TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Angewandte Mechanik

Parallel Co-Simulation for Mechatronic Systems

Dipl.-Ing. Univ. Markus Friedrich

Vollständiger Abdruck der von der Fakultät für Maschinenwesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender:

Univ.-Prof. Dr.-Ing. Manfred Hajek

Prüfer der Dissertation:

1. Univ.-Prof. Dr.-Ing. habil. Heinz Ulbrich
2. Univ.-Prof. Dr. rer.nat. habil. Martin Arnold, Martin-Luther-Universität Halle-Wittenberg

Die Dissertation wurde am 10.02.2011 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 17.07.2011 angenommen.

Abstract

The simulation of physical systems is a very important task in modern product development. Besides single domain systems also the simulation of multi domain systems as a whole has increasing importance. This work deals with coupled simulation of multi domains, especially mechanical, hydraulic and electric ones, using co-simulation: the subsystems are integrated by their own integrators and are dynamically connected at discrete macro time steps. The main aspect lies on stability improvements and parallelization to achieve time efficient simulations on multi CPU computers. Beside analytical considerations also examples of industrial relevance are given showing the power of parallel multi domain co-simulations.

Zusammenfassung

Die Simulation physikalischer Systeme ist in der modernen Produktentwicklung von hoher Bedeutung. Neben Systemen aus einer physikalischen Domain steigt auch die Bedeutung von Multi-Domain Simulationen. Diese Arbeit handelt von gekoppelten Multi-Domain Simulationen, insbesondere von mechanischen, hydraulischen und elektrischen Systemen, durch Co-Simulation: Die Subsysteme werden dabei durch ihre eigenen Integratoren integriert und zu diskreten Makrozeitpunkten dynamisch gekoppelt. Der Hauptteil dieser Arbeit liegt auf Stabilitätsuntersuchungen und Parallelisierung um möglichst kurze Simulationszeiten auf Mehrprozessor PCs zu gewähren. Neben analytischen Untersuchungen werden auch Beispiele aus dem industriellen Umfeld gezeigt, welche die Leistungsfähigkeit von parallelen Multi-Domain Co-Simulationen aufzeigen.

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Assistent am Lehrstuhl für Angewandte Mechanik der Technischen Universität München unter der Leitung meines Doktorvaters Prof. Heinz Ulbrich. Ihm gebührt mein besonderer Dank für seine stete Unterstützung in allen Bereichen sowie für das in mich gesetzte Vertrauen. Dies hat maßgeblich zum Gelingen dieser Arbeit beigetragen.

Bei Herrn Prof. Martin Arnold sowie bei Herrn Prof. Manfred Hajek bedanke ich mich herzlich für das Interesse an dieser Arbeit sowie für die Übernahme des Zweitgutachtens beziehungsweise die Übernahme des Vorsitzes in der Prüfungskommission.

Besonders zu erwähnen ist auch das hervorragende Arbeitsklima an diesem Lehrstuhl, welches allen Mitarbeitern geschuldet ist. Insbesondere die Zusammenarbeit mit den Kollegen am dem MBSim Projekt war neben meinem Thema stets, nicht nur produktiv und weiterbildend, sondern auch äußerst abwechslungsreich und erfreulich. Besonders wichtig war mir beim Erstellen dieser Arbeit immer die Verifizierung an großen industriellen Beispielen. Diese erhielt ich aus den Projekten von Thomas Engelhardt, Roland Zander sowie insbesondere von Markus Schneider. Ihnen gilt hierfür mein besonderer Dank, da diese Beispiele die vorliegende Arbeit besonders aufgewertet haben und damit zum Gelingen beitrugen. Für die kritische und äußerst gewissenhafte Durchsicht des Manuskripts bedanke ich mich herzlich bei Thorsten Schinder, Roland Zander und Markus Schneider.

Nicht zuletzt gebührt mein besonderer Dank meinen Eltern für Ihre immerwährende Unterstützung bei meinem bisherigen Werdegang.

München, im November 2011 *Markus Friedrich*

Contents

1 Introduction

1.1 Simulation in the Industrial Design Process

Simulation of technical systems has become a very important task in the last decades and an integral part of industrial product design. Even nowadays it is a still rapidly growing field in all industrial areas as well as in all physical domains. There are different reasons for this development: First, the building of test rigs is often very complex and expensive, whereas simulations can be done with low effort and cost. Second, simulation allows to analyze systems which are not build at the time of investigation. Third, it is not practical to measure any kind of physical values on test rigs, because it may not be reasonable to plug a physical sensor due to space limitations or intervention on the dynamic behavior. Whereas in simulations one can plug a virtual sensor anywhere without a change of the system. Just another and often underrated positive effect of simulation is the possibility to obtain a high level of system comprehension. This is mainly attributed to the knowledge accumulation of the physical system behavior during the assembly of the simulation model.

The results of simulation tools are only practical if they sufficiently represent the physical behavior of the real system. Hence, the modeling of basic machine parts [50] and the whole machine dynamic [77] is very important and still an ongoing process in industry and research. To make a prediction on the quality of the simulation results it is and will always be necessary to build test rigs to compare the results from simulation with the measured results of the real system. Often it will be sufficient to adjust the submodels of simulations with measurements of the corresponding machine parts, because the assembly of submodels induces insignificant errors in the entire simulation.

A drawback of the simulation of physical systems is that models with high model quality come along with a high computational effort. The aim to improve the model usually increases the computational effort even more. Hence, the development of new simulation tools or models must consider computational performance aspects. Moreover, the development of parallel algorithms for simulation tools has become an increasing topic in the last years because modern computers or workstations are nowadays equipped with more than one central processing unit (CPU).

The integral construction of modern machines or machine parts is accompanied by a close integration of different physical domains. The integration of mechanical, hydraulic and electric components is called a *mechatronic system*. Modern engineering systems include in addition also components form other physical domains like thermodynamics or aerodynamics. Hence, besides the investigation of systems belonging

to one of these domains also the simulation of multi domain systems as a whole is of major interest.

1.2 Multi Domain Simulation: Literature Survey

Today there exist not only a lot of specialized simulation tools for different domains, where [1,2,16,72] are examples for multi body, hydraulic and fluid dynamic systems, but also a lot of sophisticated multi domain simulation tools for example [17,44,47]. In both fields there is still active development and research on implementing more detailed models and opening the tools to additional domains, providing analysis of multi domain dynamic systems in high quality. The area of multi domain simulation tools can be split into two major simulation variants [41]: First, the entire system is integrated by a single solver, where the different domains must be combined using model coupling. Second, the subsystems of different domains are integrated by their own solvers and are coupled using co-simulation.

1.2.1 Model Coupling

Using model coupling, also called *close* or *strong coupling*, it is necessary that the physical model of all parts of the simulation leads to the same mathematical formulation because the whole coupled system is integrated by a single integrator. Therefore this central integrator must be feasible for the given mathematical formulation of the dynamic equations. Since many domains can be formulated as differential algebraic (DAE) or ordinary differential equations (ODE), it is possible to couple a wide range of domains using this method.

Subsystem Modeling (using DAEs or ODEs)

Very general mathematical formulations for dynamic systems are differential algebraic equations (DAEs)

$$\boldsymbol{F}(\dot{\boldsymbol{z}}, \boldsymbol{z}, \boldsymbol{s}, t) = \boldsymbol{0}, \tag{1.1}$$

where $\boldsymbol{z}$ denotes the dependent differential variables, $\boldsymbol{s}$ the dependent algebraic variables and t the independent variable being usually the time. If $\dot{\boldsymbol{z}}$ appears only linear in $\boldsymbol{F}$, being true for many systems, equation (1.1) can be reformulated in a semi-explicit form

$$\dot{\boldsymbol{z}} = \boldsymbol{f}(\boldsymbol{z}, \boldsymbol{s}, t), \tag{1.2a}$$
$$\boldsymbol{0} = \boldsymbol{g}(\boldsymbol{z}, \boldsymbol{s}, t). \tag{1.2b}$$

For both formulations, there exist a lot of different numerical solvers given for example in [27]. If no algebraic variable $\boldsymbol{s}$ exist, the DAE (1.2) becomes an ordinary differential equation (ODE)

$$\dot{\boldsymbol{z}} = \boldsymbol{f}(\boldsymbol{z}, t), \tag{1.3}$$

which can be solved by more specialized numerical integrators [26].

Rigid and flexible multi body systems [3,61], hydraulic systems [7,13] as well as continuous controllers [59] are usually modeled using ODEs (1.3) or semi-explicit DAEs (1.2). Therefore mechatronic systems can be simulated using model coupling.

Elastic single or multi body systems are governed by partial differential equations, which do no match the general DAE (1.1). However, the use of variational methods like Galerkin or Ritz [55,58] leads to ODEs (1.3). These variational methods can be applied locally, leading to finite element methods (FEM) [8,42] with a large number of freedoms or globally using for example modal ansatz functions with much less number of freedoms. Using the finite element method it is even possible to describe spatial large deformations of mechanical systems in form of ordinary differential equations [71, 80]. Therefore also elastic systems can join mechatronic simulations using model coupling.

Fluid mechanical problems can be formulated using specialized finite elements [24]. At least for large fluid mechanical systems it is however more common to use finite volume methods [79]. However, the necessary use of specialized fluid mechanic solvers for finite volume methods excludes these methods from a model couping with the above systems.

Input, Output and Interconnection (of DAE-Systems)

For multi domain simulations each of N subsystems, modeled for example using a general DAE (1.1), is extended by an input vector $\boldsymbol{u}$ and output vector $\boldsymbol{y}$

$$\boldsymbol{F}_i(\dot{\boldsymbol{z}}_i, \boldsymbol{z}_i, \boldsymbol{s}_i, t, \boldsymbol{u}_i) = \boldsymbol{0}, \tag{1.4a}$$

$$\boldsymbol{y}_i = \boldsymbol{o}_i(\boldsymbol{z}_i, \boldsymbol{s}_i, t, \boldsymbol{u}_i), \quad i = A, B, C, \ldots, N, \tag{1.4b}$$

where the output is a function $\boldsymbol{o}_i$ of the dependent variables $\boldsymbol{z}_i$ and $\boldsymbol{s}_i$, the time t and the input $\boldsymbol{u}_i$. The interconnection equations

$$\boldsymbol{u}_i = \boldsymbol{L}_i \, \boldsymbol{y}_i^* = \boldsymbol{L}_i \begin{bmatrix} \boldsymbol{y}_A \\ \vdots \\ \boldsymbol{y}_{i-1} \\ \boldsymbol{y}_{i+1} \\ \vdots \\ \boldsymbol{y}_N \end{bmatrix} \tag{1.5}$$

define the input $\boldsymbol{u}_i$ by a function of all output variables $\boldsymbol{y}_j$ except $i = j$. The elements of the incidence matrix $\boldsymbol{L}_i$ [38], being zero or one, represent a simple feed through between input and output. The global differential equation including all subsystem is again a general DAE (1.1) with

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{z}_A \\ \vdots \\ \boldsymbol{z}_N \end{bmatrix}, \quad \boldsymbol{s} = \begin{bmatrix} \boldsymbol{s}_A \\ \vdots \\ \boldsymbol{s}_N \\ \boldsymbol{u}_A \\ \vdots \\ \boldsymbol{u}_N \end{bmatrix} \text{ and } \boldsymbol{F} = \begin{bmatrix} \boldsymbol{F}_A \\ \vdots \\ \boldsymbol{F}_N \\ \boldsymbol{u}_A - \boldsymbol{L}_A \, \boldsymbol{y}_A^* \\ \vdots \\ \boldsymbol{u}_N - \boldsymbol{L}_N \, \boldsymbol{y}_N^* \end{bmatrix}. \tag{1.6}$$

Even if the subsystems contain discontinuities, for example mechanical systems with non-smooth behavior due to rigid contacts between bodies [56], *model coupling* can be used. An example of a model coupling of subsystems solved by an ODE integrator with included root finding (also often called *event detection*) is given for example in [19].

Implementation

The generation of the general DAE (1.6) can be done internal in one multi domain simulation tool like [17, 47]. In this case the multi domain simulation tool also solves this equation by a provided DAE solver. Another approach is to export the differential equations of all but one subsystems, often represented by a stand-alone tool, by code-export. The remaining subsystem or tool imports all equations by code import and solves the global system with a provided integrator.

Advantages and Disadvantages

The advantages of model couplings are:

- A unitary mathematical formulation is used for all subsystems.
- Only the numerical discretization used by the integrator is needed. No further discretization with inherent errors is introduced.
- Sophisticated accuracy and stability investigation already exists for the numerical solvers.
- The option to reuse existing single domain models, if the corresponding tool provides code export.

Disadvantages may be:

- All tools must be modeled using the same mathematical formulation.
- Specialized integration methods cannot be used for individual domains.

- In case of code export/import at least one simulation tool must provide a model import functionality and all others must provide a model export functionality.

1.2.2 Co-Simulation

Another common method for multi domain simulation is co-simulation, often being also called *solver* or *process coupling* as well as *loose* or *weak coupling* [38, 41]. In contrast to model coupling, the modeling of the different domains need not to lead to the same mathematical formulation. A discrete coupling between the individual solvers for each domain or subsystem binds all parts to one multi domain simulation.

Subsystem Modeling

Using co-simulation each domain or subsystem is solved by its own, specific and optimized numerical integrator. For simplification a general DAE (1.1) is assumed exemplary. For the dynamic interaction between the subsystems the inputs and outputs of the subsystems must be interconnected. This data exchange is done only at discrete macro time steps to be independent of the algorithm of the subsystem solvers.

Input, Output and Interconnection

Each $i = A, B, C, \ldots, N$ subsystems must be extended by an input $\boldsymbol{u}_i$ and output $\boldsymbol{y}_i$ vector, as already shown by the model coupling equation (1.4). The interconnection equations (1.5) for co-simulations are also given like in the case of model coupling.

In contrast to model coupling the subsystem equations (1.4) are not joined to a single equation. Hence, care must be taken to avoid algebraic loops on the input. If, for $N = 2$ subsystems, each output function depends on the output of the other subsystem, then these output functions yield, using the interconnection equation (1.5),

$$\boldsymbol{y}_A = \boldsymbol{o}_A(\boldsymbol{z}_A, \boldsymbol{s}_A, t, \boldsymbol{L}_A \boldsymbol{y}_B), \tag{1.7a}$$
$$\boldsymbol{y}_B = \boldsymbol{o}_B(\boldsymbol{z}_B, \boldsymbol{s}_B, t, \boldsymbol{L}_B \boldsymbol{y}_A). \tag{1.7b}$$

These equations are implicit in $\boldsymbol{y}_A, \boldsymbol{y}_B$ and can only be solved together or by an iterative method. This is called an algebraic loop in the input and has significant consequences on the feasibility of the coupling method, the convergence and the zero-stability of the global system [4, 38]. Solution strategies in case of algebraic loops are the iterative solution of the output equations or of the macro integration step as described in [38]. Alternatively the subsystems can be reformulated in a way that the implicit dependency of the input vectors $\boldsymbol{u}_i$ on the output functions $\boldsymbol{o}_i$ are eliminated. This approach called *filtering* is also described in [38] and is closely

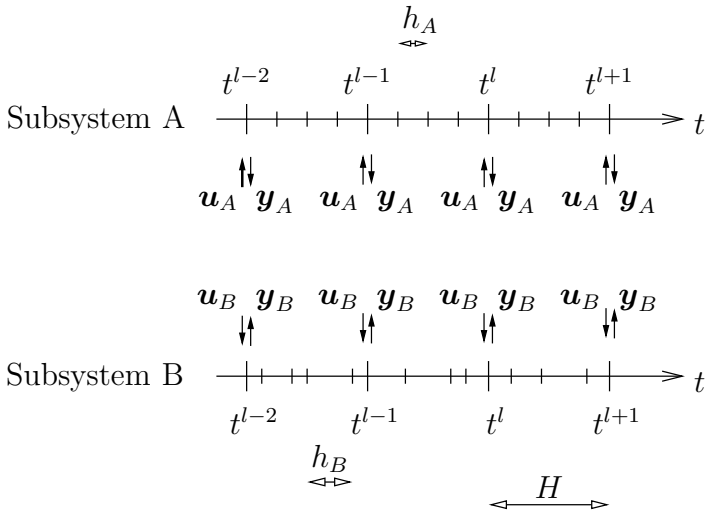


Figure 1.1: Dynamic coupling between subsystems

related to the BAUMGARTE stabilization [9] or force coupling methods in multi body simulations [43].

Macro Discretization

Since the mathematical formulation and integration of the subsystems is arbitrary, the communication between the subsystems is done only at discrete macro time steps t^l. Hence, besides the inner, micro discretization on subsystem integrator level also an outer, macro discretization needs to be imposed. At these points the subsystems stop their integration and must deal with the input and output vectors $\boldsymbol{u}_i$ and $\boldsymbol{y}_i$ as shown in Figure 1.1. Between one macro time step $t^l \rightarrow t^{l+1}$ with step size $H = t^{l+1} - t^l$ each subsystem integrates its own dynamic equations with the subsystem specific, specialized integrator which can be of fixed or variable micro step size h_i.

Extra- and Interpolation of the Input

The output values $\boldsymbol{y}_i$ are only available at the macro time steps t^l because the interconnection equations (1.5) are only evaluated at these discrete points in time. Hence, the input values $\boldsymbol{u}_i$ are also only updated at the macro time steps. The input values are however needed by the subsystem solver at each minor time step. Hence, a rule for the calculation of the input values at arbitrary times $t \in [t^l, t^{l+1}]$ must be defined depending on the known input values $\boldsymbol{u}_i^l = \boldsymbol{u}_i(t^l)$ at the discrete macro time steps t^l. This continuous input function is denoted by $\tilde{\boldsymbol{u}}_i$. As exemplary introduced for $\boldsymbol{u}_i^l$, the right side upper index denotes the discrete point of evaluation in time. Since the coupling quantities of all co-simulation examples in this work are scalar values, the calculation of the extrapolated subsystem input $\tilde{\boldsymbol{u}}$ is done per

element. Therefore in the following scalar coupling values for example $\tilde{u}$ or u are used exemplary but can directly be extended to vector-valued descriptions.

For the current macro step $t^l \rightarrow t^{l+1}$ it is always possible to use extrapolation, for example polynomial extrapolation of degree N, because the input value at the current time t^l and at the previous macro time steps $t^{l-k}, k = 1, 2, \ldots$ are known:

$$\tilde{u}_{\text{extrap.}} : \mathbb{R} \rightarrow \mathbb{R},\ t \mapsto \sum_{n=0}^{N} e_n \, (t - t^l)^n \tag{1.8}$$

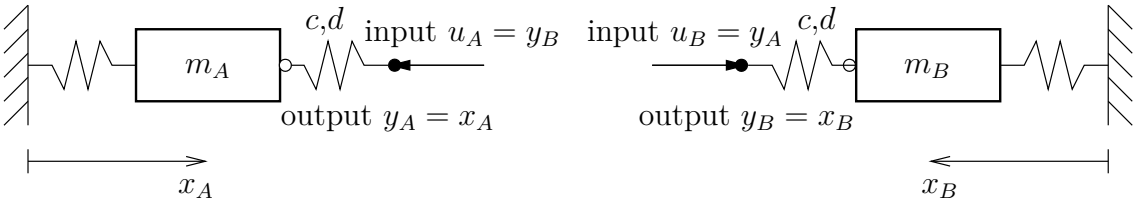
The calculation of the parameters e_n has to be done concerning good accuracy and stability properties and is shown later in detail. However, all extrapolation methods have the disadvantage that the input may comprise a point of discontinuity at the macro time steps: the extrapolated input value at the end of the current macro time step will not match the real input value at the beginning of the next macro time step. This can lead to negative effects on the integrators of the subsystems. To avoid this, [36] uses an *interpolated extrapolation* or an *extrapolated interpolation*. Both variants use only current and previous values in time but continuity is assured even at the macro time steps t^l. Another problem for polynomial based approaches, well known also from other numerical disciplines, is the induced instability especially for high order polynomials, called RUNGE's-phenomenon [60]. Therefore the advance of accuracy and the enlargement of the macro step size H is limited.

Interpolations between the time steps t^l and t^{l+1}, for example a linear interpolation

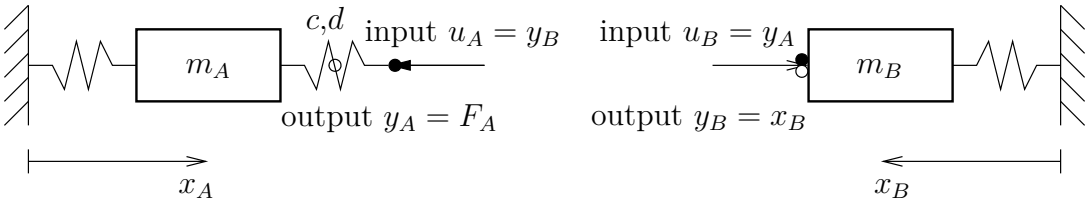
$$\tilde{u}_{\text{interp.}} : \mathbb{R} \rightarrow \mathbb{R},\ t \mapsto \frac{u^{l+1} - u^l}{H}(t - t^l) + u^l, \tag{1.9}$$

using the input values $u^l = u(t^l)$ and $u^{l+1} = u(t^{l+1})$ is another approach. Continuity even at the macro time steps is observed for this method: the interpolated input value at the end of the current macro time step equals the real input value at the beginning of the next macro time step. The main drawback is, that the value u^{l+1}, which lies in the future with respect to the current simulation interval $t \in [t^l, t^{l+1}]$, must be known. This value is commonly unknown, but if the subsystems are arranged in ahead and lagging ones, all lagging subsystems which only use inputs from ahead subsystems can interpolate their input [30]. This method however excludes the ahead subsystem from running in parallel with the lagging ones.

The evaluation of the extrapolation (1.8) or interpolation (1.9) function at the micro time step is done by the subsystem itself. Hence, the subsystems must know the type of the extra- or interpolation, all parameters and the current macro time t^l. The type does not change during the whole simulation and can be set at initialization. The macro time t^l is already known by the subsystem. Hence, the extra- or interpolation parameters, changing at every macro time step, are the input values of the subsystems.



(a) *Displacement-Displacement*



(b) *Force-Displacement*

● input action point ○ output action point

Figure 1.2: Mechanical Coupling Variants

Mechanical Coupling Variants

Mechanical subsystems coupled by a force element do not lead to an algebraic loop in the input variables because the output function $\boldsymbol{o}_i$ of at least one of two coupled subsystems is not depending on the input value $\boldsymbol{u}_i$. Figure 1.2 shows two mechanical coupling methods commonly used [15].

a) Using a so called *displacement-displacement* coupling, both subsystems, being connected by a force coupling element, export a displacement (position and/or velocity) as output variable. This value is used by the other subsystem as a kinematic input. To avoid an algebraic loop this kinematic excitation is not directly attached to a body, but to the interjacent force coupling element (for example a spring-damper). The coupling element therefore is included and evaluated in each subsystem separately. In this case the output functions $\boldsymbol{o}_i$ of both subsystems depend only on the subsystem state x_i but not on the individual input

$$y_i = x_i = o_i(\boldsymbol{z}_i). \tag{1.10}$$

b) Using the so called *force-displacement* coupling only subsystem B exports a displacement to subsystem A as described above. Subsystem A exports the force of the interjacent force coupling element (included and evaluated only in subsystem A) to subsystem B which uses this force as a kinetic excitation on the connected body. In this case only the output function o_A of subsystem A depends on the input but not the output function o_B. Considering a linear spring as coupling force element the output function o_A writes

$$y_A = F_A = c\,(x_A - x_B) = c\,(x_A - u_A) = o_A(\boldsymbol{z}_A, u_A). \tag{1.11}$$

Implementation

The implementation of a co-simulation can be split into two different approaches: a *solver coupling* and a *process coupling*.

Using a *solver coupling*, the submodel equations as well as the algorithm of the numerical integrator are exported by all subsystems but one. The remaining subsystem imports the other subsystem models and integrators as time discrete systems. The importing tool is also responsible for the connection of the input and output variables (1.5).

Using a *process coupling* each subsystem is integrated by its respective simulation tool and the communication defined by the input and output variables is left to an external process. The communication must be done using inter-process-communication (IPC) and the synchronization of the macro time progress between the subsystems is done distributed by the subsystems itself or by a global controlling process.

Advantages and Disadvantages

The general advantages of co-simulation are:

- The reuse of existing subsystems.
- The use of specialized solvers and simulation tools for each individual domain.
- No code import or export functionality is needed by the subsystems, if a *process coupling* is used.

The disadvantages may be:

- An additional macro discretization is needed, which induces additional errors.
- Numerical considerations are not as well established as for example for DAE or ODE systems.
- The subsystems need to implement an inter-process-communication and synchronization, if a *process coupling* is used.

The advantages of the iterative method compared to *filtering* or force coupling methods are:

- Arbitrary couplings between subsystems are feasible.
- Coupling constraints are exactly fulfilled, at least for a vanishing macro step size H.

Disadvantages may be:

- An increased computational effort due to the multiple call of the output functions or due to repeated macro integration steps of the subsystems.

- The subsystems must be able to reject integration steps by the request of an external process, if iterative time integration is used for the macro step. Most commercial simulation tools do not have such a functionality.

1.3 Objective and Outline

The main objective of this work is to introduce a co-simulation framework for multi domain systems. Special attention is given to the computational performance of this co-simulation on modern desktop or workstation computers. Such computers are nowadays usually equipped with more than one central processing unit (CPU). Even in the past multi processor mainboards did exist being able to take more than one CPU. However, up to now such boards are of high cost and only useful in workstation computers. Nevertheless, in the last years so called multi core CPUs have appeared even in the field of medium and low cost desktop computers. To be able to use the full computational power of such systems it is necessary to run simulations in parallel. One possibility to do this is to perform a multi domain simulation by a parallel co-simulation, being the main topic of this work.
Besides the parallelization the second main topic is the usability of the parallel co-simulation framework. It should be possible to include a large number of preexisting simulation tools of arbitrary domains in the co-simulation. The code of open source systems can be changed to meet the requirements of the co-simulation framework. For systems available only as closed source code, which is true for most commercial systems, the implementation cannot be adjusted. Hence, the introduced parallel co-simulation should only put minimal demands on the subsystems, which gives the possibility to apply it to a large number of preexisting closed source simulation tools.

In Chapter 2 a parallel co-simulation framework is introduced. Starting with the basic concept of this framework and its classification, differences are outlined compared to co-simulations known from literature. A special focus is put on the mathematical analysis of mechanical couplings concerning the local order and stability of the coupling. To achieve maximum stability, various coupling extrapolations are introduced. Some of these are numerically optimized for stability. Moreover the long time behavior of different couplings is analyzed and compared with other coupling methods known from literature.
The concept of mechanical couplings with good stability properties are transferred to hydraulic couplings as well as to control couplings. Moreover an overview is given for couplings of other domains.

Chapter 3 gives an outline on the implementation of the parallel co-simulation using the programming language C++. Besides the investigation of different inter-process-communication methods, which have a high influence on the parallel performance of the co-simulation, also the implementation of the master and the slaves are discussed. Detailed information on the implementation of open source slaves as well as the binding of commercial closed source slaves is given.

Chapter 4 starts with an academic test example to analyze the influence of different inter-process-communication methods. Using a timing chain drive the different introduced coupling extrapolations are compared concerning simulation speedup. The timing chain drive of a flat combustion engine with hydraulic chain tensioners shows an excellent speedup due to it's multi-scale character which is resolved by co-simulation. The last example gives an overview of the power of parallel co-simulations for large industrial systems: efficient simulation of a timing chain drive with a hydraulic chain tensioner including an intake and exhaust valve train with hydraulic cam phasing systems of a V6 combustion engine.

Chapter 5 gives a short overview of a further parallelization technique in simulation software: Parallelization of the algorithm. Exemplary two simulation tools are used to show the basic concept and the possibility of this method using parallelization by OpenMP and a comparison of this method with co-simulation.

Chapter 6 sums up and draws some conclusions. Finally, future potentials of parallel co-simulations are judged.

2 A Parallel Co-Simulation Framework

In the following chapter the principle method for a coupled simulation is shown in detail. In contrast to other common coupling methods for multi domain simulations as shown in Section 1.2, the main objective of this coupling variant is the possibility to run the simulation in parallel on a multi computer cluster or on multi-core machines. Moreover, it should be possible to couple subsystems which are integrated by their own integrators forcing co-simulation as preferred method. Concerning parallelization, both *solver coupling* and *process coupling* is usable. In case of a *solver coupling* the imported solvers must be parallelized internally by the importing subsystem. Even if the importing subsystem provides such a functionality, the parallelization can only run on shared memory systems because the coupled system is acting as a single process which usually cannot be distributed to different computers. Using a *process coupling* neither the limitation to shared memory systems is given nor the availability of parallelization of imported solvers is needed. Hence, a *process coupling*, using inter-process-communication for the data exchange, is used in the presented framework. Concerning parallelization a further limitation is implied: all input is limited to extrapolation methods. A common concept implying all these aspects is the master/slave concept where the subsystems act as slaves and the master controls the overall progress especially the time synchronization of the subsystems.

2.1 Master-Slave Concept

The master-slave concept is shown in Figure 2.1 in form of a block diagram. The subsystems, acting as slaves and as subsystem integrators on micro step level, are arranged in parallel. The master, arranged serial to the slaves, is responsible for the dynamic coupling of the slaves, the synchronization of the slaves, the evaluation of the physical coupling law $\boldsymbol{o}_M$, the extrapolation $\tilde{\boldsymbol{u}}_i$ and the interconnection (1.5). The advantages of the master-slave concept are:

- The main computational cost, the integration of the subsystems (slaves), can be performed in parallel.
- New coupling laws or extrapolation methods can be added to the master without a change of already existing subsystems (slaves).
- The solvers of all subsystems are treated equally and are kept time synchronous by the master.

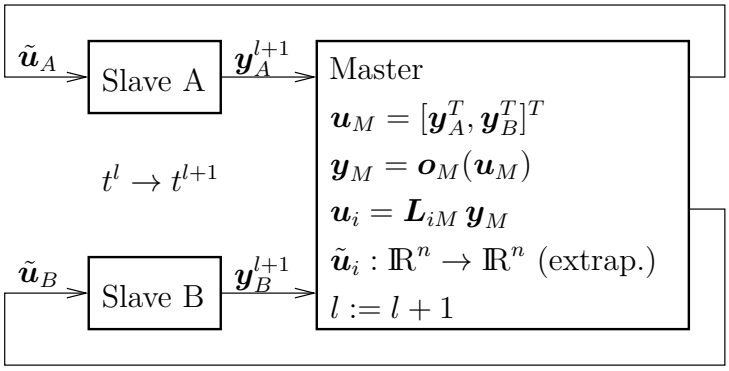


Figure 2.1: Master-slave concept

- Each subsystem is only connected to the master. This means that they do not have to know each other.

Disadvantages may be:

- Only extrapolation methods for the input can be used.
- Couplings between the slaves are restricted to the functionality of the master.

