The 5th Joint International Conference on Multibody System Dynamics
June 24 –28, 2018, Lisbon, Portugal

Efficiency of Different Co-Simulation Approaches

Jan Kraft, Tobias Meyer and Bernhard Schweizer

Institute of Applied Dynamics, Technical University Darmstadt, [kraft,meyer,schweizer]@ad.tu-darmstadt.de

ABSTRACT — *Besides the wide range of potential applications in the field of multidisciplinary simulations, co-simulation techniques may also be utilized to parallelize large monodisciplinary dynamical models. This paper focuses on the reduction of computation time that can be achieved in the simulation of multibody systems by partitioning a monolithic model into a variable number of coupled subsystems. The connection between the subsystems can be described in various ways. In this work, different subsystems are coupled by nonlinear constitutive equations (applied-force coupling approach). Exchange of coupling information takes only place at distinct macro-time points. The essential point is that the subsystems are integrated independently of each other between the macro-time points. If a Jacobi-type co-simulation scheme is used, all subsystems can be solved in parallel.*

1 Introduction

Co-simulation or solver coupling methods are used in various fields of applications. Examples can be found in [1] and [2]. The basic idea of co-simulation is to decompose an overall system into coupled subsystems. The formulation of the coupling conditions between two (or several) subsystems depends on the considered problem. In the case of mechanical systems, the decomposition of an overall model may be achieved by cutting through joints or by cutting through elements representing a physical force (torque). This leads to a coupling by reaction forces/torques [3][4] (constraint coupling) or to a connection by applied forces/torques (applied-force coupling). Co-simulation methods may further be subdivided into force/force-, force/displacement- and displacement/displacement-coupling approaches [5]. In this contribution, a force/force-decomposition approach is used and the subsystems are connected by nonlinear spring/damper-elements.

The methods presented here are weak coupling approaches, which implies that each subsystem is solved independently from the other subsystems within a macro-time step. Information (i.e. coupling variables) is only exchanged between the subsystems at certain communication-time points. The unknown coupling variables are approximated (extrapolated/interpolated) in the subsystems within a macro-time step. The independent integration of the subsystems within the macro-time steps is the essential point for parallelizing the computation.

In this manuscript, two different numerical methods for solving the coupled problem are examined: an explicit co-simulation technique and a semi-implicit integration scheme. The semi-implicit method is based on a predictor/corrector procedure, where the corrector step is carried out only once.

2 Co-Simulation Methods

To investigate the performance of explicit and semi-implicit co-simulation approaches with regard to the computation time, we use a nonlinear dynamical test model, which is described in the following subsection.

2.1 Test Model

One requirement for the test model is that it is straightforward to scale with respect to the number of degrees of freedom and with regard to the number of subsystems. Therefore, a chain of n_K masses connected by nonlinear spring/damper-elements is used as test model.

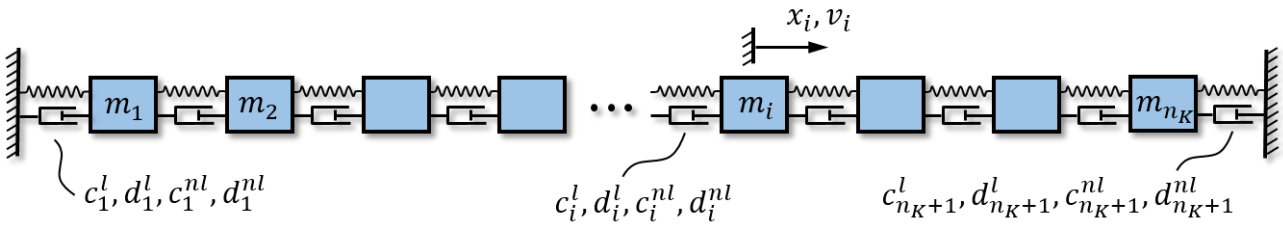

Fig. 1: Nonlinear test model: oscillator chain

Denoting the displacements of the oscillator-masses by the displacement coordinates x_i and the corresponding velocities by v_i, we obtain a system of $2n_K$ ordinary differential equations of the form

$$\dot{x}_i = v_i$$

$$\dot{v}_i = \frac{c_i^l}{m_i}(x_{i-1} - x_i) + \frac{d_i^l}{m_i}(v_{i-1} - v_i) + \frac{c_{i+1}^l}{m_i}(x_{i+1} - x_i) + \frac{d_{i+1}^l}{m_i}(v_{i+1} - v_i)$$
$$+ \frac{c_i^{nl}}{m_i}(x_{i-1} - x_i)^3 + \frac{d_i^{nl}}{m_i}(v_{i-1} - v_i)^3 + \frac{c_{i+1}^{nl}}{m_i}(x_{i+1} - x_i)^3 + \frac{d_{i+1}^{nl}}{m_i}(v_{i+1} - v_i)^3$$

with $i = 1 \ldots n_K$. We assume that the chain is fixed at both ends ($x_0 = v_0 = x_{n_K+1} = v_{n_K+1} = 0$). The model parameters are the masses m_i, the linear stiffness coefficients c_i^l, the linear damping coefficients d_i^l, the nonlinear stiffness coefficients c_i^{nl} and the nonlinear damping coefficients d_i^{nl}.

2.2 Decomposition of the Test Model

As mentioned above, the overall system is split into coupled subsystems by a force/force-decomposition approach [5]. This is achieved by cutting through certain nonlinear spring/damper-elements and by using the corresponding forces as coupling variables. The number of subsystems n_{sub} is arbitrary, but usually much smaller than the number n_K of degrees of freedom of the overall system.

