5) instead of the finite differences approach. The results are shown in Fig. A.23. It should be mentioned, that the limit  $\tau$  of the convergence criterion (2.31) of the corrector iteration has to be reduced to  $\tau = 0.01$  to obtain a stable simulation for the case of high values of the mass ratio  $\frac{m_2}{m_1} \gg 1$ . Using the finite differences approach,  $\tau = 0.33$  is sufficient.

In Fig. A.24, the performance of the fully controlled implicit co-simulation method in combination with the *sap* approach is compared to the performance of the same method but using the finite differences approximation of the interface Jacobian. It can be seen that both methods show a similar performance for the symmetric system  $m_r := \frac{m_2}{m_1} = 1$ . When the mass ratio is increased to  $m_r = 10^3$ , the average number of corrector iterations of the *sap* approach is  $\bar{n}_{cor} \approx 3$  for most of the damping values, while the *fd* approximation requires only two iterations per macro-step.

The number of convergence errors using the *sap* approach is also significantly higher. When the mass ratio is increased further to  $m_r = 10^6$ , there is a drastic discrepancy in the number of macro-steps between the two approximation techniques. It can be concluded that the simple approximation of the interface Jacobian in combination with the macro-step size controller shows a good performance for moderate mass ratios even for strongly damped coupling elements. If the mass ratio of the coupling bodies is increased over a certain level then the *fd* approximation is more efficient.



**Figure A.23:** Asymmetric two-mass oscillator: computation time, number of macro-steps and resulting global error of the fully controlled implicit co-simulation approach (*imMD*) in connection with the simple approximation of the interface Jacobian (*sap*).



**Figure A.24:** Asymmetric two-mass oscillator: total number of macro-steps, average number of corrector iterations per macro-step, and total number of convergence fails for using the approximation of the interface Jacobian by the first series term (*sap*) and by finite differences (*fd*). Results are shown for three different mass ratios  $m_r = \frac{m_2}{m_1}$ .

It should be mentioned that both approximation techniques show a similar performance for the undamped system ( $d_c = 0$ ), even for very high mass ratios. The results of the undamped system are not shown in Fig. A.24 due to the logarithmic scaling of the  $x$ -axis.

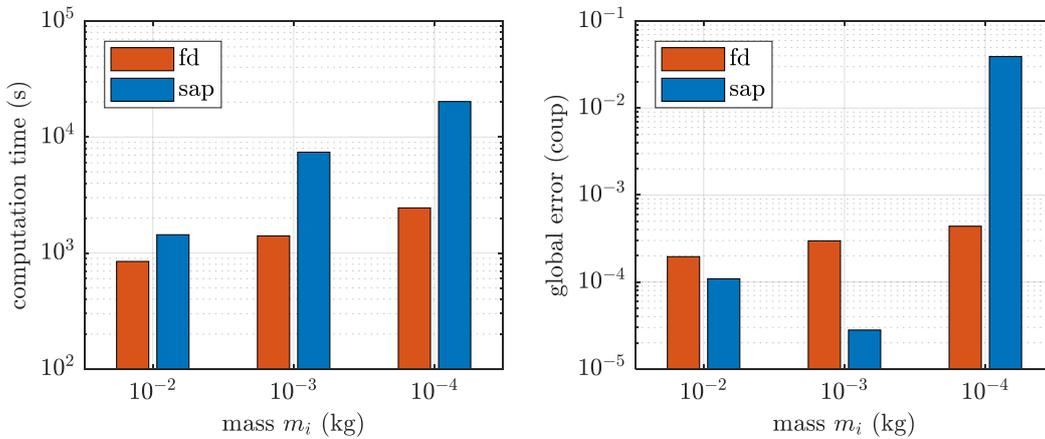
### Example 3: Nonlinear Oscillator Chain

The third example is a nonlinear oscillator chain with strong external loads and rather high damping values. The model parameters are given in Table A.4. The mass of each body is decreased successively from  $1.0e-2$  kg to  $1.0e-4$  kg to increase the numerical effort of the implicit co-simulation scheme. The initial conditions of each mass are selected randomly within the intervals  $[-1.0e-1, 1.0e-1]$  m and  $[-1.0e3, 1.0e3]$  m/s. Randomly picked 50% of the masses are excited by a modified sinus force according to Eq. (2.14) with the force amplitude  $\Delta F_{S,i} = \pm 1.0e8$  N. The angular frequency  $\Omega_{S,i}$  is selected randomly within the interval  $[5.0e3, 1.0e4]$  s<sup>-1</sup>, the exponent is set to  $A_S = 95$ . Another randomly picked 50% of the masses are affected by an impulse shaped force according to Eq. (2.13) with the parameters  $\Delta F_{I,i} = \pm 1.0e8$  N,  $\delta_I = 1.0e-6$  s and  $\Delta t_{I,i} = 1.0e-3$  s. The time  $t_{I,i}$  of occurrence of the impulse shaped force on each mass is also chosen randomly.

**Table A.4:** Test model parameters.

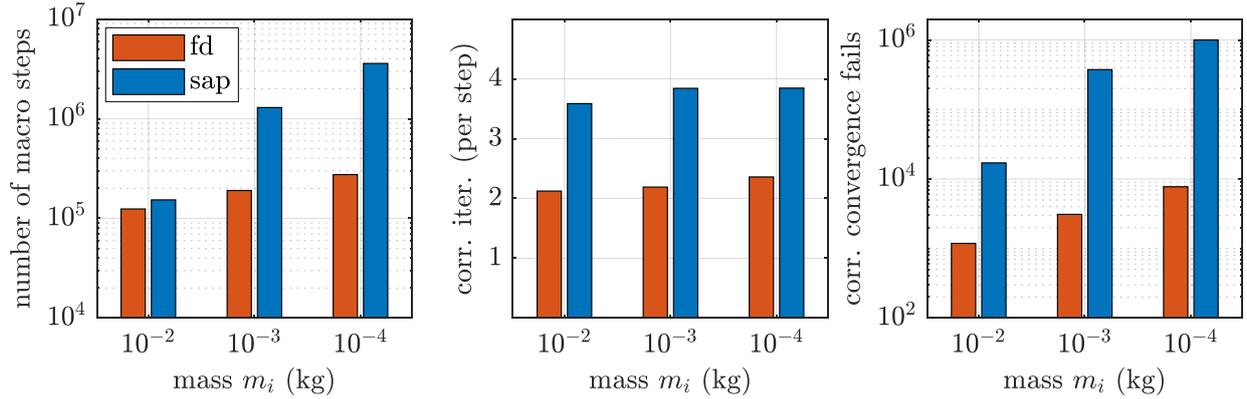
$n_K$	15 000
$n_s$	15
$t_{sim}$	$1.0e-1$ s
$c_i$	$1.0e6$ N/m
$d_i$	$1.0e4$ Ns/m
$C_i$	$1.0e11$ N/m <sup>3</sup>
$D_i$	$1.0e-2$ Ns <sup>3</sup> /m <sup>3</sup>

The model is computed with the fully controlled implicit (*imMD*) co-simulation approach with the relative error tolerance  $rtol = 10^{-4}$  and the absolute error tolerances  $atol^{pos} = rtol$  and  $atol^{vel} = 10^3 \cdot rtol$ . The constant  $\tau$  of the convergence criterion (see Eq. (2.31)) of the corrector iteration is set to  $\tau = 0.01$  in order to obtain a stable simulation with the simple approximation (*sap*) of the interface Jacobian. The resulting computation time and global errors of the coupling bodies depending on the mass  $m_i$  of each body are shown in Fig. A.25. For decreasing masses, the finite differences approximation (*fd*) performs clearly better than the *sap* approach. The statistics with regard to the number of macro-steps, the average number of corrector iterations per macro-step and the number of macro-steps which have to be repeated because the corrector does not converge within  $n_{cor}^{max} = 5$  iterations are given in Fig. A.26. The average number of corrector iterations per step is only slightly influenced by the mass  $m_i$  of each body: the *fd* approximation requires about two iterations and the *sap* approach requires about four iterations.

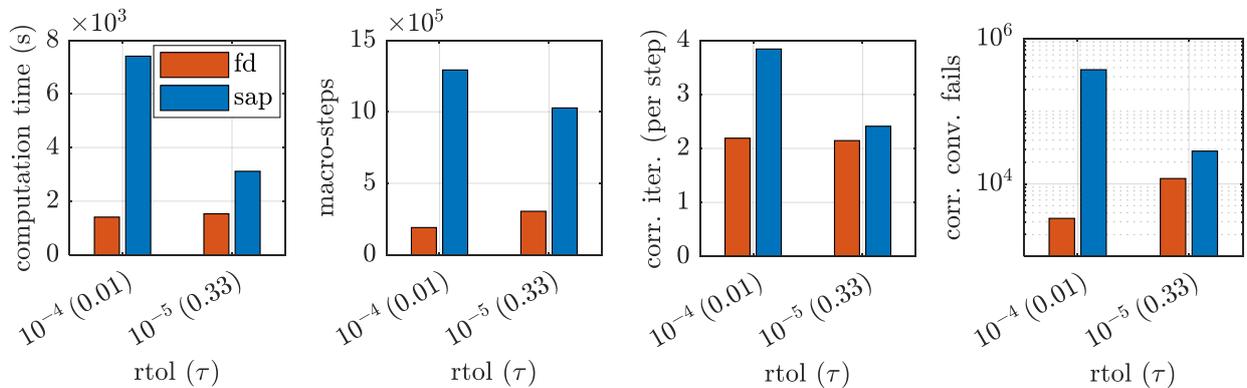


**Figure A.25:** Nonlinear oscillator chain (*imMD* approach): computation time and global error of the states of the coupling bodies (NRMSE) depending on the mass  $m_i$  of the bodies for using the approximation of the interface Jacobian by the first series term (*sap*) and by finite differences (*fd*).

The overall performance of the co-simulation using the *sap* approach can be improved significantly by defining tighter error tolerances for the macro-step size controller, as shown in Fig. A.27 exemplarily for  $m_i = 10^{-3}$  kg. A reduction of the relative error tolerance of the macro-step size controller from  $10^{-4}$  to  $10^{-5}$  allows the adjustment of the nonlinear convergence coefficient to  $\tau = 0.33$ , which has been used as a standard value for the simulations within the scope of this work. All relevant numbers and therefore also the overall computation time are decreased for the co-simulation with  $\text{rtol} = 10^{-5}$ . Considering the *fd* approach, tighter error tolerances cause an (expected) moderate increase in the number of macro-steps.



**Figure A.26:** Nonlinear oscillator chain (*imMD* approach): total number of macro-steps, average number of corrector iterations per macro-step and total number of corrector convergence fails depending on the mass  $m_i$  of the bodies for using the approximation of the interface Jacobian by the first series term (*sap*) and by finite differences (*fd*).

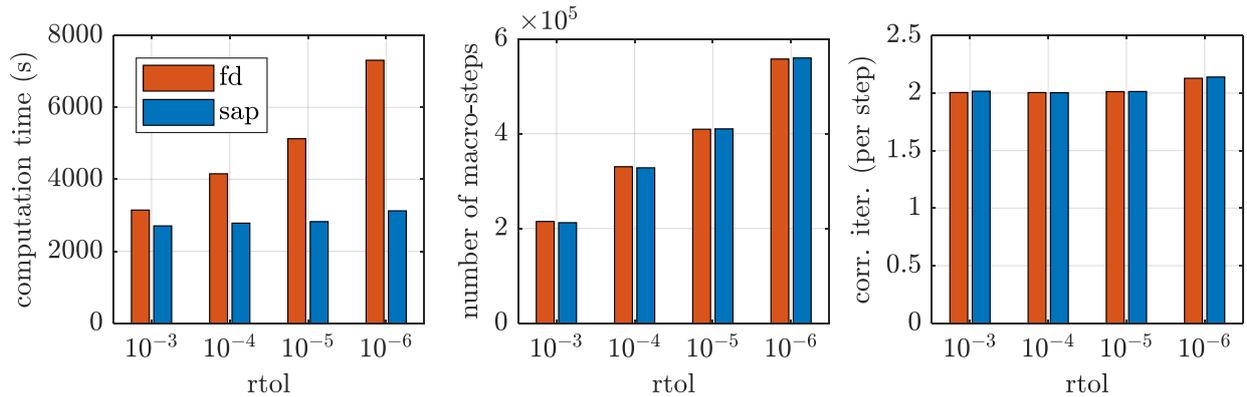


**Figure A.27:** Nonlinear oscillator chain with  $m_i = 10^{-3}$  kg: influence of the error tolerances and the coefficient  $\tau$  of the convergence criterion on the performance of the computation.

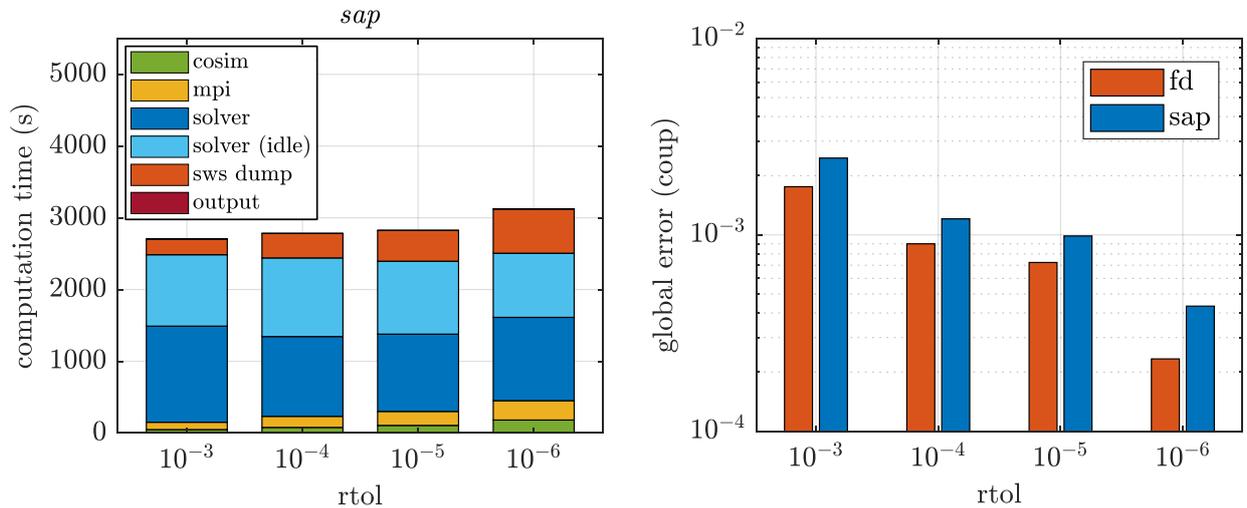
#### Example 4: Nonlinear Oscillator Chain with Impulse Shaped Forces

The nonlinear oscillator chain with impulse shaped forces acting on the coupling bodies, as described in Section 5.8, is computed with the fully controlled implicit co-simulation approach (*imMD*). The performance of the co-simulation approach using the first series term (*sap*) (see Eq. (A.5)) as an approximation of the interface Jacobian is compared to the performance of the same approach but computing the interface Jacobian by finite differences (*fd*). The results are shown in Figs. A.28 and A.29 for different error tolerances of the macro-step size controller. It

can be seen that the number of macro steps and the number of corrector iterations is almost equal for both approaches. The computation time of the co-simulation using the *sap* approach is even better, especially for tighter error tolerances. The decline of the computational efficiency of the finite differences approximation can be explained by the required subsystem integrations with perturbed approximation polynomials. It is not straightforward to define adequate perturbations of the coupling variables; not optimally selected perturbations lead to an increased subsystem integration time. The resulting global errors of the co-simulation with the *sap* approach are slightly larger, as can be seen in Fig. A.29. This could be compensated by tuning the  $\tau$  parameter of the convergence criterion (2.31).



**Figure A.28:** Nonlinear oscillator chain with impulse shaped forces (*imMD* approach): computation time, total number of macro-steps, average number of corrector iterations per macro-step for using the approximation of the interface Jacobian by the first series term (*sap*) and by finite differences (*fd*).



**Figure A.29:** Nonlinear oscillator chain with impulse shaped forces (*imMD* approach): computation time (*sap*) and global error of the states of the coupling bodies.

#### A.4. Conclusions on the Approximation of the Interface Jacobian

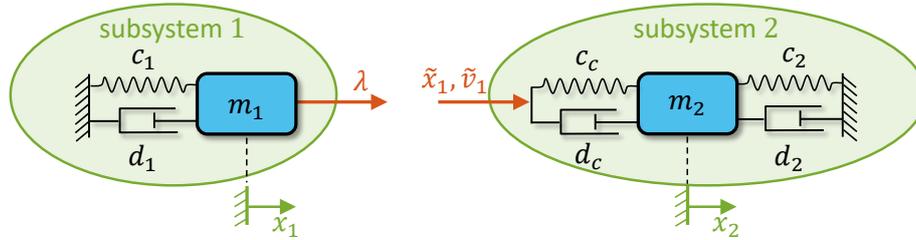
The first term of the series expansion of the solution of the interface Jacobian ODE-system, as shown exemplarily for a 1-dof-subsystem in Eq. (A.5), is used to approximate the interface Jacobian (*sap* approach). For the computation, only the mass matrix of the subsystems and the

macro-step size are required. The generalization of this method to DAE-subsystems is straightforward; in the DAE case, also the constraint Jacobian of the subsystem has to be known to apply the *sap* approach.

Numerical studies indicate that the simple approximation (*sap*) of the interface Jacobian in combination with the macro-step size controller seems to be sufficient for the implicit co-simulation of a wide range of mechanical systems. Especially for the simulation of low-damped systems with a large number of coupling variables, the *sap* approach is preferable, because it does not require subsystem integrations with perturbed approximation polynomials. The required number of cores for a full parallelization of the co-simulation model is reduced from  $n_{\text{cores}} = n_s + \sum_{L=1}^{n_s} ({}^L n_u) + 1$  when using the finite differences approximation (*fd*) to  $n_{\text{cores}} = n_s + 1$  if the *sap* approach is used ( $n_s$ : number of subsystems,  ${}^L n_u$  number of coupling variables in subsystem  $L$ ). However, for highly damped systems or models including strong nonlinearities, the finite differences approximation of the interface Jacobian is more reliable.