2.1.1 Classification Regarding Input, Output and Interconnection

The master-slave concept can be formulated using equations (1.4) and (1.5). The output (1.4b) of all $i = A, B, C, \ldots, N$ subsystems are assumed to depend only on the state $\boldsymbol{z}_i$ of the respective subsystem i

$$\boldsymbol{y}_i = \boldsymbol{o}_i(\boldsymbol{z}_i). \tag{2.1}$$

The master M is implemented as a separate subsystem without any dynamic state variable:

$$\boldsymbol{z}_M, \boldsymbol{s}_M \in \mathbb{R}^0 \text{ (stateless)}, \tag{2.2a}$$
$$\boldsymbol{y}_M = \boldsymbol{o}_M(\boldsymbol{u}_M). \tag{2.2b}$$

The output function $\boldsymbol{o}_M$ of the master represents the physical coupling law for each connection between subsystems. The interconnection equations (1.5) are defined by

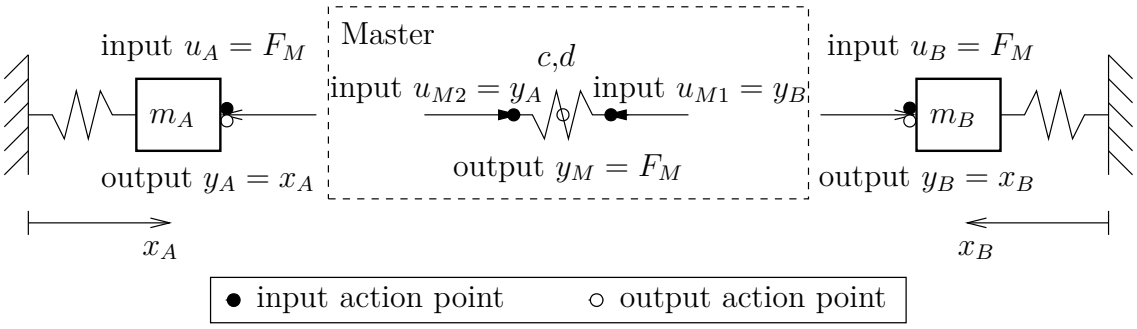


Figure 2.2: Mechanical Coupling: Master-Slave; *Force-Force*

$$\boldsymbol{u}_i = \boldsymbol{L}_i\, \boldsymbol{y}_i^* = \underbrace{[\boldsymbol{0}, \ldots, \boldsymbol{0}, \boldsymbol{0}, \ldots, \boldsymbol{0}, \boldsymbol{L}_{iM}]}_{\boldsymbol{L}_i} \begin{bmatrix} \boldsymbol{y}_A \\ \vdots \\ \boldsymbol{y}_{i-1} \\ \boldsymbol{y}_{i+1} \\ \vdots \\ \boldsymbol{y}_N \\ \boldsymbol{y}_M \end{bmatrix} = \boldsymbol{L}_{iM}\, \boldsymbol{y}_M, \tag{2.3a}$$

$$\boldsymbol{u}_M = \boldsymbol{L}_M\, \boldsymbol{y}_M^* = \underbrace{\boldsymbol{E}}_{\boldsymbol{L}_M} \begin{bmatrix} \boldsymbol{y}_A \\ \vdots \\ \boldsymbol{y}_N \end{bmatrix} = \begin{bmatrix} \boldsymbol{y}_A \\ \vdots \\ \boldsymbol{y}_N \end{bmatrix}. \tag{2.3b}$$

Since the slaves are not directly interconnected with each other (see equation (2.3a)) and the output function $\boldsymbol{o}_M$ (2.2b) of the master M depends on the input $\boldsymbol{u}_M$ but the output functions (2.1) of the slaves do not depend on their input, no algebraic loop in the input variables exists.

2.1.2 Classification of Mechanical Couplings

The presented master-slave concept for mechanical subsystem does not match the *displacement-displacement* or the *force-displacement* coupling shown in Section 1.2.2, Figure 1.2. However, this concept can be interpreted as a *force-force* coupling as shown in Figure 2.2. Both subsystems export a displacement to the master which is calculating the interjacent force coupling element o_M. The master exports the coupling force to both subsystems which use this force as a kinetic excitation on the connected body. In contrast to the *displacement-displacement* and the *force-displacement* coupling the calculation of the force coupling element is not done by any of the subsystems, but by the master. To give an example, the output functions o_i of the subsystems and the master are presented for a linear spring as

force coupling element

$$y_i = x_i = o_i(\boldsymbol{z}_i), \tag{2.4a}$$
$$y_M = F_M = c\,(x_A - x_B) = c\,(y_A - y_B) = o_M(\boldsymbol{u}_M). \tag{2.4b}$$

2.2 Calculation of the Subsystem Input

The master-slave concept uses extrapolation for the input of all subsystems. Therefore extrapolation parameters, like e_n of equation (1.8) for polynomial extrapolations are the input of the subsystems. According to the interconnection equation (2.3a) these inputs are the output of the master M. Hence, the master is not only responsible for the calculation of the coupling law $\boldsymbol{o}_M$ (2.2b) but also for the calculation of the extrapolation function $\tilde{u}$ (1.8).

This section discusses five extrapolation methods: besides the commonly known extrapolation using polynomials and the HERMITE interpolation, three further extrapolation variants are shown which are useful to optimize the accuracy and stability of the co-simulation coupling. To provide quick cross references to these extrapolations a coded name for each extrapolation is given in brackets. Moreover, for simplification, the notation is done using scalar coupling quantities without loosing generality.

2.2.1 Extrapolation Using Polynomial Interpolation (`Poly` K)

A well known and very common way for extrapolation is the polynomial interpolation of degree $K-1$ at K supporting points

$$\left(t^{l-k}, u(t^{l-k})\right), \quad k = 0 \ldots K-1 \tag{2.5}$$

for the extrapolation of u in the interval $[t^l, t^{l+1}]$ outside of the supporting points. The parameters e_n of the extrapolation function (1.8) are the solution of the equations

$$\sum_{n=0}^{K-1} e_n\,(t^{l-k} - t^l)^n = u(t^{l-k}), \quad k = 0 \ldots K-1. \tag{2.6}$$

For practical applications the parameters can be calculated easily using for example LAGRANGE-polynomials [14]. According the output function (2.2b) of the master and the coupling equations (2.3), the subsystem input u depends only on values known by the master

$$u = y_M = o_M(\boldsymbol{u}_M) = o_M(\boldsymbol{y}_A, \boldsymbol{y}_B, \ldots, \boldsymbol{y}_N). \tag{2.7}$$

Hence the master is able to calculate the extrapolation parameters e_n without knowing the slaves in detail.

The order of the extrapolation increases with the degree of the polynomial interpolation. Hence the accuracy can be increased using more supporting points. However, a problem known for polynomial interpolations are high oscillations between the supporting points for increasing degrees $K-1$. For the co-simulation this leads to stability problems which are discussed in Section 2.3.2. A similar effect is known from multi-step integration methods [26]: the order increases by the use of more steps, but the stability area of the integration method decreases.

2.2.2 Extrapolation Using HERMITE-Interpolation (`Hermite` K)

Not only using the function values at the K supporting points but also the derivatives of the function values for the polynomial interpolation is known as HERMITE-interpolation [14]. The parameters e_n of the extrapolation function $\tilde{u}$ are the solution of the following equations:

$$\sum_{n=0}^{2K-1} e_n \, (t^{l-k} - t^l)^n = u(t^{l-k}), \tag{2.8a}$$

$$\sum_{n=1}^{2K-1} n \, e_n \, (t^{l-k} - t^l)^{n-1} = \dot{u}(t^{l-k}), \quad k = 0 \ldots K-1. \tag{2.8b}$$

Compared to the polynomial extrapolation, the degree of $\tilde{u}$ is doubled using the same number of supporting points. As well as for the polynomial extrapolation, the stability deteriorates with an increasing number of supporting points.

This approach needs besides the value u (2.7) also its derivative

$$\dot{u} = \frac{\partial o_M(\boldsymbol{y}_A, \ldots, \boldsymbol{y}_N)}{\partial \boldsymbol{y}_A} \dot{\boldsymbol{y}}_A + \cdots + \frac{\partial o_M(\boldsymbol{y}_A, \ldots, \boldsymbol{y}_N)}{\partial \boldsymbol{y}_N} \dot{\boldsymbol{y}}_N \tag{2.9}$$

at the macro time steps t^l. The partial derivative of o_M with respect to $\boldsymbol{y}_i$ is known by the master, since the coupling law o_M is a quantity of the master. However, the derivative of the subsystem output $\boldsymbol{y}_i$ is unknown unless the subsystems include for each output $\boldsymbol{y}_i$ also its derivative in the output vector. This is only feasible if these derivatives are a state variable of the subsystem. Otherwise this will lead to an algebraic loop in the input variables. For mechanical couplings, using only the position x for the calculation of the coupling force, this is feasible since the velocity $v = \dot{x}$ is also a state variable. Including more than the first derivative of u by the HERMITE-interpolation will lead, for the mechanical as well as for most other domains, to an algebraic loop in the input, and is therefore not considered.

2.2.3 Constant Extrapolation Using a Linear Combination (`Const` K, R)

The methods previously described have the advantage that the degree of the extrapolation, and thereby the accuracy, can be controlled by the number K of supporting points, but there is no possibility to affect the stability of the methods directly, since there are no free parameters. This disadvantage is dealt with by a further approach: a special constant extrapolation function (1.8) for $N = 0$. This approach does not use u^l for the single parameter e_0 of equation (2.6) but a linear combination of the current and the $K-1$ previous values of u and $\dot{u}$

$$\tilde{u} = e_0 = \sum_{k=0}^{K-1} (a_k\, u^{l-k} + b_k\, \dot{u}^{l-k}\, H). \tag{2.10}$$

Note that the derivatives $\dot{u}$ are scaled by the macro step size $H = t^{l+1} - t^l$ which is assumed to be constant in this approach. For the same reason as for the HERMITE-interpolation, to avoid an algebraic loop in the input variables, higher derivatives of u must not be used.

Approximation order

The parameters a_k and b_k must fulfill some conditions for accuracy and good stability properties. To derive the conditions for accuracy a mechanical subsystem is considered. Its differential equations (1.4) are

$$\dot{\boldsymbol{q}} = \boldsymbol{T}\boldsymbol{v}, \tag{2.11a}$$
$$\boldsymbol{M}\dot{\boldsymbol{v}} = \boldsymbol{h} + \boldsymbol{B}u, \tag{2.11b}$$
$$\boldsymbol{y} = \boldsymbol{o}(\boldsymbol{z}), \tag{2.11c}$$

where $\boldsymbol{q}$ and $\boldsymbol{v}$ are the generalized positions and velocities, forming the state vector $\boldsymbol{z} = [\boldsymbol{q}^T, \boldsymbol{v}^T]^T$, $\boldsymbol{M}$ is the symmetric, positive definite mass matrix, $\boldsymbol{h}$ the force vector of the right hand side including gravitational, gyroscopic and other terms and $\boldsymbol{B}$ is the matrix projecting the co-simulation input u, being a force for mechanical systems, to the generalized velocities $\boldsymbol{v}$. For simplification only a scalar input u is used. The matrix $\boldsymbol{T}$ equals unity for most parametrizations except for some special parametrizations like quaternions. The equations (2.11) can also be given in an integral form over one macro time step:

$$\boldsymbol{q}^{l+1} = \boldsymbol{q}^l + \int_{t^l}^{t^{l+1}} \boldsymbol{T}\boldsymbol{v}dt, \tag{2.12a}$$
$$\boldsymbol{v}^{l+1} = \boldsymbol{v}^l + \int_{t^l}^{t^{l+1}} \boldsymbol{M}^{-1}\boldsymbol{h}dt + \int_{t^l}^{t^{l+1}} \boldsymbol{M}^{-1}\boldsymbol{B}udt, \tag{2.12b}$$
$$\boldsymbol{y}^{l+1} = \boldsymbol{o}(\boldsymbol{z}^{l+1}). \tag{2.12c}$$

The integrals in these equations do not depict an exact mathematical integration, but the numerical integrator of the subsystem. These equations show that the co-simulation input force u is always integrated by the subsystem leading to a corresponding velocity part in $\boldsymbol{v}$. To derive conditions for accuracy, it is assumed that for each single coupling the integral of the extrapolated input $\tilde{u}$ is equal to the integral of the input u up to a given order P concerning the macro step size H:

$$\int_{t^l}^{t^{l+1}} \boldsymbol{M}^{-1}\boldsymbol{B}u dt = \int_{t^l}^{t^{l+1}} \boldsymbol{M}^{-1}\boldsymbol{B}\underbrace{e_0}_{\tilde{u}}\, dt + \mathcal{O}(H^{P+1}) \tag{2.13}$$

The local order of the state of the subsystem $\boldsymbol{z}$ might not be of order P, since the constant extrapolation during a macro time step additionally influences the subsystem integration on micro step level. Hence a local order analysis for the subsystem state depending on the order P and concerning the extrapolation degree N is shown in Section 2.3.1.

TAYLOR expansion of the exact integral: To derive conditions for the parameters a_k and b_k a TAYLOR expansion of equation (2.13) is derived. The TAYLOR expansion of the exact integral (left hand side of equation (2.13)) yields

$$\int_{t^l}^{t^l+H} \boldsymbol{C}u dt = \int_{t^l}^{t^l+H} \left(\sum_{n=0}^{\infty} \tfrac{1}{n!} \left(\boldsymbol{C}^{(n)}\right)^l \left(t-t^l\right)^n\right) \left(\sum_{n=0}^{\infty} \tfrac{1}{n!} \left(u^{(n)}\right)^l \left(t-t^l\right)^n\right) dt \tag{2.14}$$

with the generalized position dependent term $\boldsymbol{M}^{-1}(\boldsymbol{q}(t))\,\boldsymbol{B}(\boldsymbol{q}(t))$ substituted by $\boldsymbol{C}(t)$. The upper right index written in brackets denotes the n^{th} derivative of a variable. Expanding the sum using the product of power series and integrating leads to a sorted form in the powers of H

$$\int_{t^l}^{t^l+H} \boldsymbol{C}u dt = \sum_{n=0}^{\infty}\sum_{m=0}^{n} \frac{1}{n+1}\frac{1}{m!}\frac{1}{(n-m)!} \left(\boldsymbol{C}^{(m)}\, u^{(n-m)}\right)^l H^{n+1}. \tag{2.15}$$

TAYLOR expansion of the constant extrapolation: The same is done for the constant extrapolation on the right hand side of equation (2.13). The constant value e_0 is substituted by the linear combination (2.10) and shifted in front of the integral

$$\int_{t^l}^{t^l+H} \boldsymbol{C}e_0\, dt = \sum_{k=0}^{K-1} \left(a_k\, u^{l-k} + b_k\, \dot{u}^{l-k} H\right) \int_{t^l}^{t^l+H} \boldsymbol{C} dt. \tag{2.16}$$

The values of u and $\dot{u}$ at the times $t^{l-k}, k = 0 \dots K-1$ are substituted by a TAYLOR series around t^l:

$$u^{l-k} = \sum_{n=0}^{\infty} \tfrac{1}{n!} \left(u^{(n)}\right)^l \, (-kH)^n, \tag{2.17}$$

$$\dot{u}^{l-k} = \sum_{n=0}^{\infty} \tfrac{1}{n!} \left(u^{(n+1)}\right)^l \, (-kH)^n. \tag{2.18}$$

Inserting the above equations in equation (2.16) and evaluating the integral yields

$$\begin{aligned} \int_{t^l}^{t^l+H} \boldsymbol{C} e_0 \, dt = &\left[\sum_{k=0}^{K-1} \left(a_k \sum_{n=0}^{\infty} \tfrac{1}{n!} \left(u^{(n)}\right)^l \, (-kH)^n + b_k \sum_{n=0}^{\infty} \tfrac{1}{n!} \left(u^{(n+1)}\right)^l \, (-kH)^n H \right)\right] \cdot \\ &\cdot \left(\sum_{n=0}^{\infty} \tfrac{1}{(n+1)!} \left(\boldsymbol{C}^{(n)}\right)^l H^{n+1} \right). \end{aligned} \tag{2.19}$$

Exchanging the sums in the first multiplier leads to

$$\begin{aligned} \int_{t^l}^{t^l+H} \boldsymbol{C} e_0 \, dt = &\sum_{n=0}^{\infty} \left(\tfrac{1}{n!} \left(u^{(n)}\right)^l H^n \sum_{k=0}^{K-1} (-k)^n a_k + \tfrac{1}{n!} \left(u^{(n+1)}\right)^l H^{n+1} \sum_{k=0}^{K-1} (-k)^n b_k \right) \cdot \\ &\cdot \left(\sum_{n=0}^{\infty} \tfrac{1}{(n+1)!} \left(\boldsymbol{C}^{(n)}\right)^l H^{n+1} \right). \end{aligned} \tag{2.20}$$

Expanding the outer sums using the product of power series leads to a sorted form in the powers of H

$$\begin{aligned} \int_{t^l}^{t^l+H} \boldsymbol{C} e_0 \, dt = &\sum_{n=0}^{\infty} \sum_{m=0}^{n} \frac{1}{m!} \frac{1}{(n-m+1)!} \left(\boldsymbol{C}^{(n-m)} \, u^{(m)}\right)^l \sum_{k=0}^{K-1} (-k)^m a_k \, H^{n+1} + \\ &\sum_{n=0}^{\infty} \sum_{m=0}^{n} \frac{1}{m!} \frac{1}{(n-m+1)!} \left(\boldsymbol{C}^{(n-m)} \, u^{(m+1)}\right)^l \sum_{k=0}^{K-1} (-k)^m b_k \, H^{n+2}. \end{aligned} \tag{2.21}$$

Comparison of Coefficients: To achieve order P in equation (2.13), the first P coefficients in the powers of H of the equations (2.15) and (2.21) must be equal. For order 1 this leads to the condition

$$\boldsymbol{C}^l u^l = \boldsymbol{C}^l u^l \sum_{k=0}^{K-1} a_k \quad \Leftrightarrow \quad 1 = \sum_{k=0}^{K-1} a_k \tag{2.22}$$

for the parameters a_k, where the parameters b_k are not constrained. For order 2 additionally

$$\tfrac{1}{2}(\boldsymbol{C}^l\ddot{u}^l + \dot{\boldsymbol{C}}^l u^l) = \tfrac{1}{2}\left(2\boldsymbol{C}^l\dot{u}^l\sum_{k=0}^{K-1}(b_k - ka_k) + \dot{\boldsymbol{C}}^l u^l\sum_{k=0}^{K-1}a_k\right) \tag{2.23}$$

has to be satisfied. Using equation (2.22) this can be simplified to

$$\tfrac{1}{2} = \sum_{k=0}^{K-1}(b_k - ka_k). \tag{2.24}$$

For order 3 furthermore

$$\begin{aligned}&\tfrac{1}{6}(\boldsymbol{C}^l\dddot{u}^l + 2\dot{\boldsymbol{C}}^l\ddot{u}^l + \ddot{\boldsymbol{C}}^l u^l) = \\ &\quad = \tfrac{1}{6}\left(6\boldsymbol{C}^l\ddot{u}^l\sum_{k=0}^{K-1}k\,(\tfrac{1}{2}ka_k - b_k) + 3\dot{\boldsymbol{C}}^l\dot{u}^l\sum_{k=0}^{K-1}(b_k - ka_k) + \ddot{\boldsymbol{C}}^l u^l\sum_{k=0}^{K-1}a_k\right)\end{aligned} \tag{2.25}$$

must be fulfilled which can be reduced considering (2.22) and (2.23):

$$\boldsymbol{C}^l\dddot{u}^l + 2\dot{\boldsymbol{C}}^l\ddot{u}^l = 6\boldsymbol{C}^l\ddot{u}^l\sum_{k=0}^{K-1}k\,(\tfrac{1}{2}ka_k - b_k) + \tfrac{3}{2}\dot{\boldsymbol{C}}^l\ddot{u}^l \tag{2.26}$$

In this equation, and equally in the equations for even higher orders, the unknowns $\boldsymbol{C}^l$ and u^l and their derivatives cannot be eliminated, due to the non-matching factor 2 on the left hand side and $\frac{3}{2}$ on the right for the term $\dot{\boldsymbol{C}}^l\ddot{u}^l$. This restricts the maximal order of this approximation to 2. However, with the assumption that $\boldsymbol{C}$ is constant, all derivatives of $\boldsymbol{C}$ vanish. In this case and in combination with equations (2.22) and (2.24), equation (2.26) leads to a constraint for the parameters a_k and b_k

$$\tfrac{1}{6} = \sum_{k=0}^{K-1}k\,(\tfrac{1}{2}ka_k - b_k) \tag{2.27}$$

forcing order 3.

Summarizing the conditions shows that for a constant $\boldsymbol{C}$ the order P for the constant extrapolation can be achieved if the parameters a_k and b_k fulfill the first $P \leq 2K$ constraints

$$1 = \sum_{k=0}^{K-1}a_k \tag{2.28a}$$

$$\tfrac{1}{2} = \sum_{k=0}^{K-1}(-ka_k + b_k) \tag{2.28b}$$

$$\tfrac{1}{6} = \sum_{k=0}^{K-1}k\,(\tfrac{1}{2}ka_k - b_k) \tag{2.28c}$$

Fulfilled constraints R (2.28)	1	2	3	4	...
$\boldsymbol{C} \neq$ const.	1	2	2	2	...
$\boldsymbol{C} =$ const.	1	2	3	4	...

Table 2.1: Maximal approximation order P of $\int \boldsymbol{C} u dt$ for a constant extrapolation

$$\frac{1}{24} = \sum_{k=0}^{K-1} k^2(-\tfrac{1}{6}ka_k + \tfrac{1}{2}b_k) \tag{2.28d}$$
$$\ldots = \ldots.$$

If $\boldsymbol{C}$ is not constant the maximal approximation order is limited to 2, independent of the number R of constraints being fulfilled. Table 2.1 summarizes these statements.

Since the total number of parameters is $2K$, forcing order $P \leq 2K$ will lead to $2K - P$ free parameters which can be chosen arbitrarily without changing the order of the approximation. These free parameters are used for stability improvements in Section 2.3.2.

Remark: An intuitive motivation of this approach

The value e_0 for the constant extrapolation (1.8) can be chosen as the integral mean value of u in the interval of the current macro time step $[t^l, t^{l+1}]$

$$e_0 = \frac{1}{H} \int_{t^l}^{t^l+H} u dt. \tag{2.29}$$

Since the current macro simulation time is t^l, u is not known in this interval. However, to give an approximation u can be extrapolated and used for the integral mean value e_0. Figure 2.3 shows this approach exemplarily for a linear TAYLOR extrapolation of u using the values of u and $\dot{u}$ at the current macro time step.

Evaluating this approach with the different types of extrapolations for u and numbers of supporting points gives the following values for $\tilde{u} = e_0$ (the extended `b=0` at the coded names depicts that all parameters b_k are forced to be 0 for these couplings):

- TAYLOR extrapolation around t^l:

$$N = 0 \text{ (`Const 1,1,b=0`)}: \; e_0 = u^l \tag{2.30a}$$
$$N = 1 \text{ (`Const 1,2`)}: \qquad e_0 = u^l + \tfrac{1}{2}\dot{u}^l H \tag{2.30b}$$
$$N \geq 2: \text{ not applicable for most domains since } \ddot{u} \text{ is needed} \tag{2.30c}$$

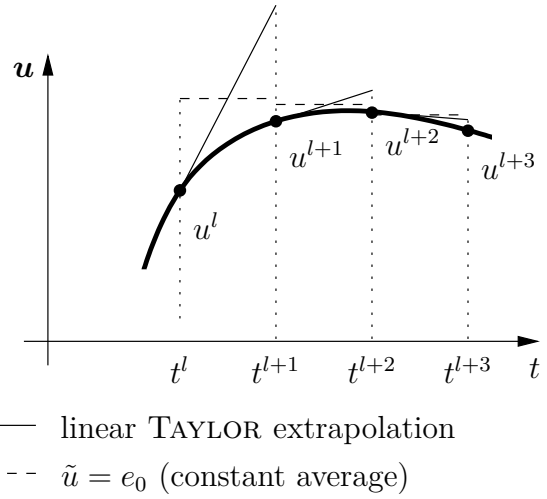


Figure 2.3: Constant extrapolation

- Extrapolation using a polynomial interpolation at K supporting points as shown is Section 2.2.1:

$$K = 1 \text{ (Const 1,1,b=0)}: \; e_0 = u^l \tag{2.31a}$$
$$K = 2 \text{ (Const 2,2,b=0)}: \; e_0 = \tfrac{1}{2}(3u^l - u^{l-1}) \tag{2.31b}$$
$$K = 3 \text{ (Const 3,3,b=0)}: \; e_0 = \tfrac{1}{12}(23u^l - 16u^{l-1} + 5u^{l-2}) \tag{2.31c}$$
$$\ldots$$

- Extrapolation using a HERMITE-interpolation at K supporting points as shown in Section 2.2.2:

$$K = 1 \text{ (Const 1,2)}: \; e_0 = u^l + \tfrac{1}{2}\dot{u}^l H \tag{2.32a}$$
$$K = 2 \text{ (Const 2,4)}: \; e_0 = \tfrac{1}{12}\left[-6u^l + 18u^{l-1} + \left(17\dot{u}^l + 7\dot{u}^{l-1}\right) H\right] \tag{2.32b}$$
$$K = 3 \text{ (Const 3,6)}: \; e_0 = \tfrac{1}{240}\left[-949u^l + 608u^{l-1} + 581u^{l-2} + \left(637\dot{u}^l + 1080\dot{u}^{l-1} + 173\dot{u}^{l-2}\right) H\right] \tag{2.32c}$$
$$\ldots$$

Equations (2.30a) to (2.32c) all match the linear combination (2.10). Hence, equation (2.10) is a general formulation including all extrapolations (2.30a) to (2.32c).

2.2.4 Linear Extrapolation Using a Linear Combination (Lin K,R)

Since the last approximation shows good results with respect to stability (see Section 2.3.2) as well as by the practical examples from Chapter 4 an extension of this

Fulfilled constraints R (2.28)	1	2	3	4	...
$\boldsymbol{C} \neq$ const.	1	2	3	3	...
$\boldsymbol{C} =$ const.	1	2	3	4	...

Table 2.2: Approximation order P of $\int \boldsymbol{C} u dt$ for a linear extrapolation

approximation to a linear extrapolation $\tilde{u}$ is considered. The aim of a linear extrapolation is mainly motivated by a higher approximation order. Point of departure is the linear extrapolation (1.8) with $N = 1$

$$\tilde{u} = e_0 + e_1 \, (t - t^l). \tag{2.33}$$

The conditions for the unknown parameters e_0 and e_1 are motivated intuitionally:

1. The linear extrapolation $\tilde{u}$ at time t^l should be equal to $u(t^l)$:

$$\tilde{u}(t^l) = u(t^l) \quad \Leftrightarrow \quad e_0 = u^l. \tag{2.34}$$

2. Under the assumption that $\boldsymbol{C}$ is constant, the integral of the constant extrapolation using a linear combination (2.10) should be equal to the integral of the linear extrapolation (2.33):

$$\int_{t^l}^{t^{l+1}} \sum_{k=0}^{K-1} (a_k \, u^{l-k} + b_k \, \dot{u}^{l-k} H) dt = \int_{t^l}^{t^{l+1}} \underbrace{\left[e_0 + e_1 \, (t - t^l) \right]}_{\tilde{u}} dt$$

$$\Leftrightarrow \quad e_1 = \frac{2}{H} \left(\sum_{k=0}^{K-1} (a_k \, u^{l-k} + b_k \, \dot{u}^{l-k} H) - u^l \right) \tag{2.35}$$

Approximation order

The order analysis of this approximation is done in the same way as for the constant extrapolation and leads to identical constraints (2.28) for the parameters a_k and b_k. In contrast to the constant extrapolation, the maximal achievable order P for $\boldsymbol{C} \neq 0$ is 3 because of the higher extrapolation degree ($N = 1$) of the input $\tilde{u}$. Table 2.2 summarizes the achieved order P, dependent on the number R of fulfilled constraints for a constant and not constant $\boldsymbol{C}$.

Remark: An intuitive motivation of this approach

The motivation of this approach is analog to the previous one. Figure 2.4 exemplarily depicts this approach for parameters a_k and b_k representing a linear TAYLOR extrapolation for u. The extrapolated input $\tilde{u}$ is equal to the linear TAYLOR extrapolation of u because the order of the TAYLOR extrapolation is equal to the degree

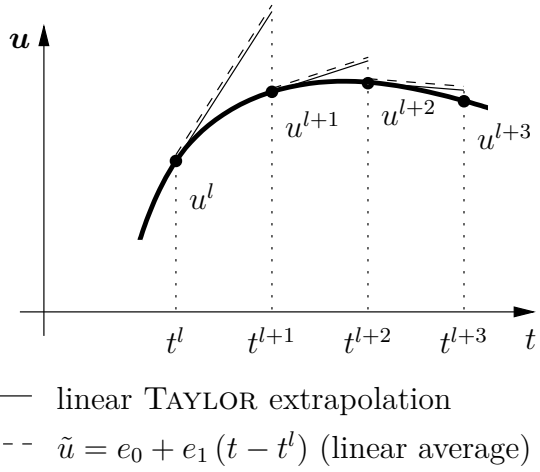


Figure 2.4: Linear extrapolation

of the linear extrapolated input $\tilde{u}$. For higher order extrapolations of u the linear extrapolation $\tilde{u}$ will form a linear average.

Remark: Forcing continuity at the macro time steps

Using a linear extrapolation, it is possible to force continuity even at the macro time steps t^l. Therefore the first condition for the unknown parameters e_0 and e_1 (2.34) is replaced by

$$\tilde{u}(t^l) = \tilde{u}^*(t^l) \tag{2.36}$$

whereas $\tilde{u}^*$ represents the linear extrapolation of the previous macro time step. The second condition (2.35) stays the same. This method is closely related to the *extrapolated interpolation* [36]. However, stability analysis and numerical tests have shown, that this approach has a bad stability character. This effect can already be divined in Figure 2.5 by the alternating slope of the extrapolation.

2.2.5 Quadratic Extrapolation Using a Linear Combination (`Quad` K,R)

The intuitive extension of the last two approaches is a quadratic extrapolation

$$\tilde{u} = e_0 + e_1 (t - t^l) + e_2 (t - t^l)^2 \tag{2.37}$$

with the conditions

$$\tilde{u}(t^l) = u(t^l) \tag{2.38a}$$
$$\dot{\tilde{u}}(t^l) = \dot{u}(t^l) \tag{2.38b}$$

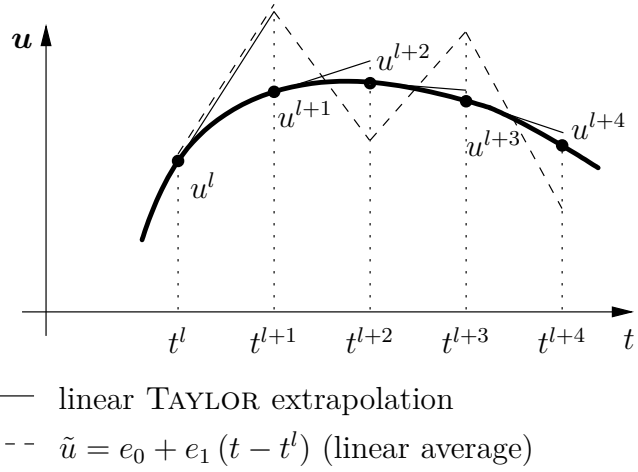


Figure 2.5: Linear extrapolation with continuity at the macro time step

$$\int_{t^l}^{t^{l+1}} \sum_{k=0}^{K-1} (a_k\, u^{l-k} + b_k\, \dot{u}^{l-k} H) dt = \int_{t^l}^{t^{l+1}} \underbrace{\left[e_0 + e_1\,(t - t^l) + e_2\,(t - t^l)^2\right]}_{\tilde{u}} dt \qquad (2.38c)$$

for the unknown parameters e_0, e_1 and e_2. This method is, like the other ones, used later in the local order and stability analysis as well as in the examples in Chapter 4, but is not considered in more detail here, since the stability analysis of this approach shows bad results compared to the other ones.

2.3 Analysis of Mechanical Couplings

The extrapolations introduced for the subsystem input $\tilde{u}$ are analyzed concerning the local order, the stability and the long time behavior. Exemplary mechanical systems are examined due to the following reasons: From the practical point of view, mechanical couplings are chosen since they represent the main point of interest for the industrial examples in Chapter 4. From the theoretical point of view they are very interesting concerning stability. This originates from the very stiff character of typical mechanical couplings: Many mechanical co-simulation couplings represent a constraint between two connected bodies. To prevent an algebraic loop in the input these couplings must be reformulated using a force coupling, being one of the *filtering* methods, see Section 1.2.2. In multi body dynamics this is called *regularization* and leads to high stiffnesses, which can lead to stability problems for time integration.

2.3.1 Local Order

A typical mathematical formulation for a dynamic mechanical subsystem is given by equations (2.11). The analysis of the local order of the co-simulation is based on the assumption that the numerical integration of all subsystems is exact. This allows

the investigation of the error induced only by the co-simulation coupling. From the practical point of view, neglecting the error of the subsystem integrators is feasible, since the integrator step sizes of the subsystems are much smaller than the macro step size of the co-simulation. Moreover the order of typical subsystem integrators for mechanical systems are high and therefore they provide good accuracy.

The local order analysis is done using a comparison of the TAYLOR expansion of the exact solution of the coupled system with the TAYLOR expansion of one macro time step using the constant or linear extrapolation from Sections 2.2.3 and 2.2.4. Since the TAYLOR expansions are getting very complex even if only a few terms are considered, only the principle way of the local order analysis and a final comparison is given in the following.