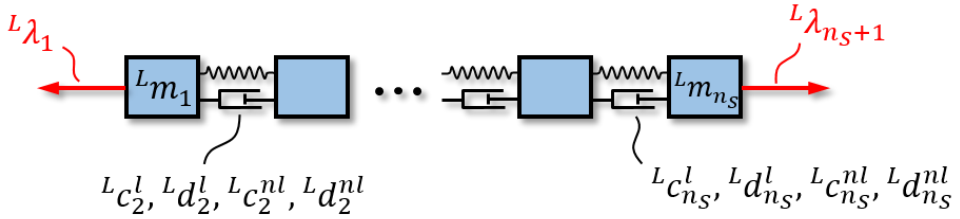

Fig. 2: Arbitrary subsystem L

The set of n_S equations of motion for the arbitrary subsystem L reads as

$$
\begin{aligned}
{}^{L}\dot{x}_j &= {}^{L}v_j \\
{}^{L}\dot{v}_j &= \frac{{}^{L}c_j^l}{{}^{L}m_j}\left({}^{L}x_{j-1} - {}^{L}x_j\right) + \frac{{}^{L}d_j^l}{{}^{L}m_j}\left({}^{L}v_{j-1} - {}^{L}v_j\right) + \frac{{}^{L}c_{j+1}^l}{{}^{L}m_j}\left({}^{L}x_{j+1} - {}^{L}x_j\right) \\
&\quad + \frac{{}^{L}d_{j+1}^l}{{}^{L}m_j}\left({}^{L}v_{j+1} - {}^{L}v_j\right) + \frac{{}^{L}c_j^{nl}}{{}^{L}m_j}\left({}^{L}x_{j-1} - {}^{L}x_j\right)^3 + \frac{{}^{L}d_j^{nl}}{{}^{L}m_j}\left({}^{L}v_{j-1} - {}^{L}v_j\right)^3 \\
&\quad + \frac{{}^{L}c_{j+1}^{nl}}{{}^{L}m_j}\left({}^{L}x_{j+1} - {}^{L}x_j\right)^3 + \frac{{}^{L}d_{j+1}^{nl}}{{}^{L}m_j}\left({}^{L}v_{j+1} - {}^{L}v_j\right)^3 - \frac{{}^{L}\lambda_j}{{}^{L}m_j} + \frac{{}^{L}\lambda_{j+1}}{{}^{L}m_j}
\end{aligned}
\tag{1}
$$

with $j = 1 \ldots n_S$ and ${}^{L}c_1^l = {}^{L}d_1^l = {}^{L}c_1^{nl} = {}^{L}d_1^{nl} = {}^{L}c_{n_S+1}^l = {}^{L}d_{n_S+1}^l = {}^{L}c_{n_S+1}^{nl} = {}^{L}d_{n_S+1}^{nl} = 0$. The coupling forces are denoted by ${}^{L}\lambda_j$ and ${}^{L}\lambda_{j+1}$; they are only required for the coupling bodies ($j = 1$ and $j = n_S$) and are set to zero for the remaining bodies (${}^{L}\lambda_2 = \cdots = {}^{L}\lambda_{n_S} = 0$).

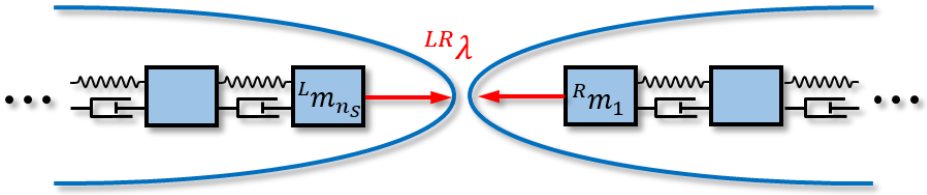

Fig. 3: Coupling of two adjacent subsystems *L* and *R*

The coupling condition for two adjacent subsystems *L* and *R* (assuming that body n_S of subsystem *L* is coupled with body 1 of subsystem *R*) is defined by

$$
\begin{aligned}
{}^{LR}g := {}^{LR}\lambda - {}^{LR}c_c^l\left({}^{R}x_1 - {}^{L}x_{n_S}\right) - {}^{LR}d_c^l\left({}^{R}v_1 - {}^{L}v_{n_S}\right) \\
- {}^{LR}c_c^{nl}\left({}^{R}x_1 - {}^{L}x_{n_S}\right)^3 - {}^{LR}d_c^{nl}\left({}^{R}v_1 - {}^{L}v_{n_S}\right)^3 = 0
\end{aligned}
\tag{2}
$$

with the coupling force ${}^{LR}\lambda = {}^{L}\lambda_{n_S+1} = {}^{R}\lambda_1$, the coupling parameters ${}^{LR}c_c^l$, ${}^{LR}d_c^l$, ${}^{LR}c_c^{nl}$ and ${}^{LR}d_c^{nl}$ and the state variables of the two coupling bodies.

2.3 Explicit Co-Simulation Scheme

To solve the decomposed system as a coupled problem by using an explicit co-simulation method, a macro-time grid is introduced. Within an arbitrary macro-step from T_N to $T_{N+1} = T_N + h_{mac}$, each subsystem is integrated using extrapolation (interpolation) polynomials of degree n_{pol} to approximate the coupling forces. After the integration of the subsystems, the resulting states of the coupling bodies are substituted into the coupling condition (2) to obtain the coupling force at the new macro-time point T_{N+1} (update of the coupling variables). The explicit co-simulation method has the advantage that a repetition of the macro-step is not necessarily required if a constant macro-step size is used (i.e. for the case that a macro-step size controller is not used). This may be an important point, if commercial subsystem solvers are used, which often do not allow solver reinitialization.

The explicit co-simulation method may be improved by using a macro-step size controller. To estimate the error of a macro-step, a second subsystem integration process within each macro-step is required. For the second integration, the sampling point $\lambda_{N+1}^{(o)}$ of the (original) approximation polynomial $\lambda^{(o)}(t)$ of the coupling force at the new macro-point T_{N+1} is modified. The modified value $\lambda_{N+1}^{(m)}$ of the approximated coupling force is obtained by increasing the extrapolation order by one $(n_{pol} + 1)$. It has to be noted that the extrapolation order is only increased to obtain the modified value for the sampling point $\lambda_{N+1}^{(m)}$, the polynomial order of the modified approximation polynomial $\lambda^{(m)}(t)$ remains n_{pol}. This process is illustrated for an arbitrary coupling force λ in

Fig. 4. To clarify the procedure of generating the approximation polynomial $\lambda^{(m)}(t)$ for the error estimator, the equations to obtain the sampling points are presented in (3) and the corresponding approximation polynomials in (4). The abbreviation P_i represents inter-/extrapolation polynomials of order i.

$$\begin{aligned} \lambda_{N+1}^{(o)} &= P_{n_{pol}}\left(\left[T_{N-n_{pol}}, \lambda_{N-n_{pol}}\right], \dots, [T_N, \lambda_N], T_{N+1}\right) \\ \lambda_{N+1}^{(m)} &= P_{n_{pol}+1}\left(\left[T_{N-n_{pol}-1}, \lambda_{N-n_{pol}-1}\right], \dots, [T_N, \lambda_N], T_{N+1}\right) \end{aligned} \tag{3}$$

$$\begin{aligned} \lambda^{(o)}(t) &= P_{n_{pol}}\left(\left[T_{N-n_{pol}+1}, \lambda_{N-n_{pol}+1}\right], \dots, \left[T_{N+1}, \lambda_{N+1}^{(o)}\right], t\right) \\ \lambda^{(m)}(t) &= P_{n_{pol}}\left(\left[T_{N-n_{pol}+1}, \lambda_{N-n_{pol}+1}\right], \dots, [T_{N+1}, \lambda_{N+1}^{m}], t\right) \end{aligned} \tag{4}$$