## B. Convergence Behavior of Explicit Co-Simulation Approaches with Feed-Through

For co-simulation models with feed-through (more precisely: direct feed-through), the subsystem output variables are direct functions of the subsystem input variables (coupling variables), see Section 2.1.2. The influence of feed-through on the convergence behavior of explicit co-simulations is subsequently discussed for Jacobi and Gauss-Seidel type schemes.



**Figure B.1:** Linear two-mass oscillator: force/displacement-decomposition approach.

For Jacobi type co-simulation schemes, the following cases have to be distinguished:

- Using a force/force-decomposition approach, the force law of the coupling element has to be known explicitly as a function of the states of the coupling bodies. After the subsystem integration, the coupling forces have to be updated at  $T_{N+1}$  in order to proceed with the next macro-step from  $T_{N+1}$  to  $T_{N+2}$ . As a consequence, the states of the coupling bodies and the coupling forces are consistent at  $T_{N+1}$ . Hence, there is no feed-through.
- Using a displacement/displacement-decomposition technique, the force law of the coupling has also to be known explicitly as a function of the states of the coupling bodies, since the coupling element has to be duplicated within this decomposition approach. After the subsystem integration, the coupling variables have to be updated at  $T_{N+1}$ . As a consequence, the states of the coupling bodies and the coupling forces are consistent at  $T_{N+1}$ . Therefore, a feed-through will not arise.
- Feed-through may only occur, if a force/displacement-decomposition approach (Fig. B.1) is used. If the coupling forces are updated at  $T_{N+1}$  by using the corresponding state variables at  $T_{N+1}$ , a feed-through will not arise. Considering the two-mass oscillator of Section 2.1.2, feed-through does not occur if the coupling force at  $T_{N+1}$  is calculated (updated) by  $\lambda_{N+1} = c_c (x_{2,N+1} - x_{1,N+1}) + d_c (v_{2,N+1} - v_{1,N+1})$ . However, if the coupling force  $\lambda_{N+1} = c_c (x_{2,N+1} - \tilde{x}_{1,N+1}^{pre}) + d_c (v_{2,N+1} - \tilde{v}_{1,N+1}^{pre})$  is used, a feed-through exists and the state variables and the coupling force (not updated) are not consistent at  $T_{N+1}$ . Considering, for instance, complex co-simulation systems with commercial software tools (black-box subsystems), where the coupling forces are not known explicitly as functions of the states of the coupling bodies, a simple force update to generate consistent variables is not possible and a feed-through is generated. In this case, consistent variables would have to be computed by a static update simulation (iteratively in case of a nonlinear model) at  $T_{N+1}$ .

For Gauss-Seidel type co-simulation approaches, the subsequent behavior is observed:

- As in case of the Jacobi type co-simulation schemes, feed-through is only detected in connection with force/displacement-decomposition. It depends on the order of the subsystem integration, whether a feed-through will entail problems (order reduction of the co-simulation) or not. If the force-driven subsystem is integrated firstly, the feed-through does not cause problems, i.e. the convergence order will not be reduced. The reason therefore is that the second subsystem is integrated with interpolated coupling variables ( $\tilde{x}_1(t) = P_\kappa(t; [T_{N+1-\kappa}, x_{1,N+1-\kappa}], \dots, [T_{N+1}, x_{1,N+1}])$  and  $\tilde{v}_1(t) = P_\kappa(t; [T_{N+1-\kappa}, v_{1,N+1-\kappa}], \dots, [T_{N+1}, v_{1,N+1}])$  in case of the two-mass oscillator example) so that consistent coupling forces are obtained in both subsystems at  $T_{N+1}$ . If, on the other hand, the base-point excited subsystem is integrated firstly, the feed-through will cause problems, i.e. will entail a reduction of the convergence order. The reason therefore is that the second subsystem is integrated with extrapolated coupling variables ( $\tilde{x}_1(t) = P_\kappa(t; [T_{N-\kappa}, x_{1,N-\kappa}], \dots, [T_N, x_{1,N}])$  and  $\tilde{v}_1(t) = P_\kappa(t; [T_{N-\kappa}, v_{1,N-\kappa}], \dots, [T_N, v_{1,N}])$  in case of the two-mass oscillator example) so that inconsistent coupling forces are obtained in both subsystems at  $T_{N+1}$ .

Concerning the local error of the co-simulation, the following convergence behavior is observed (under the assumption that the subsystems are integrated analytically or with very small numerical errors so that the error of the co-simulation is dominated by the approximation of the coupling variables):

- The same local convergence behavior is detected for both Jacobi and Gauss-Seidel type co-simulation schemes: the coupling variables converge with  $\mathcal{O}(H^{\kappa+1})$ , the velocity variables with  $\mathcal{O}(H^{\kappa+2})$  and the position variables with  $\mathcal{O}(H^{\kappa+3})$ .

With respect to the global error of the co-simulation (under the assumption that the subsystems are integrated analytically or with very small numerical errors so that the error of the co-simulation is dominated by the approximation of the coupling variables), all variables (coupling variables, velocity variables and position variables) show the same convergence rate, since the variables with the lowest convergence order determine the convergence order of all variables. Concretely, the subsequent convergence behavior is observed:

- a) If updated coupling variables are calculated at  $T_{N+1}$ , all variables will globally converge with  $\mathcal{O}(H^{\kappa+1})$ .
- b) Using a Jacobi scheme (force/displacement-coupling), without an update at  $T_{N+1}$ , all variables will converge with  $\mathcal{O}(H^\kappa)$ , since the coupling variables are inconsistent.
- c) Using a Gauss-Seidel scheme, where the force-driven subsystem is integrated firstly (force/displacement-coupling), without an update at  $T_{N+1}$ , all variables will converge with  $\mathcal{O}(H^{\kappa+1})$ . This case resembles case a), since the coupling variables are consistent.
- d) Using a Gauss-Seidel scheme, where the base-point excited subsystem is integrated firstly (displacement/force-coupling), without an update at  $T_{N+1}$ , all variables will converge with  $\mathcal{O}(H^\kappa)$ , since the coupling variables are inconsistent.

Summarizing: If inconsistent coupling variables are used, local errors are accumulated and may entail an order reduction of the global error. Following the description in [MKS21], the error of the

approximation polynomials can be expressed as

$$\mathbf{p}_{N+1}^{pre}(T_{N+1}) - \mathbf{u}(T_{N+1}) = CH^{\kappa+1}\mathbf{u}^{(\kappa+1)}(T_{N+1}) + \sum_{j=0}^{\kappa} c_j \Delta \mathbf{u}_{N-j} + \mathcal{O}(H^{\kappa+2}) \quad (\text{B.1})$$

with the error of the sampling points  $\Delta \mathbf{u}_{N-j} := \mathbf{u}_{N-j} - \mathbf{u}(T_{N-j})$  and the constants  $C$  and  $c_j$ . The first error term  $CH^{\kappa+1}\mathbf{u}^{(\kappa+1)}(T_{N+1})$  is always apparent, but will not be propagated. The second error term  $\sum_{j=0}^{\kappa} c_j \Delta \mathbf{u}_{N-j}$ , which describes the error in the supporting points, converges with  $\mathcal{O}(H^{\kappa+2})$  for the case that consistent coupling variables are used, but only with  $\mathcal{O}(H^{\kappa+1})$  if inconsistent coupling variables are applied. Hence, the second error term may entail problems, if the update is not carried out.

The above considerations are valid for co-simulation models, where the subsystems are coupled by applied-forces (i.e. by constitutive laws). Considering co-simulation systems, where the subsystems are connected by algebraic constraint equations (constraint coupling), further problems may occur, e.g. zero-stability of the co-simulation may be lost, see [Arn10, GA04, KS00, SL14b, SLL16].

## C. Numerical Stability of Explicit Co-Simulation Methods

The numerical stability of different co-simulation methods has been studied for example in [Li17, SLL15, BS11c, BS10a, BS10b]. Within this section, the theoretical background is kept to a minimum; a detailed explanation of the stability analysis of co-simulation methods can be found in [SLL15], for example.

For the studies presented within the following subsections, three different explicit co-simulation methods are compared with regard to their numerical stability. The approaches can be distinguished by the way the approximation polynomials of the coupling variables are generated:

- extrapolation of the coupling variables (*classic*) [see Section 2.4.1]
- integration of the extrapolated accelerations of the coupling bodies (*acc*) [LLSS17]
- integration of the extrapolated accelerations in combination with relaxation techniques (*accRel*) [LYL<sup>+</sup>20].

Making use of the *classic* approach, the coupling variables, which can be forces or position and velocity variables, are approximated by extrapolation polynomials defined by the values of the coupling variables at the preceding macro-time points. A detailed explanation of this method can be found in Section 2.4.1.

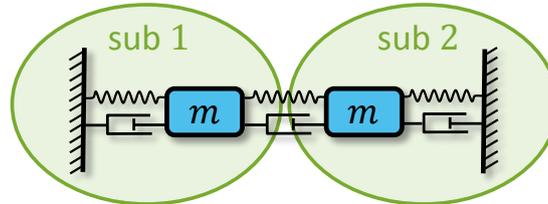
Applying the *acc* approach, the accelerations of the coupling bodies are extrapolated instead of the coupling variables. The extrapolation polynomials are then integrated over the current macro-step to obtain approximation polynomials for the velocity variables and, after a second integration, for the position variables of the coupling bodies. These polynomials are directly used as approximation polynomials for the coupling variables in the case of a displacement-coupling approach, or are substituted into the coupling conditions to obtain the coupling forces in case of a force-coupling approach. The *acc*-approach is explained in detail in [LLSS17].

The third investigated method, namely the *accRel* approach, is a combination of the *classic* and the *acc* approach. The integrated acceleration method (*acc*) is used to obtain the values of the coupling variables at the next macro-point. The approximation polynomials are generated in a second step with the help of relaxation techniques by using the values of the coupling variables at the preceding macro-time points and the obtained values at the next macro-time point. A detailed explanation of this method can be found in [LYL<sup>+</sup>20].

All three considered methods have in common that an update process has to be accomplished within each macro-step in order to develop their full potential. Within the *classic* approach, the coupling variables are updated directly by substituting the state variables of the coupling bodies into the coupling condition (Eq. 2.26). The implementation of this process is straightforward since all required variables are available to the co-simulation interface. The two approaches based on integrated accelerations, *acc* and *accRel*, require an update of the acceleration variables of the coupling bodies. The updated values can only be computed if the equations of motion of the subsystems are available to the co-simulation interface, which is in general not the case. However, the methods can be applied without an update of the acceleration variables, but the numerical stability will be significantly decreased. The convergence order, in contrast to the stability, is not influenced by the update.

### C.1. Linear Test Model

The test model for the stability analysis is a linear two-mass oscillator as shown in Fig. C.1. To investigate the numerical stability, a system of recurrence equations is derived for the considered co-simulation method and the spectral radius of this system is computed. The procedure of analyzing the numerical stability of a co-simulation method based on the linear two-mass oscillator is explained in detail for example in [SLL15] and will not be repeated here.



**Figure C.1:** Linear symmetric test model for stability analysis.

The test model for the following studies is symmetric, i. e. the two masses  $m$  are equal and the three linear spring/damper-elements have the same stiffness and damping parameters  $c$  and  $d$ .

### C.2. Result Overview

The stability analyses of the three considered explicit co-simulation methods are carried out for the different options listed in Table C.1. The stability plots are presented in Section C.3.

**Table C.1:** Considered options for the stability analysis.

Label	Description
$FF$	force/force-decomposition
$FD$	force/displacement-decomposition
$JAC$	Jacobi type co-simulation
$GS$	Gauss-Seidel type co-simulation
$u / Up$	update process is accomplished
$n / noUp$	update process is not accomplished
$\kappa_a$	polynomial degree of acceleration polynomials
$\kappa_\lambda$	polynomial degree of coupling variables polynomials
$\omega$	relaxation parameter

First of all, the convergence order of the considered co-simulation methods is determined for different approximation polynomials of the coupling variables. The results of the convergence analysis are shown in Fig. C.2. It should be mentioned that the convergence order of the *accRel* approach can be increased by selecting an optimal value for the relaxation parameter  $\omega$ :  $\omega = 1/2$  for the case  $\kappa_\lambda = 0$  and  $\omega = 1/6$  for the case  $\kappa_\lambda = 1$ . This statement holds only if the extrapolation order of the acceleration is sufficient ( $\kappa_a \geq \kappa_\lambda$ ).

The convergence analysis is carried out in terms of the local errors on position and on velocity level and also for the global error. The highest observed convergence rate on position level is achieved by the *classic* approach with quadratic approximation polynomials ( $\kappa_\lambda = 2$ ). The local

convergence rates of this method are  $\mathcal{O}(H^5)$  on position level and  $\mathcal{O}(H^4)$  on velocity level. The global error converges with order  $\mathcal{O}(H^3)$ . The only other investigated approach with a global convergence order of  $\mathcal{O}(H^3)$  is the *accRel* approach with linear approximation polynomials ( $\kappa_a = 1$  and  $\kappa_\lambda = 1$ ) and the relaxation parameter  $\omega = 1/6$ . The local error of this method converges with order  $\mathcal{O}(H^4)$  on position and on velocity level.

Convergence Order						
method	degree of approximation polynomials		convergence			
			loc $x$	loc $v$	glob	
classic	$\kappa_\lambda = 0$		3	2	1	
	$\kappa_\lambda = 1$		4	3	2	
	$\kappa_\lambda = 2$		5	4	3	
integrated acceleration	$\kappa_a = 0$		3	2	1	
	$\kappa_a = 1$		4	3	2	
integrated acceleration + relaxation	$\kappa_a = 0$	$\kappa_\lambda = 0$	$\omega = 1/6$	3	2	1
			$\omega = 1/2$	3	3	2
		$\kappa_\lambda = 1$	$\omega = 1/6$	4	3	2
			$\omega = 1/2$	4	3	2
	$\kappa_a = 1$	$\kappa_\lambda = 0$	$\omega = 1/6$	3	2	1
			$\omega = 1/2$	3	3	2
		$\kappa_\lambda = 1$	$\omega = 1/6$	4	4	3
			$\omega = 1/2$	4	3	2

**Figure C.2:** Results of the convergence analysis of different explicit co-simulation approaches.

The results of the stability analysis of the different approaches are summarized in Figs. C.3-C.6. It should be mentioned that a stability analysis of the *classic* approach has been carried out in [SLL15] and the numerical stability of the approaches *acc* and *accRel* has been examined in [LLSS17] and [LYL<sup>+</sup>20]. The novel aspect in the here presented work is the stability analysis of the explicit co-simulation approaches based on extrapolated accelerations for the case, that the update of the accelerations is not accomplished. As can be seen in Figs. C.3-C.6, the update of the accelerations has a significant effect on the numerical stability of the respective co-simulation approaches. The stability region of the methods based on extrapolated accelerations is significantly larger if the update is accomplished. This statement holds for all investigated combinations of approximation polynomials ( $\kappa_a, \kappa_\lambda$ ), decomposition techniques (force/force, force/displacement) and subsystem integration orders (parallel, sequential). However, the ability to simulate an undamped system ( $d = 0$ ) seems not to be influenced by the update process.

Force/Force – Jacobi Type							
method	degree of approximation polynomials			stability			
				$d = 0$		$d > 0$	
classic	$\kappa_\lambda = 0$			-	++++		
	$\kappa_\lambda = 1$			-	++		
	$\kappa_\lambda = 2$			+	+		
				u	n	u	n
integrated acceleration	$\kappa_a = 0$			+	+	++++	+
	$\kappa_a = 1$			+	+	++++	+
integrated acceleration + relaxation	$\kappa_a = 0$	$\kappa_\lambda = 0$	$\omega = 1/6$	+	+	++	+
			$\omega = 1/2$	+	+	++++	++
		$\kappa_\lambda = 1$	$\omega = 1/6$	+	+	++++	++
			$\omega = 1/2$	-	-	+++++	++
	$\kappa_a = 1$	$\kappa_\lambda = 0$	$\omega = 1/6$	+	+	++	+
			$\omega = 1/2$	+	+	++++	+
		$\kappa_\lambda = 1$	$\omega = 1/6$	-	-	++++	+
			$\omega = 1/2$	-	-	+++++	+

Figure C.3: Results of the stability analysis: force/force, Jacobi type.

Force/Force – Gauss-Seidel Type							
method	degree of approximation polynomials			stability			
				$d = 0$		$d > 0$	
classic	$\kappa_\lambda = 0$			-	+++++		
	$\kappa_\lambda = 1$			-	++++		
	$\kappa_\lambda = 2$			+	++		
				u	n	u	n
integrated acceleration	$\kappa_a = 0$			+	+	+++++	+++
	$\kappa_a = 1$			+	+	+++++	+
integrated acceleration + relaxation	$\kappa_a = 0$	$\kappa_\lambda = 0$	$\omega = 1/6$	+	+	+++++	+++
			$\omega = 1/2$	+	+	+++++	+++++
		$\kappa_\lambda = 1$	$\omega = 1/6$	-	-	+++++	+++
			$\omega = 1/2$	-	-	+++++	++
	$\kappa_a = 1$	$\kappa_\lambda = 0$	$\omega = 1/6$	+	+	++++	++
			$\omega = 1/2$	+	+	+++++	+++
		$\kappa_\lambda = 1$	$\omega = 1/6$	-	-	+++++	++
			$\omega = 1/2$	-	-	+++++	++

Figure C.4: Results of the stability analysis: force/force, Gauss-Seidel type.