TAYLOR Expansion of the Exact Solution

The TAYLOR-expansion of the state at time $t^{l+1} = t^l + H$ around t^l is given by

$$\boldsymbol{q}(t^l + H) = \boldsymbol{q}^{l+1} = \boldsymbol{q}^l + \dot{\boldsymbol{q}}^l H + \tfrac{1}{2}\ddot{\boldsymbol{q}}^l H^2 + \tfrac{1}{6}\dddot{\boldsymbol{q}}^l H^3 + \cdots, \tag{2.39a}$$

$$\boldsymbol{v}(t^l + H) = \boldsymbol{v}^{l+1} = \boldsymbol{v}^l + \dot{\boldsymbol{v}}^l H + \tfrac{1}{2}\ddot{\boldsymbol{v}}^l H^2 + \tfrac{1}{6}\dddot{\boldsymbol{v}}^l H^3 + \cdots. \tag{2.39b}$$

The values $\boldsymbol{q}^l$ and $\boldsymbol{v}^l$ are known since they are the initial state with respect to the current macro time step. Considering $\boldsymbol{T} = \boldsymbol{T}(\boldsymbol{q})$, $\boldsymbol{M} = \boldsymbol{M}(\boldsymbol{q})$, $\boldsymbol{h} = \boldsymbol{h}(\boldsymbol{q}, \boldsymbol{v}, t)$ and $\boldsymbol{B} = \boldsymbol{B}(\boldsymbol{q})$, the first derivatives $\dot{\boldsymbol{q}}^l$ and $\dot{\boldsymbol{v}}^l$ for the exact solution result from equation (2.11). The input $\boldsymbol{u}$ is given by the coupling law $\boldsymbol{o}_M$ only invoking positions $\boldsymbol{q}$

$$\boldsymbol{u} = \boldsymbol{y}_M = \boldsymbol{o}_M(\boldsymbol{u}_M) = \boldsymbol{o}_M(\boldsymbol{y}) = \boldsymbol{o}_M(\boldsymbol{o}(\boldsymbol{q})) =: \boldsymbol{r}(\boldsymbol{q}). \tag{2.40}$$

Higher derivatives of $\boldsymbol{q}$ and $\boldsymbol{v}$ at time t^l can be evaluated applying the chain rule in tensor notation:

$$\ddot{\boldsymbol{q}}^l = \langle \boldsymbol{T}_{\boldsymbol{q}}, \dot{\boldsymbol{q}}, \boldsymbol{v} \rangle^l + \langle \boldsymbol{T}, \dot{\boldsymbol{v}} \rangle^l, \tag{2.41a}$$

$$\begin{aligned} \ddot{\boldsymbol{v}}^l = &\langle \langle \boldsymbol{M}^{-1}, \boldsymbol{h} \rangle_{\boldsymbol{q}}, \dot{\boldsymbol{q}} \rangle^l + \langle \langle \boldsymbol{M}^{-1}, \boldsymbol{h} \rangle_{\boldsymbol{v}}, \dot{\boldsymbol{v}} \rangle^l + \langle \boldsymbol{M}^{-1}, \boldsymbol{h} \rangle_t^l + \\ &+ \langle \langle \boldsymbol{M}^{-1}, \boldsymbol{B} \rangle_{\boldsymbol{q}}, \dot{\boldsymbol{q}}, \boldsymbol{r} \rangle^l + \langle \boldsymbol{M}^{-1}, \boldsymbol{B}, \boldsymbol{r}_{\boldsymbol{q}}, \dot{\boldsymbol{q}} \rangle^l, \end{aligned} \tag{2.41b}$$

$$\dddot{\boldsymbol{q}}^l = \langle \boldsymbol{T}_{\boldsymbol{qq}}, \dot{\boldsymbol{q}}, \dot{\boldsymbol{q}}, \boldsymbol{v} \rangle^l + \langle \boldsymbol{T}_{\boldsymbol{q}}, \ddot{\boldsymbol{q}}\boldsymbol{v} + \dot{\boldsymbol{q}}\dot{\boldsymbol{v}} \rangle^l + \langle \boldsymbol{T}_{\boldsymbol{q}}, \dot{\boldsymbol{q}}, \dot{\boldsymbol{v}} \rangle^l + \langle \boldsymbol{T}, \ddot{\boldsymbol{v}} \rangle^l, \tag{2.41c}$$

$$\dddot{\boldsymbol{v}}^l = \cdots \tag{2.41d}$$

$$\cdots = \cdots$$

The lower right indices $\boldsymbol{q}$, $\boldsymbol{v}$ and t denote the partial derivatives (for example $\boldsymbol{x}_{\boldsymbol{q}} := \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{q}}$) with respect to this variable. Hence, the term $\boldsymbol{T}_{\boldsymbol{q}}$ is for example a tensor of degree 3. The outer tensor product between a and b is represented by ab and the inner tensor product (scalar product) is depicted by $\langle a, b \rangle$. Backward substitution of equations (2.41) in (2.39) leads to the TAYLOR expansion of the state values at the end of the current macro time step t^{l+1} only depending on known values at time t^l.

TAYLOR Expansion of the Constant Extrapolation Using a Linear Combination

The TAYLOR expanded state $\boldsymbol{q}$ and $\boldsymbol{v}$ of the co-simulation at time t^{l+1} is already given by equation (2.39). For a constant extrapolation $\tilde{\boldsymbol{u}} = \boldsymbol{e}_0$ the unknown state derivatives are

$$\dot{\boldsymbol{q}}^l = \langle \boldsymbol{T}, \boldsymbol{v} \rangle^l, \tag{2.42a}$$

$$\dot{\boldsymbol{v}}^l = \langle \boldsymbol{M}^{-1}, \boldsymbol{h} \rangle^l + \langle \boldsymbol{M}^{-1}, \boldsymbol{B}, \boldsymbol{e}_0 \rangle^l, \tag{2.42b}$$

$$\ddot{\boldsymbol{q}}^l = \langle \boldsymbol{T}_q, \dot{\boldsymbol{q}}, \boldsymbol{v} \rangle^l + \langle \boldsymbol{T}, \dot{\boldsymbol{v}} \rangle^l, \tag{2.42c}$$

$$\begin{aligned} \ddot{\boldsymbol{v}}^l = &\langle \langle \boldsymbol{M}^{-1}, \boldsymbol{h} \rangle_q, \dot{\boldsymbol{q}} \rangle^l + \langle \langle \boldsymbol{M}^{-1}, \boldsymbol{h} \rangle_v, \dot{\boldsymbol{v}} \rangle^l + \\ &+ \langle \boldsymbol{M}^{-1}, \boldsymbol{h} \rangle_t^l + \langle \langle \boldsymbol{M}^{-1}, \boldsymbol{B} \rangle_q, \dot{\boldsymbol{q}}, \boldsymbol{e}_0 \rangle^l, \\ \dots = &\dots \end{aligned} \tag{2.42d}$$

The TAYLOR expansion of the linear combination (2.10) fulfilling R constraints (2.28) for the parameters a_k and b_k yields

$$\boldsymbol{e}_0 = \sum_{n=0}^{P-1} \tfrac{1}{(n+1)!} \boldsymbol{u}^{(n)l} H^n + \mathcal{O}(H^P). \tag{2.43}$$

The derivatives of $\boldsymbol{u}$ up to $P-1$ can be calculated applying the chain rule on equation (2.40). Resubstituting equations (2.40), (2.43) and (2.42) in (2.39) yields the TAYLOR expanded state $(\boldsymbol{q}^{l+1}, \boldsymbol{v}^{l+1})$ for the co-simulation.

TAYLOR Expansion of the Linear and Quadratic Extrapolation Using a Linear Combination

The TAYLOR expanded state $\boldsymbol{q}$ and $\boldsymbol{v}$ of the co-simulation at time t^{l+1} for these co-simulation coupling variants are fully analog to the last one. However the extrapolated input (2.33) or (2.37) is linear or quadratic according to equations (2.34) and (2.35) or (2.38). Claiming R constraints (2.28) to be fulfilled and backward substitution all in equation (2.39) yields the TAYLOR expansion of one macro time step.

Coefficient Comparison and Summarizing

Comparing the coefficients of the powers of H between the last three TAYLOR expansions of the co-simulation and the TAYLOR expansion of the exact solution results in the local order of a single co-simulation macro integration step with step size H. Table 2.3 summarizes the local order for $\boldsymbol{q}$ and $\boldsymbol{v}$ giving the number of fulfilled constraints R (2.28) and the extrapolation degree $N = 0 \dots 2$ using the linear combination (2.10).

Extrapolation degree N		Fulfilled constraints R 1	2	3	4
0: const.	$\boldsymbol{q}$	2	2	2	2
	$\boldsymbol{v}$	1	2	2	2
1: lin.	$\boldsymbol{q}$	-	3	3	3
	$\boldsymbol{v}$	-	2	3	3
2: quad.	$\boldsymbol{q}$	-	-	4	4
	$\boldsymbol{v}$	-	-	3	4

Table 2.3: Local order of $\boldsymbol{q}$ and $\boldsymbol{v}$

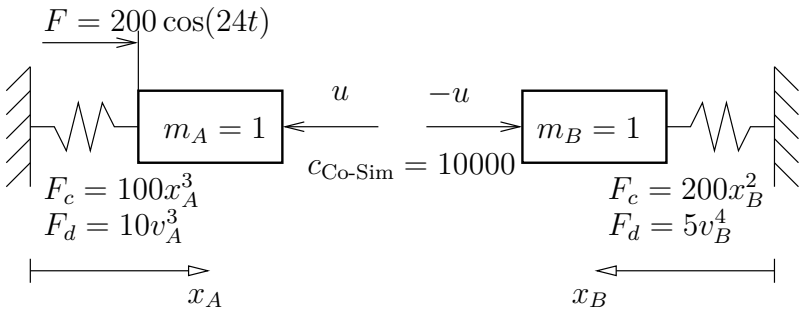
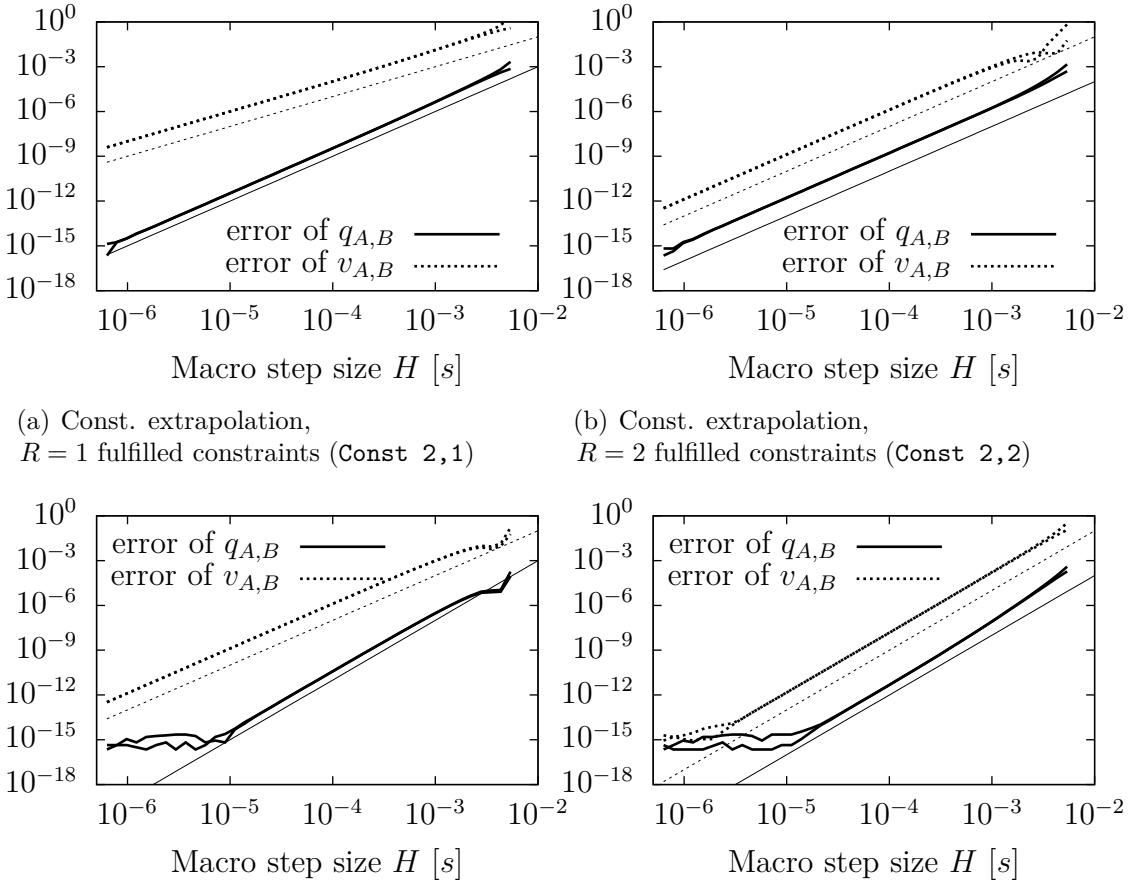


Figure 2.6: Test system for numerical local order analysis

This analysis shows that increasing the extrapolation degree N leads to a higher local order of the state vector of the subsystems. In case of $R = N+1$ fulfilled constraints, the local order of the generalized position $\boldsymbol{q}$ is one larger than the local order of the generalized velocity $\boldsymbol{v}$ because $\boldsymbol{q}$ is the integral of $\boldsymbol{Tv}$ which does not depend on the input $\boldsymbol{u}$. Therefore only the local order of the generalized velocity $\boldsymbol{v}$ can be increased by one using $R > N+1$ fulfilled constraints, independent whether $\boldsymbol{M}^{-1}\boldsymbol{B}$ is constant or not. Hence, concerning the local order of the state variables, the restriction to a constant term $\boldsymbol{M}^{-1}\boldsymbol{B}$ is irrelevant because the maximal local order is restricted also by the nonlinear character of a mechanical system in general.

Numerical Verification

The theoretical results of the local order are confirmed by a numerical study for the nonlinear example shown in Figure 2.6. This example comprises two subsystems each being a one mass oscillator. The subsystems are coupled by the co-simulation with the constant or linear extrapolated input u. Both, the spring and the damping element of each one mass oscillator are nonlinear and the mass of the subsystem A is actuated by a harmonic kinetic excitation. The coupling law o_M of the co-simulation is a linear spring without damping to prevent an algebraic loop if $\dot{u}$ is also used. The integration of the subsystems on micro step level is done with a RUNGE-KUTTA integrator of order 5 with a micro step size $h \ll H$ allowing the solution of the subsystem integrators to be interpreted as exact. The values u at the current and the



(a) Const. extrapolation, $R = 1$ fulfilled constraints (`Const 2,1`)

(b) Const. extrapolation, $R = 2$ fulfilled constraints (`Const 2,2`)

(c) Linear extrapolation, $R = 2$ fulfilled constraints (`Lin 2,2`)

(d) Linear extrapolation, $R = 3$ fulfilled constraints (`Lin 2,3`)

Figure 2.7: Numerical error and local order, represented by the mean gradient, of the test system

$K - 1 = 1$ previous macro time steps evaluated within the linear combination (2.10) have to be exact to analyze the local error (This is analog to the local error analysis of multi step integration methods [26]). These reference values are calculated by a pretended numerical integration of the coupled system (without a co-simulation) using the same numerical integrator and micro step size as described above. At the initial state both subsystem springs as well as the linear coupling spring are stretched.

Figures 2.7(a)-2.7(d) depict the numerical local error of q and v for both subsystems depending on the macro time step size H. The thin straight lines indicate the theoretical analytic order as given in Table 2.3. Based on the full logarithmic axis of these plots the order is qualified by the gradient of the straight lines. The vertical offset of the straight lines represents the local error coefficient which has not been analyzed analytically. Therefore, the offsets of the thin lines are chosen such that they are near to the numerical solution. The numerical model confirms the local order of the approximations for u except for very low macro step sizes H and high orders where the numerical solution runs into the limitation of the computational

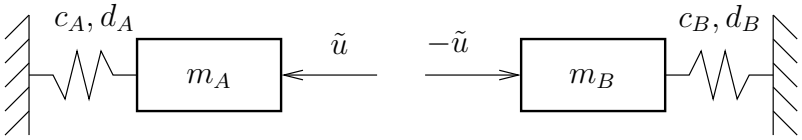


Figure 2.8: Test system for stability analysis

accuracy at about 10^{-15}.

2.3.2 Stability

Besides accuracy requirements, the stability of the co-simulated system is the main restriction for the largest possible macro step size H. If the step size is larger than the maximal stable step size $H_{\max}$, the coupling input u will oscillate with a persistent increase of the amplitude and the whole system which will get unstable. The main aspects leading to instabilities are the coupling stiffnesses and the extrapolation of the input u. Since a high coupling stiffness is often physically motivated, only the extrapolation of the input, being a numerical requirement, can be used to improve the stability. The analysis and comparison of co-simulation coupling variants regarding stability extends the results presented in [45].

Test System and its Solution

In analogy to the stability analysis of standard numerical integration methods [26], the stability of the co-simulation is analyzed using the simple linear mechanical test system depicted in Figure 2.8. This test system can be interpreted as an extended Dahlquist test equation. It consists of two subsystems $i = A, B$, each comprising a one mass oscillator with mass m_i, stiffness c_i and damping d_i. The two masses are coupled using the parallel co-simulation framework. Hence, each mass is excited by a kinetic extrapolated input force $\tilde{u}$ (1.8) of polynomial degree N. The coupling law o_M is chosen as a linear spring (2.4b). For the stability analysis the integration of the subsystem is considered to be exact. The differential equations for the subsystems are

$$m_i\,\ddot{x}_i + d_i\,\dot{x}_i + c_i\,x_i = \mp\tilde{u} = \mp\sum_{n=0}^{N} e_n\,(t-t^l)^n, \tag{2.44}$$

where the upper sign belongs to subsystem A and the lower sign to subsystem B. The homogeneous part $\underline{x}_h$ of the analytical solution of equation (2.44) and its first derivative are

$$\underline{x}_{h,i}(t) = \underline{k}_i\,e^{\underline{\lambda}_i\,t}, \tag{2.45a}$$

$$\dot{\underline{x}}_{h,i}(t) = \underline{k}_i\,\underline{\lambda}_i\,e^{\underline{\lambda}_i\,t}, \tag{2.45b}$$

$$\text{with } \underline{\lambda}_i = j\omega_i - \delta_i, \quad \delta_i = \frac{d_i}{2m_i}, \quad \omega_i^2 = \frac{c_i}{m_i} - \delta_i^2, \tag{2.45c}$$

if the damping $d_i < \sqrt{4 m_i\, c_i}$ is considered to be weak. The underscores represent complex values and j the imaginary unit. The real particular part x_p and its derivative are

$$x_{p,i}(t) = \sum_{n=0}^{N} p_n\,(t - t^l)^n, \tag{2.46a}$$

$$\dot{x}_{p,i}(t) = n \sum_{n=1}^{N} p_n\,(t - t^l)^{n-1} \tag{2.46b}$$

using the unknown parameters p_n. Considering the right hand side of equation (2.44) yields

$$\text{for } N = 0: \begin{cases} x_{p,i}(t) = \underbrace{\mp \dfrac{e_0}{c_i}}_{p_0} \\ \dot{x}_{p,i}(t) = 0 \end{cases} \tag{2.47a}$$

$$\text{and for } N = 1: \begin{cases} x_{p,i}(t) = \underbrace{\mp \dfrac{c_i\, e_0 - d_i\, e_1}{c_i^2}}_{p_0} \underbrace{\mp \dfrac{e_1}{c_i}}_{p_1} (t - t^l) \\ \dot{x}_{p,i}(t) = \mp \dfrac{e_1}{c_i} \end{cases} . \tag{2.47b}$$

The parameter e_0, in case of a constant extrapolation with $N = 0$, is defined by equation (2.10). For $N = 1$, giving the linear extrapolation, the parameters e_0 and e_1 are defined by equations (2.34) and (2.35). Substituting t with t^l in $\underline{x}_i = \underline{x}_{h,i} + x_{p,i}$ and in its derivative yields the complex state at time t^l dependent on the complex parameter $\underline{k}_i$

$$\underline{x}_i^l = \underline{k}_i\, e^{\underline{\lambda}_i\, t^l} + x_{p,i}(t^l), \tag{2.48a}$$

$$\underline{\dot{x}}_i^l = \underline{k}_i\, \underline{\lambda}_i\, e^{\underline{\lambda}_i\, t^l} + \dot{x}_{p,i}(t^l). \tag{2.48b}$$

Substituting t with $t^{l+1} = t^l + H$ in $\underline{x}_i = \underline{x}_{h,i} + x_{p,i}$ and in its derivative results in the complex state at the end of the current macro time step $t^l + H$

$$\underline{x}_i^{l+1} = \underline{k}_i\, e^{\underline{\lambda}_i (t^l + H)} + x_{p,i}(t^l + H), \tag{2.49a}$$

$$\underline{\dot{x}}_i^{l+1} = \underline{k}_i\, \underline{\lambda}_i\, e^{\underline{\lambda}_i (t^l + H)} + \dot{x}_{p,i}(t^l + H). \tag{2.49b}$$

In these equations the explicit appearance of the time t^l can be eliminated using the state at time t^l (2.48) and the particular part (2.46) at time t^l and $t^l + H$

$$\underline{x}_i^{l+1} = e^{\underline{\lambda}_i\, H} \underbrace{\left(\underline{k}_i\, e^{\underline{\lambda}_i\, t^l} + x_{p,i}(t^l)\right)}_{\underline{x}_i^l} - e^{\underline{\lambda}_i\, H} \underbrace{x_{p,i}(t^l)}_{p_0} + \underbrace{x_{p,i}(t^l + H)}_{\sum_{n=0}^{N} p_n\, H^n}, \tag{2.50a}$$

$$\underline{\dot{x}}_i^{l+1} = e^{\underline{\lambda}_i\, H} \underbrace{\left(\underline{k}_i\, \underline{\lambda}_i\, e^{\underline{\lambda}_i\, t^l} + \dot{x}_{p,i}(t^l)\right)}_{\underline{\dot{x}}_i^l} - e^{\underline{\lambda}_i\, H} \underbrace{\dot{x}_{p,i}(t^l)}_{p_1} + \underbrace{\dot{x}_{p,i}(t^l + H)}_{n \sum_{n=1}^{N} p_n\, H^{n-1}} . \tag{2.50b}$$

Hence, the state at the new macro time step t^{l+1} is a function depending on the old state at time t^l, the subsystem parameters, the parameters of the extrapolated input u and the macro step size H. The state at an arbitrary macro time step can therefore be calculated using a recurrence formula. Moreover, equation (2.50) shows that the old states occur linearly, since the parameters p_n depend only linearly on the parameters e_n which itself are linear on the input u depending linearly on the position x_i. The new subsystem state at time t^{l+1} depends on the current state at t^l and according to equation (2.10) also on the $k = 1 \dots K-1$ previous states at the times t^{l-k}. Introducing the vector

$$\begin{aligned} \boldsymbol{r}^l = [&x_A^l, \dot{x}_A^l, x_B^l, \dot{x}_B^l, x_A^{l-1}, \dot{x}_A^{l-1}, x_B^{l-1}, \dot{x}_B^{l-1}, \dots, \\ &x_A^{l-K+1}, \dot{x}_A^{l-K+1}, x_B^{l-K+1}, \dot{x}_B^{l-K+1}]^T, \end{aligned} \tag{2.51}$$

the linear recurrence formula for the new state is written in matrix notation

$$\boldsymbol{r}^{l+1} = \boldsymbol{R}\boldsymbol{r}^l \tag{2.52}$$

with the stability function $\boldsymbol{R}$ which decides about the system stability. The matrix $\boldsymbol{R} \in \mathbb{R}^{4K \times 4K}$ is evaluated using the real part of equation (2.50) substituting equations (2.47), (2.10) or (2.34) and (2.35) as well as (2.4b) dependent on the desired extrapolation degree N. The constant matrix $\boldsymbol{R}$ depends on the number of used macro time steps K, the linear combination parameters a_k and b_k, the macro step size H and the subsystem parameters ω_i, δ_i, c_i and c. Exemplarily, the first entry of $\boldsymbol{R}$ is shown for $N = 0$:

$$R_{1,1} = \frac{(c_A + a_0\, c)\,(\delta_A\, \sin(\omega_A\, H) + \omega_A\, \cos(\omega_A\, H)) - a_0\, c\omega_A\, e^{\delta_A\, H}}{c_A\, \omega_A} e^{-\delta_A\, H} \tag{2.53}$$

Note that $R_{1,1}$ does not depend on the number of supporting points K, but other entries of $\boldsymbol{R}$ will depend on it.

Stability Condition

The sequence of vectors $\boldsymbol{r}^l$ in equation (2.52) is bounded if the matrix $\boldsymbol{R}$ has a spectral radius equal or less than 1 [14]:

$$\rho(\boldsymbol{R}) := \max(|\mathrm{eig}(\boldsymbol{R})|) \leq 1. \tag{2.54}$$

Then also the states of the subsystems x_A, v_A and x_B, v_B are bounded and the co-simulated test system is stable. In the following the properties of $\boldsymbol{R}$ are analyzed.

To reduce the number of parameters the following substitutions are introduced to equation (2.52)

$$\omega_i\, H = \hat{\omega}_i, \quad \delta_i\, H = \hat{\delta}_i, \quad c_A = C, \quad c_B = VC, \quad c = FC, \tag{2.55}$$

$$\boldsymbol{p}^l = \boldsymbol{G}\boldsymbol{r}^l \text{ with } \boldsymbol{G} = \mathrm{diag}(1, H, 1, H, \dots, 1, H), \tag{2.56}$$

which yields the new linear recurrence formula

$$\boldsymbol{p}^{l+1} = \boldsymbol{P}\boldsymbol{p}^{l}, \tag{2.57}$$

having no explicit dependency on H and no dependency on the new substitution variable C. The substitutions (2.55) and the scaling of the velocity terms in equation (2.56) with H do not influence the eigenvalues of $\boldsymbol{P}$ since the scaling leads to a similar matrix [14]. Hence the eigenvalues of the matrix $\boldsymbol{R}$ and $\boldsymbol{P}$ are equal:

$$\boldsymbol{P} = \boldsymbol{G}\boldsymbol{R}\boldsymbol{G}^{-1} \;\Rightarrow\; \mathrm{eig}(\boldsymbol{P}) = \mathrm{eig}(\boldsymbol{R}) \Rightarrow \rho(\boldsymbol{P}) = \rho(\boldsymbol{R}) = \max(|\mathrm{eig}(\boldsymbol{P})|). \tag{2.58}$$

Since conservative subsystems impose the highest requirements on the co-simulation concerning stability, the subsystems are assumed to be undamped

$$d_A = d_B = \delta_A = \delta_B = \hat{\delta}_A = \hat{\delta}_B = 0. \tag{2.59}$$

Moreover the subsystems are assumed to have nearly the same eigenfrequencies and stiffnesses

$$\omega_A \approx \omega_B := \omega \quad \Leftrightarrow \quad \hat{\omega}_A \approx \hat{\omega}_B =: \hat{\omega}, \tag{2.60}$$

$$V = \frac{c_A}{c_B} \approx 1. \tag{2.61}$$

Industrial problems show that many mechanical systems meet these assumptions. The above limitations leads to a matrix $\boldsymbol{P}$ only depending on the scaled eigenfrequency $\hat{\omega}$ (2.60), the stiffness ratio F (2.55) as well as the extrapolation parameters a_k and b_k of the input.

Stability Optimization

Restricting the number of supporting points to $K \leq 3$, the maximal degree of the extrapolation to $N \leq 2$ and the number of fulfilled constraints to $R = 1 \ldots 2K$ yields a total number of 37 different coupling types by permutation of the coupling variants. Since for each coupling method $2K - R \geq 0$ parameters a_k and b_k are not defined by the method, a total number of 23 couplings have at least one free parameter which can be used to optimize the stability. The other couplings cannot be optimized for stability and represent mainly the methods described in Sections 2.2.1 and 2.2.2 or in Sections 2.2.3 and 2.2.4 with $2K - R = 0$. In the following only coupling types are considered having proven good stability and accuracy properties. The coupling variants with a quadratic extrapolation ($N = 2$) are not presented, because these have shown bad stability properties. The variants with three supporting points $K = 3$ have shown a considerable worse stability characteristic than the one with $K = 2$ and are also not shown for this reason. Finally the variants with $R = 1$ are also neglected because of their low local order. The coded names of these couplings are extended by `Opt`, since free parameters are used for stability optimizations.

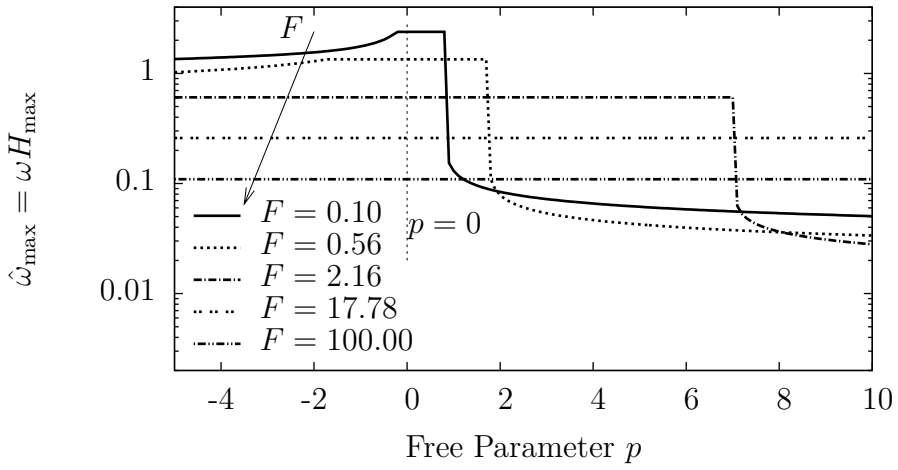


Figure 2.9: Largest stable step size $\hat{\omega}_{\max}$; `Const 2,3,Opt`

Constant Extrapolation with $K = 2$ and $R = 3$ (`Const 2,3,Opt`): The constant extrapolation from Section 2.2.3 with $K = 2$ supporting points and $R = 3$ fulfilled constraints leads to $2K - R = 1$ free parameter for the parameters a_k and b_k. The $R = 3$ constraints (2.28) can be eliminated analytically using one free parameter p

$$a_0 := -\tfrac{1}{3}(6p - 2), \tag{2.62a}$$
$$a_1 := \tfrac{1}{3}(6p + 1), \tag{2.62b}$$
$$b_0 := \tfrac{1}{6}(6p + 5), \tag{2.62c}$$
$$b_1 := p. \tag{2.62d}$$

The largest stable macro step size $H_{\max}$

$$\omega H_{\max} = \hat{\omega}_{\max} = \max\{\hat{\omega} : \rho(\boldsymbol{P}) \leq 1\} \tag{2.63}$$

for a varying parameter p and different fixed values of F is depicted in Figure 2.9. For large values of F, representing a very stiff co-simulation coupling, the parameter p has no influence on the maximal macro step size, but for smaller values of F this parameter must be restricted to $p = 0$ to achieve the best stability property. Hence, the optimal values of the parameters a_k and b_k are given by

$$a_0 = \tfrac{2}{3}, \quad a_1 = \tfrac{1}{3}, \quad b_0 = \tfrac{5}{6}, \quad b_1 = 0, \tag{2.64}$$

independent of the ratio F. The maximum value $\hat{\omega}_{\max}$, dependent on F is shown in Table 2.4

Linear Extrapolation with $K = 2$ and $R = 3$ (`Lin 2,3,Opt`): The same is done for the linear extrapolation from Section 2.2.4 with $K = 2$ supporting points and $R = 3$ fulfilled constraints (2.28), which also leads to one free parameter p. The corresponding maximal scaled macro step sizes $\hat{\omega}_{\max}$ are depicted in Figure 2.10.