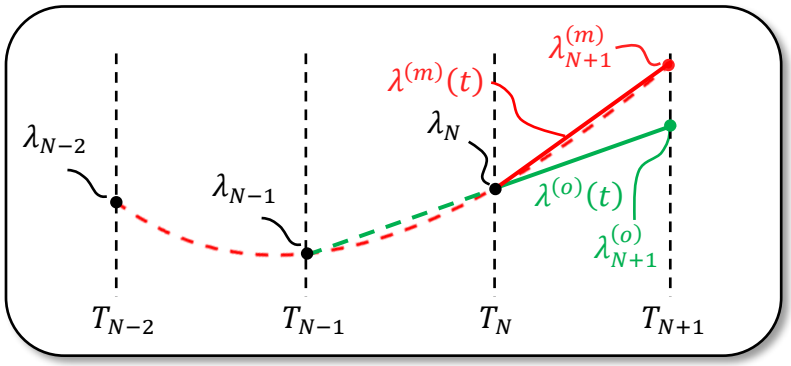

Fig. 4: Approximation polynomials (linear) of an arbitrary coupling force λ, green: original approximation polynomial, red: modified approximation polynomial for the error estimation

The resulting states of the two subsystem integration processes (with the original and the modified approximation polynomial) are used to estimate the error ε of the macro-step according to the following equation

$$\varepsilon = \sqrt{\sum_{\substack{\text{coupling} \\ \text{bodies}}} \left(\left(\frac{C_\varepsilon \cdot \left(x_{N+1,i}^{(o)} - x_{N+1,i}^{(m)} \right)}{\text{atol}_i + \text{rtol} \cdot x_{N+1,i}^{(o)}} \right)^2 \right)}. \tag{5}$$

The error constant C_ε depends on the extrapolation order and the step sizes of the previous macro-steps. The position variables $x_{N+1,i}^{(o)}$ and $x_{N+1,i}^{(m)}$ are the displacements of body i at T_{N+1} obtained by the subsystem integration with the original and with the modified approximation polynomial of the coupling force. The error ε is the estimated local error on position level. An error estimator on velocity level can be constructed in the same way. The absolute and relative error tolerances atol_i and rtol are user defined values. The new macro-step size is determined according to

$$h_{mac}^{new} = 0.9 \cdot \varepsilon^{-\frac{1}{n_{pol}+3}} \cdot h_{mac}^{old}, \tag{6}$$

where $n_{pol} + 3$ is the convergence order of the co-simulation method on position level. A detailed description of this macro-step size controller can be found in [6].

2.4 Semi-Implicit Co-Simulation Scheme

A detailed description of the implemented semi-implicit co-simulation procedure can be found in [5]. The basics of the approach are only briefly explained next.

The macro-time grid is assumed to be equidistant (macro-step size $h_{mac} = const.$). As mentioned above, the presented semi-implicit co-simulation method is based on a predictor/corrector approach with only one corrector step. An arbitrary macro-time step from T_N to T_{N+1} is explained next for a co-simulation with two subsystems L and R (assuming that body n_S of subsystem L is coupled with body 1 of subsystem R).

Within the predictor step, each subsystem is integrated twice from T_N to T_{N+1}: firstly with the predicted (extrapolated) coupling force $\lambda^p(t)$ ($= {}^{LR}\lambda^p = {}^{L}\lambda^p_{n_S+1} = {}^{R}\lambda^p_1$) and secondly with the perturbed predicted coupling force $\lambda^\Delta(t)$ ($= {}^{LR}\lambda^\Delta = {}^{L}\lambda^\Delta_{n_S+1} = {}^{R}\lambda^\Delta_1$). Note that for the reason of a concise representation, the subsystem indices have been omitted. The sampling point λ^Δ_{N+1} of the perturbed coupling force at T_{N+1} is obtained by adding a small, user-defined perturbation $\Delta\lambda$ to the sampling point λ^p_{N+1} of the predicted coupling force as given in (7).

$$\begin{aligned} \lambda^p_{N+1} &= P_{n_{pol}}\left(\left[T_{N-n_{pol}}, \lambda_{N-n_{pol}}\right], \dots, [T_N, \lambda_N], T_{N+1}\right) \\ \lambda^\Delta_{N+1} &= \lambda^p_{N+1} + \Delta\lambda \end{aligned} \tag{7}$$

$$\begin{aligned} \lambda^p(t) &= P_{n_{pol}}\left(\left[T_{N-n_{pol}+1}, \lambda_{N-n_{pol}+1}\right], \dots, [T_{N+1}, \lambda^p_{N+1}], t\right) \\ \lambda^\Delta(t) &= P_{n_{pol}}\left(\left[T_{N-n_{pol}+1}, \lambda_{N-n_{pol}+1}\right], \dots, \left[T_{N+1}, \lambda^\Delta_{N+1}\right], t\right) \end{aligned} \tag{8}$$

With the predicted state variables $\boldsymbol{z}^p$ and the perturbed predicted states $\boldsymbol{z}^\Delta$ at T_{N+1}, the partial derivatives of the states with respect to the coupling variables can be approximated by finite differences

$$\left.\frac{\partial \boldsymbol{z}_c}{\partial\lambda}\right|_{\lambda^p_{N+1}} = \lim_{\Delta\lambda\to 0}\frac{\boldsymbol{z}_c(\lambda^\Delta) - \boldsymbol{z}_c(\lambda^p)}{\Delta\lambda} \approx \frac{\boldsymbol{z}^\Delta_c - \boldsymbol{z}^p_c}{\Delta\lambda}. \tag{9}$$

Note that partial derivatives only have to be calculated for the states $\boldsymbol{z}_c$ of the coupling bodies.

The approximated partial derivatives obtained in the predictor step are utilized to compute the improved (corrected) coupling force. Therefore, the coupling condition (2) is considered as a function of the coupling force λ at T_{N+1} and expanded in a Taylor series. Choosing λ^p_{N+1} as expansion point and neglecting higher-order terms $\mathcal{O}(\lambda^2)$, one obtains the linearized coupling condition

$$\begin{aligned} g^{lin}(\lambda) &:= g\left(\lambda^p_{N+1}\right) + \left.\frac{\partial g}{\partial\lambda}\right|_{\lambda^p_{N+1}}\left(\lambda - \lambda^p_{N+1}\right) \\ &= \lambda^p_{N+1} - c^l_c\left({}^{R}x^p_1 - {}^{L}x^p_{n_s}\right) - d^l_c\left({}^{R}v^p_1 - {}^{L}v^p_{n_s}\right) - c^{nl}_c\left({}^{R}x^p_1 - {}^{L}x^P_{n_s}\right)^3 - d^{nl}_c\left({}^{R}v^p_1 - {}^{L}v^p_{n_s}\right)^3 \\ &\quad + \left[1 - c^l_c\left(\left.\frac{\partial {}^{R}x^p_1}{\partial\lambda}\right|_{\lambda^p_{N+1}} - \left.\frac{\partial {}^{L}x^p_{n_s}}{\partial\lambda}\right|_{\lambda^p_{N+1}}\right) - d^l_c\left(\left.\frac{\partial {}^{R}v^p_1}{\partial\lambda}\right|_{\lambda^p_{N+1}} - \left.\frac{\partial {}^{L}v^p_{n_s}}{\partial\lambda}\right|_{\lambda^p_{N+1}}\right)\right. \\ &\quad - 3\,c^{nl}_c\left({}^{R}x^p_1 - {}^{L}x^p_{n_s}\right)^2\left(\left.\frac{\partial {}^{R}x^p_1}{\partial\lambda}\right|_{\lambda^p_{N+1}} - \left.\frac{\partial {}^{L}x^p_{n_s}}{\partial\lambda}\right|_{\lambda^p_{N+1}}\right) \\ &\quad \left. - 3\,d^{nl}_c\left({}^{R}v^p_1 - {}^{L}v^p_{n_s}\right)^2\left(\left.\frac{\partial {}^{R}v^p_1}{\partial\lambda}\right|_{\lambda^p_{N+1}} - \left.\frac{\partial {}^{L}v^p_{n_s}}{\partial\lambda}\right|_{\lambda^p_{N+1}}\right)\right]\left(\lambda - \lambda^p_{N+1}\right) = 0\ . \end{aligned} \tag{10}$$