Force/Displacement – Jacobi Type							
method	degree of approximation polynomials			stability			
				$d = 0$		$d > 0$	
classic	$\kappa_\lambda = 0$			-		+++++	
	$\kappa_\lambda = 1$			-		++	
	$\kappa_\lambda = 2$			-		+	
				u	n	u	n
integrated acceleration	$\kappa_a = 0$			-	-	+++++	+
	$\kappa_a = 1$			-	-	+++++	+
integrated acceleration + relaxation	$\kappa_a = 0$	$\kappa_\lambda = 0$	$\omega = 1/6$	-	-	++	+
			$\omega = 1/2$	-	+	++++	++
		$\kappa_\lambda = 1$	$\omega = 1/6$	-	-	+++++	+
			$\omega = 1/2$	-	-	+++++	++
	$\kappa_a = 1$	$\kappa_\lambda = 0$	$\omega = 1/6$	-	-	++	+
			$\omega = 1/2$	+	+	++++	+
		$\kappa_\lambda = 1$	$\omega = 1/6$	-	-	+++++	+
			$\omega = 1/2$	-	-	+++++	+

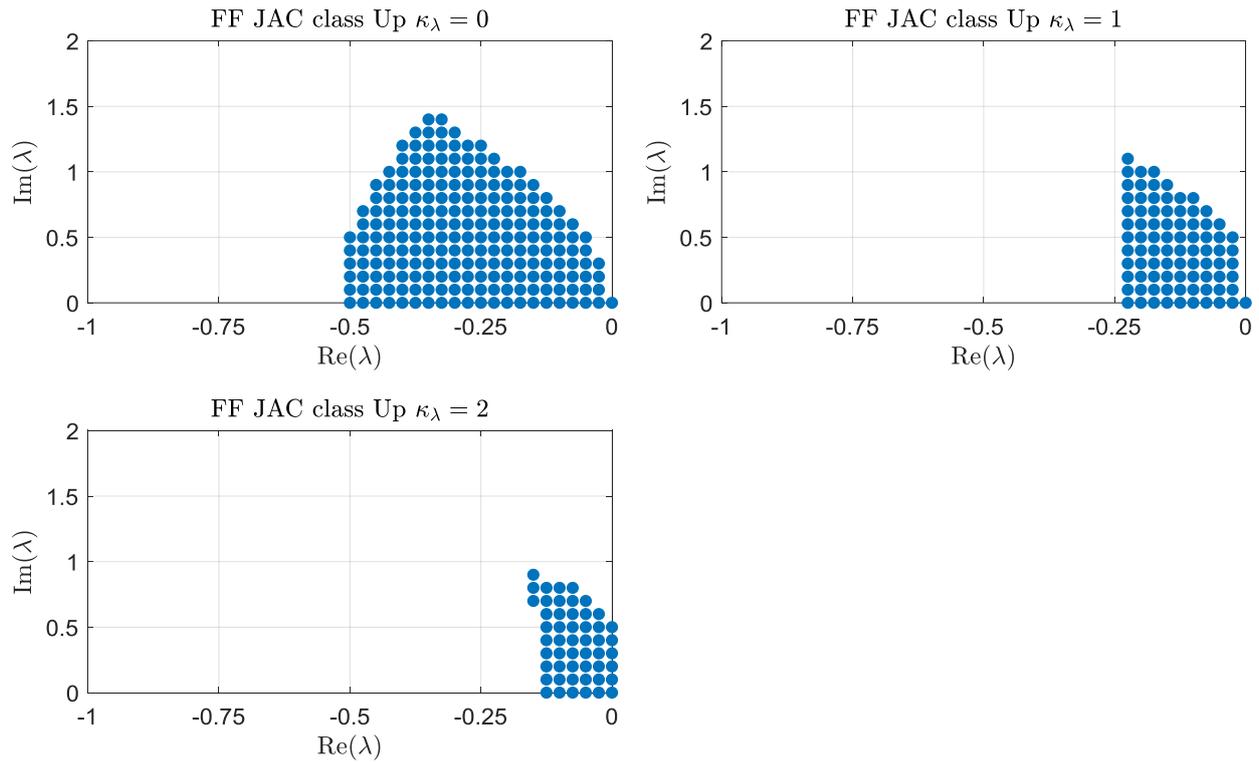
Figure C.5: Results of the stability analysis: force/displacement, Jacobi type.

Force/Displacement – Gauss-Seidel Type							
method	degree of approximation polynomials			stability			
				$d = 0$		$d > 0$	
classic	$\kappa_\lambda = 0$			-		+++++	
	$\kappa_\lambda = 1$			-		+++++	
	$\kappa_\lambda = 2$			+		++	
				u	n	u	n
integrated acceleration	$\kappa_a = 0$			+	+	+++++	+++
	$\kappa_a = 1$			-	-	+++++	++
integrated acceleration + relaxation	$\kappa_a = 0$	$\kappa_\lambda = 0$	$\omega = 1/6$	-	-	++++	+++
			$\omega = 1/2$	+	+	+++++	++++
		$\kappa_\lambda = 1$	$\omega = 1/6$	-	-	+++++	+++
			$\omega = 1/2$	-	-	+++++	+++
	$\kappa_a = 1$	$\kappa_\lambda = 0$	$\omega = 1/6$	-	-	++++	++
			$\omega = 1/2$	+	+	+++++	+++
		$\kappa_\lambda = 1$	$\omega = 1/6$	-	-	+++++	++
			$\omega = 1/2$	-	-	+++++	++

Figure C.6: Results of the stability analysis: force/displacement, Gauss-Seidel type.

### C.3. Stability Plots

The stability plots below show the spectral radius  $\rho$  as a function of the (dimensionless) real and imaginary part of eigenvalue  $\lambda$  of subsystem 1. The blue points mark the numerically stable regions ( $\rho \leq 1$ ). It should be mentioned that the  $\text{Re}(\lambda) = 0$  axis represents an undamped system ( $d = 0$ ). The physical interpretation of the plots is that the damping of the system is increased from right to left and the stiffness is increased from the bottom to the top. As already mentioned, a detailed description of how the the stability analysis is carried out, can be found in [SLL15], for example.



**Figure C.7:** Numerical stability: force/force, Jacobi type, *classic*.

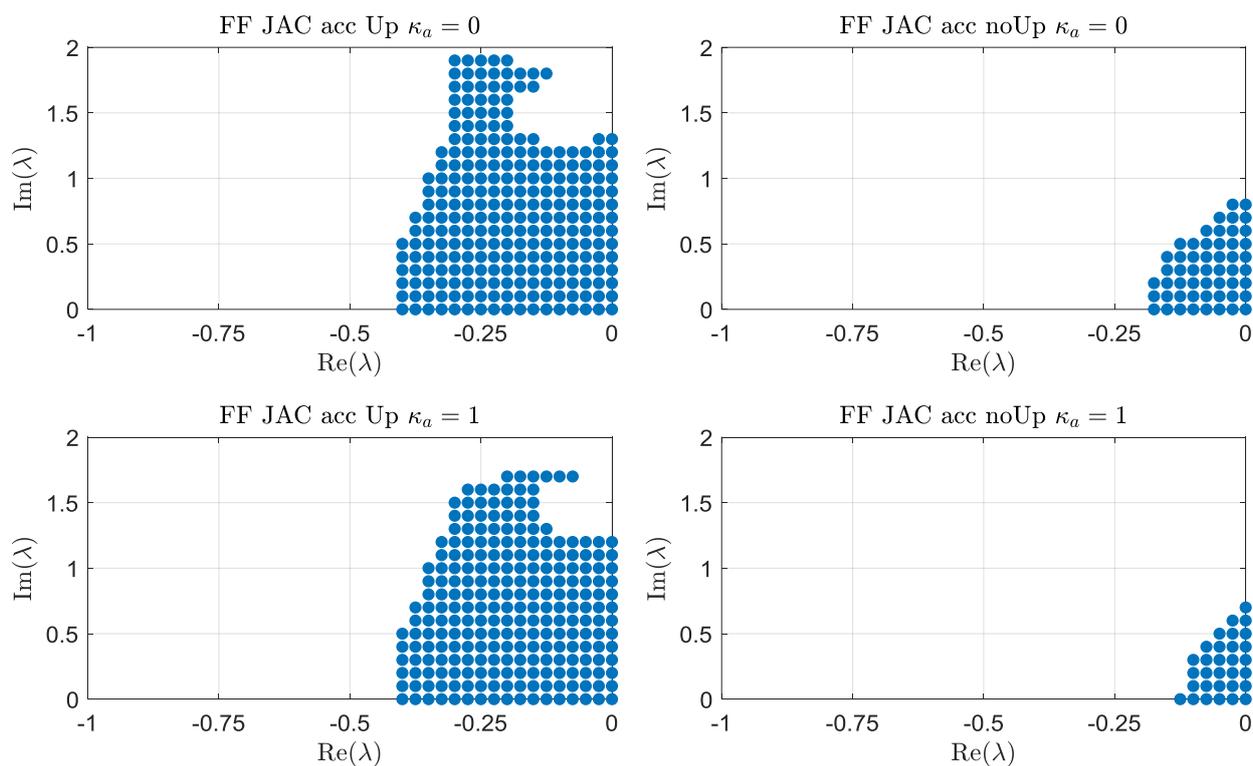


Figure C.8: Numerical stability: force/force, Jacobi type, *acc*.

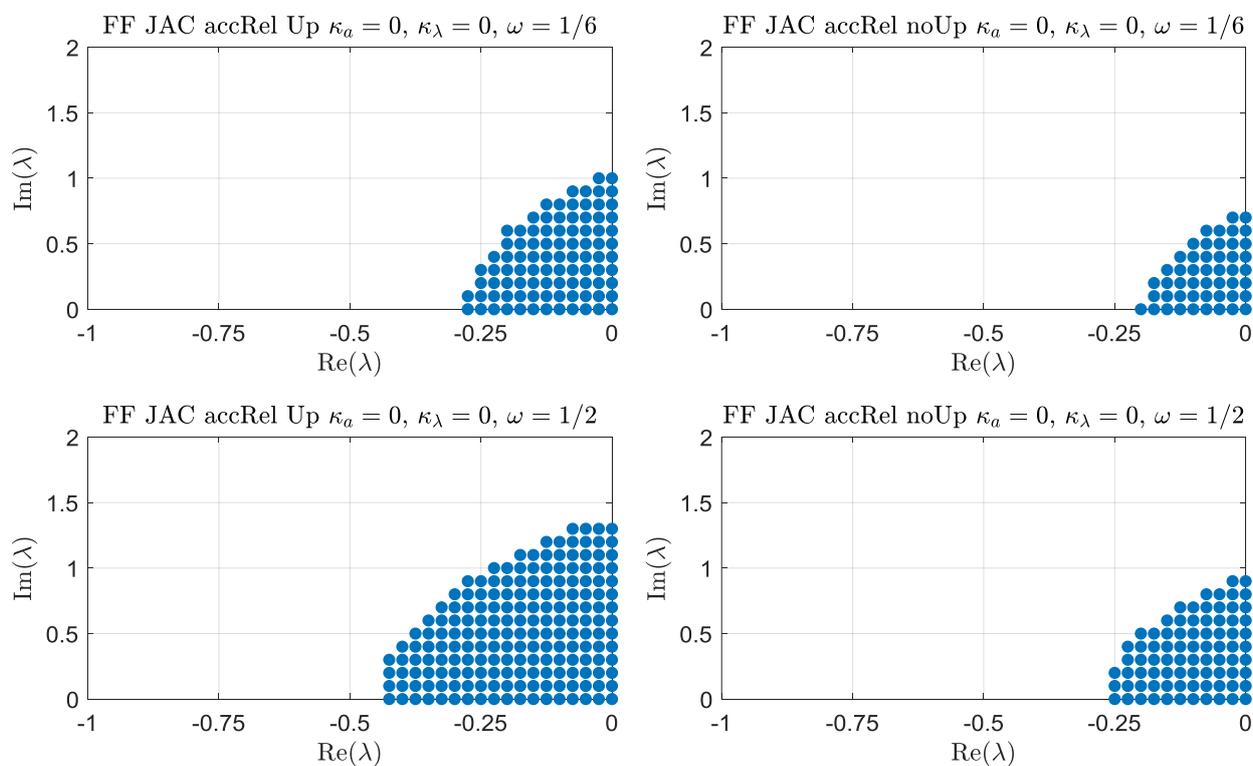


Figure C.9: Numerical stability: force/force, Jacobi type, *accRel*,  $\kappa_a = 0$ ,  $\kappa_\lambda = 0$ .

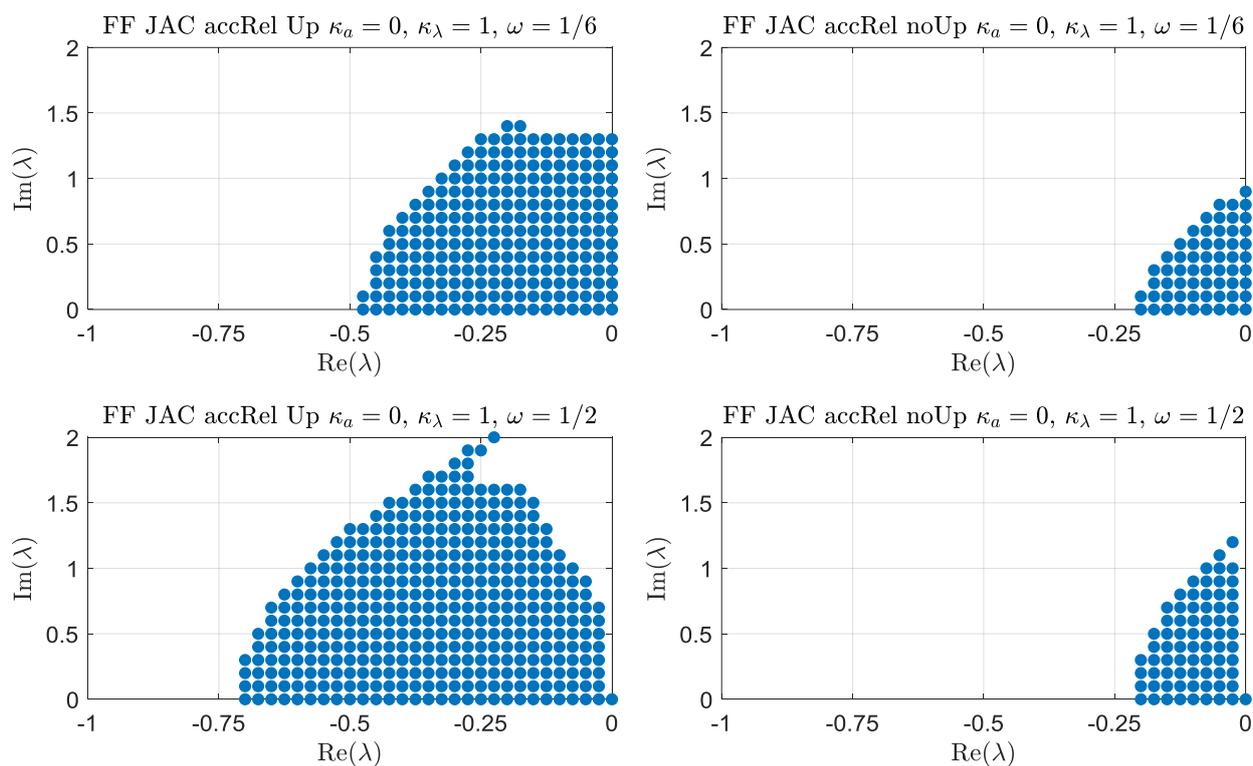


Figure C.10: Numerical stability: force/force, Jacobi type, *accRel*,  $\kappa_a = 0, \kappa_\lambda = 1$ .

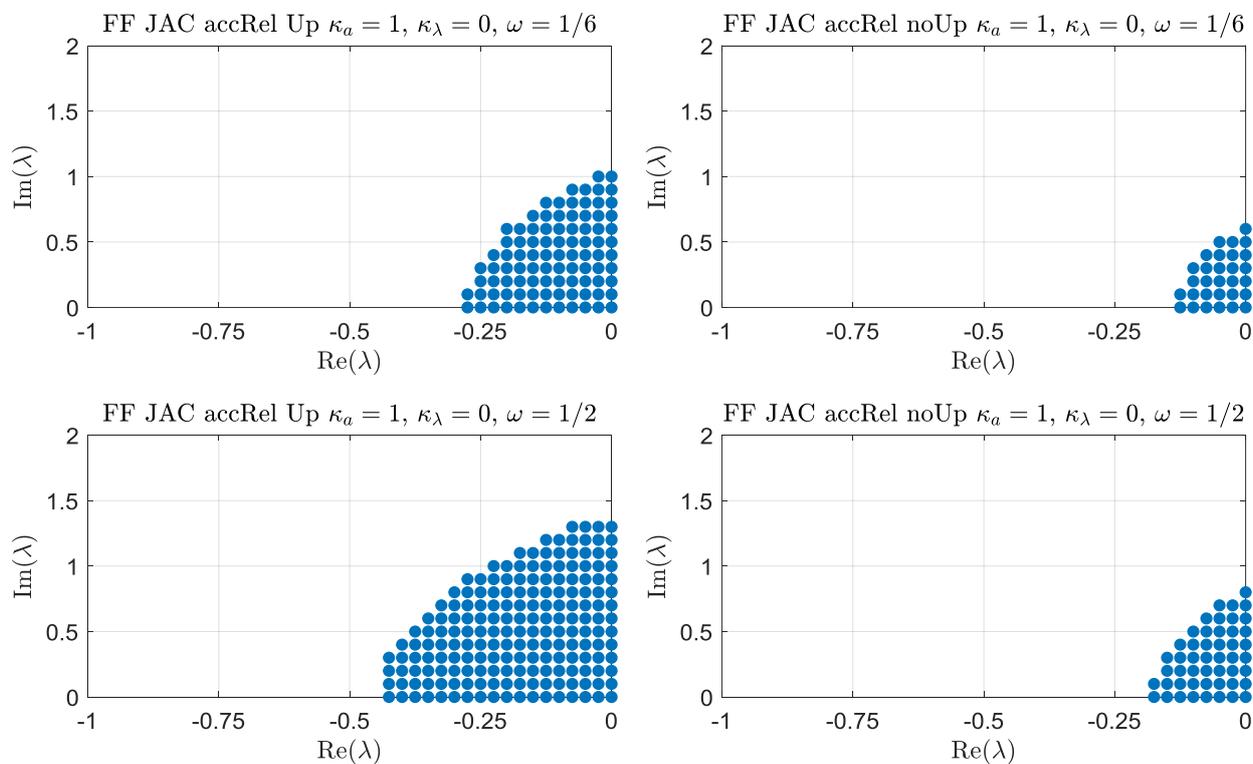


Figure C.11: Numerical stability: force/force, Jacobi type, *accRel*,  $\kappa_a = 1, \kappa_\lambda = 0$ .