F	$\hat{\omega}_{\max}$	p at $\hat{\omega}_{\max}$
0.10	2.38	(see Figure 2.9)
0.56	1.34	(see Figure 2.9)
3.16	0.606	(see Figure 2.9)
17.8	0.259	(see Figure 2.9)
100.0	0.109	(see Figure 2.9)

Table 2.4: Maximum $\hat{\omega}_{\max}$; `Const 2,3,Opt`; compare to Figure 2.9

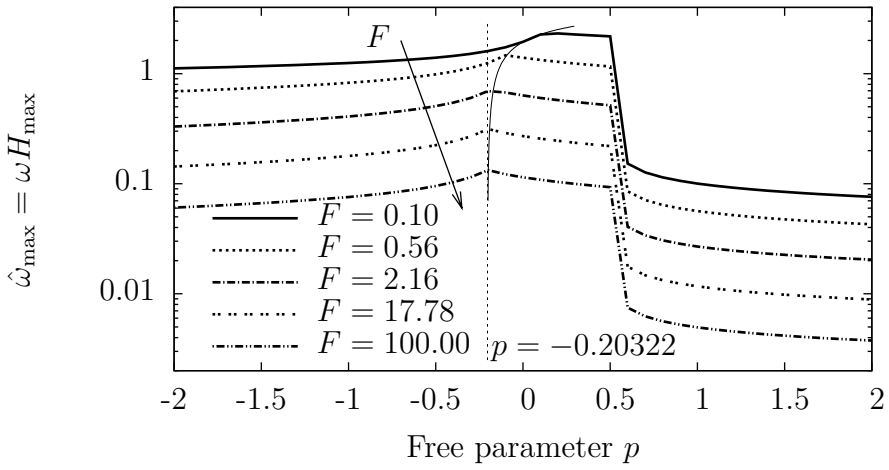


Figure 2.10: Largest stable step size $\hat{\omega}_{\max}$; `Lin 2,3,Opt`

For each ratio F the maximum value $\hat{\omega}_{\max}$ is always at an individual point, which is listed in Table 2.5. In contrast to the constant extrapolation there exists no single parameter p where best stability can be achieved for all ratios F. Considering that stability problems only arise for stiff couplings, where F is large, and that the parameters p at the maximum values $\hat{\omega}_{\max}$ for large ratios F are close together, it is a good compromise to use $p = -0.20322$ as the general choice. This corresponds to $F = 100$ and yields

$$a_0 = 1.0731067, \quad a_1 = -0.0731067, \quad b_0 = 0.6301133, \quad b_1 = -0.20322, \tag{2.65}$$

as best general parameters for this coupling.

Constant Extrapolation with $K = 2$ and $R = 2$ (`Const 2,2,Opt`): Forcing only $R = 2$ constraints (2.28) to be fulfilled leads to two free parameters p_1 and p_2 for

$$a_0 := -\tfrac{1}{2}(2p_2 + 2p_1 - 3), \tag{2.66a}$$
$$a_1 := \tfrac{1}{2}(2p_2 + 2p_1 - 1), \tag{2.66b}$$
$$b_0 := p_2, \tag{2.66c}$$
$$b_1 := p_1. \tag{2.66d}$$

F	$\hat{\omega}_{\max}$	p at $\hat{\omega}_{\max}$
0.100	2.3607	0.12446
0.562	1.4745	-0.097519
3.16	0.72003	-0.18132
17.8	0.31417	-0.19994
100.0	0.13335	-0.20322

Table 2.5: Maximum $\hat{\omega}_{\max}$; `Lin 2,3,Opt`; compare to Figure 2.10

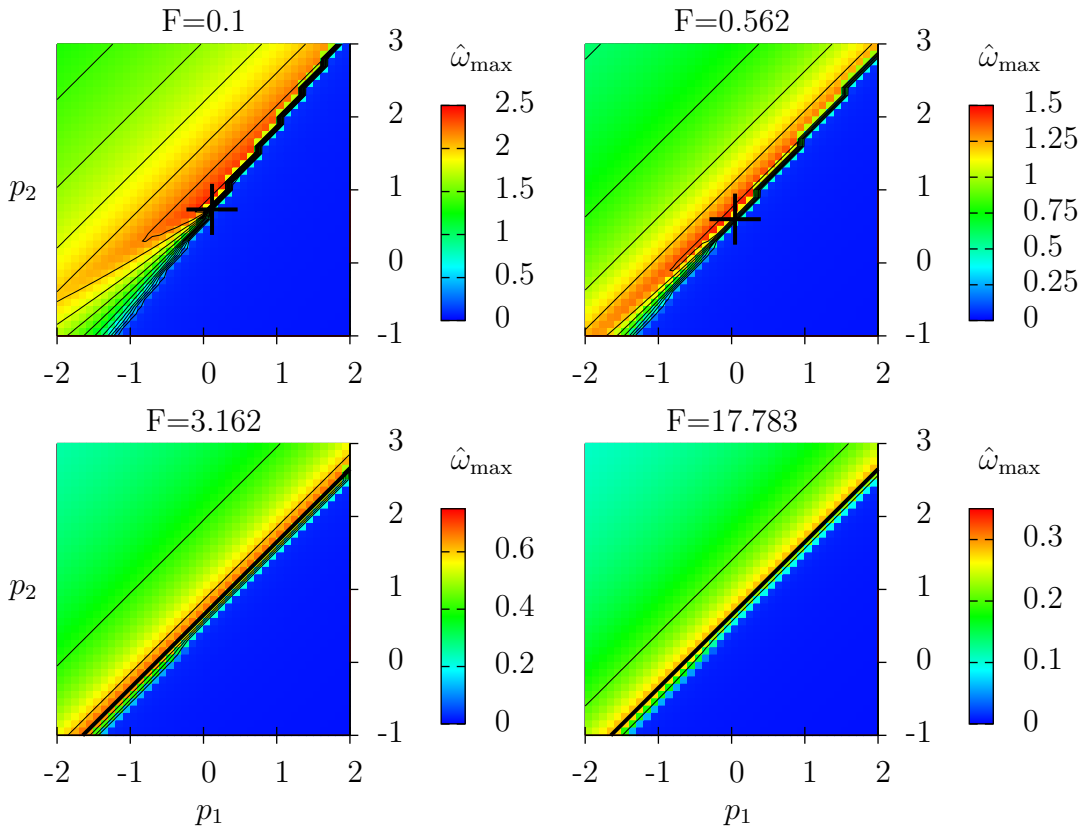


Figure 2.11: Largest stable step size $\hat{\omega}_{\max}$; $N = 0$, $K = 2$ and $R = 2$

The contour plots in Figure 2.11 depicts the largest step size $\hat{\omega}_{\max}$ depending on p_1 and p_2 for various scaled co-simulation stiffnesses F. The colors represent the value of $\hat{\omega}_{\max}$ in the plane of the parameters p_1 and p_2. The black lines represent contour lines of equal $\hat{\omega}_{\max}$. For low ratios F the maximal stability is at a single point marked by a bold cross. For high ratios F maximal stability is achieved not at a single point but within a straight line from the bottom left to the top right, marked by a bold line. Table 2.6 lists the maximum values of $\hat{\omega}_{\max}$ in dependency of F. A compromise for the best parameter set, independent on the ratio F, are the values $p_1 = -0.2$ and $p_2 = 0.363$. Hence, the best general parameters chosen for this coupling are

$$a_0 = 1.3370, \quad a_1 = -0.33700, \quad b_0 = 0.363, \quad b_1 = -0.2. \tag{2.67}$$

F	$\hat{\omega}_{\max}$	p_1, p_2 at $\hat{\omega}_{\max}$
0.100	2.5347	0.11734, 0.73499
0.562	1.5858	0.049051, 0.601361
3.16	0.76531	straight line (see Figure 2.11)
17.8	0.33281	straight line (see Figure 2.11)
100.0	0.14120	straight line

Table 2.6: Maximum $\hat{\omega}_{\max}$; $N = 0$, $K = 2$ and $R = 2$; compare to Figure 2.11

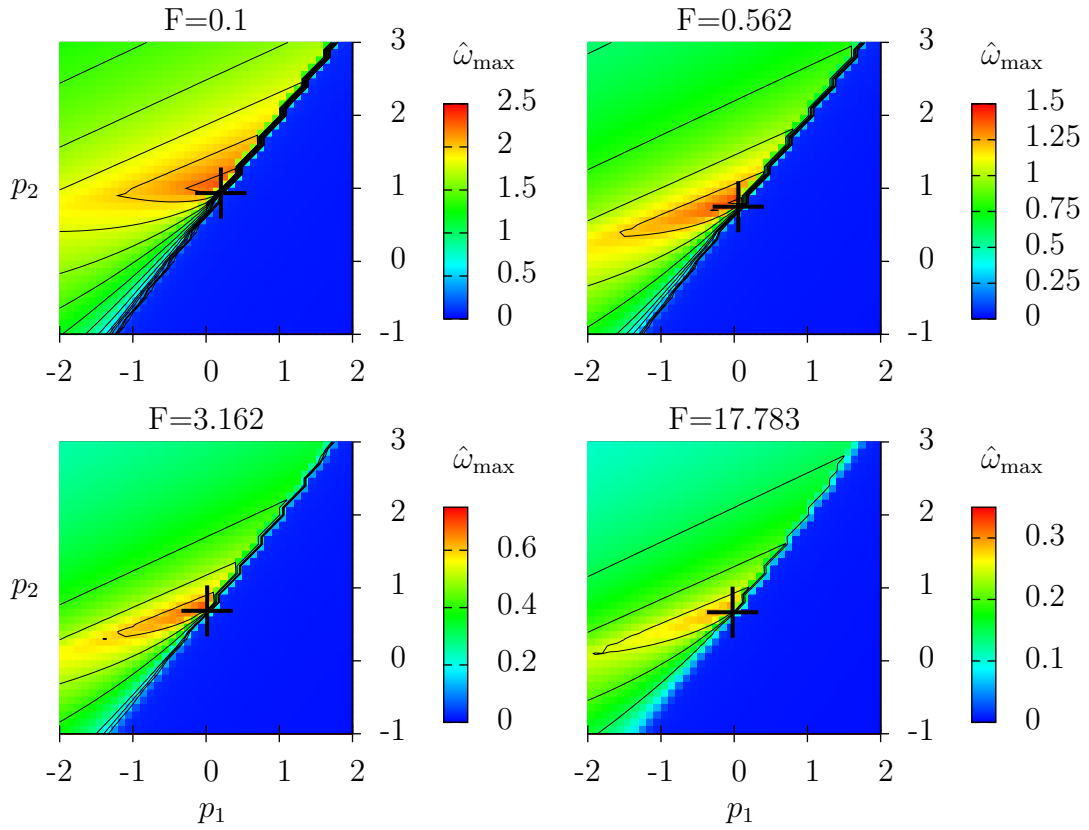


Figure 2.12: Largest stable step size $\hat{\omega}_{\max}$; $N = 1$, $K = 2$ and $R = 2$

Linear Extrapolation with $K = 2$ and $R = 2$ (`Lin 2,2,Opt`): The second coupling variant having two free parameters is the linear extrapolation from Section 2.2.4 with $K = 2$ supporting points and $R = 2$ fulfilled constraints. Figure 2.12 shows plots for the maximal step size $\hat{\omega}_{\max}$ for this coupling variant. For each ratio F there is a single point of maximum, shown in Table 2.7 and marked by a bold cross in Figure 2.12. Since higher ratios F are more problematic concerning stability, the values $p_1 = -0.0069$ and $p_2 = 0.667$ are chosen as best general point independent of the parameter F. Hence, the parameters

$$a_0 = 0.83990, \quad a_1 = 0.1601, \quad b_0 = 0.667, \quad b_1 = -0.0069 \tag{2.68}$$

are used for this type of coupling, defining the best general stability for the analyzed coupled system.

F	$\hat{\omega}_{\max}$	p_1, p_2 at $\hat{\omega}_{\max}$
0.100	2.4054	0.20301, 0.93711
0.562	1.5373	0.057105, 0.749684
3.16	0.75832	0.014735, 0.686343
17.8	0.32978	-0.020791, 0.666899
100.0	0.14071	-0.0069416, 0.6667712

Table 2.7: Maximum $\hat{\omega}_{\max}$; $N = 1$, $K = 2$ and $R = 2$; compare to Figure 2.12

Coupling Type	R	$F = 100$	$F = 17.8$	$F = 3.16$
`Const 2,3,Opt`	3	0.109 ↘ 22%	0.259 ↘ 21%	0.606 ↘ 19%
`Lin 2,3,Opt`	3	0.133	0.314	0.720
`Const 2,2,Opt`	2	6% ↘ 0.141 ↘ 0%	5% ↘ 0.333 ↗ 1%	4% ↘ 0.765 ↗ 2%
`Lin 2,2,Opt`	2	0.141	0.330	0.750

Table 2.8: Comparison of $\hat{\omega}_{\max}$

Comparison

Table 2.8 summarizes the maximal values $\hat{\omega}_{\max}$ for different F from the Tables 2.4 to 2.7. In case of $R = 3$ fulfilled constraints there is an increase of about 20% for the largest macro step size $H_{\max} = \frac{\hat{\omega}_{\max}}{\omega}$ if a linear extrapolation is used instead of a constant extrapolation. In contrast for $R = 2$, there is nearly no change in stability depending on the extrapolation degree, being constant or linear. Reducing the number of fulfilled constraints R from 3 to 2 leads to a small increase of about 5% in case of a linear extrapolation. However, according to Table 2.3 the local order of the generalized velocity is decreased by one.

Figure 2.13 compares the stability not only at some discrete ratios F but at arbitrary ones. Moreover this figure compares the last four optimized variants with the variants from Section 2.2, without free parameters a_k or b_k. This plot shows the maximal scaled macro step size $\hat{\omega}_{\max} = \omega H_{\max}$ depending on the ratio F. The thin dotted lines in this plot represent the level lines of equal stability which are generated as follows: The two eigenfrequencies of the coupled, undamped, analytical test system of Figure 2.8 under the conditions (2.60) and (2.61) as well as the substitution (2.55) are

$$\omega_1 = \omega, \quad \omega_2 = \omega\sqrt{2F+1}. \tag{2.69}$$

Defining the level L of equal stability to

$$L = \max(\omega_1, \omega_2)H = \omega\sqrt{2F+1}H, \tag{2.70}$$

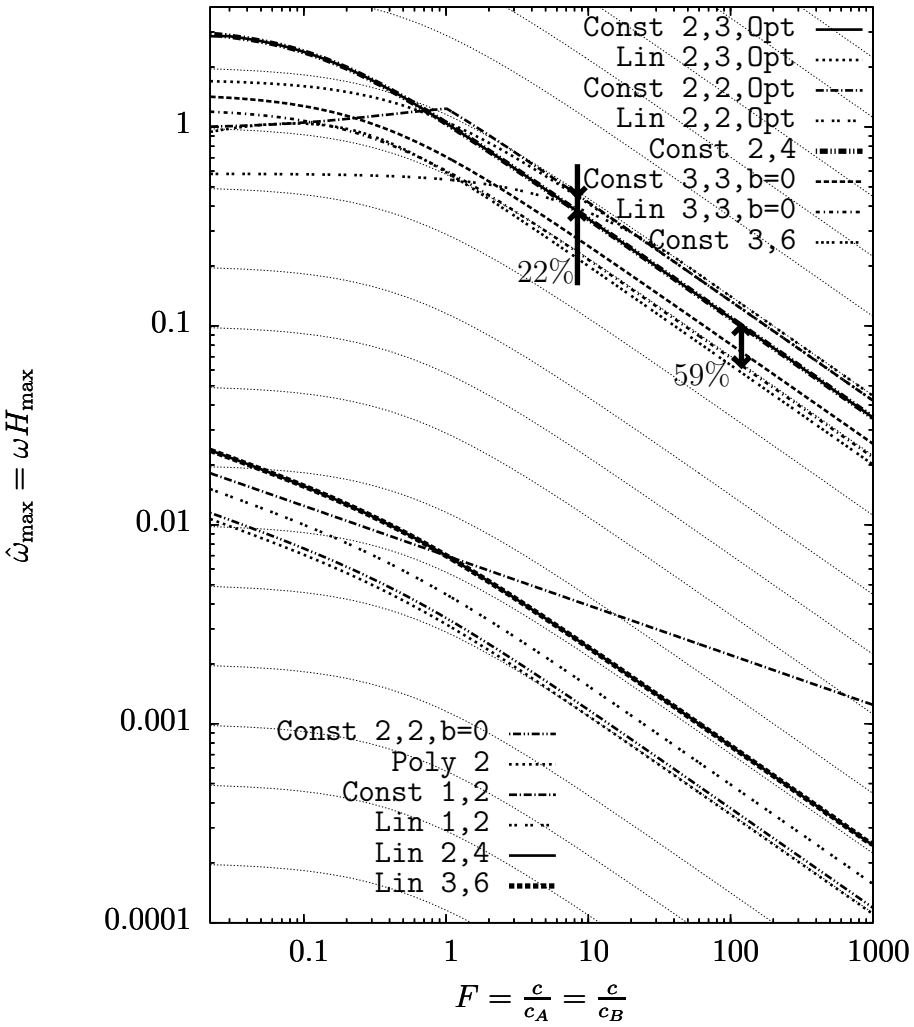


Figure 2.13: Stability plot

the level lines of equal stability L are defined by

$$\omega H_L = \hat{\omega}_L = \frac{L}{\sqrt{2F+1}}. \tag{2.71}$$

Hence, if the stability curve of a coupling is always near one level line, then this coupling has a nearly equal stability property independent on the ratio F.

Figure 2.13 shows two groups of couplings: one with relatively low and one with relatively high stability. The couplings with low stability are not very useful for a parallel co-simulation because the low macro step size H will lead to bad parallel speedups due to frequent communications. The other curves are discussed in more detail:

1. The constant extrapolation using a HERMITE interpolation (`Const 2,4`) is the most stable classic coupling. The maximal macro step size H is about

59% larger than for the linear extrapolation using a quadratic polynomial (`Lin 3,3,b=0`).

2. The most stable coupling for stiff connections ($F \gg 1$) is the linear extrapolation using $K = 2$ supporting points and two free parameters for optimization (`Lin 2,2,Opt`). However this coupling does not have an optimal behavior for $F < 10$.

3. A good compromise between stability for high and low ratios F are the couplings `Lin 2,3,Opt` and `Const 2,2,Opt`.

4. The constant extrapolation using $K = 2$ supporting points and one free parameter ($R = 3$, `Const 2,3,Opt`) shows a nearly equal stability compared to the HERMITE interpolation `Const 2,4`.

5. The systematic optimization of coupling variants has increased the stability of about 22% without a lose in accuracy because the decrease from $R = 4$ to $R = 3$ fulfilled constraints does not influence the local order according to Table 2.3.

The stability is only tested concerning the given linear test system having also a linear force coupling between the subsystems. Such analysis using linear systems are common in numerical mathematics, but nonlinear complex examples can show other stability properties. Hence, in Section 4.2 a numerical benchmark will be presented with respect to stability and parallel computing performance for different coupling methods using an industrial complex and nonlinear example. This example demonstrates that the qualitative stability properties, regarded for the linear test system, also qualifies for a complex nonlinear example.

2.3.3 Long Time Behavior

The last analysis considers the long time behavior of co-simulations based on the linear example from the previous section, Figure 2.8. Three different co-simulations are compared: First, the master-slave concept or *force-force* coupling according to Figure 2.2, using the most stable coupling methods from Section 2.3.2. Second and third, the classical *force-displacement* and the *displacement-displacement* coupling, see Figure 1.2, using a constant and linear extrapolation of the input with a spring-damper as coupling element. Whereas the first represent the optimized methods used in this work, the last two are commonly used co-simulation methods from literature [70].

The linear recurrence equation (2.57) is used as starting point for the analysis of the master-slave (*force-force*) coupling. An analogous recurrence equation can be generated using the test system for the *displacement-displacement* and the *force-displacement* couplings, which leads to different matrices $\boldsymbol{P}$. The value of $\boldsymbol{p}$ of

equation (2.57) at time t^l is given without recursion depending on the initial value $\boldsymbol{p}^0$ by

$$\boldsymbol{p}^l = \underbrace{\boldsymbol{P}\boldsymbol{P}\cdots\boldsymbol{P}}_{l \text{ times}}\boldsymbol{p}^0 = \boldsymbol{P}^l\boldsymbol{p}^0. \tag{2.72}$$

Note that $\boldsymbol{P}^l$ means $\boldsymbol{P}$ to the power of l, whereas all other top right indices l represent a value at time t^l. For the analysis of the long time behavior of the discrete co-simulation, a modified ordinary differential equation is investigated which has the same solution as the discrete co-simulation at the discrete macro time steps $t^l, l = 0 \ldots \infty$. Such a procedure is commonly known as *backward error analysis* and used for example for investigations on numerical integrators for HAMILTON systems [25, 40]. The continuous modified solution $\tilde{\boldsymbol{p}}$ of equation (2.72) yields

$$\tilde{\boldsymbol{p}}(t) = \boldsymbol{P}^{t/H}\boldsymbol{p}^0 \tag{2.73}$$

with $t^0 = 0$ and equidistant macro time steps $t^l = lH$ with step size H. Reformulating this solution using the matrix exponential and the matrix logarithm gives

$$\tilde{\boldsymbol{p}}(t) = e^{\ln(\boldsymbol{P}^{t/H})}\boldsymbol{p}^0 = e^{t/H\cdot\ln\boldsymbol{P}}\boldsymbol{p}^0. \tag{2.74}$$

Finally, equation (2.74) is the solution of the modified linear first order ordinary differential equation [5]

$$\dot{\tilde{\boldsymbol{p}}} = \underbrace{\tfrac{1}{H}\ln\boldsymbol{P}}_{\boldsymbol{A}}\cdot\tilde{\boldsymbol{p}} \tag{2.75}$$

with the initial condition $\tilde{\boldsymbol{p}}(t^0 = 0) = \boldsymbol{p}^0$. This differential equation can be analyzed using the eigenvalues of the constant system matrix $\boldsymbol{A} = \frac{1}{H}\ln\boldsymbol{P}$. The eigenvalues of $\boldsymbol{P}$ were already analyzed for the stability considerations in Section 2.3.2. Since the matrix $\boldsymbol{P}$ is diagonalizable $\boldsymbol{P} = \boldsymbol{T}\boldsymbol{D}\boldsymbol{T}^{-1}$ with $\boldsymbol{D} = \operatorname{diag}(\operatorname{eig}(\boldsymbol{P}))$, the matrix logarithm of $\boldsymbol{P}$ is defined according to [21] as

$$\ln\boldsymbol{P} = \boldsymbol{T}\,(\ln\boldsymbol{D})\,\boldsymbol{T}^{-1}. \tag{2.76}$$

Hence, the eigenvalues of $\ln\boldsymbol{P}$ are equal to the logarithm of the eigenvalues of $\boldsymbol{P}$

$$\operatorname{eig}(\ln\boldsymbol{P}) = \ln(\operatorname{eig}(\boldsymbol{P})). \tag{2.77}$$

This can be proved by setting up the eigenvalue problem for the matrix $\ln\boldsymbol{P}$ using equation (2.76) and the transformation matrix $\boldsymbol{T}$:

$$(\ln\boldsymbol{P})\cdot\boldsymbol{T} = \boldsymbol{T}\ln\boldsymbol{D}, \tag{2.78a}$$

$$\boldsymbol{T}\,(\ln\boldsymbol{D})\underbrace{\boldsymbol{T}^{-1}\cdot\boldsymbol{T}}_{E} = \boldsymbol{T}\ln\boldsymbol{D}. \tag{2.78b}$$

Stability is claimed for the comparison of the three different co-simulation methods, which leads to the condition (2.54). Since the logarithm of a complex number with

a spectral radius less than or equal to 1 always leads to a real part less than or equal to 0,

$$\rho(\boldsymbol{P}) = \max(|\mathrm{eig}\boldsymbol{P}|) \leq 1 \quad \Rightarrow \quad \Re(\mathrm{eig}\boldsymbol{A}) = \Re\left(\tfrac{1}{H}\ln(\mathrm{eig}\boldsymbol{P})\right) \leq 0, \tag{2.79}$$

the solution of the modified differential equation (2.75) is always damped or energy conservative.

Subsystems with Equal Eigenfrequencies

The highest requirements on the numerical integration is given for undamped subsystems $\hat{\delta}_A = \hat{\delta}_B = 0$. In this case and the assumption that the eigenfrequencies of the two subsystems A and B are equal $\hat{\omega}_A = \hat{\omega}_B = \hat{\omega}$ and a *force-force* coupling the two eigenvalues of $\boldsymbol{P}$ satisfy

$$\mathrm{eig}\boldsymbol{P} = [e^{+j\hat{\omega}}, e^{-j\hat{\omega}}, \dots], \tag{2.80}$$

independent of the extrapolation used for the input. Hence, two eigenvalues of the modified equation (2.75) are the conjugate complex values $\pm j\hat{\omega}$, which represent the eigendynamics of the modified equation. The eigenfrequency of the test system without a co-simulation coupling but with a rigid connection between the two bodies is

$$\hat{\omega}_n = \omega_n\, H = \sqrt{\frac{c_A + c_B}{m_A + m_B}} H. \tag{2.81}$$

Using the substitutions (2.45c) and (2.55) yields

$$\hat{\omega}_n = \sqrt{\frac{C + VC}{\frac{C}{\omega_A^2} + \frac{VC}{\omega_B^2}}} H = \sqrt{\frac{1+V}{\omega_B^2 + V\omega_A^2}} \omega_A\, \omega_B\, H = \sqrt{\frac{1+V}{\hat{\omega}_B^2 + V\hat{\omega}_A^2}} \hat{\omega}_A\, \hat{\omega}_B. \tag{2.82}$$

Hence, the eigenfrequency of the modified equation of the *force-force* coupling equals the eigenfrequency $\hat{\omega} = \hat{\omega}_A = \hat{\omega}_B$ of the reference system being coupled by a constraint. The same applies for the damping: the reference system as well as the co-simulation *force-force* coupling is conservative. Therefore, at least for subsystems having equal eigenfrequencies the co-simulation *force-force* coupling does not influence the solution compared to the reference system. However, the *displacement-displacement* as well as the *force-displacement* coupling does not show the same effect: both couplings induce a damping to the subsystems and shift the eigenfrequency of the modified equation. Since the analytic investigation, especially for eigenvalues, of these methods is very complex, this effect is shown numerically in Figure 2.14 for a constant extrapolation of the input using a phase diagram and a Poincaré plot. The *force-force* plot is generated with the most stable coupling method (`Const 2,3,Opt`) from Section 2.3.2 and a step size being about 5 times smaller than the maximal stable step size for this method. The two other plots are generated with the same step size and a damping of the coupling being the lowest

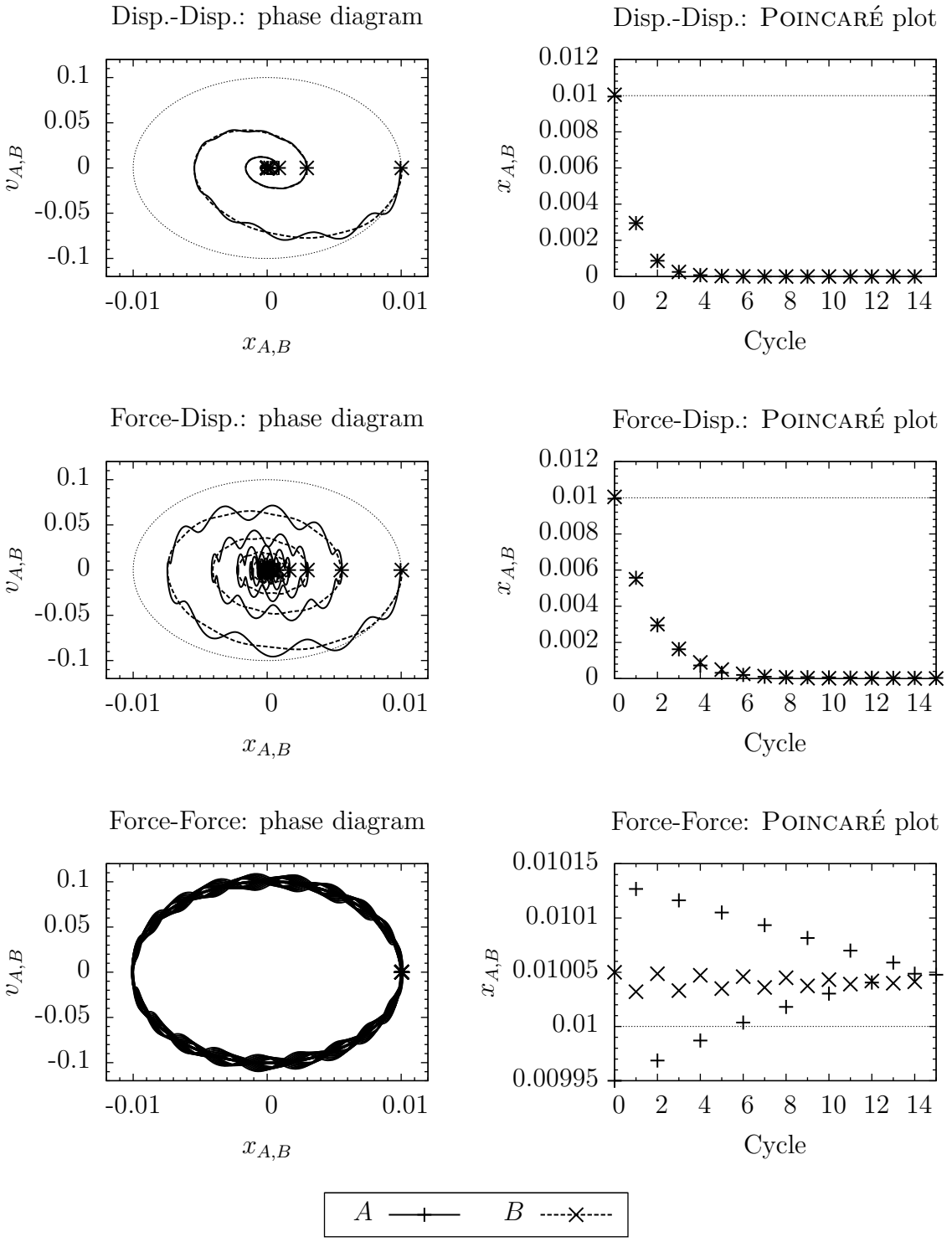


Figure 2.14: Comparison of different co-simulation methods: constant extrapolation

feasible leading to a stable solution. The stiffness ratio $V = 1$ is chosen for all three plots and an initial condition forcing a deviation of both subsystems springs and the coupling spring.

The *displacement-displacement* coupling leads to a very high damping of the subsystems: during a macro time step the input, considered as a kinematic excitation, is hold constant. Therefore the damping part of the coupling law acts on both subsystems like a damping against the environment. The *force-displacement* coupling results in a lower damping: the damping part of the coupling law acts only on one subsystem. The *force-force* coupling leads to an undamped system as expected.

The orbit of the reference system being rigidly coupled is reflected by the thin dotted line. The orbit of the *force-force* coupling does not match this orbit, but leads to a stationary behavior after the high frequent oscillations of the coupling are damped out (see Figure 2.14 bottom left). For the other couplings a comparison with the orbit is not possible, since they are highly damped.

Qualitatively the same effect is regarded for a linear extrapolation of the input as shown in Figure 2.15. The phase diagrams of all three variants look like an undamped stationary system, but the POINCARÉ plots show again, that the *force-force* coupling is the only one which leads to a stationary orbit. The two other methods are damped, even if damping is orders of magnitudes lower than for the constant coupling extrapolation.

The *displacement-displacement* as well as the *force-displacement* coupling can be used with larger macro step sizes than the *force-force* coupling without getting unstable. However these two methods may have a very high damping influence which also leads to a frequency shift compared to the real solution and so to qualitatively wrong behaviors.

Subsystems with Different Eigenfrequencies

If the eigenfrequencies of the subsystems are different, $\hat{\omega}_A \neq \hat{\omega}_B$, all three coupling variants lead to damped subsystems and do not reflect the exact eigenfrequency $\hat{\omega}_n$ (2.82) of the rigidly coupled system. To compare the deflection of the three different methods with respect to the exact solution, the eigenfrequency error $e_{\tilde{\omega}} = \tilde{\omega} - \hat{\omega}_n$ and the damping error $e_{\tilde{\delta}} = \tilde{\delta} - \hat{\delta}_n = \tilde{\delta}$ of the modified equations are plotted for different spring stiffness rations V and fixed ratios $k_\omega = \frac{\hat{\omega}_B}{\hat{\omega}_A}$. These errors are plotted for a stiff coupling ($F = 100$) in Figure 2.16. The same qualitative effect as for equal eigenfrequencies occurs: the *force-force* coupling gives the best approximation for the exact solution, no error is visible in this plot. The *force-displacement* has a medium error and the *displacement-displacement* coupling has the largest deviation.