Solving equation (10) for the coupling force λ yields the corrected coupling force λ^c_{N+1}. In general, the predicted state variables and the predicted coupling force will not fulfill the coupling condition. The corrected coupling force (together with the corrected state variables), however, fulfills at least the linearized coupling condition (10). The subsystem integration within the corrector step is carried out by making use of the corrected coupling force $\lambda^c(t) = P_{n_{pol}}\left(\left[T_{N-n_{pol}+1}, \lambda_{N-n_{pol}+1}\right], \dots, [T_{N+1}, \lambda^c_{N+1}], t\right)$.

The corrected state variables together with the corrected coupling forces will in general not fulfill the nonlinear coupling conditions. To achieve consistent coupling forces, an update of the coupling forces at T_{N+1} is useful. Therefore, the corrected state variables of the coupling bodies are substituted into the coupling condition (2) in order to calculate updated coupling forces.

2.5 Subsystem Solver

The subsystems are solved with the IDA solver from the SUNDIALS (Suite of Nonlinear and Differential/Algebraic Equation Solvers) package [7]. This implicit DAE solver is based on a variable-order variable-coefficient BDF implementation combined with either direct (sparse) or iterative methods for solving the linear system within the Newton iteration. For the present studies, the direct sparse linear solver (KLU [8]) is used.

3 Remarks on the Computation Time

3.1 Parallelized Computation

Within a macro-step, each subsystem is integrated independently. Exchange of information takes only place before or after the subsystem integration processes. Therefore, all subsystems can be solved in parallel. The parallelized implementation is realized with a hybrid MPI-openMP code: each subsystem is executed by a MPI rank. The different integration processes of each subsystem within a macro-time step – these are the additional integrations for the error estimation in case that a macro-step size controller is used and the subsystems integrations with the perturbed coupling variables for the semi-implicit approach – are carried out in openMP threads that are spawned within each MPI rank (Fig. 5). The simulations for this work have been carried out on a high performance computer (Lichtenberg High Performance Computer of the TU Darmstadt) so that all subsystem integrations could be fully parallelized.

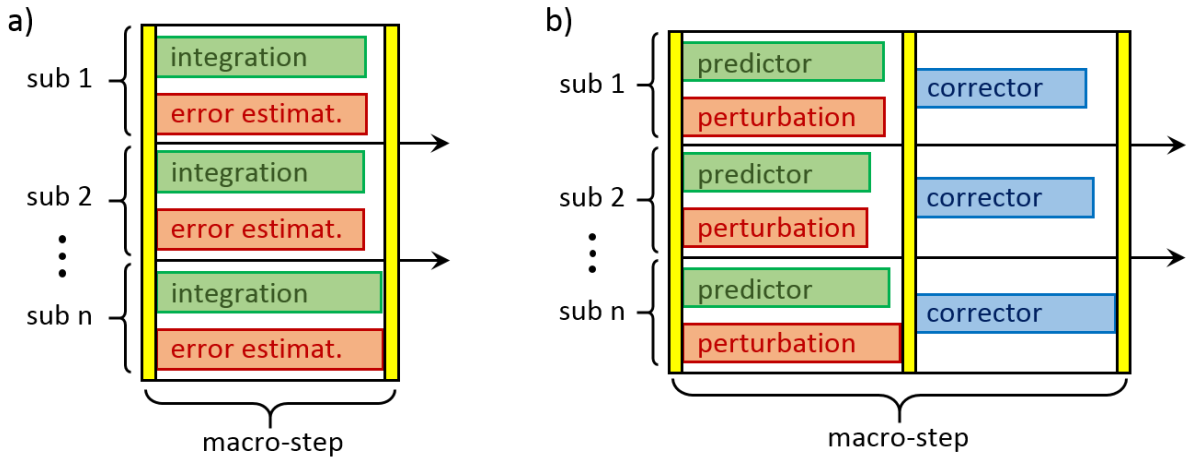

Fig. 5: Parallelization scheme: a) explicit co-simulation (with macro-step size controller) and b) semi-implicit co-simulation

Applying a parallel implementation, the simulation time is usually strongly reduced. The computation time for the co-simulation can be estimated by

$$T_{cos}^{(expl)} \approx \frac{T_{mon}}{n_{sub}^{P}} + C^{(expl)} \quad \text{and} \quad T_{cos}^{(semi)} \approx 2\frac{T_{mon}}{n_{sub}^{P}} + C^{(semi)} \,, \tag{11}$$

where T_{mon} denotes the computation time of the monolithic model and n_{sub} the number of subsystems. P represents the scaling factor of the computation time of the multibody implementation with respect to the number of degrees of freedom. For typical multibody systems, the value of P is between one and three, depending on the formulation of the equations of motion and the solving strategy. The overhead caused by the synchronization of parallel threads and additional calculations due to the co-simulation approach (e.g. solving equations (9) and (10)

for the semi-implicit method) is summarized in the parameter C. The formula for T_{cos} implies the assumption that the overall system is split into equal-sized subsystems, so that the integration times for the different subsystems are similar.

3.2 Micro-Step Size Limitation

Depending on the subsystem solver, there are different options to control the micro-step size (subsystem solver step size). The most obvious possibility is to enforce the subsystem solver to stop exactly at the end of each macro-step ("exact stop", $t_{sub} = T_{N+1}$). The problem with this option is that the step-size controller of the subsystem solver will be interrupted. This may lead to a significant increase of the number of micro-steps. A completely unrestricted micro-step size on the other hand is also undesirable, because then it can happen that the micro-step size of a subsystem is larger than the macro-step size. As a result, the subsystem will not be evaluated in each macro-step.