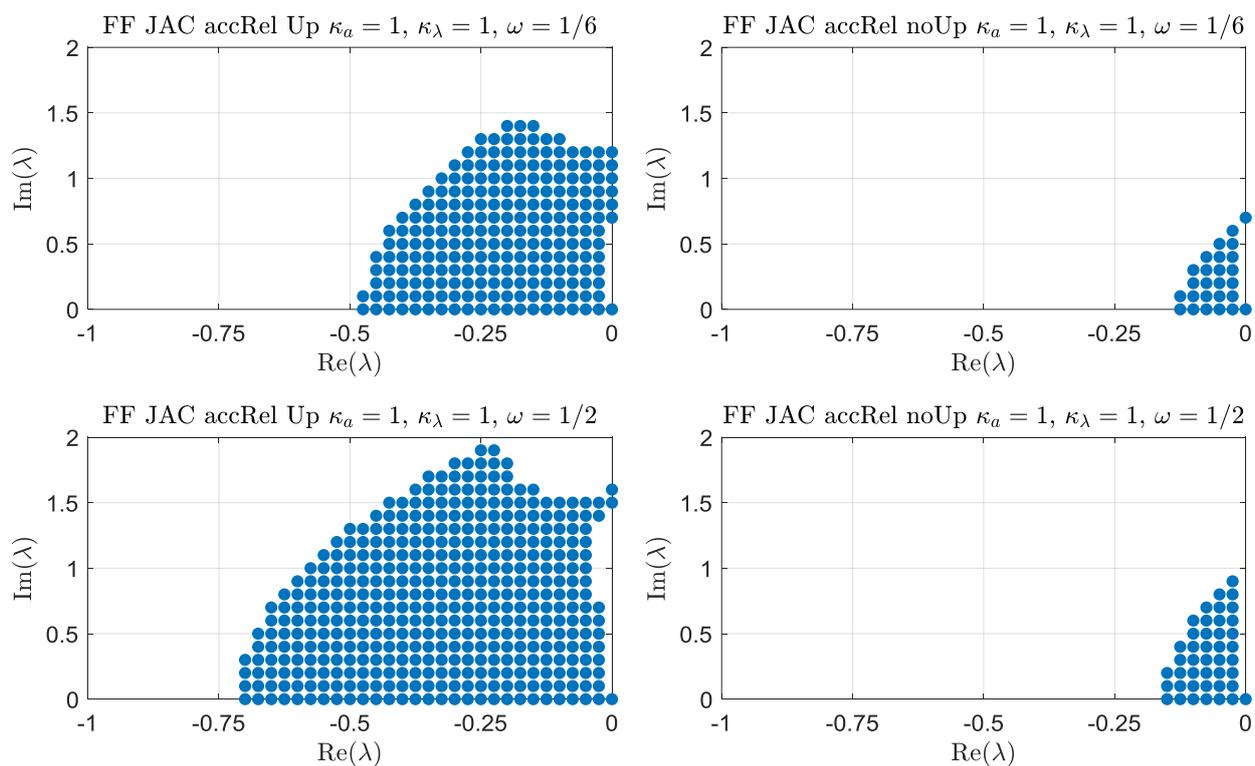


Figure C.12: Numerical stability: force/force, Jacobi type, *accRel*,  $\kappa_a = 1, \kappa_\lambda = 1$ .

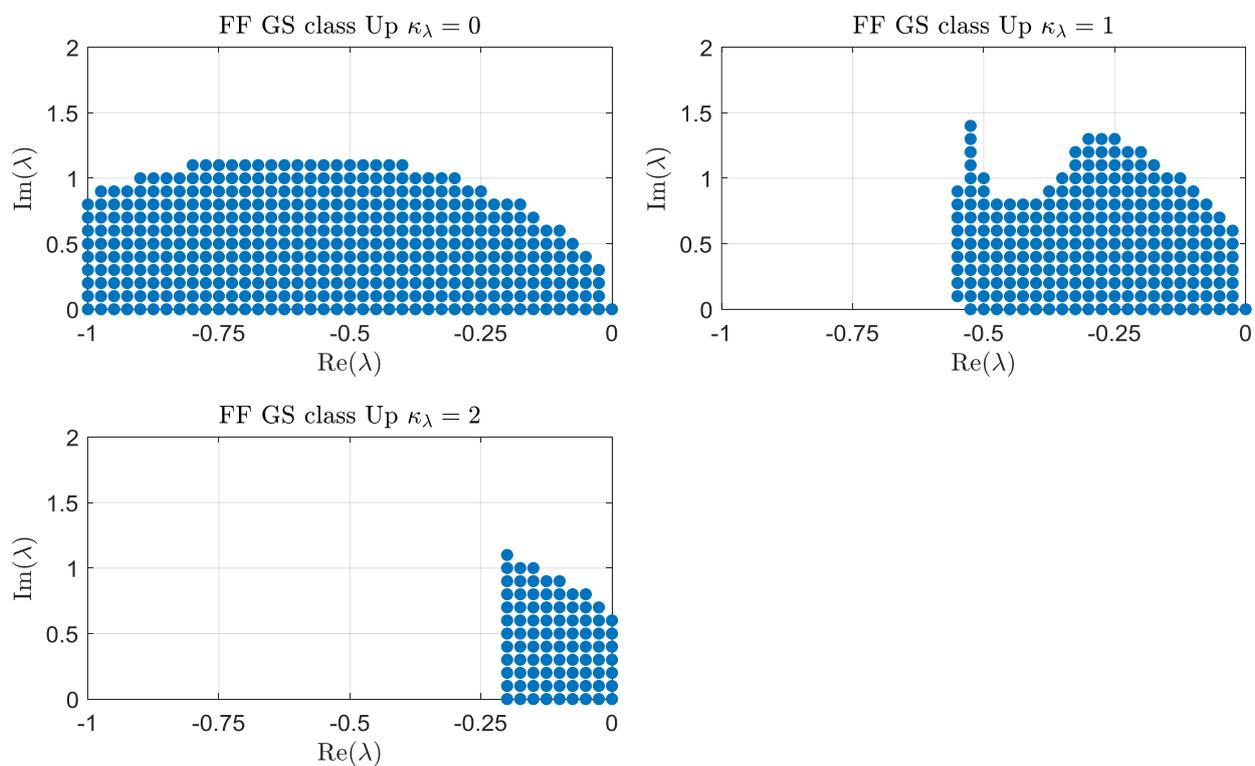


Figure C.13: Numerical stability: force/force, Gauss-Seidel type, *classic*.

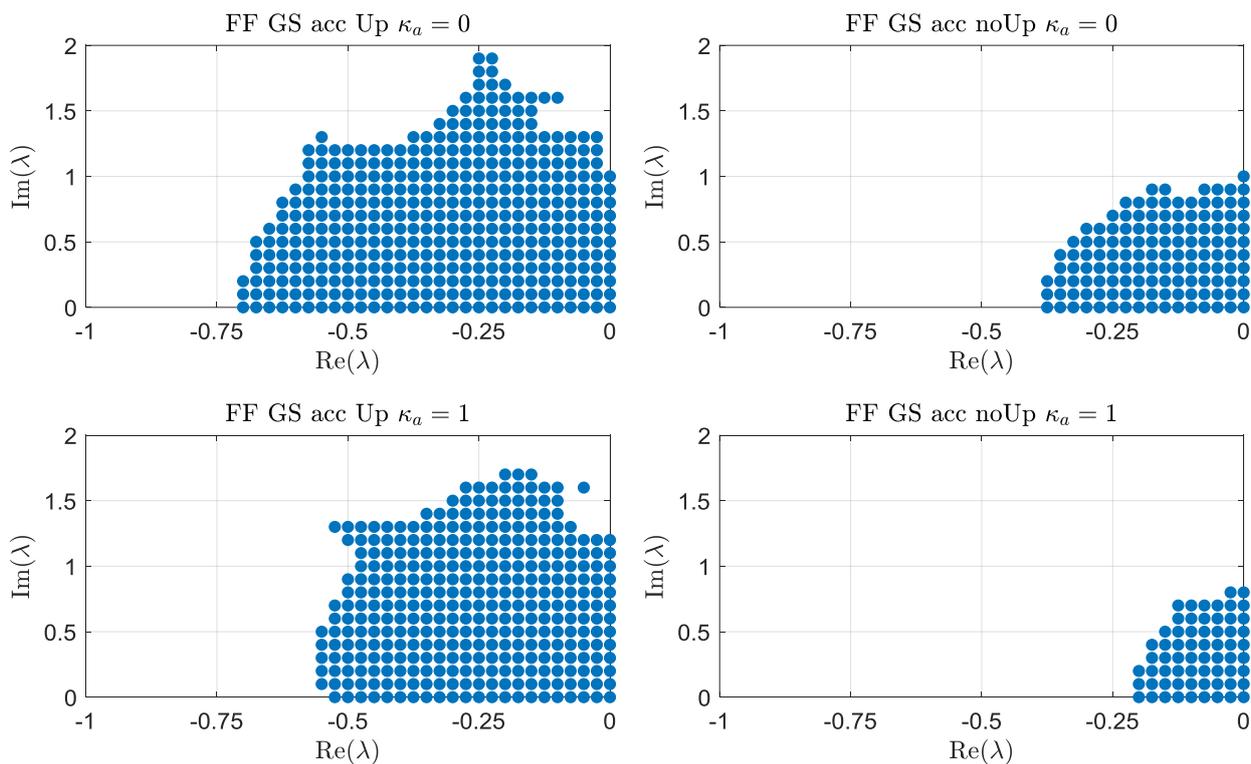


Figure C.14: Numerical stability: force/force, Gauss-Seidel type, *acc*.

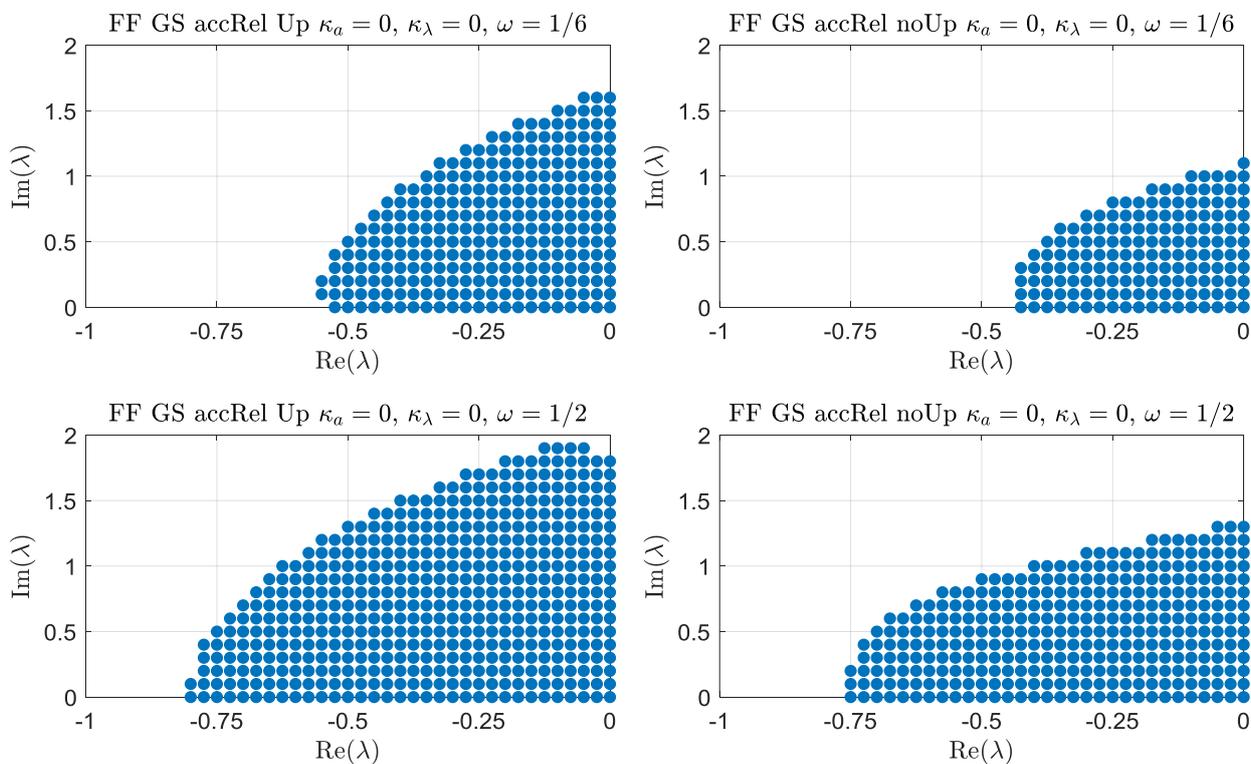


Figure C.15: Numerical stability: force/force, Gauss-Seidel type, *accRel*,  $\kappa_a = 0$ ,  $\kappa_\lambda = 0$ .

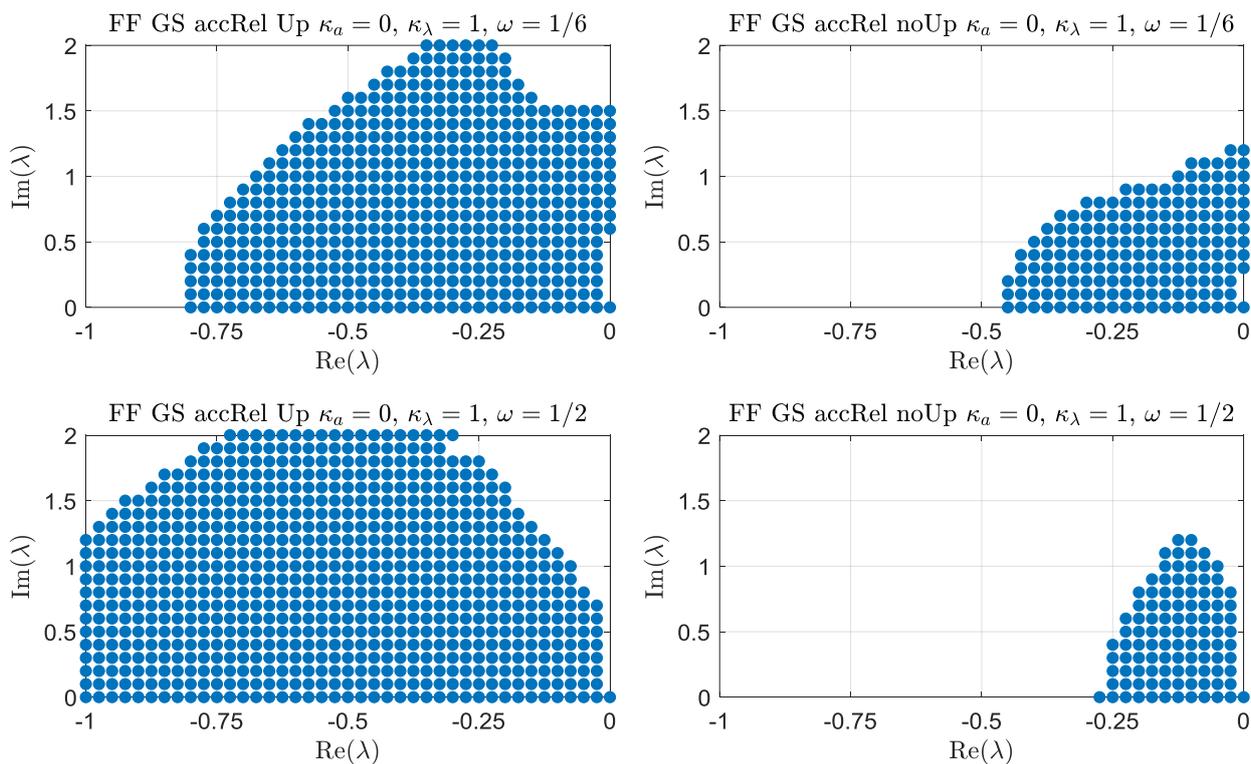


Figure C.16: Numerical stability: force/force, Gauss-Seidel type, *accRel*,  $\kappa_a = 0$ ,  $\kappa_\lambda = 1$ .