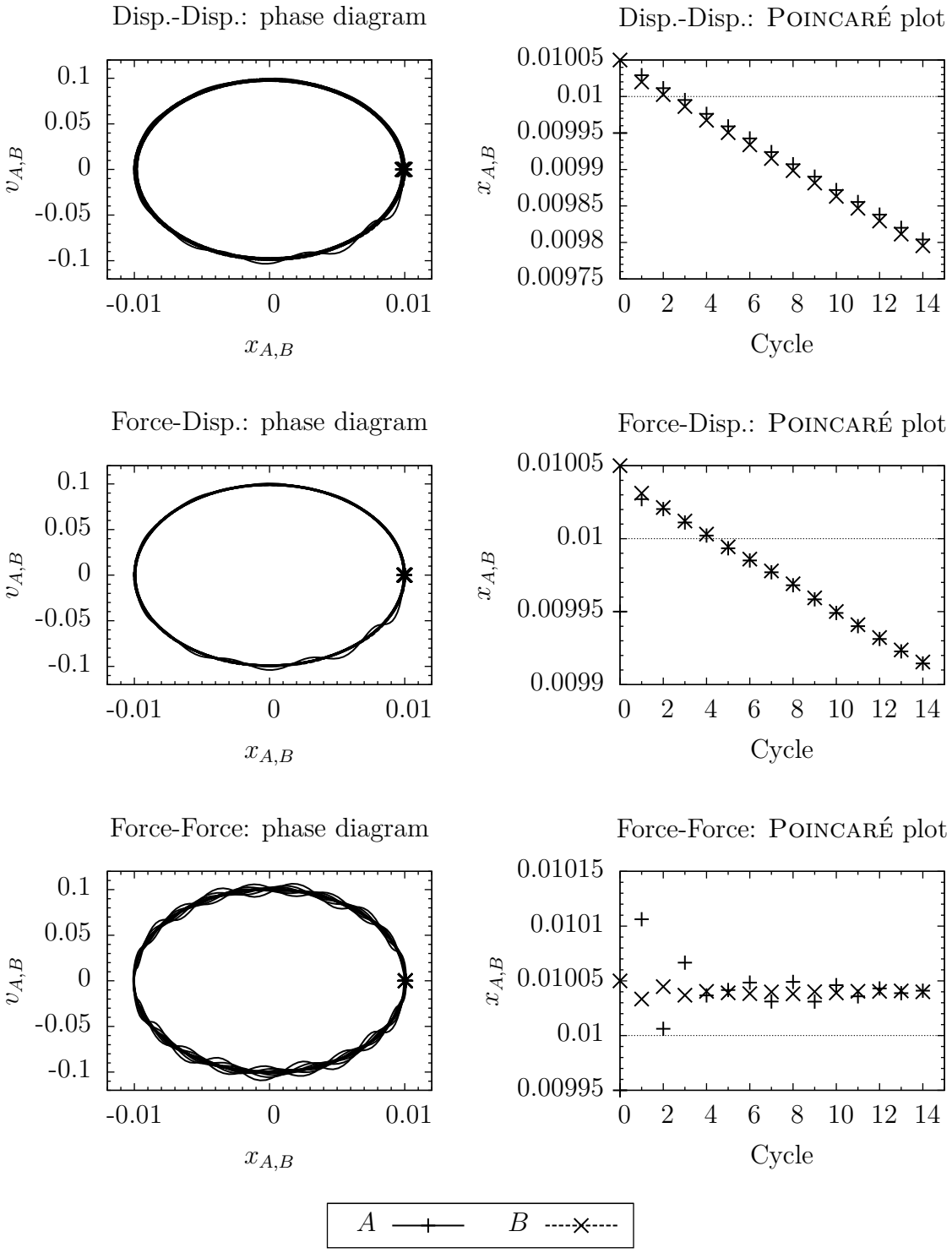


Figure 2.15: Comparison of different co-simulation methods: linear extrapolation

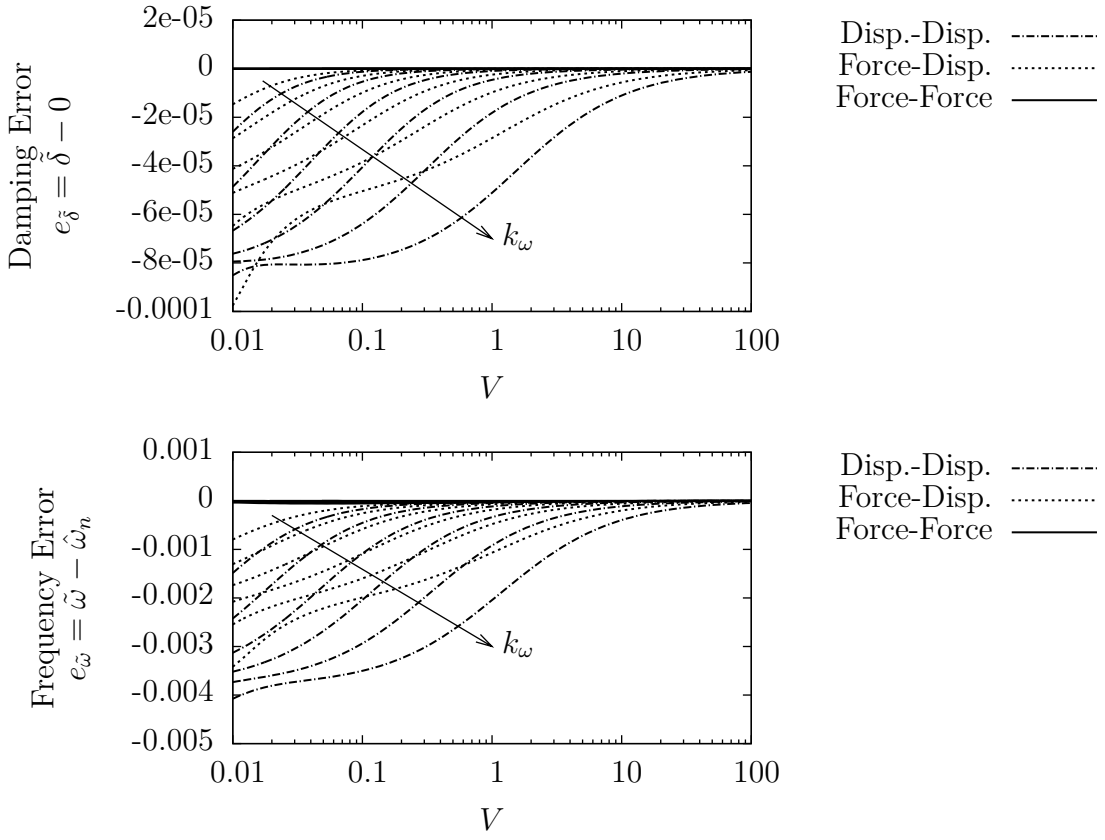


Figure 2.16: Damping and eigenfrequency error of the modified equation: linear extrapolation

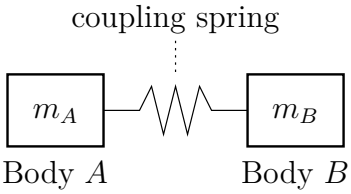
2.4 Extension to Hydraulic Couplings

Although the co-simulation framework is introduced as a domain independent approach, Section 2.3 analyzes only mechanical couplings in detail. Hence, the investigations for this section can not be directly applied to other domains. For reusing the methods and results for hydraulic systems it is reasonable to convert the mathematical equation of a hydraulic co-simulation coupling to a formulation which equals mechanical couplings. Then, the local order estimations, the stability and long time behavior of mechanical couplings can be transferred to hydraulic ones.

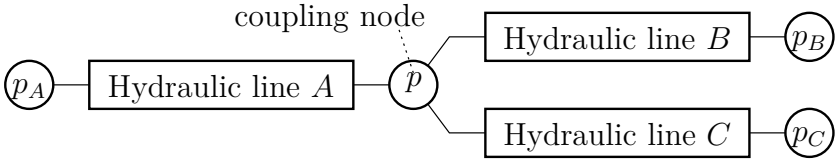
The basic dynamic elements and a coupling element is depicted in Figure 2.17 for the mechanical and hydraulic domain. The mathematical formulation of the mechanical subsystems can be represented by equation (1.4). The output $\boldsymbol{y}$ only depends on the subsystems state

$$\boldsymbol{z}_{\text{mech}} = [\boldsymbol{q}^T, \boldsymbol{v}^T]^T \tag{2.83}$$

which consists of the generalized positions $\boldsymbol{q}$ and velocities $\boldsymbol{v}$. The co-simulation coupling law, which only depends on the input of the master being the output of



(a) Mechanical coupling



(b) Hydraulic coupling

Figure 2.17: Mechanical and hydraulic couplings

the subsystems, is defined by an elastic element (see Figure 2.17(a)). In case of a linear spring, the coupling law yields

$$u_{\text{mech}} = F = c\,(y_A - y_B) + c\,\Delta x \tag{2.84}$$

where $F = u_{\text{mech}}$ is the co-simulation coupling force, c the coupling stiffness and Δx the unloaded spring length. The mathematical formulation of the hydraulic subsystems can also be represented by equation (1.4). The hydraulic subsystem state

$$\boldsymbol{z}_{\text{hyd}} = [\boldsymbol{Q}^T, \boldsymbol{p}^T]^T \tag{2.85}$$

typically consists of the hydraulic fluid flow $\boldsymbol{Q}$ and the hydraulic node pressure $\boldsymbol{p}$. The hydraulic coupling law for an elastic hydraulic node (see Figure 2.17(b)) is given according to [13] by

$$\dot{p} = \frac{E(p)}{V_K} \sum_{i=A}^{N} Q_i, \tag{2.86}$$

where Q_i is the fluid flow into the coupling node of the connected hydraulic line i, $E(p)$ the bulk modulus of the fluid and p, V_K are the fluid pressure and volume of the coupling node. In contrast to the algebraic mechanical coupling law (2.84), the hydraulic coupling law (2.86) is an ordinary differential equation of first order. Hence, the analysis for the mechanical co-simulation coupling cannot be applied to the hydraulic ones without reformulation:
Integrating equation (2.86) by separation of the independent variables yields

$$\frac{dp}{dt} = \frac{E(p)}{V_K} \sum_{i=A}^{N} Q_i(t) \quad \Rightarrow \quad \int_{p^0}^{p} \frac{dp}{E(p)} = \frac{1}{V_K} \sum_{i=A}^{N} \int_{t^0}^{t} Q_i(t)dt. \tag{2.87}$$

The calculation of the individual integrals i on the right hand side of equation (2.87) is accomplished by the subsystem, respectively the slave, each hydraulic line belongs to, because the fluid flow Q_i is a state variable of subsystem i. Hence, introducing an abstract position variable V_i with $\frac{d}{dt}V_i = Q_i$ in the subsystem and the constant V_i^0, one obtains from equation (2.87)

$$\int_{p^0}^{p} \frac{dp}{E(p)} = \frac{1}{V_K} \sum_{i=A}^{N} (V_i - V_i^0). \tag{2.88}$$

The missing evaluation of the integral on the left hand side can be done analytically dependent on the used function for the bulk modulus $E(p)$ and leads to an explicit algebraic expression for the hydraulic coupling node pressure p.

2.4.1 Constant Bulk Modulus

A constant bulk modulus is frequently used in hydraulic simulation, especially if the pressure level of the system is high, the fraction of air in the fluid is small or no temporary pressure drops to low pressures appears. In such a case, the bulk modulus can be interpreted as constant because it changes only insignificant at high pressures. Hence, equation (2.88) can be simplified to

$$u_{\text{hyd.}} = p = \frac{E}{V_K} \sum_{i=A}^{N} V_i + p_*^0. \tag{2.89}$$

All integration constants V_i^0 and p^0 are summarized to the initial value p_*^0. Extending the output vector $\boldsymbol{y}$ of the subsystem by the abstract volume V

$$\boldsymbol{y} = \boldsymbol{o}\left(\boldsymbol{z}_{\text{hyd}} = [\boldsymbol{Q}^T, \boldsymbol{p}^T]^T, V\right) \tag{2.90}$$

leads to identical formulations for the hydraulic coupling (2.89) and the mechanical coupling (2.84): the input, being an algebraic expression, only depends on the output $\boldsymbol{y}$ of the subsystems. Hence, the analysis for mechanical test systems can be carried over to hydraulic couplings.

Besides the input u, also the derivative $\dot{u}$ is required by some coupling variants from Section 2.2. Since the time derivative of equation (2.89)

$$\dot{u} = \dot{p} = \frac{E}{V_K} \sum_{i=A}^{N} Q_i = f(\boldsymbol{y}) \tag{2.91}$$

only depends on the output $\boldsymbol{y}$ of the subsystems, these couplings are also applicable for hydraulic systems.

2.4.2 Stepped Bulk Modulus

If the conditions for a constant bulk modulus are not met, piecewise constant extrapolation during macro time steps might be used. Since the pressure p^{l-1} and the abstract volume V^{l-1} at the previous macro time step are known by the master, equation (2.87) can be integrated over the current macro time step

$$\int_{p^{l-1}}^{p^l} \frac{dp}{E(p^{l-1})} = \frac{1}{E(p^{l-1})} \int_{p^{l-1}}^{p^l} dp = \frac{1}{V_K} \sum_{i=A}^{N} \int_{t^{l-1}}^{t^l} Q_i(t)dt. \tag{2.92}$$

This yields the node pressure at the current macro time step:

$$p^l = \frac{E(p^{l-1})}{V_K} \sum_{i=A}^{N} (V_i^l - V_i^{l-1}) + p^{l-1} \tag{2.93}$$

The first derivative of $u = p$ is given by equation (2.91) with $E = E(p^{l-1})$.

2.4.3 Pressure Dependent Bulk Modulus

According to [7], a common formulation of the pressure dependent bulk modulus is given by

$$E(p) = \frac{\tilde{E}\,(1+\nu)}{1 + \frac{\tilde{E}\nu}{\kappa} \cdot \tilde{p}^{(\kappa^{-1})} \cdot p^{-(1+\kappa^{-1})}} = \frac{a}{1+bp^c} \quad , \quad a, b, c \in \mathbb{R}, \tag{2.94}$$

where ν is the volume fraction of air in the fluid at the reference pressure $\tilde{p}$, κ the isentropic expansion factor and $\tilde{E}$ the bulk modulus of the airless fluid. Then equation (2.88) can be integrated analytically

$$\begin{aligned} \int_{p^0}^{p} \frac{dp}{E(p)} = \frac{1}{a} \int_{p^0}^{p} (1+bp^c)dp = \underbrace{\frac{1}{a}\left[p - p^0 + \tfrac{b}{c+1}\left(p^{c+1} - \left(p^0\right)^{c+1}\right)\right]}_{g(p)} = \\ = \frac{1}{V_K} \sum_{i=A}^{N} (V_i - V_i^0). \end{aligned} \tag{2.95}$$

This leads to an implicit algebraic equation in p which cannot be solved explicitly due to the rational exponent $c+1$ of p. However numerical experiments have shown that a solution is found by a Newton-Solver in 2 or 3 iteration steps with analytic Jacobian evaluations:

$$f(p) = g(p) - \frac{1}{V_K} \sum_{i=A}^{N} (V_i - V_i^0) \stackrel{!}{=} 0, \tag{2.96a}$$

$$f'(p) = g'(p) = \frac{1+bp^c}{a} = \frac{1}{E(p)} \tag{2.96b}$$

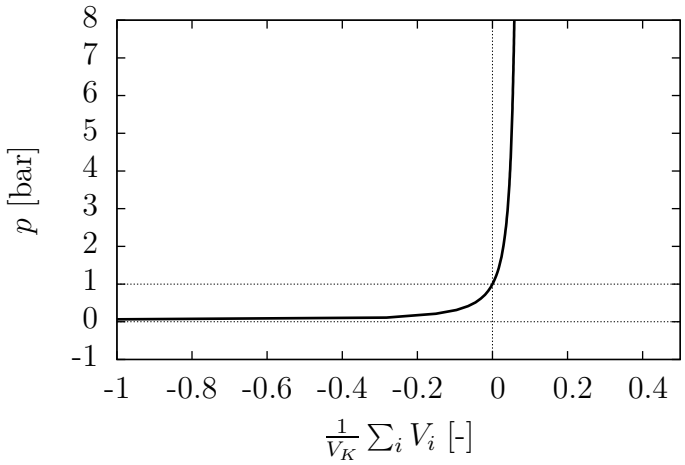


Figure 2.18: Stiffness of the hydraulic coupling node

Figure 2.18 shows the pressure p of the coupling node depending on the volume fraction of fluid flowing in or out of the node with respect to the node volume V_K, exemplarily for some realistic values: $\tilde{E} = 821\frac{N}{mm^2}$, $\nu = 0.08$, $\kappa = 1.4$, $\tilde{p} = p^0 = 1\text{bar}$. The nonlinear character as well as the very high stiffness at high pressures are revealed. They are responsible for the failure of the NEWTON-method when the initial point is chosen poorly. However, this is bypassed using the node pressure p^{l-1} from the last macro time step as initial value. The first derivative of $u = p$ is again given by equation (2.91) with $E = E(p)$.

In summary, all co-simulation extrapolation methods which have been developed for mechanical couplings can be applied to hydraulic couplings even if a pressure dependent bulk modulus is used. Therefore the abstract volume $V_i = \int Q_i \, dt$ is defined within the corresponding subsystem i and both V_i and Q_i must be included in the output vector of the subsystem. Moreover, at least for a pressure dependent bulk modulus, the master must implement a numerical solver for the iterative evaluation of the coupling pressure.

2.5 Control Couplings

Since the number of physical systems containing active elements is increasing, the modeling and simulation of active systems is very important [78]. Hence, a third kind of very common couplings in the field of multi domain simulations are control couplings. In contrast to mechanical and hydraulic systems, where the input and output was introduced due to mathematical or implementation considerations, the input and output is already an essential part of control theory: A controller reads the sensor output of the system, calculates the input signal using the control law and sends it to an active element of the system. Hence, control couplings have an information flow only in one direction. The point of action of the control input and output need not to be the same point nor at the same subsystem. Schematically

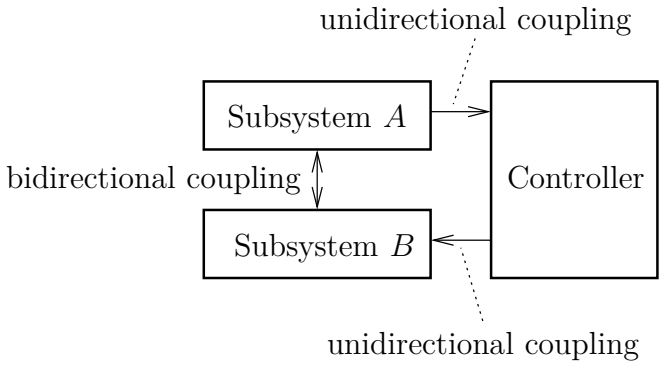


Figure 2.19: Two subsystems with a controller

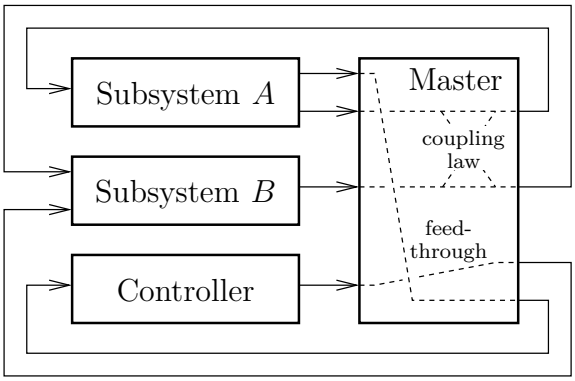


Figure 2.20: Master-slave concept with two subsystems and a controller

this is depicted in Figure 2.19: two subsystems are coupled using a bidirectional connection, for example a mechanic spring, and the controller communicates only unidirectional but between different subsystems. Additionally, arbitrary information concerning the physical measurement, for example accelerations, may be used.

2.5.1 Embedding to the Master-Slave Concept

To use a control subsystem in the master-slave concept of the presented co-simulation framework, it is necessary to formally restructure the input and output signals of a controller using the intermediate master as shown in Figure 2.20. The master is responsible for the calculation of the coupling law (for example a mechanical spring) including the coupling extrapolation for the bidirectional coupling between the subsystems A and B. For the input and output of the controller, the master simply feeds-through the output of subsystem A to the input of the controller and the output of the controller to the input of subsystem B. Hence, a control subsystem can be integrated to the master-slave concept like other subsystems. Since the special coupling approximations for mechanical and hydraulic couplings cannot be used for a unidirectional control coupling, the local order and stability of control couplings should be investigated separately.

2.5.2 Input Extrapolation, Local Order and Stability

The analysis of the time discretization and the extrapolation of the input, induced by the macro time steps, is not relevant for control couplings concerning co-simulation because at least digital controlling already implies a discretization in time, called *sampling*: the continuous system output, gained by sensors, is sampled and the controller is acting as a discrete, digital real-time controller. Therefore, the discretization introduced by the co-simulation is already part of the physical model of the controller, even if the control sample rate will differ from the co-simulation macro step size. The same applies for the extrapolation of the input: a digital controller keeps the controller output (input of the subsystem) constant during a sampling interval. Hence, the simplest form of the co-simulation input extrapolation (constant extrapolation) represents already the exact behavior of the real system. All appearing effects, like aliasing [10], are not specific to co-simulation of controlled systems, but have to be solved already for the underlying controller.

The last missing part is the stability analysis of control couplings. The sampling rates of typical controllers are in the range of about 1kHz to 10kHz. A typical macro time step for a co-simulation of mechanical subsystem is about $1 \cdot 10^{-6}s$, which equals 1000kHz. Since a co-simulation macro time step is significantly smaller than the sampling rate of the controller and the controller is stable at the controlling sampling rate, it will also be stable at the co-simulation macro time step. However, this analysis is topic of control theory [59].

Another problem, relevant for every discrete subsystem, is the clash of more than one discretization in time: the co-simulation macro time steps and the time steps of the discrete system. There exist different possibilities to resolve this problem [28]. The easiest way is to restrict the sampling step size of the discrete system to be an integer multiple of the co-simulation macro time step size.

2.6 Other Domains

This section presents a general approach for co-simulation couplings for arbitrary domains conforming the master-slave concept.

2.6.1 Domain Specific Modeling Interfaces

A well known way for defining domain specific interfaces is the *bond graph* approach [34], which is motivated by the energy flow between connected components. Similar is the approach used by the MODELICA® language [47,54]: The coupling variables are split into potential and flow variables. Potential variables P are defined to coincide at the coupling points usually often called KIRCHHOFF nodes

$$P_i = P_j, \quad i \neq j;\ i,j \in [1 \dots N]. \tag{2.97}$$

Domain	Potential Variable	Flow Variable	Basic Dynamic Element; Output	Basic Coupling Element; Input
Translational mechanic	Position x	Force F	$m\ddot{x} = \sum F$ $y = [x, \dot{x}]^T$	$u = F = c_t\, \Delta x$
Rotational mechanic	Angle φ	Torque M	$J\ddot{\varphi} = \sum M$ $y = [\varphi, \dot{\varphi}]^T$	$u = M = c_r\, \Delta\varphi$
Hydraulic	Pressure p	Volume flow Q	$\frac{\rho L}{A}\dot{Q} = \Delta p$ $y = [V, Q]^T$	$\dot{p} = \frac{E}{V_K}\sum Q$ $u = p = \frac{E}{V_K}\sum V$
Thermal	Temperature T	Heat flow $\bar{Q}$	$C_t\, \dot{T} = \bar{Q}$ $y = T$	$u = \bar{Q} = \lambda\frac{A}{d}\, \Delta T$
Electric	Electric potential V	Electric current $\imath$	$L_e\, i = \Delta V$ $C_e\, \Delta\dot{V} = \imath$	$\imath = G\, \Delta V$
Magnetic	Magnetic potential Ψ	Magnetic flux Φ	no basic dynamic elements	$\Phi = G_m\, \Delta\Psi$

with mass m, translational stiffness c_t, moment of inertia J, rotational stiffness c_r, fluid density ρ, line length L, cross section area A, abstract volume V, bulk modulus E, node volume V_K, heat capacity C_t, heat conductivity λ, thermal conduction length d, electric inductance L_e, electric capacitance C_e, electric conductance G and magnetic permeance G_m.

Table 2.9: Interface variables used by MODELICA

In contrast, flow variables f sum-up zero at a KIRCHHOFF node

$$\sum_{i=1}^{N} f_i = 0. \tag{2.98}$$

Table 2.9 shows the potential and flow variables for some domains used by MODELICA. Further, it presents the differential equations for the basic dynamic elements and the equations for the basic, mostly algebraic, coupling elements for each domain. Note that the hydraulic coupling element is the only one which is not an algebraic equation, but hydraulic couplings were already investigated separately in Section 2.4.

2.6.2 Selection of Input and Output Variables

To prevent an algebraic loop in the input variables, the subsystems output $\boldsymbol{y}$ for the master-slave concept is restricted to functions only depending on the state of the subsystems. These state variables are always appearing together with their time derivative in Table 2.9. An input can be any variable appearing in the differential equation of the subsystem, but must not appear in the output function. Moreover, the master must be able to calculate the input for the subsystems using only their

states. These two assumptions are fulfilled for the mechanic, hydraulic and thermal domain. Note, that for the hydraulic domain the abstract volume V must be introduced as shown in Section 2.4. To ensure this characteristic, couplings in the thermal domain are restricted to co-simulation couplings not depending on the derivative of the coupling value

$$\dot{u} = \dot{\tilde{Q}} = \lambda \tfrac{A}{d} \, \Delta \dot{T}, \tag{2.99}$$

because the derivative of the temperature $\dot{T}$ is not a state value and thus is not included in the output vector of a thermal subsystem.

The last missing domains in Table 2.9 are electric and magnetic systems. These two domains are closely related concerning modeling and their physical interaction. However the modeling of these domains differs a lot from the previous domains. For the electric domain, it depends on the basic dynamic element whether the state variable is the potential variable V or the flow variable $\imath$. Hence, there is no a priori definition of the state variables for this domain. This is also reflected in the typical modeling of electric or magnetic networks: they are build by setting up the KIRCHHOFF equations (node rule and mesh rule) [39] and not by adding new elements to the global equation structure as done for example by mechanical systems. Since the electric and magnetic domain is not the focus of this work, the couplings inside this domain are not discussed in further detail.

2.6.3 Inter Domain Coupling

Using couplings inside a given domain it is possible to couple different simulation tools of the same domain using co-simulation. Moreover the model of a given domain can be split into different subsystems whereby the possibility of a simulation speedup arises if a parallel co-simulation is used. Besides this benefit, co-simulations are mostly used to couple models of different domains, but till now only the coupling inside a domain was shown.

For multi domain simulation, the physical model of the overall system must also include the physical model of the interface between two domains. For example a hydraulic system is mostly used to actuate a mechanical system. The physical interface between these two domains is for example a hydraulic piston. The models of such inter domain interfaces are typically built into one of the involved domains. In case of a hydraulic-mechanical system, the hydraulic system has a model of a piston included. Hence, the hydraulic system can even handle simple mechanical systems like a piston. In this case the co-simulation between a hydraulic and mechanic system can be done using the same couplings as for pure mechanical subsystems. The same applies for inter domain co-simulations between other domains with own specific coupling elements.

3 Implementation

In the last chapter the physical and mathematical modeling of subsystems of different domains and especially of a parallel co-simulation framework has been discussed in detail. To solve the global co-simulated system numerically an implementation in form of a computer program is necessary. This implementation is non trivial, since it must deal with different preexisting subsystems as well as with multi threads or processes and communication methods to be able to handle parallelization. Hence, this section shows the principle implementation of the parallel co-simulation using the master-slave concept.

Based on the need for performance, scalability and code reusing the language of choice for the co-simulation is the C++ programming language. This language offers besides the object orientated structure also very good access to operating system functions which are necessary for different types of inter-process-communication. Moreover, linking C++ code with other preexisting simulation tools or libraries works, no matter if they are implemented in C++, C or Fortran being the most common languages for technical simulation tools.

3.1 Inter-Process-Communication (IPC)

The master-slave concept is implemented using a *process coupling* to be able to share the computational work of the subsystems not only on multi-core computers but also on different computers in a cluster. Hence, an inter-process-communication is required, transferring the input and output variables between the subsystems and the master. The principle methods for inter-process-communication and the main operating systems providing these methods are shown in Table 3.1 according to [73].

The *file* method is feasible for the exchange of very large datasets with low communication frequency. It offers the advantage that the exchanged data is automatically archived in the file. If the file is stored on a network file system, the communication can also be done between computers arranged in a network.
Using *signals* it is only possible to send a very limited amount of information to another process. Mostly, this method is only used to notify other processes about events. This type of communication is very fast but is not available across networks.
*Socket*s or *internet-socket*s are one of the most known inter-process-communication variants. The endpoints of a communication over Internet-Protocol (IP) based networks are called *socket*s, forming the basics of the communication in the World Wide Web (WWW). It is self-evident that this kind of communication is available across

IPC Method	Providing Operating Systems
file	nearly all operating systems
signal	most operating systems
socket	most operating systems
pipe	POSIX-systems, MICROSOFT-WINDOWS, ...
shared memory	POSIX-systems, MICROSOFT-WINDOWS, ...
semaphore, mutex	POSIX-systems, MICROSOFT-WINDOWS, ...
message passing	depending on the availability of for example MPI or PVM

Table 3.1: IPC methods

Communication Method	*Round Trip Delay Time*
socket (*ethernet*; local network; not routed)	≈ 0.140 ms
socket (*loopback*; host intern)	≈ 0.035 ms
signal, *shared memory* (host intern)	$\ll 0.001$ ms

Table 3.2: *Round trip delay time* of IPC-methods

multi computers but also inside one computer using the so called *loopback*-interface. The communication speed depends highly on the used interface (*loopback*, *ethernet*, ...) but also on the transport protocol (for example *TCP* or *UDP*). In contrast to *signal*s the communication speed is several magnitudes slower. Table 3.2 shows the typical *round trip delay time* of different communication methods which describes the time delay between sending a message and receiving the answer.
The IPC methods listed above are implemented on nearly all modern operating systems. The following methods are provided by all operating systems conforming to the POSIX-standard [33], for example GNU/LINUX, BSD, MAC OS X and others, as well as by MICROSOFT®-WINDOWS®.
*Pipe*s are similar to *file*s but with direct feed-through between the writing and the reading process without storing the transmitted data on the file-system. They are mostly used in shell scripts and not very common for inter-process-communication in case of co-simulation.
Using *POSIX shared memory* [33] one declares an area of memory accessible to other processes in addition to the allocating process. Thereby, the need for synchronization of read/write accesses arises like for threaded programs. *Shared memory* is only available on one unique host computing the entire simulation but not across computers in a network. The use of *system memory* for communication without an administrative overhead makes this method the fastest inter-process-communication method.
*Semaphore*s and *mutex*es are not inter-process-communication methods for the exchange of data but for the synchronization of the chronologic flow of the processes and of the read/write access to shared data. *Binary semaphore*s are closely related to *mutex*es and are used to lock and unlock the access to shared data atomically, preventing inconsistent data values. *Counting semaphore*s can be used effectively to synchronize the program flow in different processes offering their usage in the

Socket	*Message Passing*	*Shared Memory*
+ **single- and multi-host** + automatic synchronization	+ **single- and multi-host** + operating system comprehensive + automatic synchronization + easy to use	+ **very fast**
- operating system dependent - **slow communication**	- overhead due to general approach - **slow communication** (depending on the implementation)	- operating system dependent - only single host - manual synchronization needed (more complex to implement)

Table 3.3: Benefits and drawbacks of different IPC-methods

master-slave concept of the co-simulation.

Another type of inter-process-communication, differing from the previous ones, is *message passing*. It is a high level language for the above IPC methods. Internally *message passing* is implemented using the IPC methods provided by the operating system. The implementation is either done directly in the programming language of a specific hardware or operating system, or as an external library using it through library calls. In both cases *message passing* is used to send messages, which can be signals, simple data or complex data structures, from a sender to one or more recipients. Depending on the routine used to send or receive a message, there are synchronization mechanisms already included by a blocking mechanism. A very common *message passing* method is the *Message Passing Interface (MPI)* [48]. Current implementations of MPI are MPICH2 [49], OpenMPI [53] (the successor of LamMPI) or Boost.MPI [12], all available for POSIX-systems, Microsoft-Windows and also for some specialized hardware in high performance computing. Due to the specialized implementation of the MPI standard for each system, the hardware can be optimally used without changing the source code of a program. Since MPI is available for different operating systems, a heterogeneous pool of computers can be connected to a single parallel computer. Another type of common *message passing* is the Parallel Virtual Machine (PVM) [22] which is closely related to MPI but is an independent standard.
The main advantages and disadvantages of the most important IPC methods, which are useful for co-simulations, are summarized in Table 3.3 where the most important aspects are shown in bold font. A benchmark for these IPC methods in case of a co-simulation using the master-slave concept in given in Section 4.1.