Another possibility is to restrict the subsystem solver step size by the macro-step size ($h_{mic} \leq h_{mac}$) and to stop the solver when it steps beyond a macro-time point ($t_{sub} \geq T_{N+1}$). The states of the coupling bodies are interpolated at the macro-time point T_{N+1}. Within the next macro-step, the subsystem solver starts at the point t_{sub} at which it has stopped before. In this case, the subsystem solver step size controller does not get affected by the co-simulation. However, the interval between the end of a macro-step T_{N+1} and the point t_{sub} at which the solver stops is integrated with the coupling force from the previous macro-step. This leads to an additional numerical error and an increased discontinuity in the coupling force. In our simulations, we observed that the restriction $h_{mic} \leq h_{mac}$ yields good results.

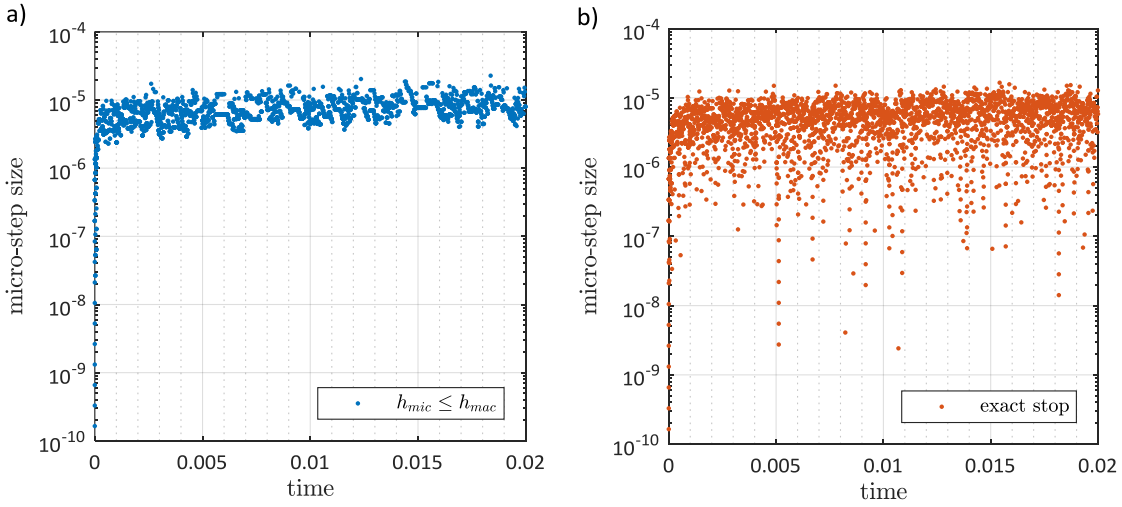

Fig. 6: Micro-step size limitation: a) $h_{mic} \leq h_{mac}$ and b) integration is stopped at each macro-time point exactly

Fig. 6 shows the micro-step size of an arbitrary subsystem solver of an explicit co-simulation carried out with the two different micro-step size limitations. As can be seen, the micro-step size of the co-simulation in which the subsystem solver is stopped exactly at each macro-time point (Fig. 6b) shows large fluctuations. The average micro-step size is therefore smaller than for the co-simulation in which the micro-step size is restricted by $h_{mic} \leq h_{mac}$.

3.3 Macro-Step Size

Considering classical time integration methods, the numerical error decreases and the computation time increases when the time step size is reduced. For co-simulation methods, the numerical error also decreases with

the macro-step size, of course, but the relation of the computation time and the macro-step size is not necessarily as straightforward as for classical time integration methods.

Assuming that the co-simulation is carried out in parallel, the overall computation time is the sum of the subsystem integration time (i.e. of the slowest subsystem integration process) and the time required for synchronization and data exchange between the subsystems. The macro-step size can affect both parts of the overall computation time. On the one hand, a larger macro-step size decreases the number of synchronization points and the data transfer between the subsystems; therefore, it reduces the computation time. On the other hand, a larger macro-step size increases the discontinuities in the coupling variables at the macro-time points. Because of these discontinuities, the subsystem solver has to reduce the micro-step size at the beginning of each macro-step. As a result the overall computation time may increase. A small macro-step size will limit the micro-step size and therefore also increase the number of micro-steps and the overall computation time.

For the determination of an appropriate macro-step size, it is necessary to know what the dominating part of the overall computation time is. If the bottleneck is the data transfer – this may be the case when there is a large number of subsystems or coupling variables – then the macro-step size should be chosen as large as possible. If the dominating factor of the computation time are the subsystem integration processes (which is mostly the case when multibody systems are coupled) then an appropriate macro-step size is a compromise between small discontinuities and minimal limitation of the micro-step size.

3.4 Differences in Subsystem Computation Times

The computation time of each subsystem solver within each macro-step varies between the subsystems. Even if all subsystems have approximately the same computation time for the overall simulation, as in our test case, there are different computation times for each subsystem in each macro-step as shown in Fig. 7b). The effect of these "local" computation time fluctuations on the overall computation time depends strongly on the macro-step size.

Fig. 7 shows a detailed view of the computation time of an explicit co-simulation with a constant macro-step size. The subsystems are physically identical. The green bars show the computation time of each subsystem in each macro-step. The macro-steps are indicated by the black lines. The red bars mark the slowest subsystem integration process in each macro-step.

In Fig. 7a) the macro-step size is chosen relatively large so that each subsystem solver makes many micro-steps (~20-30) within each macro-step. The computation times of the subsystems within each macro-step are similar.

Fig. 7b) shows results obtained with a reduced macro-step size. Here, the subsystem solver makes only a small number of micro-steps (~1-4) within each macro-step. This results in relatively large differences of the subsystem computation times. Considering that each subsystem is assigned to one core, the hardware utilization is low, because the cores are idling a large proportion of the time.

The macro-step size in Fig. 7c) is assumed to be smaller than the micro-step size that is chosen by the step size controller of the subsystem solver. Due to the restriction $h_{mic} \leq h_{mac}$, the micro-step size is limited by the macro-step size. Each subsystem solver takes only one micro-step per macro-step. As a result the computation time for all subsystems is almost equal. The price for the good hardware utilization is the increased number of micro-steps.