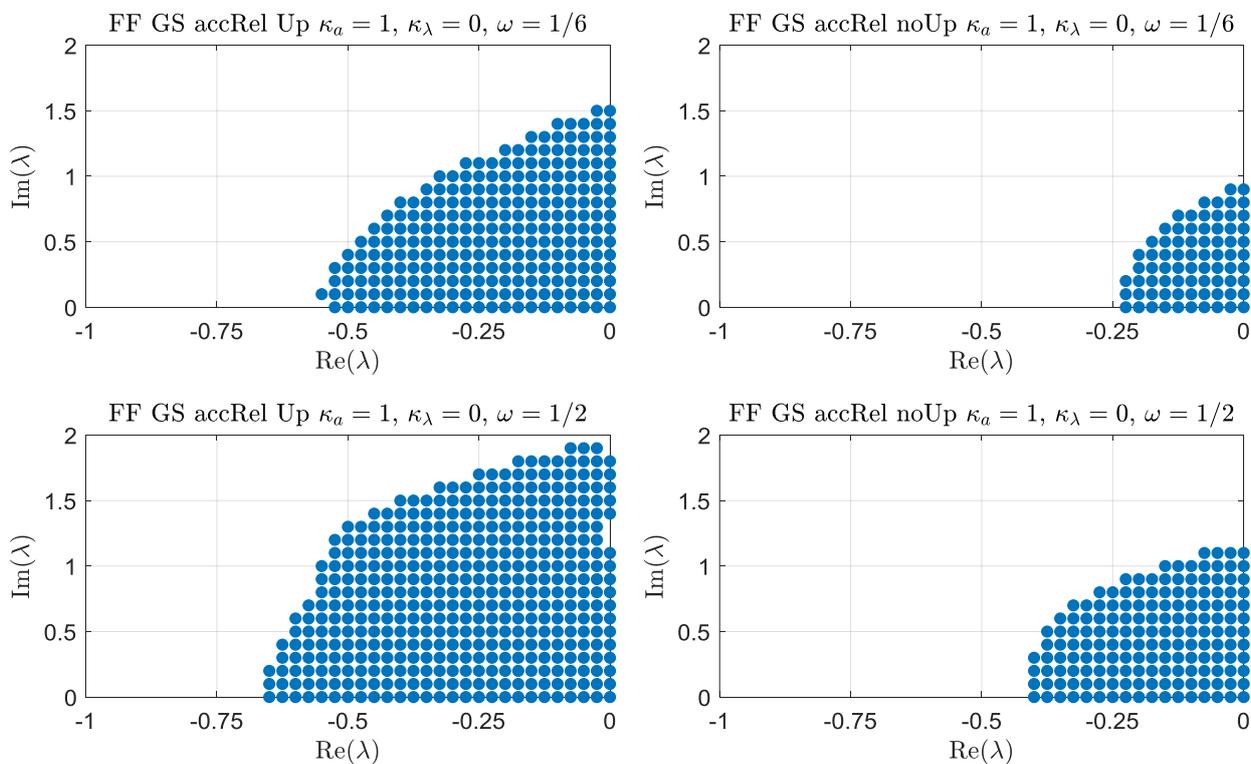
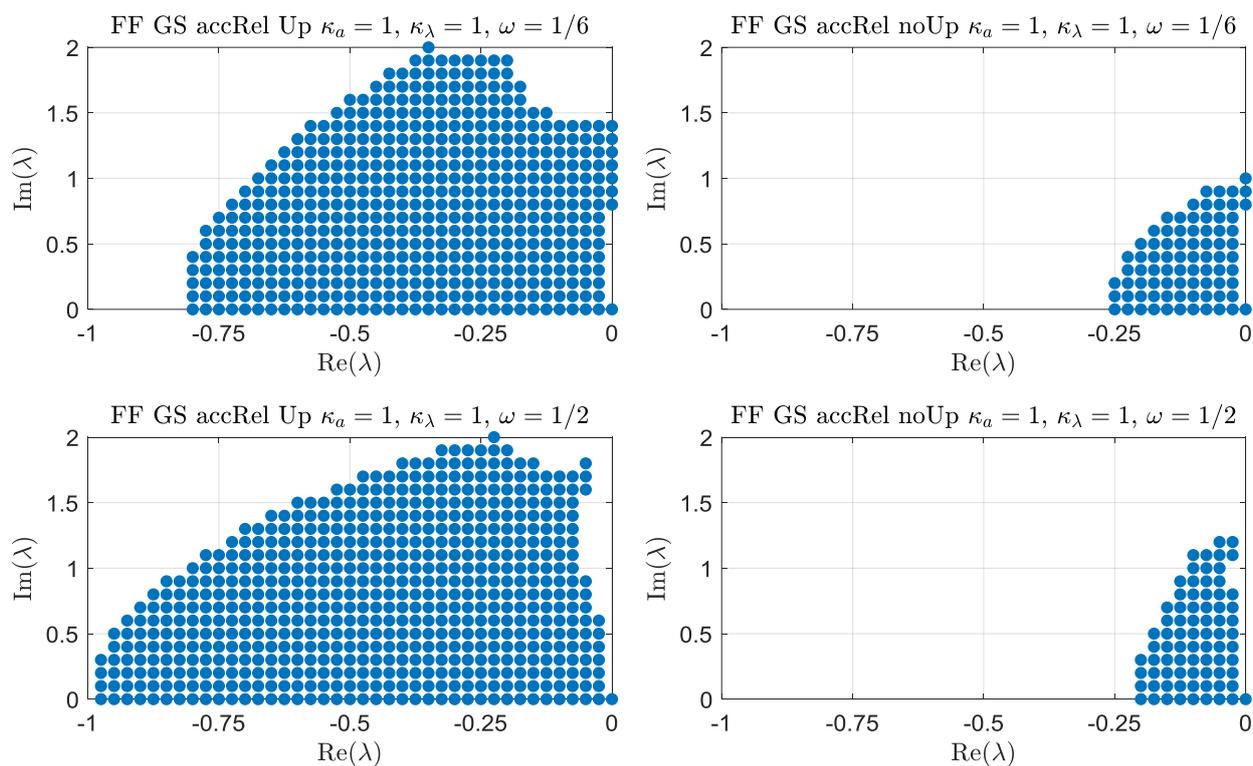
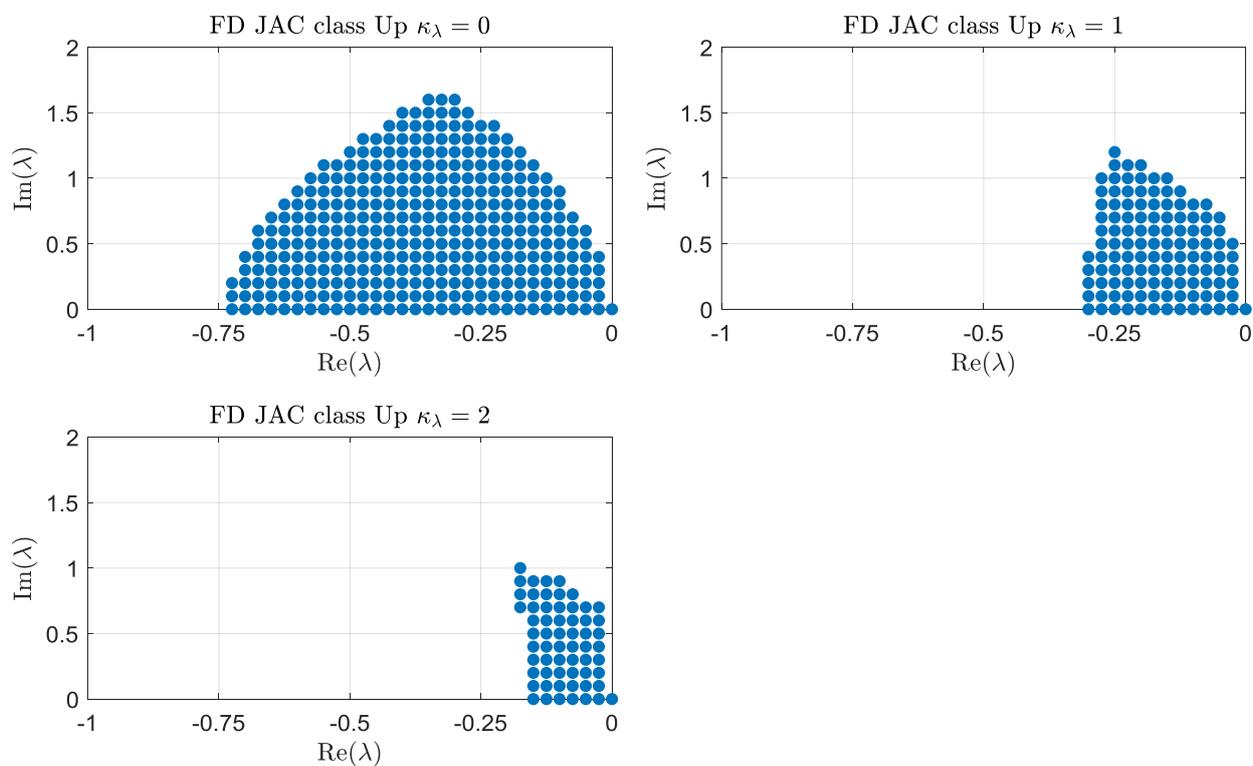


Figure C.17: Numerical stability: force/force, Gauss-Seidel type, *accRel*,  $\kappa_a = 1$ ,  $\kappa_\lambda = 0$ .



**Figure C.18:** Numerical stability: force/force, Gauss-Seidel type, *accRel*,  $\kappa_a = 1$ ,  $\kappa_\lambda = 1$ .



**Figure C.19:** Numerical stability: force/displacement, Jacobi type, *classic*.

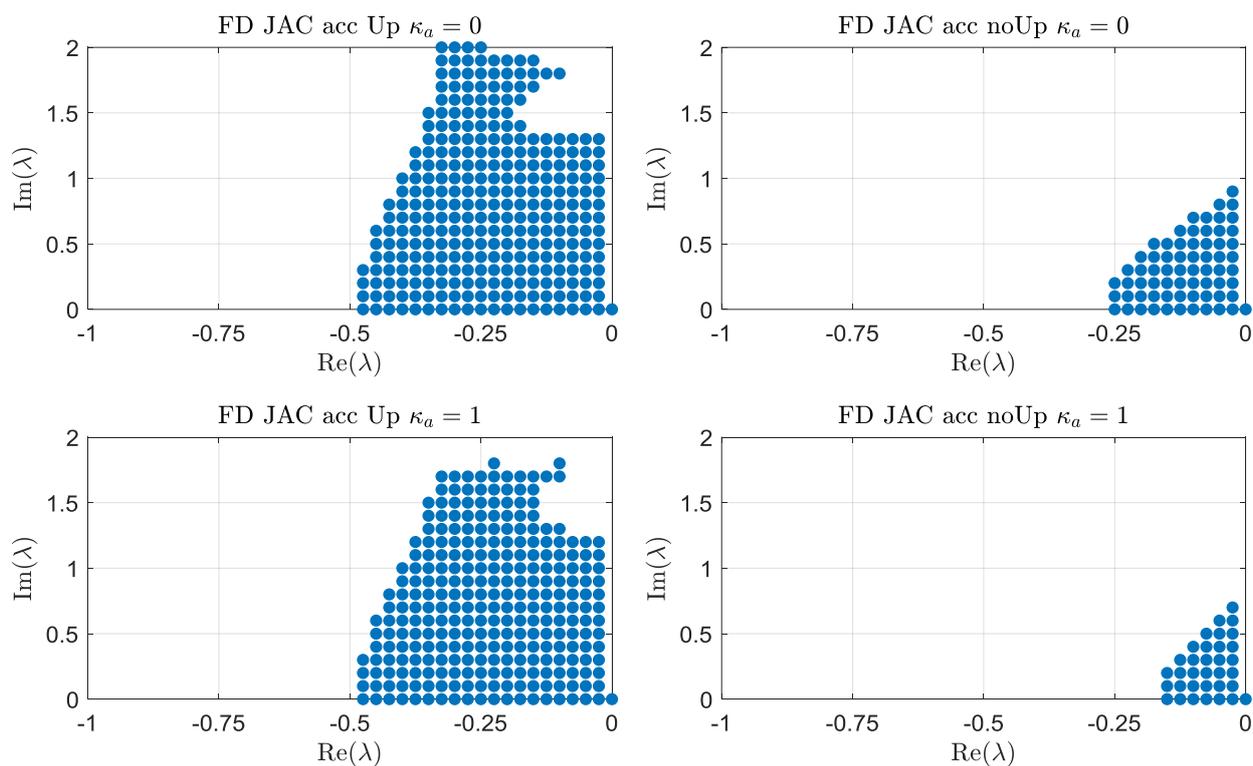


Figure C.20: Numerical stability: force/displacement, Jacobi type, *acc*.

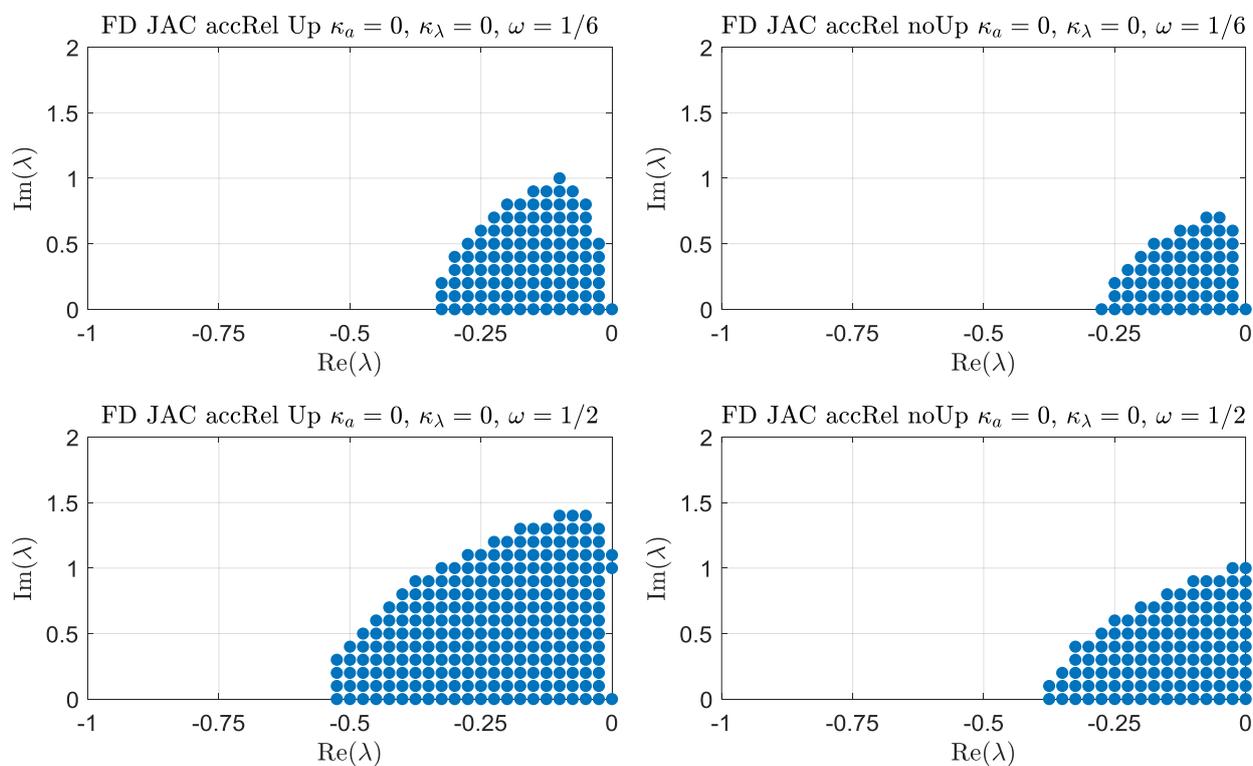


Figure C.21: Numerical stability: force/displacement, Jacobi type, *accRel*,  $\kappa_a = 0$ ,  $\kappa_\lambda = 0$ .

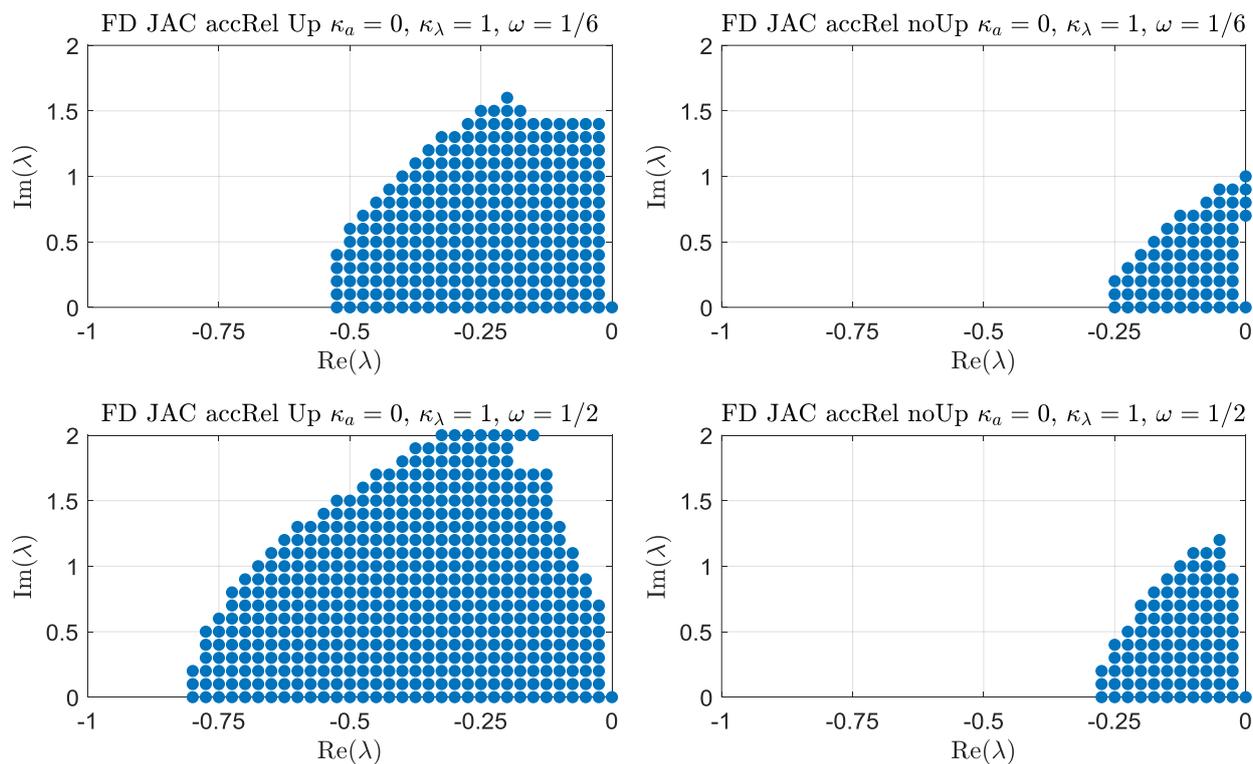


Figure C.22: Numerical stability: force/displacement, Jacobi type, *accRel*,  $\kappa_a = 0, \kappa_\lambda = 1$ .