3.2 MDPCosim

In practical examples of co-simulations the number of subsystems acting as slaves is limited to a small value of usually less than 10. Therefore the use of high performance computer systems with hundreds or thousands of central processing units (CPUs) is not reasonable. Hence, the use of several computers in parallel in form of a small computer cluster or modern shared memory multi-CPU or multi-core workstations are good approaches to parallelize co-simulations. If there are subsystems which themselves use massive parallelization the total number of CPUs may be much higher, but the number of subsystems is mostly still limited to only several ones.

The implementation of the master-slave concept developed in this work is called *m*ulti-*d*isciplinary *p*arallel *co-sim*ulation (MDPCosim) and supports currently the IPC methods *shared memory*, PVM, LamMPI and OpenMPI. The actual inter-process-communication is chosen at compilation time of the master and the communication classes of the slaves. The topmost part of the co-simulation is a shell script named `mdpcosim`. This script is responsible for the potential requirement of initialization of the inter-process-communication method as well as for the invocation of the master and the slave processes. The master and all slaves are separate programs to be able to run on different hosts in a computer cluster. In case of *shared memory* the `mdpcosim` script calls the master and all slave programs. In case of PVM the `mdpcosim` script calls the master program, being the entry point for PVM, and this program spawns the slave programs using PVM function calls. Using MPI the script calls `mpiexec` with a parameter set for the master and all slaves. Thereby, the initialization and process startup of the different inter-process-communication methods are enclosed from the user, calling always only the `mdpcosim` script without noticing the used IPC method which was defined previously.

3.3 Master

Figure 3.1 shows a UML [11, 51] diagram of the class structure used by the master. The **main** subroutine comprises a collection of ***Connection*** and **SlaveCom** objects. The ***Connection*** class provides a pure virtual function *`updateu()`* which must be implemented by a derived class using the coupling law (2.2b) including the extrapolation. For each connection between subsystems a separate object of type ***Connection*** is generated. The individual connections are grouped by an intermediate abstract class for the different domains. The `updateu()` function calculates and sets the new input $\boldsymbol{u}$ using the output values $\boldsymbol{y}$ being set by the **SlaveCom** class. This class holds the input and output vectors as attributes.

For each slave, representing an individual subsystem, the main routine of the master generates also an object of type **SlaveCom** being responsible for the communication between the master and the appropriate slave, mainly the exchange of input and output. For this communication the routines `recv` and `send` are always used independent of the connection type, the IPC method or the subsystem. Hence, changing the

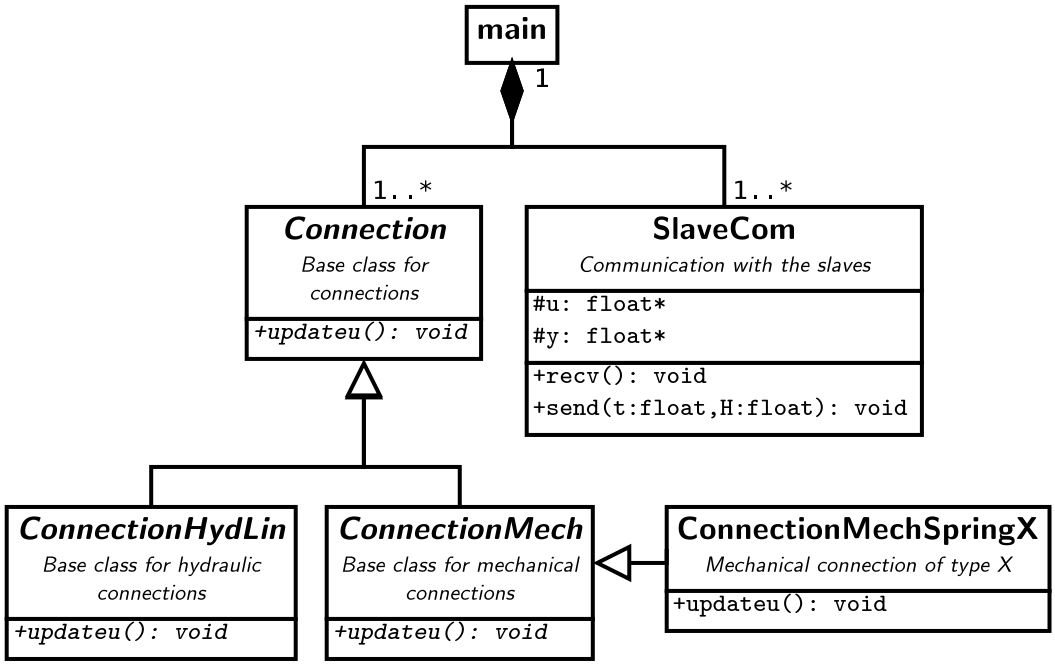


Figure 3.1: UML diagram of the master

implementation of these routines gives the possibility to change the inter-process-communication method used by the co-simulation for all subsystems without the need to change existing subsystems. The `recv` function is blocking the execution of the master until the required data is completely sent by the subsystems and thereby guarantees the synchronization of the subsystems and the master.

The configuration of the master consists of three files:

`mastersys.dat` defines the subsystems and the name of their individual configuration files `<slave>.dat`. Moreover, this file contains the hostname of the machine where the subsystem is executed in case of an IPC method being able for inter-host-communication.

`masterpar.dat` is a simple file defining only the simulation end time and the constant macro time step size H.

`mastercon.dat` defines all connections between the subsystems. Each connection consists of the type, the number of connected subsystems, the indices of the input $\boldsymbol{u}$ and output $\boldsymbol{y}$ vector the connection acts on as well as connection specific parameters, for example the mechanical stiffness.

Besides management functionality the main subroutine of the master implements the outer macro time integration loop. A flowchart diagram of this loop is depicted in Figure 3.2. After parsing the configuration files and initializing all objects of type **SlaveCom** and **Connection**, the initial output vectors $\boldsymbol{y}$ of the subsystems are received. Afterwards all connections calculate the input $\boldsymbol{u}$ for the first macro time step at $t^0 = 0$. This input is sent, using the defined IPC method, to the subsystems using the `send` function of **SlaveCom**. The following call of the `recv` function is

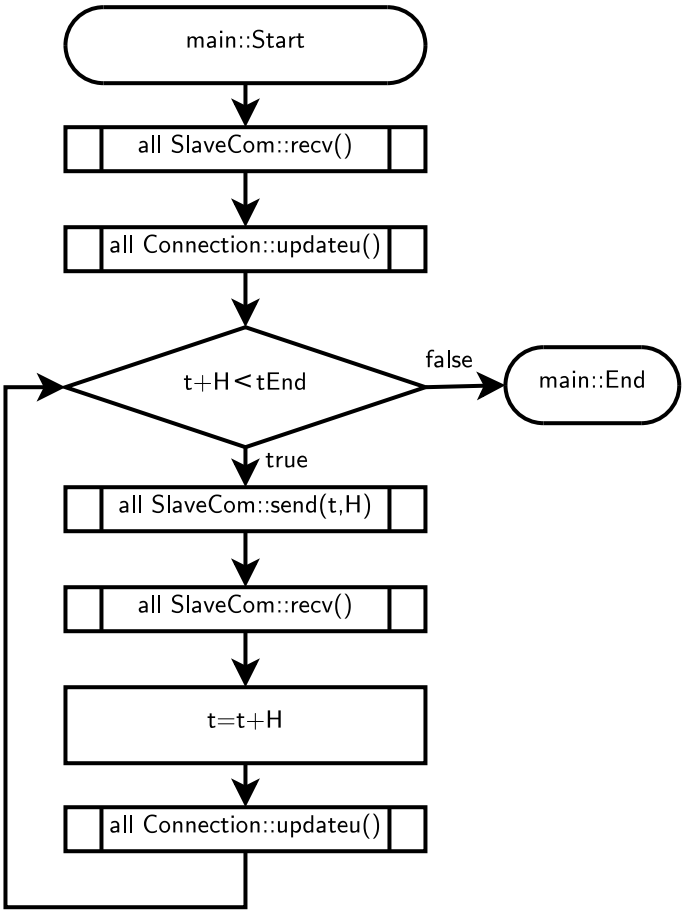


Figure 3.2: Flow chart diagram of the main routine of the master

blocking further execution until all subsystem have finished the integration of the current macro time step and have sent their new outputs to the master. At this point one macro time step integration is finished. The macro time is increased by the macro time step size H and the loop starts again after the connections have updated their input vector $\boldsymbol{u}$.

3.4 Slaves

Figure 3.3 shows an UML diagram of the basic communication classes used to extend a preexisting simulation tool by the functionality needed to communicate with the master of the co-simulation. The virtual class ***Slave*** implements a `send` and a `recv` function which are responsible for sending and receiving the input and output data. Alike for **SlaveCom** of the Master, the `recv` function is implemented as a blocking function responsible for the synchronization between slaves and master at the macro time steps. The virtual function `updatey` must be implemented by each slave. This function collects all output values from the subsystem and copies these into the output vector $\boldsymbol{y}$ which is provided as an attribute by the class ***Slave***.

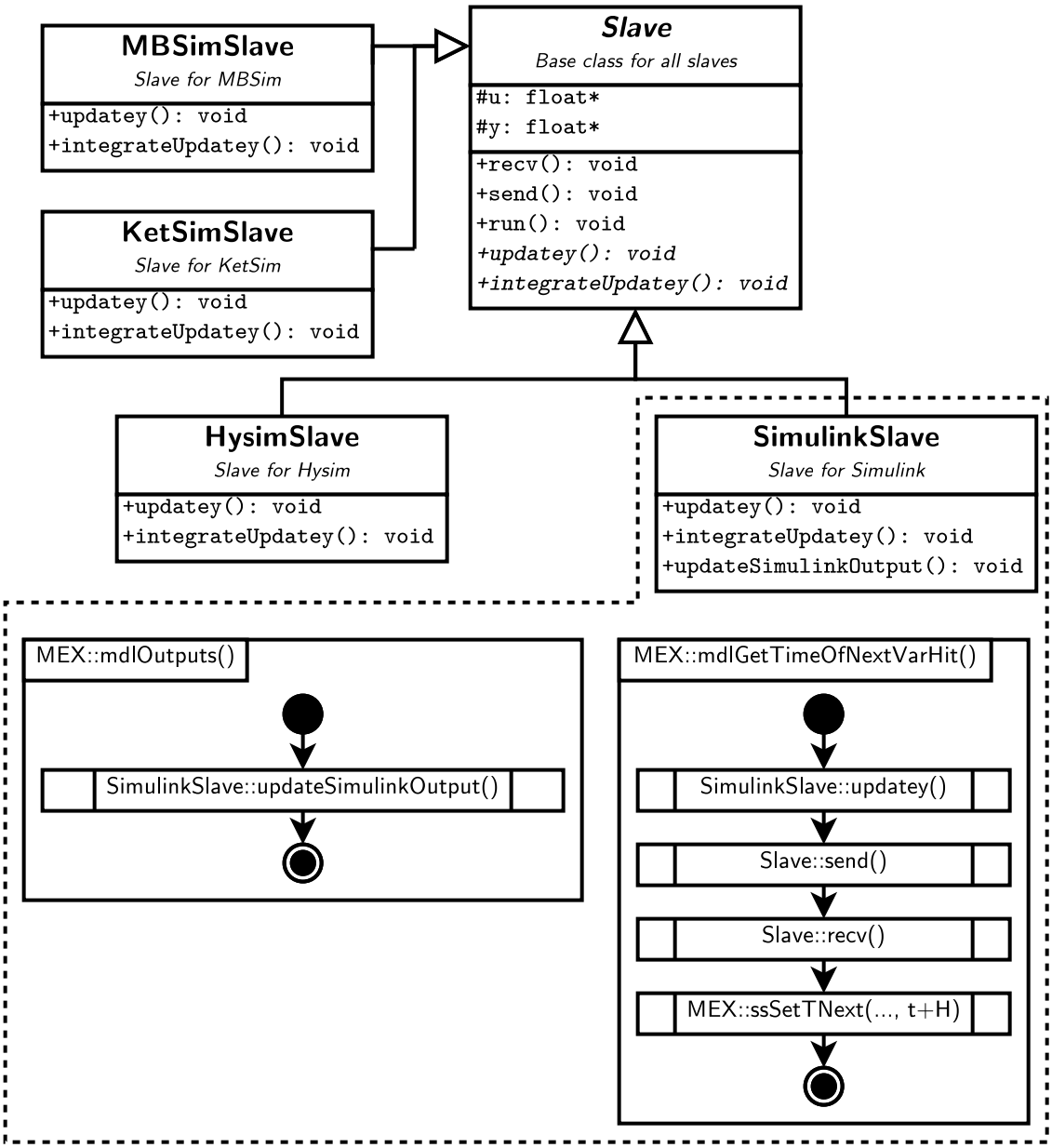


Figure 3.3: UML diagram of the slave

Each slave requires a configuration file `<slave>.dat`. Besides the number of input and output variables this file also defines subsystem specific parameters or input files needed for example for the initialization of the subsystem startup. Moreover, the connection of the individual elements of the input and output vectors to subsystem model variables are defined in this configuration file.

Each subsystem must implement the counterpart loop to the main routine of the master, see Figure 3.2. Such a macro time step loop in defined by the convenience function `run` defined in **Slave**. This function calls at simulation start one times the abstract function *`updatey`* and repeatedly, one times for each macro time step, the abstract function *`integrateUpdatey`*. Hence, a slave must only implement these

two functions. The *integrateUpdatey* function must carry out one macro time step integration of the subsystem and querying afterwards for the new output, which is done by calling *updatey*.

This procedure is only possible if the subsystem is able to carry out a single macro time integration step and to continue later after an update of the subsystem input. This simple procedure is only for few preexisting tools possible and shown for some in Sections 3.4.2 to 3.4.4. Especially for closed source simulation tools the process of implementing the tools as a co-simulation slave is more complex and shown exemplarily in Section 3.4.5 for MATLAB/Simulink. But first the general requirements on a preexisting simulation tool to attend the master-slave co-simulation is shown.

3.4.1 Subsystem Requirements

A main aspect of the introduced master-slave co-simulation is the possibility to include a large number of different simulation programs as slaves. Hence, the requirements on a subsystem should be minimal: First, the integrator of the subsystem simulation tool must be able to stop the integration at discrete times and to call a user defined function being responsible for the communication with the master process. Second, functions from external software libraries (C, C++ or Fortran) must be callable from a user function. This is necessary because all inter-process-communication methods described in Section 3.1 are implemented as functions defined in the C, C++ or Fortran language. Alternatively tools which provide access to standardized IPC methods like MPI can be used. However, this restricts the slave to this IPC method.

These two requirements are fulfilled for many commercial software tools. The first requirement is already satisfied if the simulation tool is able to handle time discrete events. The second is often fulfilled since external user functions written in C, C++ or Fortran are widely used to extend simulation tools. Two such systems are for example the multi disciplinary simulation tool Dymola® [17] and the multi body simulation software SIMPACK® [72]. Both are able to join the parallel co-simulation framework based on an interface being very similar to the implementation for Simulink shown in Section 3.4.5.

3.4.2 MBSim

The simulation program MBSim [46] is a software for the dynamical analysis of physical systems. The main focus of this software is the analysis of nonsmooth multi body systems being characterized by uni- and bilateral contacts and impacts leading to discrete jumps of the system velocities [18, 56]. An introduction to the basic concept of MBSim is given in [62]. Besides rigid multi body systems MBSim is also extended to elastic bodies [65, 80, 81] as well as to hydraulic systems [68] and control models.

Different specialized time integration methods are available in MBSim: Time-stepping schemes [18, 74, 75] with fixed step size solve impacts without the resolution of impact times. Step size controlled integrators with root finding [26, 27, 29] can be selected in addition to detect the times of impacts and to solve the impact equations at this discrete times. The contact solution in MBSim is implemented in form of a fixed point problem using a proximal point notation of the set-values force laws [18], where several numeric solvers are applicable.

Since MBSim is available as open source software, the integrator is accessibly on programming level. Hence, it can be modified slightly to match the requirements for using the convenience functions `run` and *`integrateUpdatey`* introduced in Section 3.4. At initialization time, pointers to the input and output elements are stored. These are used in the `updatey` function to read the output data from MBSim and in the `integrateUpdatey` function to set the input data of MBSim before the integration up to the next macro time step.

3.4.3 KetSim

The simulation of timing chain drives is state of the art for the analysis of the interaction between the crank shaft and the cam shafts in combustion engines. Current research activities on chain drive simulation are for example acoustic and friction optimizations. Timing chain drives can be incorporated in standard multi body simulation tools. However, a large amount of specialized code needs to be included to take care of the typical structure of chain drives to avoid unnecessary overhead for example in the contact search algorithm. Several commercial tools have chosen this way [1, 57, 72]. Another approach is to develop specialized chain drive tools like for example KetSim [20, 32, 35] or AVL Excite Timing Drive [6]. The advantage of this method is the freedom of the design of the simulation tool which can be utilized to achieve high computational efficiency. The drawback is that even for an interaction with other mechanical systems a co-simulation is required due to in general small capabilities beside the simulation of chain drive components. An advantage especially of KetSim is the use of rigid, unilateral contact laws between the chain links and the wheels or chain guidings.

The numerical integration in KetSim is done using a step size controlled Runge-Kutta integrator of order 2/3 [26] which is extended by a root finding algorithm to detect the shift points of opening and closing rigid contacts. The rigid contact problem is formulated as a linear complementary problem solved with the Lemke-LCP solver [20].

The correct approach of the discrete macro time steps is done using events in time causing small changes of the integrator code because the original software is not able to interrupt at discrete points in time. These points must be treated separately from those related to mechanical contact roots to be efficient. Such changes are possible in KetSim since it was available as source code for this work. After this

modification the implementation and configuration of KetSim as a slave is analog to the MBSim-slave shown in Section 3.4.2.

3.4.4 Hysim

Hysim [13] was originally developed as research software aiming to reduce the simulation time of hydraulic systems by exchanging elastic hydraulic elements like hydraulic nodes by bilateral rigid or, in cases where cavitation must be accounted for, by unilateral rigid elements. In analogy to multi body systems with uni- and bilateral rigid contact models, this approach avoids high stiffnesses of small fluid volumes. The simulation of hydraulic chain tensioners [32] including the verification against measurements of dynamical actuated tensioners [37] have become the main scope of Hysim in industrial applications.

The numerical integration, root finding and contact solving in Hysim is nearly equal to the implementation used in KetSim. The same applies for the modification of Hysim allowing to carry out macro time integration steps and the implementation of the **HysimSlave** class being used by the co-simulation.

3.4.5 MATLAB-Simulink

MATLAB®-Simulink® [44] is a well known and widely used simulation tool where the model is build using block diagrams. The main field of application is the design of controllers often used in hardware in the loop simulations. Besides the integration of continuous models, being formulated as ordinary differential equations or differential algebraic equations, Simulink is able to integrate discrete models even with different sampling rates. Also mixtures of continuous and discrete models can be used.

Simulink provides a large number of numerical integrators. The different solvers include integrators with fixed and variable step size, explicit and implicit ones, integrators for continuous, discrete or mixed continuous-discrete systems, with constant or variable order as well as single- or multistep integrators. Important with respect to the co-simulation is the availability of integrators for discrete systems. These integrators are able to handle the stopping points requested by the macro time integration.

In contrast to subsystems being available as source code it is not possible to setup this slave using the convenience functions from the ***Slave*** class. Hence, the implementation of Simulink as a co-simulation slave is more complex, see Figure 3.3. One possibility to extend Simulink is to use so called S-Functions: an S-Function acts in Simulink like a normal block, whereas the internal dynamic of an S-Function is implemented by the user using a programming language like C or C++. For the co-simulation a special S-Function block is communicating between Simulink and the master of the co-simulation. The output vector $\boldsymbol{y}$ of the Simulink-subsystem is the input of the S-Function block and the output of the S-Function block is

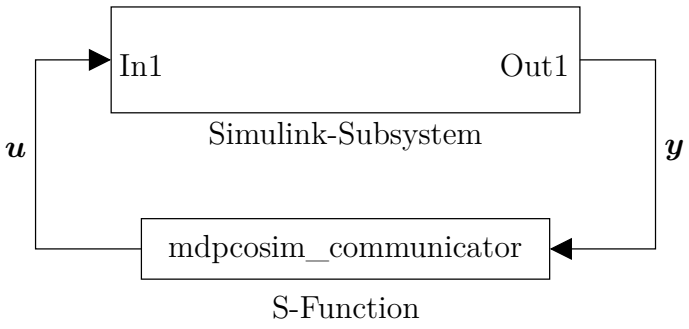


Figure 3.4: SIMULINK subsystem and IO communicator for the co-simulation

the input vector $\boldsymbol{u}$ of the SIMULINK-slave, compare to Figure 3.4. Besides the implementation of different functions for the initialization, starting and termination of the S-Function block, the two most important functions `MEX::mdlOutputs` and `MEX::mdlGetTimeOfNextVarHit` of the S-Function block are shown in Figure 3.3. The SIMULINK function `MEX::mdlOutputs` is called for each S-Function block at arbitrary micro time integration steps, so continuous access must be ensured by this function. Hence, this function is responsible for the calculation of the extrapolated input $\tilde{\boldsymbol{u}}$ (1.8) at the current micro time t using the slave input values $\boldsymbol{u}$. The `MEX::mdlGetTimeOfNextVarHit` function is called by SIMULINK for a block when the current simulation time reaches the next sample point of this discrete block. This function is used to update the output values $\boldsymbol{y}$ of the slaves by reading the input of the S-Function block (`SimulinkSlave::updatey`), sending own output to the master (`Slave::send`), blocking the SIMULINK simulation until the new input has been received (`Slave::recv`) and setting the next SIMULINK sample point to the next macro time step (`MEX::ssSetTNext`). Sending and receiving of data is done again by the abstract base class ***Slave*** and can thus be exchanged without changing the S-Function block being specific for the SIMULINK-slave.

4 Examples

The modular implementation of the parallel co-simulation framework in Chapter 3 opens this framework to a wide range of subsystems and examples, from academic test examples up to large and complex industrial applications. The examples in this section reach from benchmark problems for inter-process-communications and coupling extrapolations to complex multi domain examples in the area of combustion engines. All examples use the introduced software MDPCosim from Section 3.2, in conjunction with the slaves listed in Section 3.4.

4.1 IPC-Benchmark

Since inter-process-communication is a pure information technology topic it does not influence the result of the numerical solution. However, the communication has a significant effect on the overall simulation time of the co-simulation. Especially, small macro time steps force very frequent communication. Hence, the time overhead for the communication can get significant with respect to the pure simulation time in case of frequent inter-process-communication. Moreover co-simulations on a computer cluster connected for example by the *ethernet*-interface will be much more critical than co-simulations on a single host multi processor computer. This effect was already addressed theoretically by the *round trip delay time* in Table 3.2.

4.1.1 Model Setup

To analyze the communication overhead of different methods and communication architectures, a very simple test model and a valve train including a chain drive are used as benchmark problems.

Simple Test System

The simple test system for the IPC benchmark is the linear mass-spring oscillator used for the stability analysis in Section 2.3.2, Figure 2.8. This system is co-simulated using a relatively small and a relatively large macro step size to show the influence of the macro step size on the communication overhead.

IPC-Method	Single-Host 1 CPU	Single-Host 2 CPUs	Multi-Host 1 CPU per Host
SHM	1.00	1.56	-
PVM	1.00	1.56	1.56
LamMPI	1.00	1.53	1.56

Table 4.1: IPC benchmark: simple test system: speedup using large macro step size

IPC-Method	Single-Host 1 CPU	Single-Host 2 CPUs	Multi-Host 1 CPU per Host
SHM	5.48	5.11	-
PVM	1.00	1.10	0.35
LamMPI	(0.02)	(0.02)	(0.02)

Table 4.2: IPC benchmark: simple test system: Speedup using small macro step size

Valve Train with Chain Drive

The second IPC benchmark example is more practical: a valve train, for co-simulation purposes separated into the intake and exhaust part as own subsystems modeled using MBSim and a timing chain drive modeled using KetSim. The macro step size is the maximal stable step size. The subsystems are coupled by two rotational springs between the intake and exhaust shaft on one side and the corresponding chain wheels on the other side.

4.1.2 Comparison of IPC Methods

Both test systems are co-simulated with the different IPC methods introduced in Section 3.1, on single host and multi host architectures and compared regarding the simulation time.

Simple Test System

Table 4.1 and 4.2 show the simulation speedup

$$\Psi = \frac{t_{1\text{CPU}}}{t_{n\text{CPU}}} \tag{4.1}$$

of the simulation time $t_{n\text{CPU}}$ using a co-simulation with n CPUs with respect to a reference simulation done using PVM on a single host one CPU machine for a large and a small macro step size. The three different IPC methods, *shared memory* (SHM), PVM and LamMPI, are compared using a simulation on a single host with one or two CPUs and on a multi host cluster using one CPU per host. As outlined in Section 3.1 *shared memory* is available only for single host simulation.

IPC-Method	Number of CPUs	Time [min]
SHM	2 (single host)	51
PVM	2 (multi host)	52
LamMPI	2 (multi host)	(312)

Table 4.3: IPC benchmark: valve train: simulation time

In case of a large macro step size only few communications are needed and all three IPC methods perform nearly equally, compare to Table 4.1. The simulation is by a factor of about 1.5 faster if the subsystems are calculated in parallel on different CPUs. This is related to the very low fraction of communication overhead with respect to the simulation time of the subsystems. In case of a small macro step size, see Table 4.2, the shared memory (SHM) IPC method performs better by a factor of about 5 than PVM. This is related to the communication using shared memory which is much faster than the host internal communication used by PVM in case of single host communication. The LamMPI implementation even increases simulation times. This reflects, that LamMPI has a very large overhead showing up especially by sending very small data packages very frequently. Other implementations of MPI like OpenMPI, being the next-generation MPI implementation, may perform better but are not tested here since it was not implemented at the time of this benchmark investigation.

The relation of computation time to communication time is called *task granularity*. It has a very important effect on the overall performance of the co-simulation. To be efficient, the amount of communication should be as low as possible and the computation time per macro time step should be high, giving a coarse *task granularity*.

Valve Train with Chain Drive

Table 4.3 shows the simulation time for the valve train including a chain drive. The fastest simulation is the one using shared memory as inter-process-communication. The *task granularity* is even coarse enough so that the much higher *round trip delay time* in case of multi host parallelization (PVM) has only a minimal negative effect on the total simulation time. The LamMPI implementation of MPI shows again poor results.

Summary

All tests with different IPC methods have shown the expected result: shared memory is the fastest communication in case of a single host. However, at least for realistic industrial examples, the parallelization on multi host clusters is also feasible if fast inter-host-communication methods like PVM are used.

Since co-simulation is mostly used with only a few subsystems and modern workstations already have up to 4 or 8 CPUs in a shared memory architecture, it is

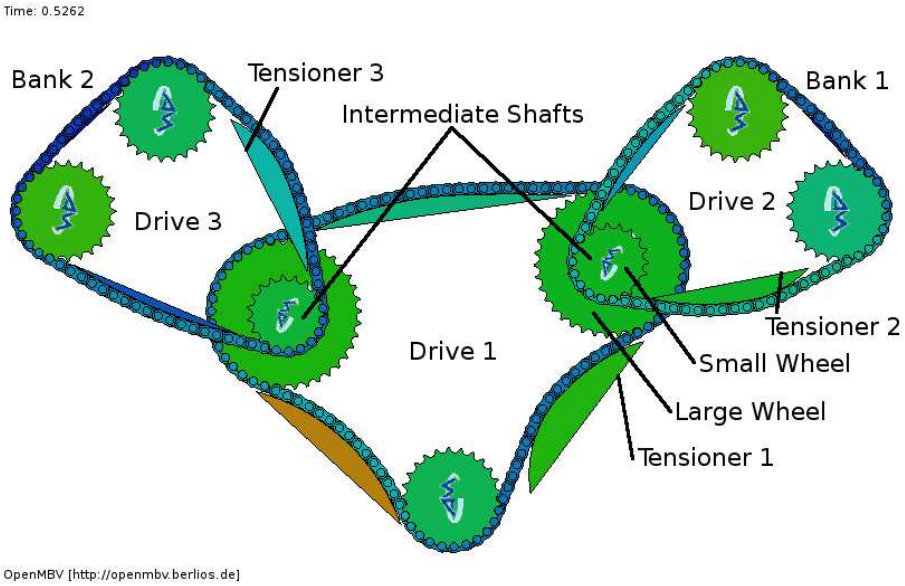


Figure 4.1: Timing chain drive with 3 drives

advisable to perform parallel co-simulations using shared memory inter-process-communication on modern multi CPU workstations. The following examples are all calculated using this IPC method.

4.2 Coupling-Benchmark

The different types of extrapolation for the co-simulation coupling have already been analyzed analytically concerning the stability in Section 2.3.2. The most stable coupling leads to the largest macro step size and to the minimal overall computational time. However, this analytical stability analysis is only based on a linear test system. Therefore, a benchmark of the different coupling variants applied to a complex, non linear example of industrial relevance is necessary for further analysis. Moreover, the parallel speedup and the simulation results of the different variants are analyzed with respect to a serial not co-simulated reference simulation. Therefore an example is used which can be calculated with and without a co-simulation.

4.2.1 Model Setup

Figure 4.1 shows the principle setup of the simulation of the chain drive of a V6 engine visualized by the program OpenMBV [52]. The lower wheel resides on the crank shaft which follows a kinematic rotational nonuniformity representing the varying load on the crank shaft. Chain drive 1 transmits a torque from the crank shaft wheel to the exhaust and intake cam shaft of bank 1 and 2, using the intermediate shafts and wheels and drive 2 and 3. The boundary conditions of the

cam shafts are kinetic excitations representing the torque load of the valve trains. The chain is prestressed using one hydraulic chain tensioner per drive.

The reference simulation is defined using KETSIM calculating all three drives and the chain tensioners in one simulation. The other simulations are realized using the presented parallel co-simulation framework with different types of extrapolations for the coupling. Therefore, the chain drive is split into the three drives by cutting the intermediated shafts between the small and large wheels. The resulting three independent chain drives again are simulated using KETSIM. In each KETSIM chain drive a coupling interface is added at the point of intersection at the wheels. Each interface exports the translational and rotational position and velocity and imports a force or torque as a piecewise constant or linear kinetic boundary condition defining the subsystem excitation due to the coupling interaction. The stiffnesses of the translational linear co-simulation coupling elements are $c_{\mathrm{tra}} = 6.6 \cdot 10^9 \frac{N}{m}$, and the rotational stiffnesses are $c_{\mathrm{rot}} = 6.3 \cdot 10^4 \frac{Nm}{\mathrm{rad}}$. These values are the real elasticities of the small part of the shaft in between the small and large wheel. The co-simulation is running on a four core, shared memory computer, using *shared memory* as inter-process-communication.

4.2.2 Comparison of Coupling Extrapolations

The first row of Table 4.4 shows the total simulation time of the whole chain drive using KETSIM without co-simulation. The other rows show a selection of co-simulations with different types of extrapolations for the coupling force. The simulations are grouped in constant and linear extrapolations. Moreover, each coupling variant is tested using two different macro time steps H: the first is the maximal stable macro step size and the second the one with the minimal simulation time. This minimal time was found by carrying out several runs with different macro step sizes and choosing the best. The simulation with the maximal stable macro step size is not the fastest one because this simulation is acting near the stability barrier which may induce high frequent oscillations having a negative effect on the step size controlling of the subsystem integrators.

Table 4.4 shows that the coupling variants being optimized for stability lead to the highest macro step size as well as to the minimal simulation time. Moreover, the optimized coupling variants, especially those having only one free parameter, `Const 2,3,Opt` and `Lin 2,3,Opt`, have a very low difference in simulation time between the simulation with the maximal possible macro step size and the fastest simulation with this coupling variant. This is a very helpful property, because the manual detection of the maximal feasible macro step size by the user is easy whereas the detection of the step size leading to the minimal simulation time is very time consuming: To detect the maximal possible macro step size one starts at a high step size and the simulation will fail very early due to instabilities. Decreasing the step size until the simulation will not fail leads to the maximal macro step size. The detection of the minimal simulation time needs several full simulations. Hence, if the simulation time with the maximal possible macro step size H is very close to the

Coupling variant	Simulation Time [s]	Macro Step Size H $[10^{-6}s]$
Without co-simulation	458	-
Constant Extrapolation:		
Spring-Damper $d_{t/r} = 5000/50$	1032	2
Spring-Damper $d_{t/r} = 5000/50$	604	1
`Const 3,6`	711	4
`Const 3,6`	321	2
`Const 2,3,Opt`	216	6
`Const 2,3,Opt`	208	5
`Const 2,2,Opt`	272	8
`Const 2,2,Opt`	213	5
Linear Extrapolation:		
`Lin 2,4`	274	5
`Lin 2,4`	204	4
`Lin 1,2`	433	4
`Lin 1,2`	239	3
`Lin 2,3,Opt`	189	6
`Lin 2,3,Opt`	189	5
`Lin 2,2,Opt`	574	8
`Lin 2,2,Opt`	209	4

Table 4.4: Simulation times of the three chain drives

minimal simulation time, a nearly optimal macro step size covering the simulation time can be found very fast.