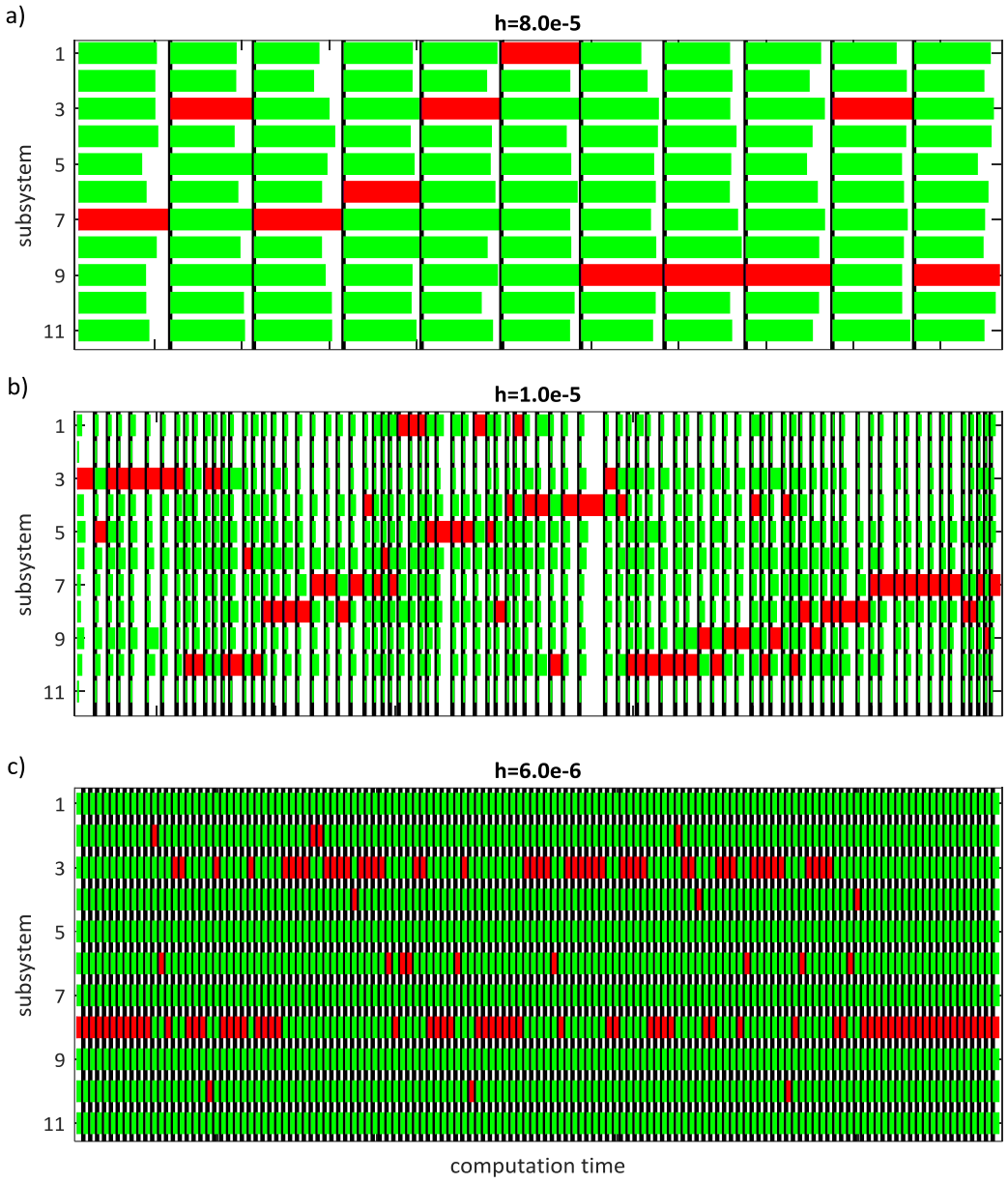

Fig. 7: Subsystem computation times for different macro-step sizes: a) ~20-30 micro-steps per macro-step, b) ~1-4 micro-steps per macro-step, c) 1 micro-step per macro-step

4 Simulation Results

To investigate different co-simulation methods, the following two parameter sets for the test model are defined:

Parameter Set 1 ("model 1")	Parameter Set 2 ("model 2")
$n_K = 7750$	
$n_{sub} = 47$	
$m_i = 1.0e0$	
$c_i^l = c_{c_i}^l = 1.0e7$	
$d_i^l = d_{c_i}^l = 1.0e0$	
$c_i^{nl} = c_{c_i}^{nl} = 1.0e9$	
$d_i^{nl} = d_{c_i}^{nl} = 1.0e\text{-}2$	
No external forces	External forces F_{Si} acting on 5% of all bodies

Tab. 1: Test model parameters

The external force F_{Si} is defined by the force law

$$F_{Si} = \frac{1}{2}\Delta F_{Si}\left[\tanh\left(\frac{t-t_i}{\delta}\right) - \tanh\left(\frac{t-(t_i+\delta)}{\delta}\right)\right].$$

This may be considered as an approximation of an impact force acting on body i over a short time interval Δt with an amplitude ΔF_{Si} at t_i. The initial positions and velocities of the bodies are chosen randomly in the interval $[-0.1, +0.1]$ and $[-100, +100]$.

4.1 Convergence Analysis

The convergence behavior of the explicit co-simulation method is investigated by varying the (constant) macro-step size and by evaluating the global and the local error of the state variables.

Fig. 8 shows the results for the convergence analysis of model 1. Co-simulations are carried out with linear (blue curve) and quadratic (red curve) approximation polynomials for the coupling variables. The subsystem solver tolerance is set to 1.0e-12 to minimize the numerical errors that are introduced by the subsystem solvers. As can be seen, the local error converges with order $n_{pol} + 3$ on position level and with order $n_{pol} + 2$ on velocity level. The global error converges with $n_{pol} + 1$.

When the subsystem solver tolerance is increased to 1.0e-6, the overall accuracy of the co-simulation is limited by the subsystem errors. This can be clearly observed for a co-simulation of model 2 with quadratic approximation polynomials for the coupling variables (Fig. 9 right hand side, red curve), where the error remains almost constant when the macro-step size is decreased below 1.0e-5. Due to the micro-step size restriction, the subsystem accuracy of model 1 is implicitly increased for small macro-step sizes; therefore, this effect cannot be observed very clearly for model 1.

The dashed curves in Fig. 9 show the error of a co-simulation with the restriction that the subsystem solver stops exactly at each macro-time point. As expected, a co-simulation with this restriction produces slightly lower errors than a co-simulation with the limitation $h_{mic} \leq h_{mac}$. The black dashed lines in Fig. 9 show the errors of a monolithic simulations of both models for comparison.