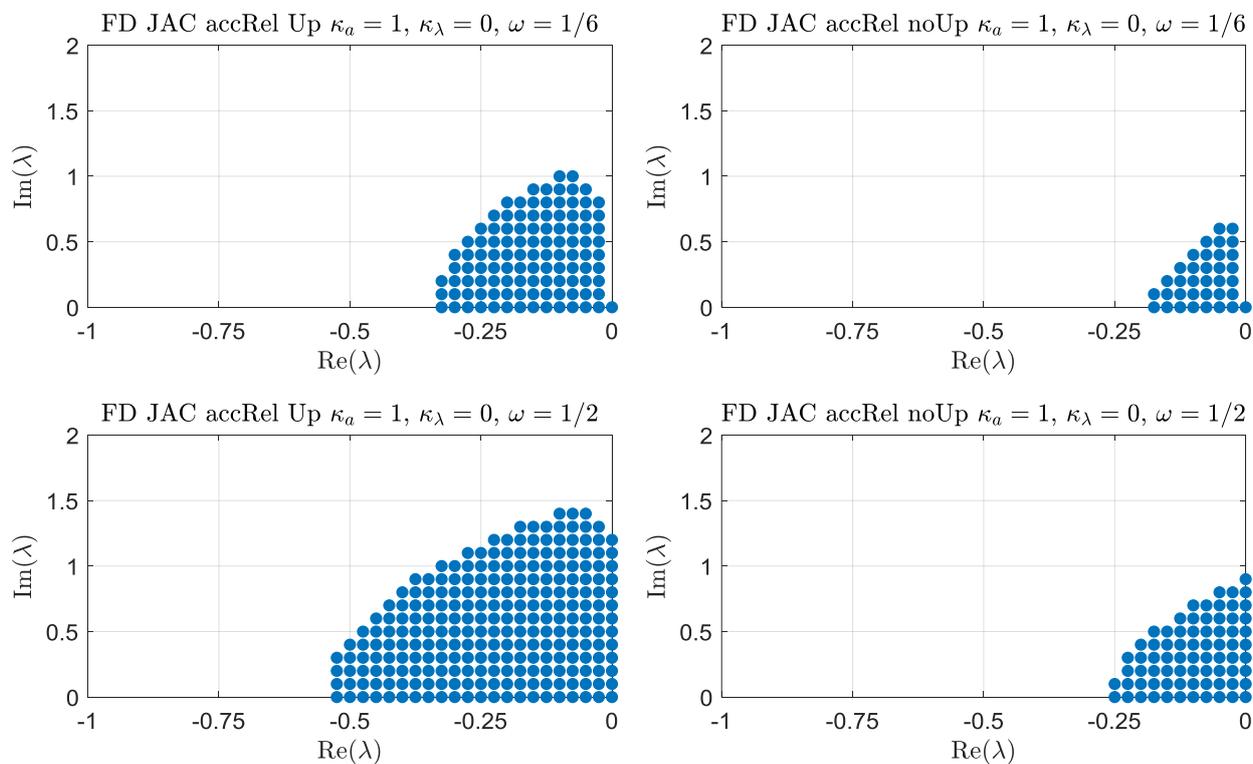


Figure C.23: Numerical stability: force/displacement, Jacobi type, *accRel*,  $\kappa_a = 1, \kappa_\lambda = 0$ .

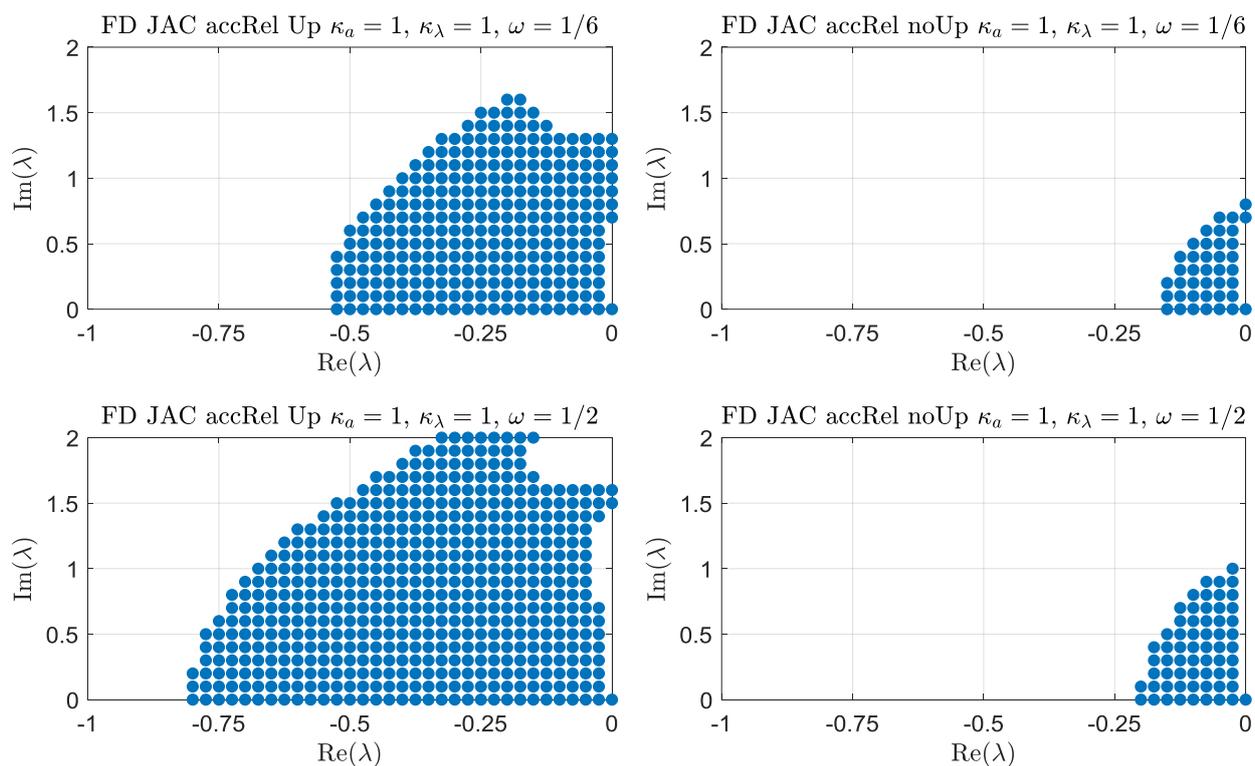


Figure C.24: Numerical stability: force/displacement, Jacobi type, *accRel*,  $\kappa_a = 1, \kappa_\lambda = 1$ .

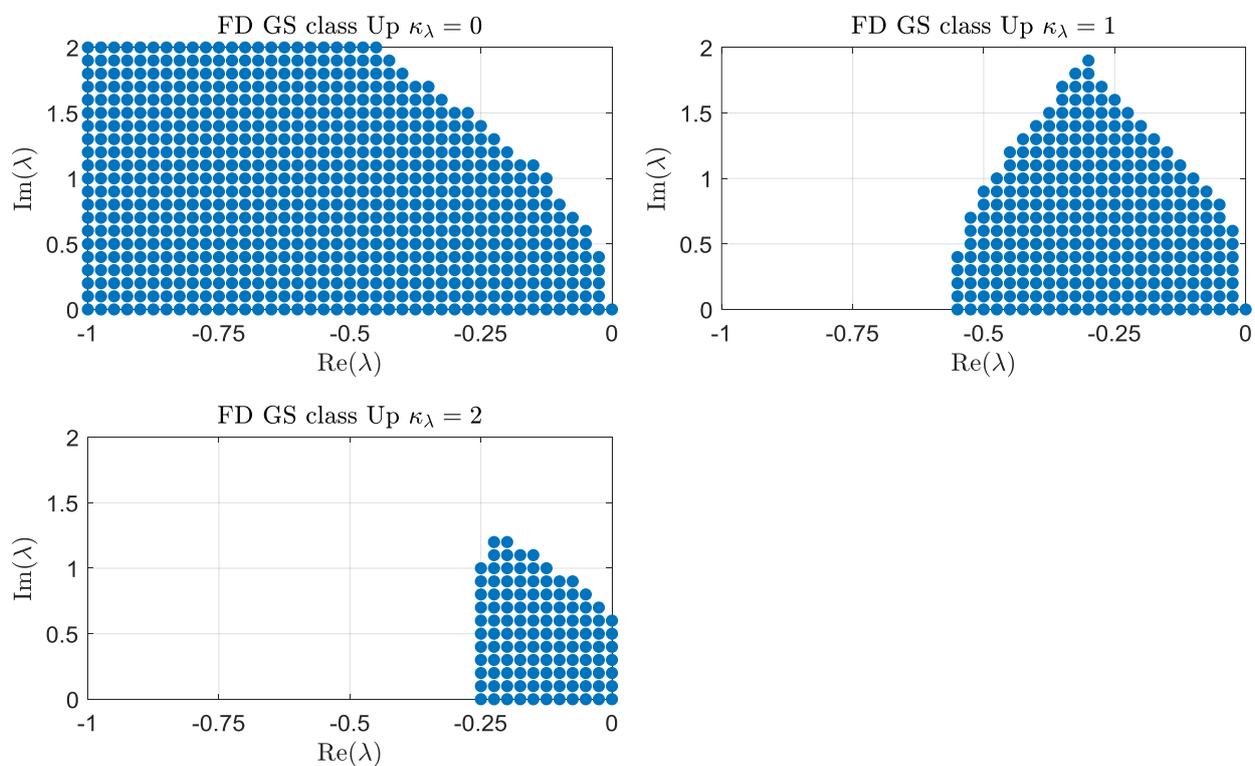
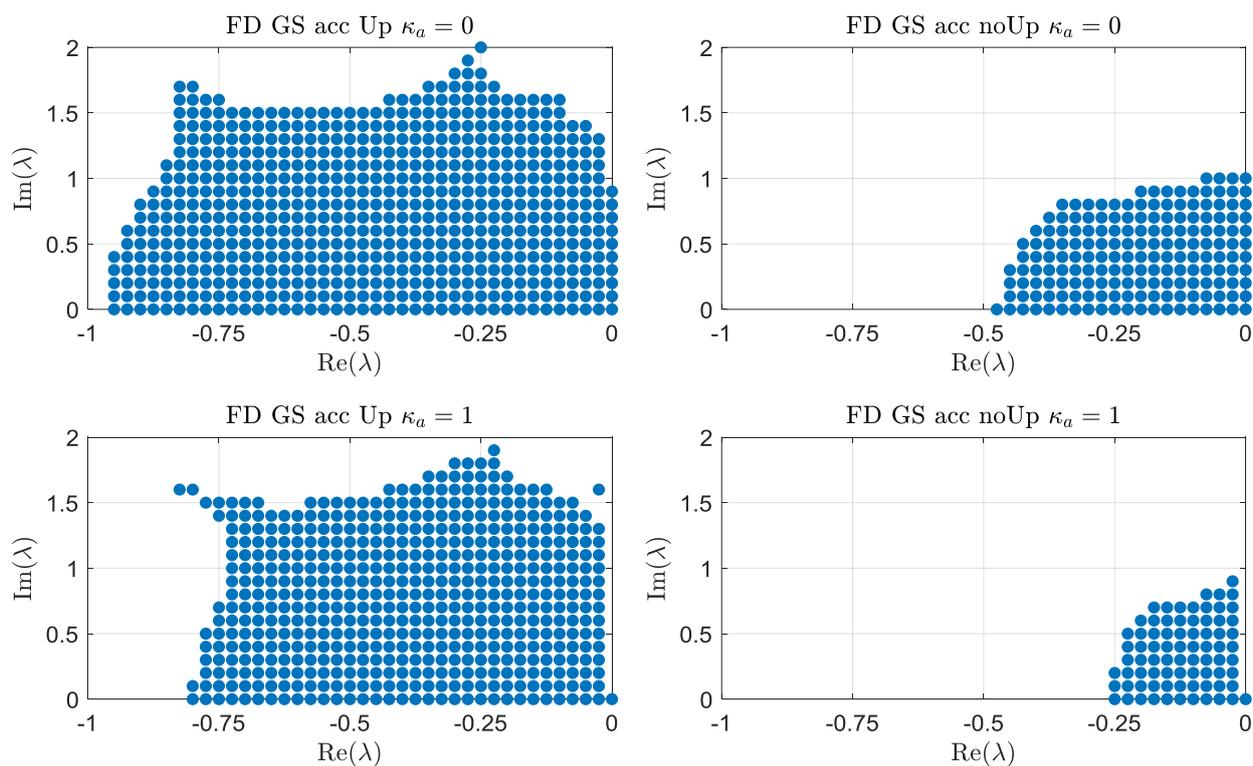
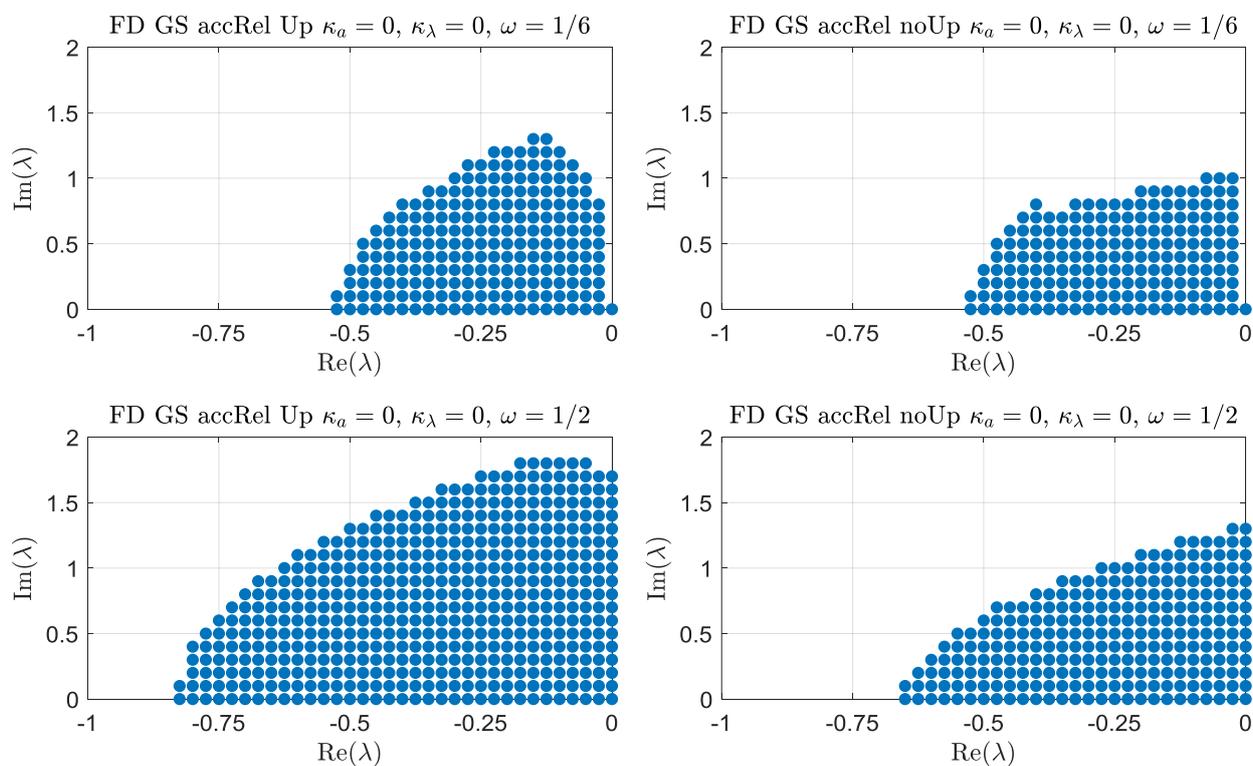


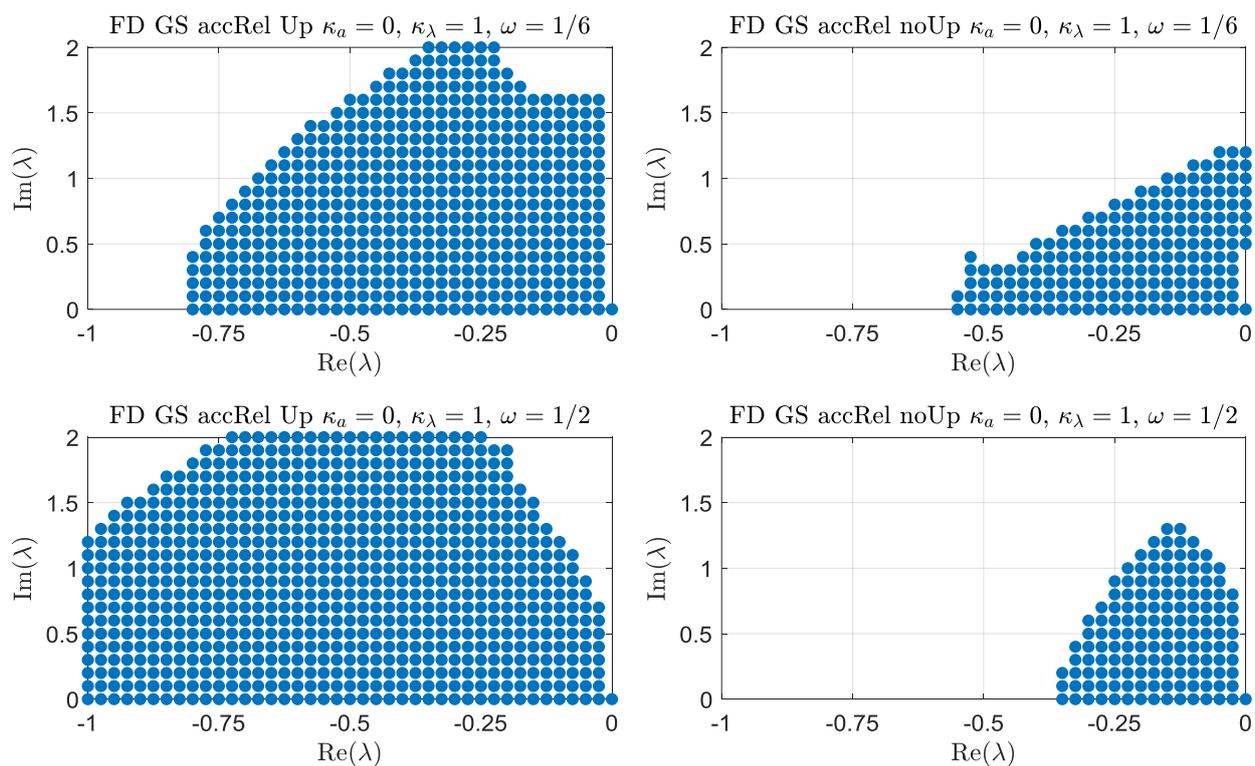
Figure C.25: Numerical stability: force/displacement, Gauss-Seidel type, *classic*.