User selectable parameters like the macro step size H above are even more problematic if a spring-damper-coupling is used besides a spring-coupling. Such a co-simulation is shown in Table 4.4 by the variant 'Spring-Damper' using a simple extrapolation variant. In this case the damping part can be used to stabilize the co-simulation, but the best numerically inspired value for this damping is unknown and must be estimated by the user. This is not the case for the optimized couplings since they do not need any other numerical parameter except the macro step size H.

Speedup Effects

The simulation time of the most practical coupling `Lin 2,3,Opt` is about 189s, which is a factor of about 2.4 faster than the simulation without co-simulation. Comparing the CPU time of the three drives shows, that drive 1 needs about a factor of 1.9 more computational time than drive 2 and 3 which have about the same computational effort. This correlates with the larger number of chain links in drive 1. Hence, the total simulation time of $458s$ of the drive without co-simulation can be distributed approximately to the three drives

$$T_{\text{all}} \approx T_1 + T_2 + T_3 \approx 1.9T_2 + 2T_2 \approx 458s \Rightarrow T_2 \approx T_3 \approx 117s, T_1 \approx 224s. \quad (4.2)$$

Therefore the total simulation time of any kind of parallel co-simulation regardless of any overhead should be expected to by at least $\max(T_1, T_2, T_3) = 224s$. The paradox why the simulation time $189s$ of the best parallel co-simulation is even less can be explained by the following aspects:

Integrator decoupling The shift points of closing or opening contacts between the chain links and the wheels must be detected by the integrator. This search needs extra calls to the right hand side of the simulation model. In case of no co-simulation a shift point in one of the subsystems forces the call of the right hand side to the entire system. In case of a co-simulation each subsystem integrator is searching only own shift points including the call to the own smaller right hand side. The other integrators can proceed as normal. This effect as well as the decoupling of the step size control is known as *integrator decoupling*.

Contact decoupling The contacts between the chain links and the wheels are modeled using rigid contacts. Without co-simulation the two wheels on the intermediate shaft are connected using a constraint. Each contact on the large wheel is directly coupled with each contact on the small wheel. During co-simulation these contacts are decoupled by the elastic co-simulation coupling. In the first case the contact solver has to solve a coupled problem with $N_1 + N_2$ rigid contacts whereas a co-simulation decouples the system into independent contact problems of size N_1 and N_2 being much easier and faster to solve.

Filtering Dynamical vibrations with frequencies higher than $\frac{1}{H}$ can not be exchanged between subsystems due to the macro step size H. As a result some effects with high frequency, which are not interesting for the global dynamics, are eliminated by the co-simulation.

Besides the minimal simulation time estimated in equation (4.2), there exist two common ways to compare the parallel speedup of a program with theoretical reachable values: the law of Amdahl and the law of Gustafson [31]. Both laws assume that the program can be split into an arbitrary number of parallel portions. For co-simulation this is far from reality, since the number of parallel portions is restricted to the number of subsystems. Hence, the theoretical reachable speedups of both laws may be much higher than the real speedup in case of a parallel co-simulation. Amdahl states that the maximal speedup

$$\Psi \leq \frac{1}{f + \frac{1-f}{p}} \tag{4.3}$$

is defined using the fraction of serial code f and the number p of subsystems on individual CPUs. Gustafson defines the maximal speedup

$$\Psi \leq \frac{pT_p + T_s}{T_p + T_s} \tag{4.4}$$

using a slightly different approach using the fraction of serial simulation time T_s, the parallel simulation time T_p and also the number p of CPUs. In case of a negligible

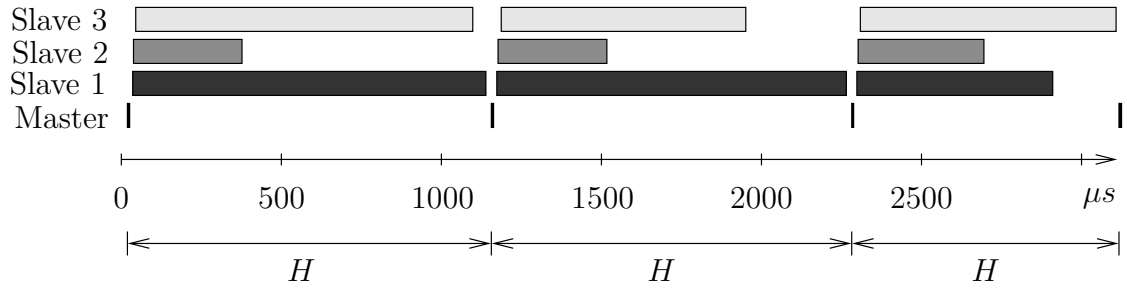


Figure 4.2: Thread profiling of the V6 engine

fraction of serial code, as it is true for the parallel co-simulation, both laws state $\Psi = p$. Hence, the achieved parallel co-simulation speedup of about 2.42 is a good value compared to the theoretical reachable speedup of 3.

Thread Profiling

Figure 4.2 shows the thread activity for the co-simulation of the chain drive over the real time. Slave 1 has nearly always the highest computational effort, even though this can change in some macro time steps. The alternating computational effort per macro step is related to the inconstant computational effort of the slaves which itself is mainly related to the step size control of the subsystem integrators and the contact search. Moreover, Figure 4.2 shows that the slaves perform well in parallel and the master, which must run in serial with respect to all slaves and ensures the synchronization, has a very low computational effort.

4.2.3 Simulation Results

Good speedups are an important benefit of co-simulation but are pointless if the simulation results are not nearly equal to the original non parallelized program. A typical parallelization technique is for example the parallel coding of an algorithm. In this case the results of the non parallel and a parallel program are equal up to the numerical accuracy of the computer. However this is not the case for co-simulation, because adding elasticities at the coupling points physically changes the system and might lead to a change of the simulation result which can be larger by magnitudes than numerical errors. Hence, it is important to compare the co-simulated example with a not co-simulated reference example. Figure 4.3 shows on the left the oscillation angle $\Delta\alpha$ of the exhaust cam shaft of bank 1 for the reference simulation without co-simulation and for the co-simulation with the optimal coupling variant `Lin 2,3,Opt`. The comparison of this value is chosen exemplarily, because the cam shaft oscillation is of major interest for timing drives in combustion engines. This figure shows that only at some points slight differences between the two simulations are distinguishable. For a more detailed error analysis the error given by the absolute difference between the two simulations is shown in Figure 4.3 on the right. The

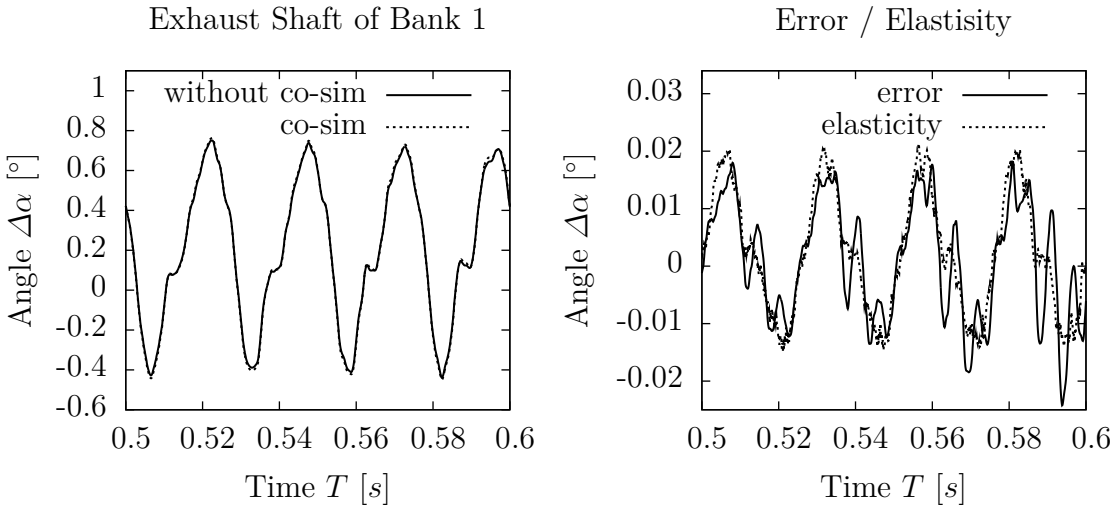


Figure 4.3: Comparison of the simulation results: V6 engine

error is only about two magnitudes smaller than the dynamic oscillation and thus magnitudes larger than typical numerical errors of about 10^{-5}. However, looking at the relative rotational twist of the elasticity of the co-simulation coupling which is also plotted in Figure 4.3 on the right, shows that this error is mainly attributed to the change of the model needed by the co-simulation. This elastic deflection between the two wheels on the intermediate shaft is caused by the propagation of the changing torque of the valve train having a peak torque of about 20Nm which leads to a rotational deformation of about 0.02°. Since the added elasticity was motivated physically the co-simulation can also be interpreted as a model extension using an elastic intermediate shaft instead of a rigid shaft.

4.3 Multi Scale Problem

The decoupling of the subsystem integrators is a very important aspect concerning the maximum speedup of a co-simulation, even if the co-simulation is run in serial. Especially if the time scales of the subsystems are very different, often called multi scale problems, the speedup related to the decoupling of the multi scale problem can be higher than the speedup related to the parallelization of a co-simulation. In this case a parallel co-simulation will lead to very good speedups because of the combination of both effects. An example of such a multi scale system is given in this section.

4.3.1 Model Setup

In timing chain drives, the chain is often prestressed using a hydraulic chain tensioner. Such chain tensioners can be modeled by the chain drive simulation software

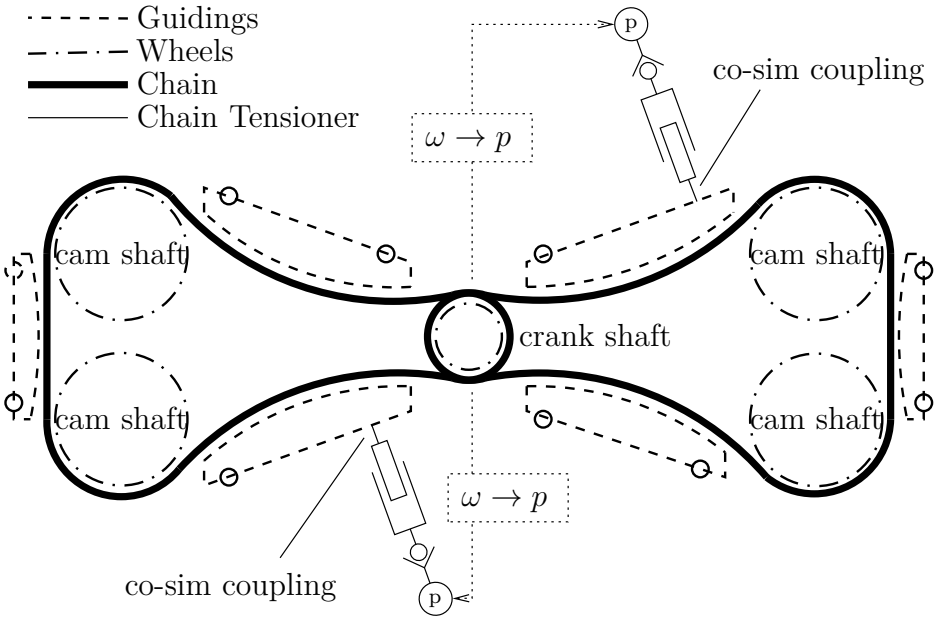


Figure 4.4: Flat engine chain drive with two hydraulic chain tensioners

KetSim using a limited number of provided models. Since the chain tensioner has a significant effect on the dynamics of timing drives and there are many variants of hydraulic chain tensioners, the representation of arbitrary chain tensioners with high model quality is needed. Such modular chain tensioners can be modeled using the hydraulic simulation tool Hysim [13, 32]. Using these tools, the need for a co-simulation between the chain tensioner and the chain drive simulation software arises.

As an example, a chain drive of a flat engine, depicted in Figure 4.4, simulated with KetSim and hydraulic chain tensioners simulated with Hysim is used to show the effects of multi scale problems. The crank shaft in the middle is kinematically excited representing the load of the crank shaft. The boundary condition on the four cam shafts are kinetic torque excitations representing the load of the valve train being not simulated in this example. Moreover, the hydraulic chain tensioners have an oil pressure boundary which itself depends on the rotational speed of the crank shaft.

Since the chain tensioners include small oil volumes which must be calculated elastically to represent the dynamic behavior sufficient the hydraulic model is getting much stiffer compared to the mechanical model of the chain. This leads to different scales of both models causing numerical and computation time problems which can be bypassed using a co-simulation.

The point of intersection in this co-simulation is the contact point between the piston of the chain tensioners and the corresponding guiding. Since this unilateral contact does not separate it can be approximated by a non separating local contact stiffness. This can be represented by the co-simulation coupling law. Moreover, the two chain tensioners can be calculated by different slaves and the left and right chain drive

Model	Simulation time [h]
Reference Simulation	15.37
Parallel Co-Simulation (2 subsystems)	1.86
Parallel Co-Simulation (4 subsystems)	1.46

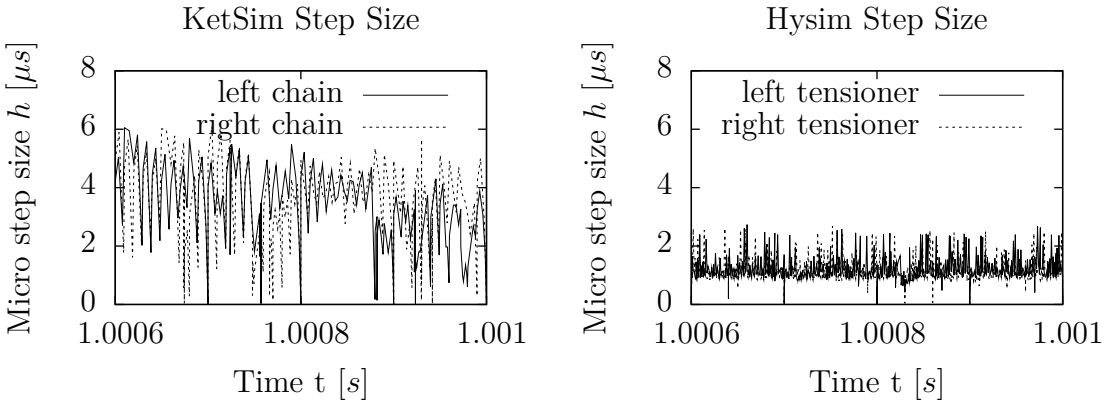
Table 4.5: Simulation time of the chain drive and chain tensioner

can also be split into two subsystems if another co-simulation coupling between the two wheels on the crank shaft is added. The optimized constant extrapolation `Const 2,3,Opt` is used for the co-simulation coupling.

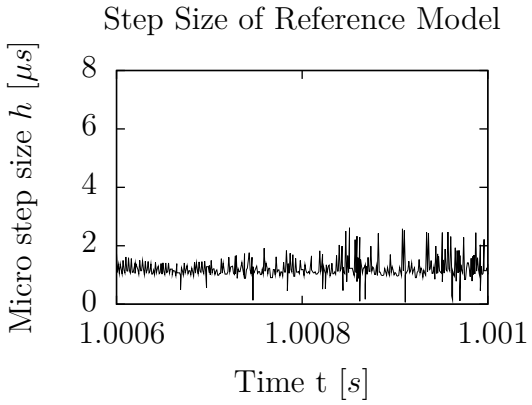
Besides defining a co-simulation the whole system can also be integrated by the software HYSIM using model coupling. In this case the HYSIM integrator includes the whole KETSIM chain drive model as a component [19]. This simulation defines the reference simulation without a decoupling of the subsystem and without the possibility of parallelization.

4.3.2 Computing Time

The total computing time t_{total} of a dynamical system is mainly attributed to the product of the number of time steps N, required for a given time interval, and the computing time of a single time step t_{step}. In case of a model coupling the whole chain drive has to be integrated with the very small step size (many time steps), needed for the very stiff hydraulic chain tensioner, although this would not be necessary for the mechanical part. Using a co-simulation both subsystems are integrated by their own integrators with their own typical step sizes, which leads to a different number of required steps for each subsystem. Hence the total computing time of a co-simulation is approximately the sum of $N_i\, t_{\text{step},i}$ of all subsystems. This leads to a speedup if the number of required time steps N_i differs highly, being the case for multi scale problems. Moreover, using co-simulation, the simulation of the chain drives and the chain tensioner can run in parallel in two or four threads depending on the substructure definition. The computation times for all simulation variants are shown in Table 4.5. The parallel co-simulation with 2 subsystems gives a speedup of about 8.26 compared to the reference simulation. Due to the average CPU load of about 85% for both subsystems during the parallel co-simulation a maximal speedup of about $2 \cdot 0.85 = 1.7$ can be related to the parallel execution. The remaining speedup of about 4.8 results mainly from the decoupling of the multi scale problem. Splitting the simulation into four subsystems leads to an additional speedup of about 1.27 with respect to the co-simulation with two subsystems. In this case no further speedup related to a multi scale problem arises because the micro step sizes of the two chain drives are nearly equal and the micro step sizes of the two tensioners are also nearly equal. The benefit of the extended parallelization from two to four CPUs is not very high because the two tensioner subsystems can only load a CPU with about 30%, since the calculation of one macro time step for each chain drive takes much more time than for the tensioner subsystems.



(a) Co-simulation with four subsystems



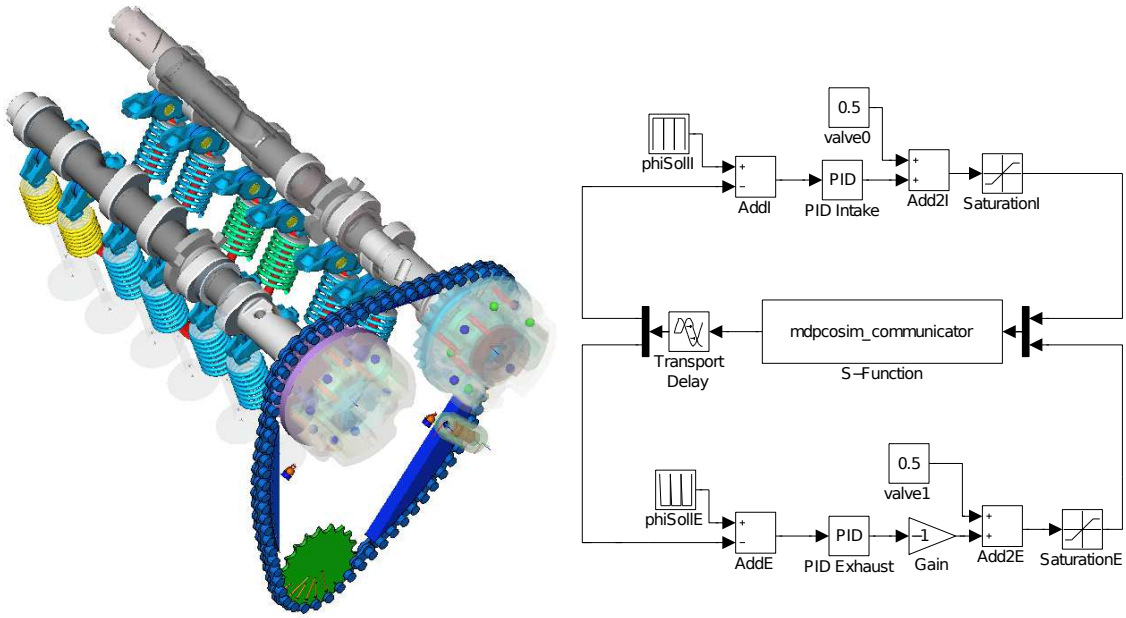
(b) Reference simulation (no co-simulation)

Figure 4.5: Variable micro step size for chain drive system of Figure 4.4

Figure 4.5 shows the variable step size of the subsystem integrators. Figure 4.5(a) left depicts that the step sizes of the chain drives calculated in KetSim are in average equal but by a factor of about 2.4 larger than the step sizes of the chain tensioner depicted in Figure 4.5(a) on the right. The step size of the not co-simulated reference simulation is the minimal required step size of the overall system and is shown in Figure 4.5(b). Hence, the chain drive, having a much higher computational effort per time step than the tensioner, is forced to the small step size of the tensioner part.

4.3.3 Simulation Results

Comparing the simulation results of the reference simulation with the parallel co-simulation shows that the influence of the co-simulation approach is negligible for the global dynamics. In contrast to the example in Section 4.2, no additional elasticity needs to be added at the co-simulation coupling point because the cutting point between the tensioner piston and the guiding is already modeled linear elastic in the reference simulation. This elasticity is used as co-simulation coupling law.



(a) V6 engine, bank 1 (b) Cam phasing controller

Figure 4.6: Bank 1 of a V6-engine and cam phasing controller

4.4 Multi Domain Example

The examples in Section 4.1 and 4.2, are both purely mechanical where the subsystems are mechanically coupled. The multi scale problem in Section 4.3 is already a multi domain example because the mechanical timing chain drive is coupled with a hydraulic chain tensioner. However, the couplings between the subsystems again are mechanical because the domain interface between mechanic and hydraulic systems is included by the model of the piston in the hydraulic domain, compare to Section 2.6.3. The example in this section uses mechanical, hydraulic and control subsystems as well as mechanical, hydraulic and feed through control couplings.

4.4.1 Model Setup

Figure 4.6(a) illustrates the setup of the system visualized by OpenMBV [52]. The chain drive of the system is the same as drive 2 in Figure 4.1. The wheel on the intermediate shaft (lower wheel) is actuated using a kinematic boundary. This boundary represents the load of the crank shaft as well as the transmission behavior of the chain drive 1 of Figure 4.1 between the crank shaft and the intermediate wheel. The valve train comprises a full mechanical model including valve contacts and springs, whereas the gas forces in the combustion chambers are not regarded. The oil pressure of the oil supply is a variable boundary condition using a function of the intermediate shaft rotational speed. Moreover, this system includes a hydraulic chain tensioner and two hydraulic cam phasing systems on the intake and exhaust shaft. The modeling of the components of this system is described in detail in [66,

#	Subsystem / Slave	Program
#1	chain drive	KetSim
#2	intake cam phasing	MBSim
#3	exhaust cam phasing	MBSim
#4	hydraulic chain tensioner	MBSim
#5	intake valve train	MBSim
#6	exhaust valve train	MBSim
#7	cam phasing controller	Simulink

#	Connection	Type
#1	intake chain wheel - intake cam phasing stator	mechanical
#2	exhaust chain wheel - exhaust cam phasing stator	mechanical
#3	tensioner guiding - tensioner piston	mechanical
#4	oil supply - intake cam phasing - exhaust cam phasing	hydraulic
#5	exhaust cam phasing - chain lubrication - chain tensioner	hydraulic
#6	intake cam phasing rotor - intake cam shaft	mechanical
#7	exhaust cam phasing rotor - exhaust cam shaft	mechanical
#8	exhaust cam phasing relative angle - exhaust controller input	feed through
#9	exhaust cam phasing relative velocity - exhaust controller input	feed through
#10	intake cam phasing relative angle - intake controller input	feed through
#11	intake cam phasing relative velocity - intake controller input	feed through
#12	exhaust controller output - exhaust 4/3 way valve control edge	feed through
#13	intake controller output - intake 4/3 way valve control edge	feed through

Table 4.6: Subsystems and connections

67, 69]. The original model of this system does not include a controller for the cam phasing systems, they are acting at their end stops. Hence, two controllers for the relative angle of the two cam phasing systems are added. Since a detailed model of a real cam phasing controller was not available for this work the simple PID controllers, shown in Figure 4.6(b) including the S-Function communication block, are used.

The original model of this system is a co-simulation of two subsystems: the chain drive and the rest of the system excluding the cam phasing controller which is not included in the original model. The focus of this example lies on the splitting of the system into more and more subsystems with the aim to speedup the simulation dramatically using a parallel co-simulation on a multi core machine. Lastly the controller is added. The final, fully split system consists of 7 subsystems and 13 co-simulation connections. The single subsystems and connections are listed in Table 4.6 together with the respective simulation programs and connection types. The system is split incrementally in the following order:

2 Subsystems: The chain drive is modeled in KetSim and both valve trains including the individual cam phasing systems and the chain tensioner are modeled as one subsystem in MBSim.

3 Subsystems: The valve train is split into the intake and exhaust valve train

Number of Subsystems	Hyd. Coupling	Speedup
2	$E(p) = \text{const}$	1.0
3	$E(p) = \text{const}$	1.9
4	$E(p) = \text{const}$	2.1
4	$E(p)$: stepped	2.1
4	$E(p) \neq \text{const}$	2.1
6	$E(p) = \text{const}$	3.4
7	$E(p) = \text{const}$	3.4

Table 4.7: Speedup of the co-simulation

including their cam phasing system. The chain tensioner is part of the exhaust model.

4 Subsystems: The chain tensioner is split from the exhaust model as a separate subsystem.

6 Subsystems: The intake and exhaust cam phasing systems are split from the intake and exhaust cam shafts.

7 Subsystems: The control subsystem is modeled in one SIMULINK subsystem and includes two controllers for the intake and exhaust cam phasing systems. This step does not further split but extends the model towards an active control.

Each splitting is attended by adding co-simulation connections of appropriate type at the intersection points. The mechanical and hydraulic couplings use the extrapolation `Lin 2,3,Opt` which provided fast and accurate simulations in the previous examples.

4.4.2 Computing Time

All calculations are run on a 4 CPU shared memory workstation computer with a macro step size $H = 1 \cdot 10^{-5}$. Table 4.7 shows the speedup factors of the system divided into the subsystems described above in relation to the original model with only two subsystems. The splitting of the valve train gives a very good speedup of 1.9, because the valve train has a significant higher computational effort than the chain drive and the cutting results in two similar subsystems having nearly the same computational effort.
However, the splitting of the chain tensioner from the exhaust valve train gives only a moderate additional speedup of about 1.1, in total 2.1 compared to the two subsystems variant, because the tensioner model is very small and does not have a considerably different time scale as it has been in the multi scale problem of Section 4.3. The model with 4 subsystems has been tested with the three different modelings of the hydraulic coupling node bulk modulus E described in Section 2.4.1-2.4.3. Even the variant which needs an iterative numerical solution for the pressure p

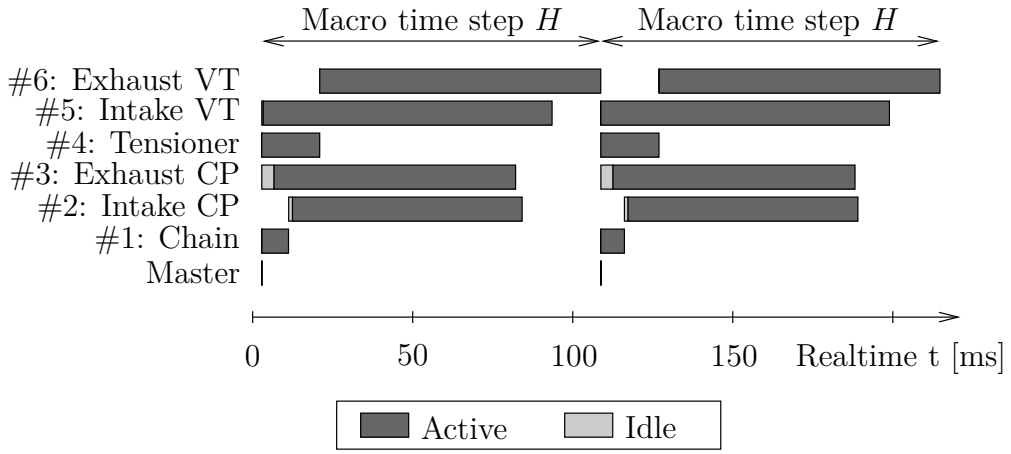


Figure 4.7: Thread profiling of the 6 subsystems and the master with same priorities

at each macro time steps does not lead to a significant increase of the simulation time. The simulation result is also not influenced in an appreciable matter because the pressure change has comparatively slow dynamics at the hydraulic coupling nodes. The cam phasing systems have again a relevant computational effort. Hence, the cutting of the cam phasing systems from the cam shafts leads to a total speedup of about 3.4 on a 4 CPU machine.
Adding the controller increases the total computational effort of the model only marginal since the simulation of the control subsystem can be performed very fast.

4.4.3 Thread Profiling

Figure 4.7 shows the thread profiling of the co-simulation with 6 subsystems. The bars show the active times from the reception of the input from the master until the sending of the output back to the master. At the times where no bar is drawn, the subsystem has to wait because the other subsystems have not yet finished the last macro time step. The total sleep time, called idle time, of a thread during one macro time step is shown by the lighter bars. Note that the point in real time where a process sleeps is not shown, only the amount of time during one macro time step is depicted by the lighter bars.
Since there are 6 subsystems (slaves, threads) on a 4 CPU machine some threads must run in serial or are forced to sleep by the multitasking feature of the operating system. To run more threads at a time than CPUs are available the operating system switches frequently between the execution of the threads, which results in a quasi parallel execution called multitasking. Such switches are called *context switch* and take a relatively long time for heavy weight threads [76] such as co-simulation subsystems. The sequence and execution time of the different threads in case of a multitasking operation is the challenge of the so called *scheduler*. The optimal strategy for scheduling is a continuous research topic in information technology [76]. Current implementations like *preemptive scheduler*s are not optimal in the given scenario because the threads with the highest computational effort not always receive

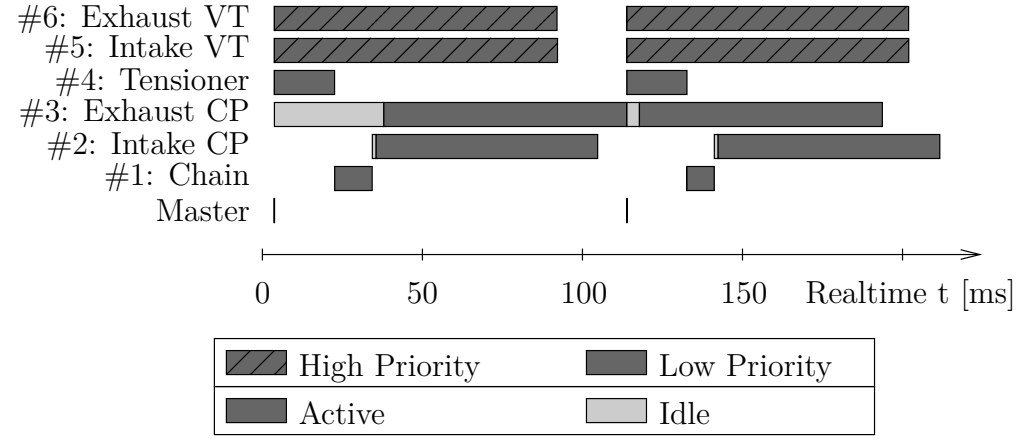


Figure 4.8: Thread profiling of the 6 subsystems and the master with different priorities

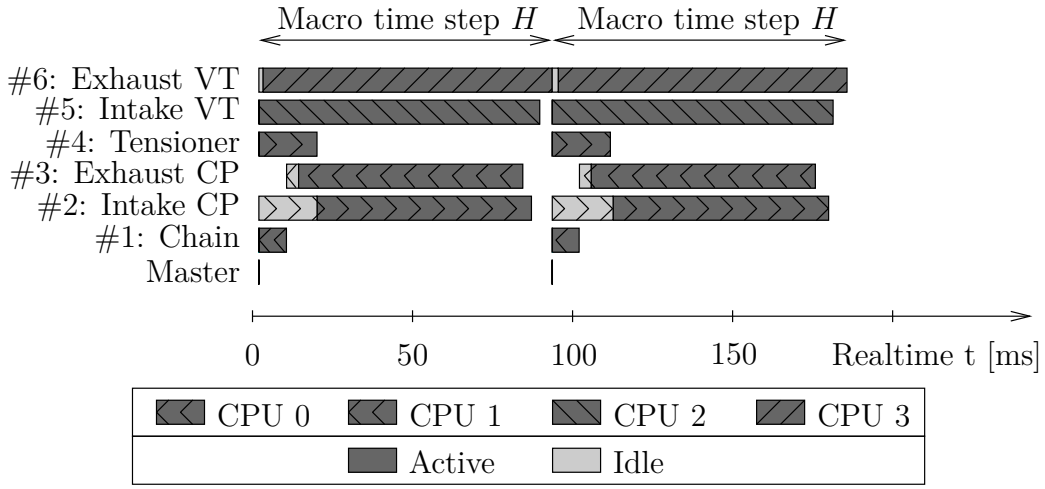


Figure 4.9: Thread profiling of the 6 subsystems and the master with applied *process affinity*

the full CPU time from the beginning of a macro time step, see for example the exhaust VT in Figure 4.7. One possibility to influence the process scheduling of the operating system is to apply a *process priority* to each process or thread. Figure 4.8 shows the thread profiling of the same simulation but with a high process priority for the valve train subsystems having the highest computational effort and a low priority for others. This changes the scheduling a lot, but has no relevant positive effect on the speedup because the automatic distribution of the threads to the CPUs is still not optimal. Another possibility is to apply a so called *process affinity* to the threads, where a process is limited to run on a specific CPU. For the given system one associates a subsystem with a very low and a middle computational effort to one CPU and exclusively assigns the subsystems with the highest effort each to a separate CPU. The resulting thread profiling is shown in Figure 4.9. The total simulation time reduces by setting the *process affinity* because now the distribution of the threads to the CPUs is optimal. A disadvantage of this method is that the *process affinity* must be applied by the user. However, modern operating systems

Number of Subsystems	Allowed CPU for Subsystem #1	#2	#3	#4	#5	#6	Speedup
2	all	all					1.0
6	all	all	all	all	all	all	3.4
6	0	1	0	1	2	3	4.0

Table 4.8: Speedup of the co-simulation with applied *process affinity*

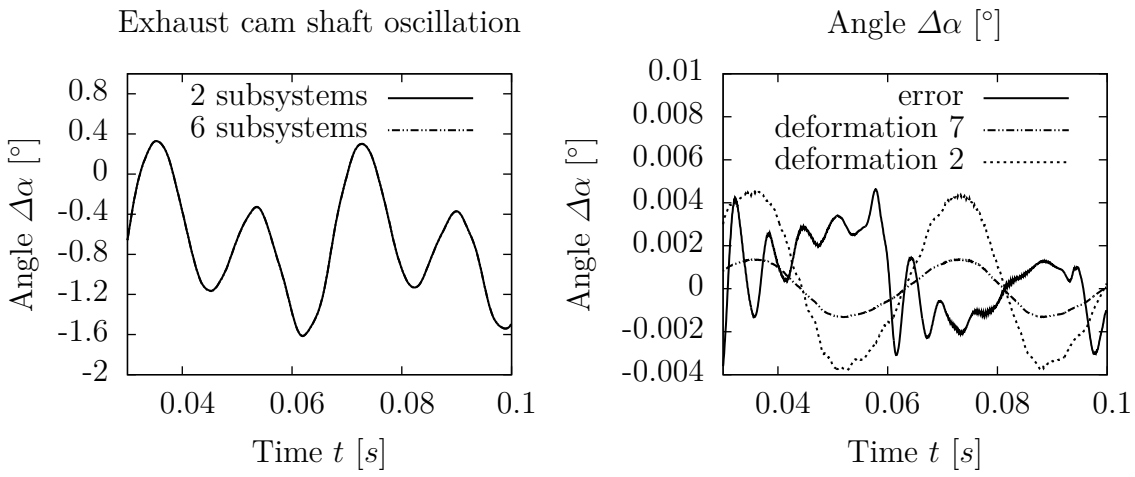
are able to change the *process affinity* dynamically during execution. Hence, it is feasible to measure the CPU time of each thread for one macro time step and to use this information to set the optimal *process affinity* automatically. Using this method it is also feasible to change the affinity when the ratio of the computational effort for two or more subsystems changes during the simulation.