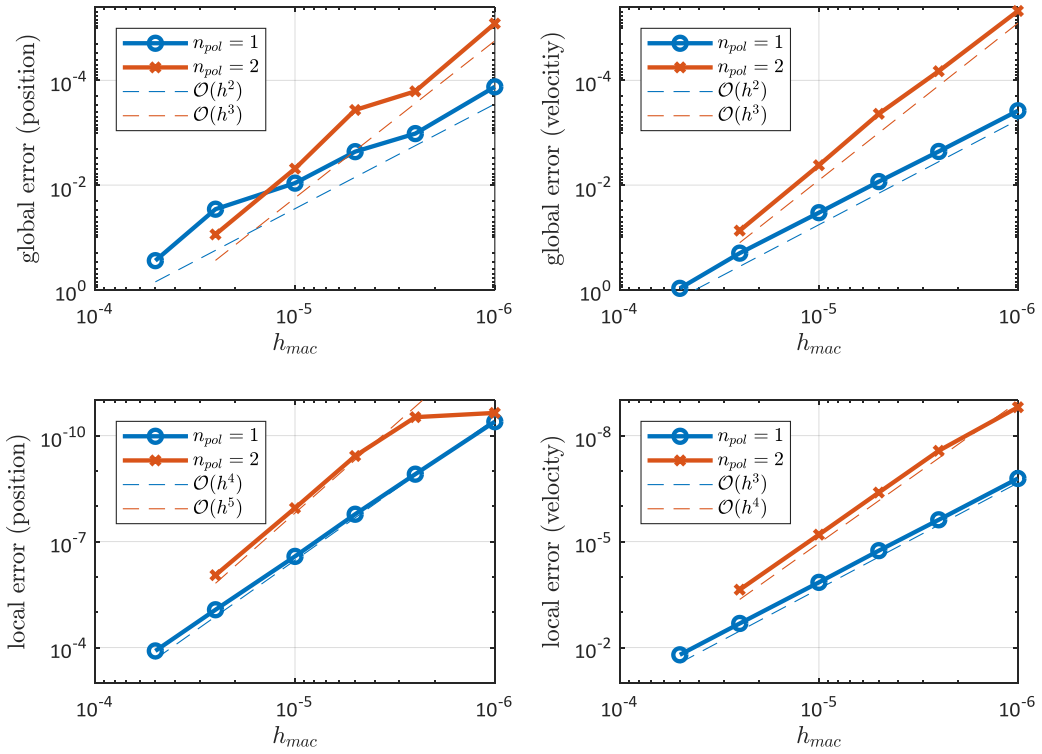

Fig. 8: Convergence analysis: explicit co-simulation (subsystem error tolerance 1.0e-12)

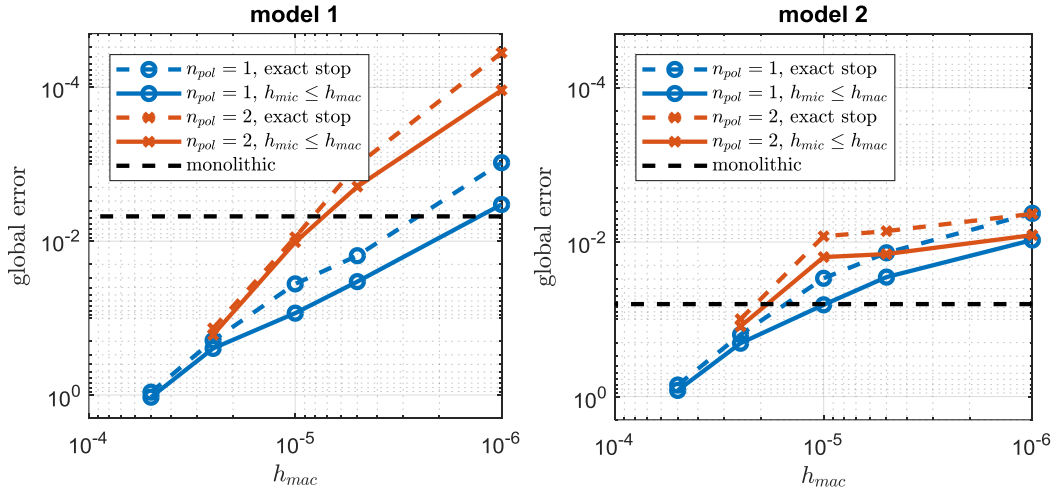

Fig. 9**:** Convergence analysis: explicit co-simulation (subsystem error tolerance 1.0e-6)

The convergence order of the semi-implicit co-simulation method is the same as for the explicit method. The advantage of the semi-implicit approach are the smaller errors and the increased numerical stability.

Fig. 10 shows a comparison of the numerical errors of the explicit (dashed curves) and the semi-implicit co-simulation method. As can be seen, the error of the semi-implicit method is for both models significantly smaller, especially for large macro-step sizes. In addition, the macro-step size can be increased to $h_{mac} = 7.5\text{e-}5$ for the semi-implicit method, while the explicit method needs a macro-step size of $h_{mac} \leq 5.0\text{e-}5$ with linear approximation polynomials or $h_{mac} \leq 2.5\text{e-}5$ with quadratic polynomials order to achieve a stable co-simulation. The black dashed lines in Fig. 10 show the error of monolithic simulations of the two models for comparison. The monolithic simulations are carried out with an error tolerance of 1.0e-6, the same error tolerance is used for the subsystem solvers.

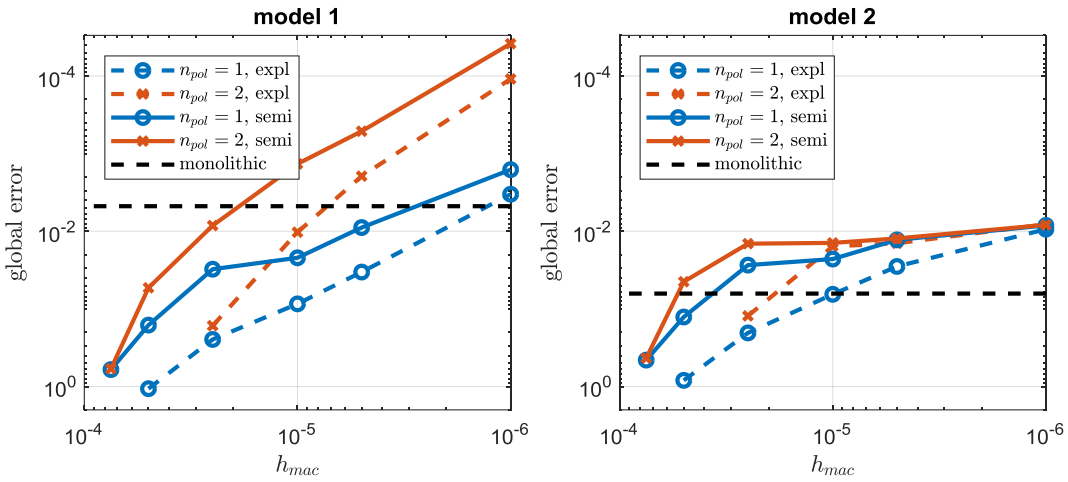

Fig. 10: Convergence analysis: semi-implicit co-simulation (subsystem error tolerance 1.0e-6)

4.2 Analysis of the Computation Time

4.2.1 Computation Time of the Monolithic Simulation

As mentioned before, the scaling factor of the computation time of a multibody system with respect to the number of degrees of freedom is typically between 1 and 3. There exist, for instance, recursive $\mathcal{O}(n)$ algorithms [9] that scale linear. Generally, when the equations of motion are formulated in absolute coordinates and the system is integrated numerically with a BDF method, the dominating factor in the computation time is the linear system that has to be solved within the Newton iterations. The system matrices in our test cases are sparse, because of the simple structure of the test model. This sparsity is exploited by the KLU linear solver resulting in an almost linear scaling of the computation time for model 1. Model 2 is – because of the excitation by the impact forces – stronger affected by the nonlinearities. Therefore, more Newton iterations are needed in each solver step.