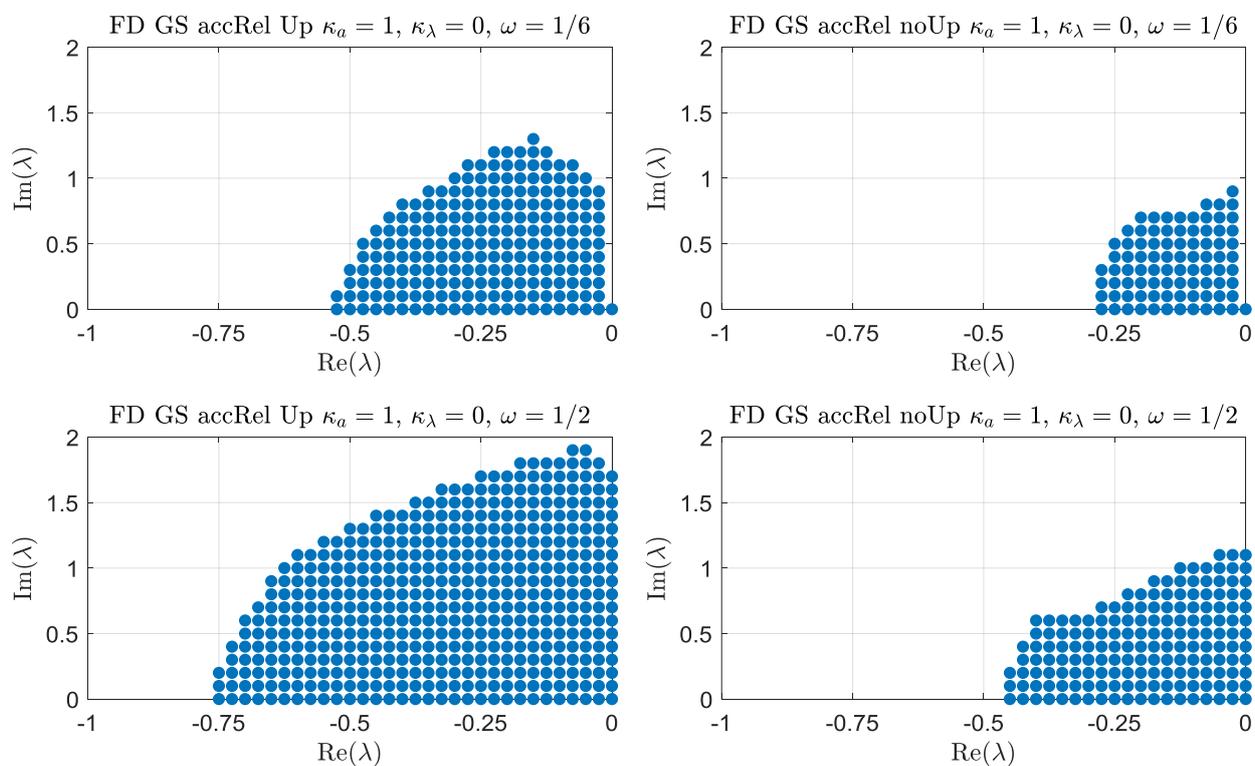
**Figure C.26:** Numerical stability: force/displacement, Gauss-Seidel type, *acc*.



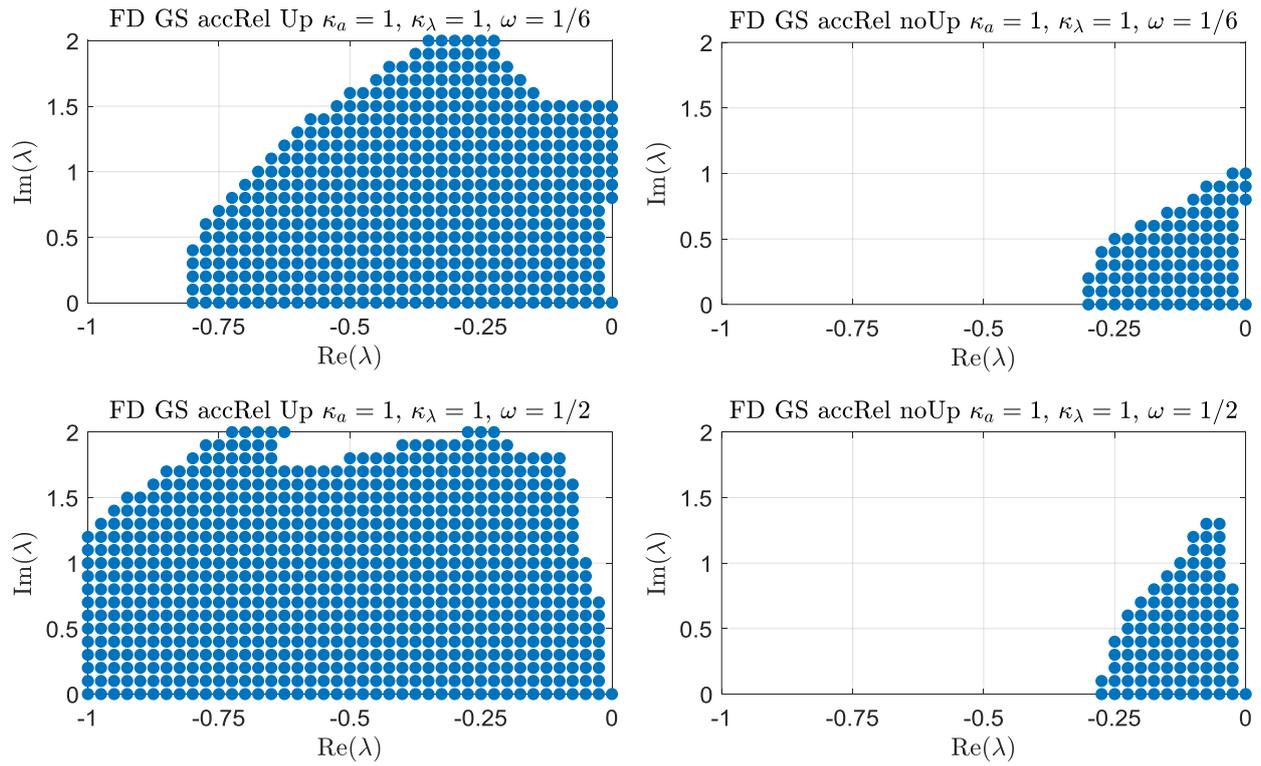
**Figure C.27:** Numerical stability: force/displacement, Gauss-Seidel type, *accRel*,  $\kappa_a = 0$ ,  $\kappa_\lambda = 0$ .



**Figure C.28:** Numerical stability: force/displacement, Gauss-Seidel type, *accRel*,  $\kappa_a = 0$ ,  $\kappa_\lambda = 1$ .



**Figure C.29:** Numerical stability: force/displacement, Gauss-Seidel type, *accRel*,  $\kappa_a = 1$ ,  $\kappa_\lambda = 0$ .



**Figure C.30:** Numerical stability: force/displacement, Gauss-Seidel type, *accRel*,  $\kappa_a = 1, \kappa_\lambda = 1$ .

---

## References

- [ACS13] Martin Arnold, Christoph Clauss, and Tom Schierz. Error analysis and error estimates for co-simulation in fmi for model exchange and co-simulation v2.0. *Archive of Mechanical Engineering*, 60(1):75 – 94, 2013.
- [AFF06] Martin Arnold, Andreas Fuchs, and Claus Führer. Efficient corrector iteration for dae time integration in multibody dynamics. *Computer methods in applied mechanics and engineering*, 195(50-51):6958–6973, 2006.
- [AFk16] Christian Andersson, Claus Führer, and Johan Åkesson. Efficient predictor for co-simulation with multistep sub-system solvers. Technical Report 1, Centre for Mathematical Sciences, Lund University, 2016.
- [AHSS03] S.A. Al-Hiddabi, B. Samanta, and A. Seibi. Non-linear control of torsional and bending vibrations of oilwell drillstrings. *Journal of Sound and Vibration*, 265(2):401 – 415, 2003.
- [AMA<sup>+</sup>18] P Antunes, H Magalhães, J Ambrósio, J Pombo, and J Costa. A co-simulation approach to the wheel–rail contact with flexible railway track. *Multibody System Dynamics*, pages 1–28, 2018.
- [AMM13] Mattia Alioli, Marco Morandini, and Pierangelo Masarati. Coupled multibody-fluid dynamics simulation of flapping wings. In *ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages V07BT10A014–V07BT10A014. American Society of Mechanical Engineers, 2013.
- [APP<sup>+</sup>12] Jorge Ambrósio, Joao Pombo, Manuel Pereira, Pedro Antunes, and António Mósca. A computational procedure for the dynamic analysis of the catenary-pantograph interaction in high-speed trains. *Journal of Theoretical and Applied Mechanics*, 50(3):681–699, 2012.
- [APRP09] Jorge Ambrósio, João Pombo, Frederico Rauter, and Manuel Pereira. A memory based communication in the co-simulation of multibody and finite element codes for pantograph-catenary interaction simulation. In Carlo L. Bottasso, editor, *Multibody Dynamics*, volume 12 of *Computational Methods in Applied Sciences*, pages 231 – 252. Springer Netherlands, 2009.
- [Arn07] Martin Arnold. Multi-rate time integration for large scale multibody system models. In Peter Eberhard, editor, *IUTAM Symposium on Multiscale Problems in Multibody System Contacts*, volume 1 of *IUTAM Bookseries (closed)*, pages 1 – 10. Springer, 2007.
- [Arn09] Martin Arnold. Numerical methods for simulation in applied dynamics. In Martin Arnold and Werner Schiehlen, editors, *Simulation Techniques for Applied Dynamics*, volume 507 of *CISM Courses and Lectures*, pages 191 – 246. Springer, 2009.

- [Arn10] M. Arnold. Stability of sequential modular time integration methods for coupled multibody system models. *Journal of Computational and Nonlinear Dynamics*, 5, 2010.
- [BCP96] Kathryn Eleda Brenan, Stephen L Campbell, and Linda Ruth Petzold. *Numerical solution of initial-value problems in differential-algebraic equations*, volume 14. Siam, 1996.
- [BKEFDF<sup>+</sup>17] Abir Ben Khaled-El Feki, Laurent Duval, Cyril Faure, Daniel Simon, and Mongi Ben Gaid. Choptrey: contextual online polynomial extrapolation for enhanced multi-core co-simulation of complex systems. *Simulation*, 93(3):185–200, 2017.
- [BS10a] Martin Busch and Bernhard Schweizer. Explicit and implicit solver coupling: Stability analysis based on an eight-parameter test model. *Proceedings in Applied Mathematics and Mechanics*, 10:61 – 62, 2010.
- [BS10b] Martin Busch and Bernhard Schweizer. Numerical stability and accuracy of different co-simulation techniques: Analytical investigations based on a 2-dof test model. In *The 1st Joint International Conference on Multibody System Dynamics*, Lappeenranta, Finland, May 2010.
- [BS11a] Martin Busch and Bernhard Schweizer. Coupled simulation of multibody and finite element systems: an efficient and robust semi-implicit coupling approach. *Archive of Applied Mechanics*, 82:723 – 741, 2011.
- [BS11b] Martin Busch and Bernhard Schweizer. An explicit approach for controlling the macro-step size of co-simulation methods. In *7th European Nonlinear Dynamics Conference*, Rome, Italy, July 2011.
- [BS11c] Martin Busch and Bernhard Schweizer. Stability of co-simulation methods using hermite and lagrange approximation techniques. In *ECCOMAS Thematic Conference - Multibody Dynamics 2011*, Brussels, Belgium, July 2011.
- [BS19] Michael Burger and Stefan Steidel. Local extrapolation and linear-implicit stabilization in a parallel coupling scheme. In *IUTAM Symposium on Solver-Coupling and Co-Simulation*, pages 43–56. Springer, 2019.
- [Bus12] Martin Busch. *Zur effizienten Kopplung von Simulationsprogrammen*. PhD thesis, Universität Kassel, 2012.
- [Bus16] Martin Busch. Continuous approximation techniques for co-simulation methods: Analysis of numerical stability and local error. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 96(9):1061–1081, 2016.
- [Bus19] Martin Busch. Performance improvement of explicit co-simulation methods through continuous extrapolation. In *IUTAM Symposium on Solver-Coupling and Co-Simulation*, pages 57–80. Springer, 2019.

- [BWZH13] M Benedikt, D Watzenig, J Zehetner, and A Hofer. Nepce—a nearly energy preserving coupling element for weak-coupled problems and co-simulation. In *IV International Conference on Computational Methods for Coupled Problems in Science and Engineering, Coupled Problems*, 2013.
- [CCMB00] J Cuadrado, J Cardenal, P Morer, and E Bayo. Intelligent simulation of multibody dynamics: space-state and descriptor methods in sequential and parallel computing environments. *Multibody System Dynamics*, 4(1):55–73, 2000.
- [Cha89] F-Y Chang. The generalized method of characteristics for waveform relaxation analysis of lossy coupled transmission lines. *IEEE Transactions on Microwave Theory and Techniques*, 37(12):2028–2038, 1989.
- [CI94] Mariesa L Crow and MD Ilić. The waveform relaxation method for systems of differential/algebraic equations. *Mathematical and computer modelling*, 19(12):67–84, 1994.
- [DM06] Jack Dongarra and Kaj Madsen. *Applied Parallel Computing: State of the Art in Scientific Computing*, volume 3732. Springer Science & Business Media, 2006.
- [DML69] George E. Duvall, R. Manvi, and Sherman C. Lowell. Steady shock profile in a one-dimensional lattice. *Journal of Applied Physics*, 40(9):3771–3775, 1969.
- [DPN10] Timothy A Davis and Ekanathan Palamadai Natarajan. Algorithm 907: Klu, a direct sparse solver for circuit simulation problems. *ACM Transactions on Mathematical Software (TOMS)*, 37(3):36, 2010.
- [DSN12] Makarand Datar, Ilinca Stanciulescu, and Dan Negrut. A co-simulation environment for high-fidelity virtual prototyping of vehicle systems. *International Journal of Vehicle Systems Modelling and Testing*, 7(1):54, 2012.
- [EEHJ96] Kenneth Eriksson, Don Estep, Peter Hansbo, and Claes Johnson. *Computational differential equations*, volume 1. Cambridge University Press, 1996.
- [ESF98] Edda Eich-Soellner and Claus Führer. *Numerical methods in multibody dynamics*. Vieweg+Teubner Verlag, 1998.
- [Fea83] Roy Featherstone. The calculation of robot dynamics using articulated-body inertias. *The International Journal of Robotics Research*, 2(1):13–30, 1983.
- [Feh68] Erwin Fehlberg. Classical fifth-, sixth-, seventh-, and eighth-order runge-kutta formulas with stepsize control. *NASA TR R 287*, 1968.
- [FMI14] FMI. *The functional mockup interface*. <https://fmi-standard.org>, 2014.
- [FSU10] Markus Friedrich, Markus Schneider, and Heinz Ulbrich. A parallel co-simulation for mechatronic systems. In *The 1st Joint International Conference on Multibody System Dynamics*, Lappeenranta, Finland, May 2010.