Table 4.8 depicts the positive effect of *process affinity* settings in the speedup factor compared to the original systems with two slaves and the one with 6 slaves but without a *process affinity*. It is worth to point out that the speedup on a 4 CPU machine is 4.0 even though there is an overhead for the co-simulation, especially for the synchronization at the macro time steps. The reason why the theoretically maximal speedup can be reached is already described in Section 4.2: integrator decoupling, contact decoupling and filtering as well as the multi scale phenomena.

4.4.4 Simulation Results

Besides the speedup of the simulation also the influence of the co-simulation on the simulation results needs to be investigated. Figure 4.10(a) shows the oscillating angle of the exhaust cam shaft for the original simulation with 2 slaves and the co-simulation with 6 slaves. Though the two curves nearly coincide, the error defined as the difference between the two results, depicted in Figure 4.10(b), is much larger than typical numerical errors as shown in Section 2.3.1. The reason therefore is again the modification of the physical model: at the coupling points rigid connections must be replaced by elastic coupling laws. Hence, Figure 4.10(b) depicts exemplarily also the elastic deformations of the co-simulation coupling #7 and #2, see Table 4.6. This demonstrates that the errors between the different simulations are mainly related to the change of the model and not to the numerical accuracy of the co-simulation coupling.

The co-simulation variant with 7 subsystems including the phasing controllers are conclusively addressed. Figure 4.11 shows the set- and actual-value cam phasing angle φ for the exhaust and intake cam phasing systems as well as the position s of the corresponding 4/3 way valve. The valve is in neutral position for $s = 0.5$ and has a corresponding minimal and maximal value of $s_{\min} = 0$ and $s_{\max} = 1$ respectively. To demonstrate the work of the controller the set value exhaust cam phasing angle is linear in time and the set value intake cam phasing angle jumps at $t = 0.02s$. Both plots show that the controller has a relative large reaction time and that the



(a) Oscillating angle

(b) Error and deformation of elastic couplings

Figure 4.10: Simulation results

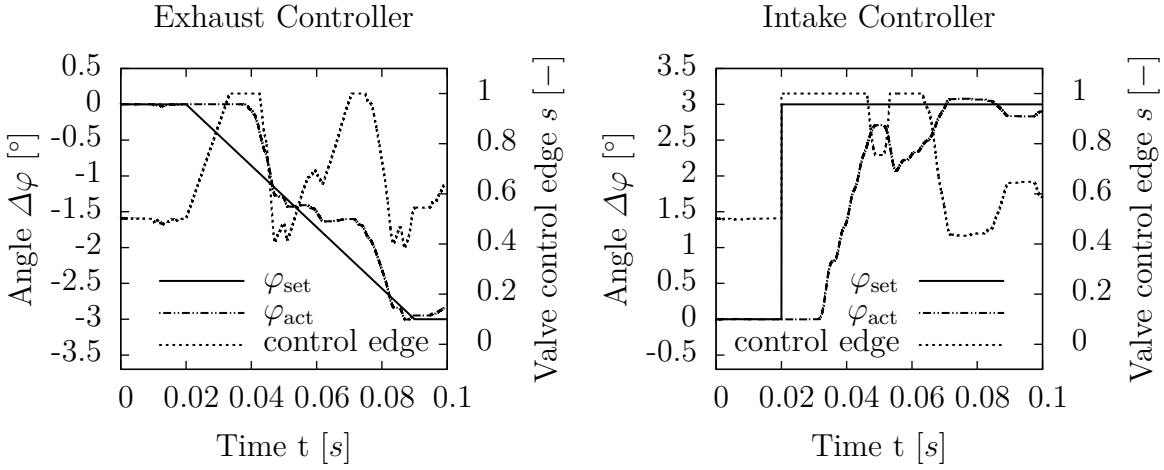


Figure 4.11: Controller results

maximal adjustment speed is very limited by the oil flow even if the valve is fully opened. Improving the controller will improve the tracking of the set value, but this is not topic of this work.

5 Parallelization of the Algorithm

The last chapters addressed the parallelization of multiple dynamic simulation software tools with a detailed derivation of methods and examples. Another variant is the parallelization of the algorithm used by the individual software. This is often called internal parallelization and has the advantage that neither a change of the model nor a split into subsystems is needed. The disadvantage of this method is that usually only small parts of the code can be reformulated and implemented utilizing parallelization. Hence, a high fraction of code cannot be parallelized which leads to low speedups as well as to high frequent communication being responsible for overhead. Another disadvantage is that the fraction of parallel code must be thread safe: if one thread is writing to memory no other thread running in parallel is allowed to read or write at the same time to the same memory. An investigation of the code for thread safety is needed which can be very time consuming for large preexisting simulation software. Two principle possibilities arise to force thread safety: reformulating the algorithm in a way that such *race conditions* are avoided or synchronizing of the corresponding code. The first might be hard to develop and implement and the second leads to an increase of the synchronization overhead. In contrast to co-simulation, internal parallelization is impossible with simulation tools only available as closed source. To compare the principle possibilities of internal parallelization and parallel co-simulation focusing on the speedup effects, a short outlook to internal parallelization of the simulation software KetSim and MBSim is given in this section.

5.1 Parallelization of KetSim

In KetSim chain links of the drive are connected to each other using local contact elasticities. The contacts between the chain links and the chain wheels as well as the chain guidings are modeled using rigid body contacts with set valued force laws [18,20,23]. Due to the design of chain drives, a chain link can not be in contact with more than one wheel or guiding. Therefore it is impossible that the contact of a chain link with a wheel or guiding is coupled with the contact of a chain link with another wheel or guiding. Hence, the rigid body contact calculation of each wheel or guiding and all adjacent chain links can be separated and thus done in parallel. Since KetSim is implemented in Fortran 77 with excessive use of global variables it is hard to implement the parallel contact calculation.

For the V6 engine example with three drives, depicted in Figure 4.1, the implementation of this parallelization leads to a fraction of serial code of $f = 0.559$. Hence, on

Method	Speedup	CPU-Load
Without Parallelization	1.000	100%
OpenMP: Two *Parallel For*-Loops	1.095	200%
OpenMP: *Parallel Section* with For-Loops	1.162	200%

Table 5.1: KetSim speedup using internal parallelization

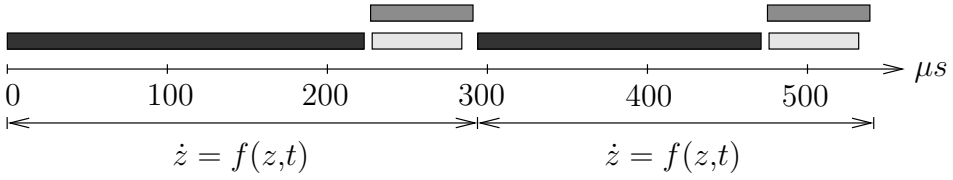


Figure 5.1: Thread profiling using internal parallelization

a $p = 2$ CPU computer the maximal speedup according to Amdahl, equation (4.3), is 1.28. Table 5.1 shows the speedup of the implementation using OpenMP [31]. Two different implementations are given: One with a parallel execution of the elements of two loops and one with two parallel sections each running one of the loops. The maximal speedup of 1.162 is pretty close to the theoretical maximum. The CPU load in case of OpenMP is 200% which shows that the synchronization by this OpenMP implemention is done using busy waits. This minimizes the overhead for synchronization but fully loads both CPUs even though the speedup is far away from 2. The thread profiling in Figure 5.1 shows the still high ratio of serial code. Of course there are still some parts of code in the serial code which can be parallelized, but the implementation if difficult. Moreover the profit will not be very good since these parts of code have shown a very low computation effort in cost profiling analysis.

5.2 Parallelization of MBSim

The implementation of internal parallelization in MBSim is more straight forward because the object orientated language C++ being used, encapsulates the data locally which resolves many problems compared to the use of global data. Being the basis for continuous variable transmission models [64], as an academic example a flexible closed band with 36 rigid body elements is used, see Figure 5.2. Since the elements can only move tangential to the band, a relative kinematic description with respect to the band is implemented [80]. Hence, the mass matrix $\boldsymbol{M}_{\text{tot}}$ of this tree structure is the sum of the element masses $\boldsymbol{M}_i$ projected by the element Jacobian $\boldsymbol{J}_i$

$$\boldsymbol{M}_{\text{tot}} = \sum_i \boldsymbol{J}_i^T \, \boldsymbol{M}_i \, \boldsymbol{J}_i. \tag{5.1}$$

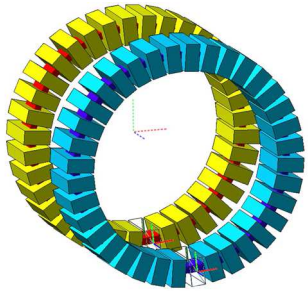


Figure 5.2: MBSim example for internal parallelization

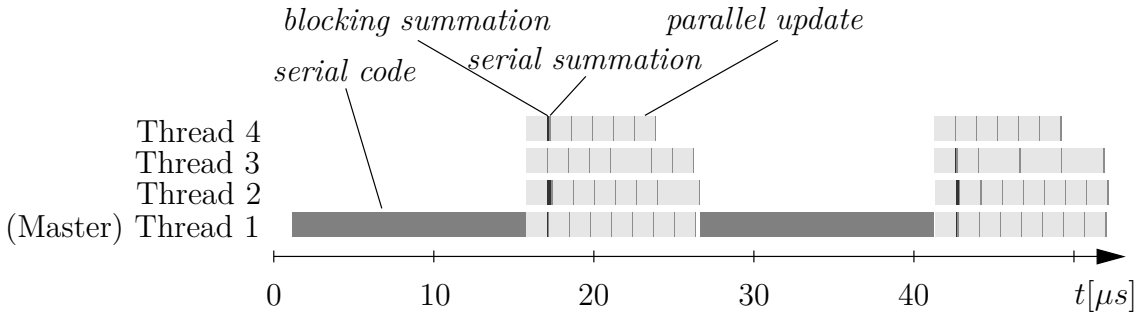


Figure 5.3: MBSim example: thread profiling

Since the calculation of the element Jacobians is very expensive, it is reasonable to parallelize the evaluation of equation 5.1 for the elements i. The summing up must be done to global memory. Hence, this process must be serialized to avoid *race conditions*. This blocking is shown in the thread profiling of Figure 5.3 by the bars *blocking summation*. Since the computational effort of each element is nearly equal, the blocking appears only after the first element in each thread. Afterwards the elements have already an offset in real time and the *serial summation* is not blocking any more. The maximal speedup according to Amdahl is 2.05 on a 4 CPU machine, where the achieved speedup is at about 1.7 [63].

5.3 Comparison: Parallelization of the Algorithm - Co-Simulation

This and the previous example show that internal parallelization is associated with a high programming effort for each individual software. The speedup factors are good compared to the factors according to Amdahl or Gustafson but are only moderate compared to the full power of p CPUs. Using a parallel co-simulation, it is much easier to achieve good speedups which are near to p, and the code of the subsystems needs no or only marginal changes. Of course it will be advisable to combine both methods in case of a co-simulation, if a subsystem is a priori implemented in parallel.

6 Conclusion

The simulation of physical systems is a very important task in modern product development. Besides the simulation of single domain systems with high quality also the simulation of multi domain systems as a whole has increasing importance. One possibility to calculate such systems is a co-simulation having the advantage that preexisting systems of different domains can be coupled. Due to the high computational effort of large multi domain systems, time efficient simulation appears to be a main challenge of the problem. Particularly with regard to current processor development, which leads nowadays to a larger number of cores rather than faster single CPUs, parallelization is a major topic. Both multi domain simulations and efficient usage of multi CPUs can be obtained by the presented parallel co-simulation framework.

Chapter 2 starts with a draft for the proposed parallel co-simulation using the fundamental master-slave concept and a classification of this concept by well known co-simulation variants from literature. The master acts as a global manager for all subsystems, being the slaves, which do the main computational work, the time integration. Beside the master-slave concept the main difference of this framework compared to others lies in the evaluation of the coupling law. It is not part of one of the subsystems but belongs to the master which evaluates the coupling law in a special discrete form. The main viewpoint for the development of this framework is a simple interface to be able to include a maximal number of preexisting simulation tools, even including commercial software. Moreover, the number of unknown numerical parameters like the macro step size is minimized. Good parallel scaling on multi CPU workstations or multi computer clusters are a central topic. An essential aspect is the stability of the coupling being discussed in detail. A numerical optimization concerning the stability of special couplings yields great improvement compared to couplings well known from literature. However, especially for large macro step sizes the accuracy of the numerical solution must be investigated, which is done by a local order analysis of different coupling variants. These analysis are applied to mechanical couplings and are taken over to hydraulic and control couplings. It is proven to be feasible to reformulate these domains in a way that they are mathematical identical to mechanical couplings. The chapter closes with a long time analysis of the constituted methods in comparison to other common co-simulation coupling variants.

In Chapter 3 an overview of the implementation of the parallel co-simulation is given. From the field of information technology a large number of inter-process-communication methods are available. These are necessary for co-simulation and proven to be more or less usable for the data exchange and synchronization between the subsystems. Besides the discussion of the usage and implementation of these IPC

methods also the basic program structure of the master and the slaves are outlined. The implementation of subsystems as co-simulation slaves is shown for preexisting open source simulation tools as well as for closed source commercial software. Closed source tools limit the co-simulation interface to the minimal interfaces common for all closed source tools.

Examples for the application of the parallel co-simulation framework are provided in Chapter 4. It starts with a simple academic example being used as a benchmark problem for different inter-process-communication methods. The investigation shows, that parallel co-simulations are only reasonable on shared memory machines or on a computer cluster providing good communication methods and if the subsystems are large enough. The different coupling variants are compared concerning the maximal parallel speedup using a timing chain drive with three subdrives of a V6 combustion engine. The results confirm the mathematical stability analysis from Chapter 2. Physical models with different time scales, known as multi scale problems, are very problematic with respect to numerical efficiency. In this case co-simulation is useful if the model can be divided into submodels each having mainly a single time scale. For the provided coupling example between a mechanical and hydraulic one this leads to a simulation speedup being larger than the number of subsystems on individual CPUs. The last example, a timing chain drive with a hydraulic chain tensioner and a valve train including a hydraulic cam phasing system, shows the full power of the parallel co-simulation framework: besides the coupling of different domains and different preexisting simulation tools also a simulation speedup using parallelization and model splitting is reached.

The work is closed by an overview on internal parallelization in Chapter 5. Exemplarily the benefit of internal parallelization is analyzed for two simulation tools. The results show that it is very hard to achieve good parallel speedups using this method. Moreover, the implementation of a parallel algorithm might be very time consuming but not very efficient, at least for multi body algorithms.

Summing up, the presented work shows that parallel co-simulation is a very helpful simulation method: it offers the possibility to couple different preexisting simulation tools, even closed source software. Moreover the co-simulation can run in parallel on a multi CPU workstation or a computer cluster. Hence, it is feasible to reuse existing models and to investigate the dynamics of coupled multi domain systems. In many cases the simulation time of the coupled system even may not be larger than the simulation time of the subsystem with the highest computational effort. If a single system can be split into subsystems a reduction of the simulation time is reachable.

The co-simulation macro step size was chosen constant for most investigation in this work. An extension to a non constant macro step size is possible but leads to extrapolation parameters being dependent on the ratio of the step sizes which may complicate the analytical analyzes a lot. However, the required macro step size controlling is topic of current research and may improve co-simulation even more.

Bibliography

[1] Adams - multibody dynamics. `http://www.mscsoftware.com/Contents/Products/CAE-Tools/Adams.aspx`.

[2] ANSYS - smart engineering simulation (FEM). `http://www.ansys.com`.

[3] Arnold, M. Numerische Verfahren in der Mehrkörperdynamik, 2004. Workshop: La Villa, Niederpöcking.

[4] Arnold, M., and Günther, M. Preconditioned dynamic iteration for coupled differential-algebraic systems. *BIT Numerical Mathematics 41* (Jan. 2001), 1–25.

[5] Arnold, V. I. *Ordinary Differential Equations*. Springer-Verlag, Berlin, 2006.

[6] AVL - powertrain systems. `http://www.avl.com`.

[7] Backé, W., and Murrenhoff, H. *Grundlagen der Ölhydraulik*, 10 ed. Institut für fluidtechnische Antriebe und Steuerungen, 1994.

[8] Bathe, K. J. *Finite Element Procedures*. Prentice Hall, India, 2007.

[9] Baumgarte, J. Stabilization of constraints and integrals of motion in dynamical systems. *Computer Methods in Applied Mechanics and Engineering* (1972), 1–16.

[10] Beauchamp, K. G. *Signal Processing: Using Analog and Digital Techniques*. George Allen & Unwin, London, 1973.

[11] Booch, G., Rumbaugh, J., and Jacobson, I. *Unified Modeling Language User Guide*, 2 ed. Addison-Wesley, 2005.

[12] Boost.MPI - a library for message passing in high-performance parallel applications. `http://www.boost.org`.

[13] Borchsenius, F. *Simulation ölhydraulischer Systeme*. VDI Fortschritt-Berichte, Reihe 8, Nr. 1005. VDI-Verlag, Düsseldorf, 2003.

[14] Bronstein, I. N., Semendjajew, K. A., Musiol, G., and Mühlig, H. *Taschenbuch der Mathematik*. Verlag Harri Deutsch, 1999.

[15] Busch, M., and Schweizer, B. Numerical stability and accuracy of different co-simulation techniques: Analytical investigations based on a 2-dof test model. In *1st Joint International Conference on Multibody System Dynamics* (Lappeenranta, Finland, May 25-27 2010).

[16] DSHplus - fluid power simulation software. `http://www.fluidon.com`.

[17] Dymola - multi-engineering modeling and simulation. `http://www.3ds.com/products/catia/portfolio/dymola`.

[18] Förg, M. *Mehrkörpersysteme mit mengenwertigen Kraftgesetzen - Theorie und Numerik*. VDI Fortschritt-Berichte, Reihe 20, Nr. 411. VDI-Verlag, Düsseldorf, 2007.

[19] Friedrich, M. Entwicklung einer Software zur Cosimulation mechanischer Systeme. Diplomarbeit, Technische Universität München, Lehrstuhl für Angewandte Mechanik, 2004.

[20] Fritz, P. *Dynamik schnellaufender Kettentriebe*. VDI Fortschritt-Berichte, Reihe 11, Nr. 253. VDI-Verlag, Düsseldorf, 1997.

[21] Gantmacher, F. R. *The theory of matrices 1*. AMS Chelsea Publishing, Rhode Island, 1959.

[22] Geist, A., Bequelin, A., and Dongarra, J. *PVM: Parallel Virtual Machine: A Users' Guide and Tutorial for Network Parallel Computing*. Scientific and Engineering Computation. MIT Press, 1994.

[23] Glocker, C. *Set-Valued Force Laws*. Springer, Berlin, Heidelberg, 2001.

[24] Gresho, P. M., and Sani, R. L. *Incompressible Flow and the Finite Element Method: Volume 1: Advection-Diffusion*. Wiley, Chichester, 1998.

[25] Hairer, E., Lubich, C., and Wanner, G. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer Series in Computational Mathematics. Springer-Verlag, Berlin, 2006.

[26] Hairer, E., Nørsett, S., and Wanner, G. *Solving Ordinary Differential Equations I - Nonstiff Problems*. Springer-Verlag, 1986.

[27] Hairer, E., and Wanner, G. *Solving Ordinary Differential Equations II - Stiff and Differential-Algebraic Problems*. Springer-Verlag, 1991.

[28] Harris, F. J. *Multirate Signal Processing for Communication Systems*. Prentice Hall, 2004.

[29] Hindmarsh, A. C. ODEPACK, a systematized collection of ODE solvers. *IMACS Transactions on Scientific Computation, Amsterdam 1* (1983), 55–64.

[30] Hippmann, G., Arnold, M., and Schittenhelm, M. Efficient simulation of bush and roller chain drives. In *Multibody Dynamics 2005, ECCOMAS Thematic Conference* (Madrid, Spain, 2005).

[31] Hoffmann, S., and Lienhard, R. *OpenMP*. Springer-Verlag, Berlin, 2008.

[32] Hösl, A. *Dynamiksimulation von Steuerkettentrieben*. VDI Fortschritt-Berichte, Reihe 12, Nr. 618. VDI-Verlag, Düsseldorf, 2005.

[33] IEEE 1003.1. IEEE standard for information technology - portable operating system interface (POSIX) base specifications, issue 7, 2008.

[34] Karnopp, D., , Margolis, D., and Rosenberg, R. *System Dynamics: Modeling and Simulation of Mechatronic Systems*, 3 ed. McGraw-Hill, 2009.

[35] Kell, T. J. *Experimentelle Schwingungsuntersuchungen an Kettentrieben*. PhD thesis, Technische Universität München, 1999.

[36] Knorr, S. Multirate-Verfahren in der Co-Simulation gekoppelter dynamischer Systeme mit Anwendung in der Fahrzeugindustrie. Diplomarbeit, Universität Ulm, Fakultät für Mathematik und Wirtschaftswissenschaften, 2002.

[37] Krüger, K., Engelhardt, T., Ginzinger, L., and Ulbrich, H. Dynamical analysis of hydraulic chain tensioners - experiments and simulation. Proceedings of the SAE 2007 World Congress, Detroit, USA.

[38] Kübler, R., and Schiehlen, W. Two methods of simulator coupling. In *Mathematical and Computer Modelling of Dynamical Systems, 6* (2000), pp. 93–113.

[39] Küpfmüller, K. *Theoretische Elektrotechnik und Elektronik*. Springer, Berlin, 2000.

[40] Leimkuhler, B., and Reich, S. *Simulating Hamiltonian Dynamics*. Cambridge Monographs on Applied and Computational Mathematics (No. 14). Cambridge University Press, Cambride, 2004.

[41] Liebig, S., Helduser, S., Stüwing, M., and Dronka, S. Die Modellierung und Simulation gekoppelter mechanischer und hydraulischer Systeme, 2001. TU Dresden.

[42] Logan, D. L. *A first course in the finite element method*. Thomson, Toronto, 2007.

[43] Lückel, J., Junker, F., and Toepper, S. Block-oriented modelling of rigid multibody systems with regard to subsystem techniques. *Advanced Multibody System Dynamics* (1993), 49–66.

[44] MATLAB/Simulink - the language of technical computing. `http://www.mathworks.de`.

[45] Mayet, J. Diskretisierung von Systemkopplungen zur parallelen Berechnung von MKS. Technische Universität München, Lehrstuhl für Angewandte Mechanik, Semesterarbeit, 2008.

[46] MBSim - multi-body simulation software. GNU Lesser General Public License `http://mbsim.berlios.de`.

[47] Modelica - modeling of complex physical systems. `http://www.modelica.org`.

[48] MPI. A message-passing interface standard - version 2.2. `http://www.mpi-forum.org/docs/mpi-2.2/mpi22-report.pdf`, 2009.

[49] MPICH2 - a high performance and widely portable implementation of the MPI standard. `http://www.mcs.anl.gov/research/projects/mpich2`.

[50] Niemann, G. *Maschinenelemente*. Springer-Verlag Berlin Heidelberg New York, 1981.

[51] OMG Object Management Group. Unified modeling language: Infrastructure. `http://www.omg.org/spec/UML/2.0/Infrastructure/PDF`, 2005.

[52] OpenMBV - open multi body viewer. GNU General Public License `http://openmbv.berlios.de`.

[53] OpenMPI - open source high performance computing. `http://www.open-mpi.org`.

[54] Otter, M., Elmqvist, H., and Mattsson, S. E. Multidomain modeling with modelica. In *Handbook of Dynamic System Modeling* (2007), P. A. Fishwick, Ed., Chapman & Hall/CRC Computer and Information Science Series, Chapman & Hall CRC, Taylor & Francis Group.

[55] Pfeiffer, F. *Einführung in die Dynamik*. B. G. Teuber Stuttgart, 1992.

[56] Pfeiffer, F., and Glocker, C. *Multibody Dynamics with Unilateral Contacts*. John Wiley & Sons, New York, 1996.

[57] RecurDyn - simulation at its best. `http://functionbay.de/index.php`.

[58] Reddy, J. N. *Energy principles and variational methods in applied mechanics*. Wiley, New York, 2002.

[59] Reuter, M., and Zacher, S. *Regelungstechnik für Ingenieure: Analyse, Simulation und Entwurf von Regelkreisen*. Vieweg, 2004.

[60] Runge, C. Über empirische Funktionen und Würfel Interpolation zwischen äquidistanten Ordinaten. In *Zeitschrift für Mathematik und Physik 46* (1901), p. 224–243.

[61] Schiehlen, W. *Multibody Systems Handbook*. Springer-Verlag, Berlin, Heidelberg, New York, 1990.

[62] Schindler, T., Förg, M., Friedrich, M., Schneider, M., Esefeld, B., Huber, R., Zander, R., and Ulbrich, H. Analysing dynamical phenomenons: Introduction to MBSim. In *Proceedings of 1st Joint International Conference on Multibody System Dynamics, Lappeenranta, May 25-27 2010* (2010).

[63] Schindler, T., Friedrich, M., and Ulbrich, H. Computing time reduction possibilities in multibody dynamics. In *Multibody Dynamics: Computational Methods and Applications*, W. Blajer, K. Arczewski, J. Fraczek, and M. Wojtyra, Eds., Computational Methods in Applied Sciences. Springer, Dordrecht, 2010.

[64] Schindler, T., Ulbrich, H., Pfeiffer, F., van der Velde, A., and Brandsma, A. Spatial simulation of pushbelt CVTs with timestepping schemes. *Applied Numerical Mathematics* (2010).

[65] Schindler, T., Zander, R., Grundl, K., Missel, R., and Ulbrich, H. Spatial MFR-FE sheave with NURBS-based contact description. In *Proceedings of 7th EUROMECH Solid Mechanics Conference* (Lisbon, Portugal, September 2009).

[66] Schneider, M., Krüger, K., Friedrich, M., and Ulbrich, H. Wechselwirkungen hydraulischer Nockenwellenversteller mit Ventil- und Steuertrieb. In *6. Fachtagung Schwingungen in Antrieben 2009 / VDI-Schwingungstechnik* (Leonberg, Oktober 2009), no. 2077, Düsseldorf : VDI-Verl., pp. 133–142.

[67] Schneider, M., Krüger, K., and Ulbrich, H. Experiments and simulation of hydraulic cam phasing systems. In *Variable Valve Optimization (2008)* (Detroit, Michigan, USA, 2008).

[68] Schneider, M., Krüger, K., and Ulbrich, H. Simulation of hydraulic systems with set-valued force laws. In *1st Joint International Conference on Multibody System Dynamics* (Lappeenranta, Finland, May 25-27 2010).

[69] Schneider, M., and Ulbrich, H. Simulation of engine valve trains containing hydraulic elements and timing chain drives using co-simulation. In *Proceedings of the 80th Annual Meeting of the Gesellschaft für Angewandte Mathematik und Mechanik e.V.* (Gdansk, Poland, Feb. 9-13 2009).

[70] Schweizer, B., and Busch, M. Explicit and implicit solver coupling: Stability analysis based on an eight-parameter test-model. In *Proceedings of the 81th Annual Meeting of the International Association of Applied Mathematics and Mechanics* (Karlsruhe, Germany, Mar. 22-26 2010).

[71] Shabana, A. A. Constrained motion of deformable bodies. *International Journal for Numerical Methods in Engineering 31* (1991), 1813–1831.

[72] SIMPACK - multi-body simulation software. `http://www.simpack.com`.

[73] Stevens, R. W. *Unix Network Programming, Volume 2: Interprocess Communications*. Prentice Hall, 1998.

[74] Stiegelmayr, A. *Zur numerischen Berechnung strukturvarianter Mehrkörpersysteme*. VDI Fortschritt-Berichte, Reihe , Nr. VDI-Verlag, Düsseldorf, 2001.

[75] Studer, C. W. *Augmented time-stepping integration of non-smooth dynamical systems*. PhD thesis, Eidgenössische Technische Hochschule Zürich, 2008.

[76] Tanenbaum, A. S. *Moderne Betriebssysteme*, 1 ed. Pearson Studium, Prentice Hall, 2002.

[77] Ulbrich, H. *Maschinendynamik*. Teubner Verlag, 1996.

[78] Ulbrich, H. Some selected research activities in mechatronic applications - case studies. In *Proceedings of 2nd International Conference on Engineering Mechanics, Structures, Engineering Geology* (Rodos Island, Greece, 2009).

[79] Versteeg, H. K. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. Prentice Hall, Harlow, 2007.

[80] Zander, R. *Flexible Multi-Body Systems with Set-Valued Force Laws*. VDI Fortschritt-Berichte, Reihe 20, Nr. 420. VDI-Verlag, Düsseldorf, 2008.

[81] Zander, R., and Ulbrich, H. Impacts on beam structures: Interaction of wave propagation and global dynamics. In *IUTAM Symposium* (Garching b. München, Germany, 2006).