- [FU09] Markus Friedrich and Heinz Ulbrich. A parallel co-simulation for multibody systems. In *2nd South-East European Conference on Computational Mechanics*, Rhodes, Greece, June 2009.
- [GA04] Bei Gu and H. Harry Asada. Co-simulation of algebraically coupled dynamic subsystems without disclosure of proprietary subsystem models. *Journal of Dynamic Systems, Measurement, and Control*, 126:1 – 13, 2004.
- [Gea91] Charles William Gear. Waveform methods for space and time parallelism. *Journal of Computational and Applied Mathematics*, 38(1-3):137–147, 1991.
- [GGLC10] Manuel González, Francisco González, Alberto Luaces, and Javier Cuadrado. A collaborative benchmarking framework for multibody system dynamics. *Engineering with Computers*, 26(1):1–9, Feb 2010.
- [GGM10] Francisco González, Manuel González, and Aki Mikkola. Efficient coupling of multibody software with numerical computing environments and block diagram simulators. *Multibody System Dynamics*, 24:237 – 253, 2010.
- [GLS88] Kjell Gustafsson, Michael Lundh, and Gustaf Söderlind. Api stepsize control for the numerical solution of ordinary differential equations. *BIT Numerical Mathematics*, 28(2):270–287, Jun 1988.
- [GnNLG11] Francisco González, Miguel Ángel Naya, Alberto Lucaces, and Manuel González. On the effect of multirate co-simulation techniques in the efficiency and accuracy of multibody system dynamics. *Multibody System Dynamics*, 25:461 – 483, 2011.
- [GTB<sup>+</sup>17] Cláudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, and Hans Vangheluwe. Co-simulation: State of the art. *Technical Report. arXiv:1702.00686*, 2017.
- [GTB<sup>+</sup>18] Cláudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, and Hans Vangheluwe. Co-simulation: a survey. *ACM Computing Surveys (CSUR)*, 51(3):49, 2018.
- [HBG<sup>+</sup>05] Alan C Hindmarsh, Peter N Brown, Keith E Grant, Steven L Lee, Radu Serban, Dan E Shumaker, and Carol S Woodward. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):363–396, 2005.
- [HSC20] Alan C Hindmarsh, Radu Serban, and Aaron Collier. *User Documentation for ida v5.3.0*. U.S. Department of Energy by Lawrence Livermore National Laboratory, 2020.
- [HSL10] Torsten Hoefler, Timo Schneider, and Andrew Lumsdaine. Accurately measuring overhead, communication time and progression of blocking and nonblocking collective operations at massive scale. *International Journal of Parallel, Emergent and Distributed Systems*, 25(4):241–258, 2010.

- [HSR04] Alan C. Hindmarsh, Radu Serban, and Daniel R. Reynolds. *User Documentation for cvode v5.3.0*. U.S. Department of Energy by Lawrence Livermore National Laboratory, 2004.
- [int18] intel. *Intel MPI Library for Linux OS Developer Reference*, 2018.
- [Jan91] J.D. Jansen. Non-linear rotor dynamics as applied to oilwell drillstring vibrations. *Journal of Sound and Vibration*, 147(1):115 – 135, 1991.
- [Jen03] J.S. Jensen. Phononic band gaps and vibrations in one- and two-dimensional mass–spring structures. *Journal of Sound and Vibration*, 266(5):1053 – 1078, 2003.
- [KS00] Ralf Kübler and Werner Schiehlen. Two methods of simulator coupling. *Mathematical and computer modelling of dynamical systems*, 6(2):93–113, 2000.
- [KS12] E. Kreuzer and M. Steidl. Controlling torsional vibrations of drill strings via decomposition of traveling waves. *Archive of Applied Mechanics*, 82(4):515–531, Apr 2012.
- [Li17] Pu Li. *On the Numerical Stability of Co-Simulation Methods*. PhD thesis, Technische Universität Darmstadt, 2017.
- [LLSS17] Pu Li, Daixing Lu, Robert Schmoll, and Bernhard Schweizer. Explicit co-simulation approach with improved numerical stability. In Bernhard Schweizer, editor, *Proceedings of the IUTAM Symposium on Solver-Coupling and Co-Simulation*, volume 35 of *IUTAM*, pages 153–201. Springer, 2017.
- [LRSV82] Ekachai Lelarasmee, Albert E Ruehli, and Alberto L Sangiovanni-Vincentelli. The waveform relaxation method for time-domain analysis of large scale integrated circuits. *IEEE transactions on computer-aided design of integrated circuits and systems*, 1(3):131–145, 1982.
- [Lu15] Daixing Lu. *Semi-implizite Co-Simulationsverfahren*. PhD thesis, Technische Universität Darmstadt, 2015.
- [LYL<sup>+</sup>20] Pu Li, Qi Yuan, Daixing Lu, Tobias Meyer, and Bernhard Schweizer. Improved explicit co-simulation methods incorporating relaxation techniques. *Archive of Applied Mechanics*, 90(1):17–46, 2020.
- [MAT17] MATLAB. *version R2017b*. The MathWorks Inc., Natick, Massachusetts, 2017.
- [MGS<sup>+</sup>17] Michal Maciejewski, I Cortes Garcia, Sebastian Schöps, Bernhard Auchmann, Lorenzo Bortot, Marco Prioli, and AP Verweij. Application of the waveform relaxation technique to the co-simulation of power converter controller and electrical circuit models. In *2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 837–842. IEEE, 2017.
- [Mil26] William Edmund Milne. Numerical integration of ordinary differential equations. *The American Mathematical Monthly*, 33(9):455–460, 1926.

- [MKL<sup>+</sup>17] Tobias Meyer, Jan Kraft, Pu Li, Daixing Lu, and Bernhard Schweizer. Error estimation approach for controlling the macro step-size for explicit co-simulation methods. In *Proceedings of the 7th GACM Colloquium on Computational Mechanics for Young Scientists from Academia and Industry*, Stuttgart, Germany, October 11-13 2017.
- [MKLS19] Tobias Meyer, Jan Kraft, Daixing Lu, and Bernhard Schweizer. Error estimation approach for controlling the communication step-size for explicit co-simulation methods. In *IUTAM Symposium on Solver-Coupling and Co-Simulation*, volume 35 of *IUTAM*, pages 217–241. Springer, 2019.
- [MKS21] T. Meyer, J. Kraft, and B. Schweizer. Co-Simulation: Error Estimation and Macro-Step Size Control. *Journal of Computational and Nonlinear Dynamics*, 16(4), 02 2021. 041002.
- [MS07] Ashish Mohan and SK Saha. A recursive, numerically stable, and efficient simulation algorithm for serial robots. *Multibody System Dynamics*, 17(4):291–319, 2007.
- [N<sup>+</sup>59] Nathan Mortimore Newmark et al. A method of computation for structural dynamics. In *Journal of the Engineering Mechanics Division*, volume 85, pages 67–94. American Society of Civil Engineers, 1959.
- [NCDL11] Miguel Naya, Javier Cuadrado, Daniel Dopico, and Urbano Lugris. An efficient unified method for the combined simulation of multibody and hydraulic dynamics: Comparison with simplified and co-integration approaches. *Archive of Mechanical Engineering*, LVIII(2):223 – 243, 2011.
- [Now18] Gerrit Edgar Nowald. *Numerical Investigation of Rotors in Floating Ring Bearings using Co-Simulation*. PhD thesis, Technische Universitaet Darmstadt, 2018.
- [NSMH14] Dan Negrut, Radu Serban, Hammad Mazhar, and Toby Heyn. Parallel computing in multibody system dynamics: Why, when, and how. *Journal of Computational and Nonlinear Dynamics*, 9(4):041007, 2014.
- [NTM<sup>+</sup>12] Dan Negrut, Alessandro Tasora, Hammad Mazhar, Toby Heyn, and Philipp Hahn. Leveraging parallel computing in multibody dynamics. *Multibody System Dynamics*, 27(1):95–117, 2012.
- [OCC77] Nicolae Orlandea, Milton A Chace, and Donald Albert Calahan. A sparsity-oriented approach to the dynamic analysis and design of mechanical systems—part 1. *Journal of Engineering for Industry*, 99(3):773–779, 1977.
- [Ope15] OpenMP Architecture Review Board. *OpenMP 4.5 API C/C++ Syntax Reference Guide*, 2015.
- [PA12] Joao Pombo and Jorge Ambrósio. Multiple pantograph interaction with catenaries in high-speed trains. *Journal of Computational and Nonlinear Dynamics*, 7(4):041008, 2012.

- [PGKT18] Albert Peiret, Francisco González, József Kövecses, and Marek Teichmann. Multi-body system dynamics interface modelling for stable multirate co-simulation of multiphysics systems. *Mechanism and Machine Theory*, 127:52–72, 2018.
- [PKB08] Kosmas Petridis, Andreas Klein, and Michael Beitelschmidt. Asynchronous method for the coupled simulation of mechatronic systems. *Proceedings in Applied Mathematics and Mechanics*, 8(1):10521 – 10522, 2008.
- [RGN19] Jarkko Rahikainen, Francisco González, and Miguel Ángel Naya. An automated methodology to select functional co-simulation configurations. *Multibody System Dynamics*, pages 1–25, 2019.
- [RH93] Krishnan Radhakrishnan and Alan C Hindmarsh. Description and use of Isode, the livermore solver for ordinary differential equations. *NASA Reference Publication*, 1327, 1993.
- [RSP<sup>+</sup>17] Antonio Recuero, Radu Serban, Bryan Peterson, Hiroyuki Sugiyama, Paramsothy Jayakumar, and Dan Negrut. A high-fidelity approach for vehicle mobility simulation: Nonlinear finite element tires operating on granular material. *Journal of Terramechanics*, 72:39–54, 2017.
- [SA12] Tom Schierz and Martin Arnold. Stabilized overlapping modular time integration of coupled differential-algebraic equations. *Applied Numerical Mathematics*, 62(10):1491–1502, 2012.
- [SBAS17] Fabio Schneider, Michael Burger, Martin Arnold, and Bernd Simeon. A new approach for force-displacement co-simulation using kinematic coupling constraints. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 97(9):1147–1166, 2017.
- [SBC<sup>+</sup>07] J. C. Samin, O. Brüls, J. F. Collard, L. Sass, and P. Fiset. Multiphysics modeling and optimization of mechatronic multibody systems. *Multibody System Dynamics*, 18:345 – 373, 2007.
- [Sch15] Robert Schmoll. *Co-Simulation und Solverkopplung: Analyse komplexer multiphysikalischer Systeme*. PhD thesis, Universität Kassel, 2015.
- [SG75] Lawrence F Shampine and Marilyn K Gordon. *Computer solution of ordinary differential equations: the initial value problem*. Freeman, 1975.
- [Sha73] LF Shampine. Local extrapolation in the solution of ordinary differential equations. *Mathematics of Computation*, 27(121):91–97, 1973.
- [Sha80] Lawrence F Shampine. Implementation of implicit formulas for the solution of odes. *SIAM Journal on Scientific and Statistical Computing*, 1(1):103–118, 1980.
- [Ske77] Stig Skelboe. The control of order and steplength for backward differentiation methods. *BIT Numerical Mathematics*, 17(1):91–107, 1977.

- [SKSP17] Severin Sadjina, Lars T Kyllingstad, Stian Skjong, and Eilif Pedersen. Energy conservation and power bonds in co-simulations: non-iterative adaptive step size control and error estimation. *Engineering with Computers*, 33(3):607–620, 2017.
- [SL14a] Bernhard Schweizer and Daixing Lu. Predictor/corrector co-simulation approaches for solver coupling with algebraic constraints. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 2014.
- [SL14b] Bernhard Schweizer and Daixing Lu. Semi-implicit co-simulation approach for solver coupling. *Archive of Applied Mechanics*, 84(12):1739–1769, 2014.
- [SL15] Bernhard Schweizer and Daixing Lu. Stabilized index-2 co-simulation approach for solver coupling with algebraic constraints. *Multibody System Dynamics*, 34(2):129–161, 2015.
- [SLL15] Bernhard Schweizer, Pu Li, and Daixing Lu. Explicit and implicit cosimulation methods: Stability and convergence analysis for different solver coupling approaches. *Journal of Computational and Nonlinear Dynamics*, 10(5):051007, 2015.
- [SLL16] Bernhard Schweizer, Pu Li, and Daixing Lu. Implicit co-simulation methods: Stability and convergence analysis for solver coupling approaches with algebraic constraints. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 96(8):986–1012, 2016.
- [SLLM15] Bernhard Schweizer, Pu Li, Daixing Lu, and Tobias Meyer. Stabilized implicit co-simulation methods: solver coupling based on constitutive laws. *Archive of Applied Mechanics*, 85(11):1559–1594, 2015.
- [SM10] Sukhpreet Singh Sandhu and John McPhee. Partitioned dynamic simulation of multibody systems. *Journal of Computational and Nonlinear Dynamics*, 5(3):031007, 2010.
- [SM11a] Tommaso Solcia and Pierangelo Masarati. Efficient multirate simulation of complex multibody systems based on free software. In *ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, Washington, DC, USA, August 28-31 2011. ASME International.
- [SM11b] Tommaso Solcia and Pierangelo Masarati. Multirate simulation of complex multibody systems. In *Multibody Dynamics 2011*, Brussels, Belgium, July 2011.
- [Sof14] MSC Software. *Adams Co-Simulation Interface (ACSI)*, 2014.
- [SON<sup>+</sup>17] Radu Serban, Nicholas Olsen, Dan Negrut, Antonio Recuero, and Paramsothy Jayakumar. A co-simulation framework for high-performance, high-fidelity simulation of ground vehicle-terrain interaction. In *Conference: NATO AVT-265 Specialists Meeting, Vilnius, Lithuania (May 2017)*, 2017.
- [SP19] Severin Sadjina and Eilif Pedersen. Energy conservation and coupling error reduction in non-iterative co-simulations. *Engineering with Computers*, 2019.

- [SR97] Lawrence F Shampine and Mark W Reichelt. The matlab ode suite. *SIAM journal on scientific computing*, 18(1):1–22, 1997.
- [SS11] Robert Schmoll and Bernhard Schweizer. Co-simulation of multibody and hydraulic systems: Comparison of different coupling approaches. In *ECCOMAS Thematic Conference - Multibody Dynamics 2011*, Brussels, Belgium, July 2011.
- [SV96] Vladimir Stejskal and Michael Valášek. *Kinematics and dynamics of machinery*. M. Dekker, 1996.
- [SW81] Hans J Stetter and Ewa Weinmüller. On the error control in ode solvers with local extrapolation. *Computing*, 27(2):169–177, 1981.
- [TWY<sup>+</sup>16] Jialin Tian, Chunming Wu, Lin Yang, Zhi Yang, Gang Liu, and Changfu Yuan. Mathematical modeling and analysis of drill string longitudinal vibration with lateral inertia effect. *Shock and Vibration*, 2016:1–8, 03 2016.
- [TZL19] Liping Tang, Xiaohua Zhu, and Hongzhi Lin. Effects of axial impact load on the dynamics of an oilwell drillstring. *Advances in Mechanical Engineering*, 11(3):1687814019836876, 2019.
- [VHK<sup>+</sup>16] Mariano Vázquez, Guillaume Houzeaux, Seid Koric, Antoni Artigues, Jazmin Aguado-Sierra, Ruth Arís, Daniel Mira, Hadrien Calmet, Fernando Cucchietti, Herbert Owen, et al. Alya: Multiphysics engineering simulation toward exascale. *Journal of computational science*, 14:15–27, 2016.
- [VKV04] Ondrej Vaculin, Wolf R. Krüger, and Michael Valasek. Overview of coupling of multibody and control engineering tools. *Vehicle System Dynamics*, 41(5):415 – 429, 2004.
- [VM11] Michael Valasek and Ladislav Mraz. Parallelization of multibody system dynamics by heterogeneous multiscale method. In *ECCOMAS Thematic Conference - Multibody Dynamics 2011*, Brussels, Belgium, July 2011.
- [WMH03] Jinzhong Wang, Zheng-Dong Ma, and Gregory M Hulbert. A gluing algorithm for distributed simulation of multibody systems. *Nonlinear dynamics*, 34(1-2):159–188, 2003.
- [WSVOR85] John A White, Alberto Sangiovanni-Vincentelli, Farouk Odeh, and A Ruehli. *Waveform relaxation: Theory and practice*. Electronics Research Laboratory, College of Engineering, UCB, 1985.
- [YC00] AS Yigit and AP Christoforou. Coupled torsional and bending vibrations of actively controlled drillstrings. *Journal of sound and vibration*, 234(1):67–83, 2000.
- [ZEK<sup>+</sup>14] Zhenkai Zhang, Emeka Eyisi, Xenofon Koutsoukos, Joseph Porter, Gabor Karsai, and Janos Sztipanovits. A co-simulation framework for design of time-triggered automotive cyber physical systems. *Simulation modelling practice and theory*, 43:16–33, 2014.

## Publications

J. Kraft, S. Klimmek, T. Meyer and B. Schweizer. "Implicit Co-Simulation and Solver-Coupling: Efficient Calculation of Interface-Jacobian and Coupling Sensitivities/Gradients", In: *Journal of Computational and Nonlinear Dynamics* (2021).

T. Meyer, J. Kraft and B. Schweizer. "Co-Simulation: Error Estimation and Macro-Step Size Control", In: *Journal of Computational and Nonlinear Dynamics*, 16(4):041002 (2021).

J. Kraft, T. Meyer and B. Schweizer. "Parallel Co-Simulation Approach with Macro-Step Size and Order Control Algorithm", In: *Proceedings of the ASME 2019 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Volume 6: 15th International Conference on Multibody Systems, Nonlinear Dynamics, and Control*. Anaheim, California, USA. August 18–21, (2019).

J. Kraft, T. Meyer and B. Schweizer. "Reduction of the Computation Time of Large Multibody Systems with Co-simulation Methods", In: *Proceedings of the IUTAM Symposium on Solver-Coupling and Co-Simulation*, IUTAM Bookseries, Volume 35, Springer, pp. 131–152 (2019).

J. Kraft, T. Meyer and B. Schweizer. "Efficiency of Different Co-Simulation Approaches". In: *The 5th Joint International Conference on Multibody System Dynamics*, Lisbon, Portugal, June 24–28 (2018).

J. Kraft and B. Schweizer. Manolis Papadrakakis, E. O. & Schrefler, B. (Eds.). "Reduction of Computation Time by Parallelization Incorporating Co-Simulation Techniques", In: *Proceedings of The VII International Conference on Computational Methods for Coupled Problems in Science and Engineering*, International Center for Numerical Methods in Engineering (CIMNE), pp. 779–787 (2017).

J. Kraft and B. Schweizer. "Efficient Parallelization of Multibody Systems Incorporating Co-Simulation Techniques", In: *Proceedings of the 4th Joint International Conference on Multibody System Dynamics*, Montreal, Canada, May 29–June 2 (2016).