Since the scaling factor of the computation time with respect to the degrees of freedom is higher for model 2 – at least within the considered range of degrees of freedom – we expect also a greater benefit of a parallel co-simulation for model 2. Fig. 11 shows the computation time of monolithic simulations of the two test models. The parameters are given in Table 1, only the number of masses n_K is varied.

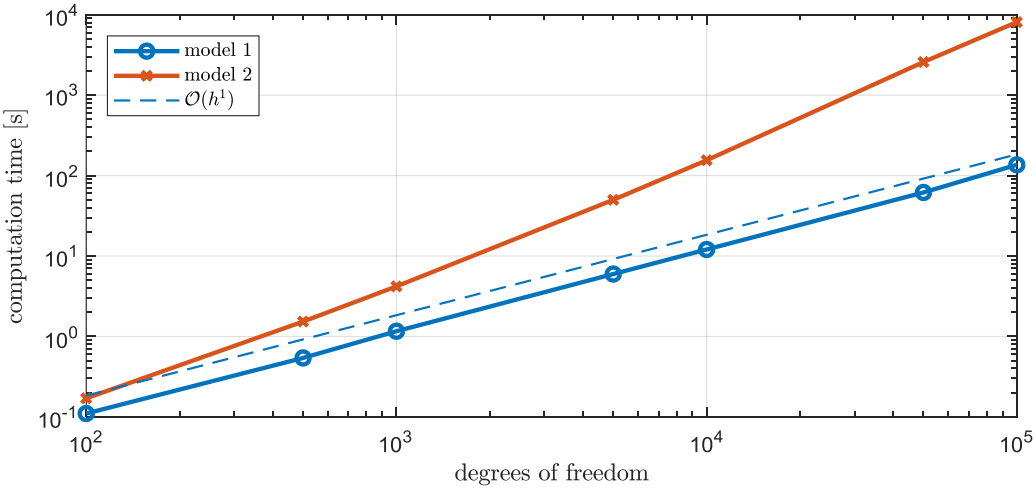

Fig. 11: Monolithic simulation: scaling of the computation time with the number of degrees of freedom

4.2.2 Influence of the Number of Subsystems on the Computation Time

The choice of the number of subsystems, in which a model should be subdivided for achieving the minimal computation time, depends on several aspects. It can be restricted by the topology of the model or limited by the available computer architecture. When there are no restrictions, it is a tradeoff between the reduction of the computation time of each subsystem due to the reduced subsystem size and the cost of the synchronization and data transfer between the subsystems.

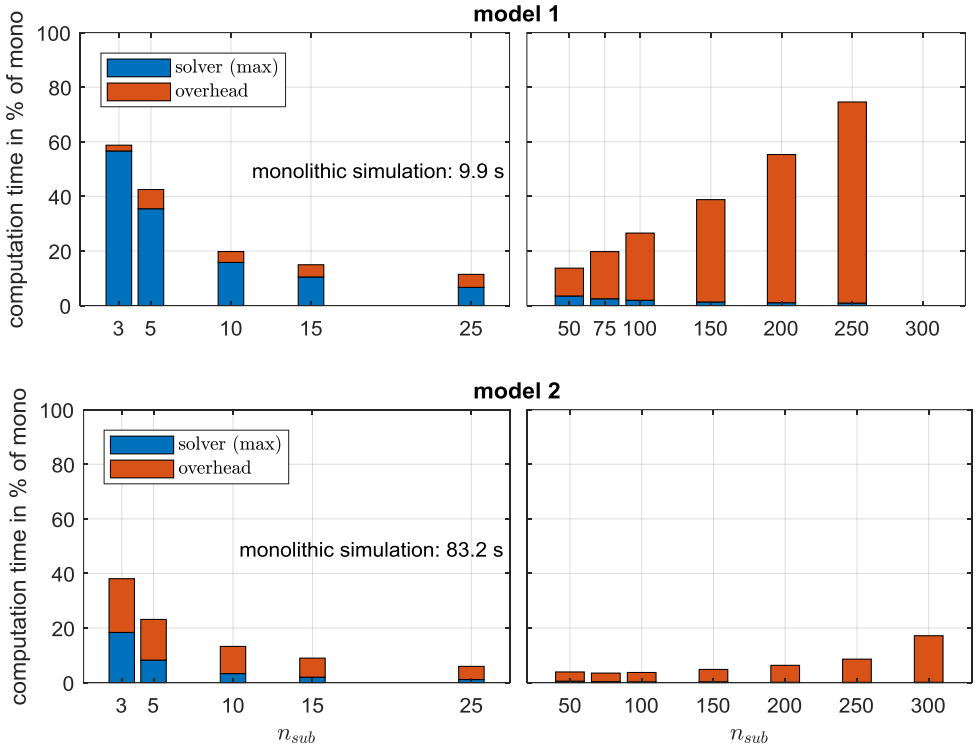

Fig. 12: Explicit co-simulation: computation time depending on the number of subsystems

Fig. 12 shows the computation time of explicit co-simulations ($n_{pol} = 2, h_{mac} = 1.0e\text{-}5, h_{mic} \leq h_{mac}$) with a various number of subsystems. The blue bars show the computation time of the subsystem solver and the red bars the overhead. As expected, the solver time per subsystem decreases as the number of subsystems increases.

The overhead that appears in co-simulations of model 2 with a small number of subsystems results from different subsystem integration times within each macro-step (cf. Fig. 7b). For model 1, the subsystem solver usually makes one micro-step per macro-step (cf. Fig. 7c). Therefore, the integration times of all subsystems are almost equal in each macro-step. The number of micro-steps for model 1 is – mainly because of the micro-step size limitation – almost doubled compared to the monolithic simulation. Hence, a co-simulation with three subsystems takes about 60% of the computation time of the monolithic simulation instead of the 33% that would be expected.

When the number of subsystems is increased over a certain level, the overhead caused by network traffic and synchronization becomes the dominating factor of the overall computation time.

4.2.3 Influence of the Macro-Step Size on the Computation Time

The following results are obtained by co-simulations with 47 subsystems. Fig. 13 shows the computation time as a function of the macro-step size of explicit co-simulations of both test models with quadratic approximation polynomials for the coupling variables. The qualitative observation is the same for both models. When the limitation $h_{mic} \leq h_{mac}$ is applied, the computation time decreases as the macro-step size is decreased until it

reaches a minimum. This is the optimal macro-step size, at which the discontinuities in the coupling variables are small and the macro-step size is large enough to not interfere with the micro-step size. When the macro-step size is decreased further, the micro-step size will be restricted by the macro-step size resulting in an increased computation time.

The computation time (Fig. 13 dashed curves) of co-simulations with the restriction that the subsystem integration stops at each macro-time point exactly behaves different. In this case, the key point is that the step size controller of the subsystems solver is interrupted at each macro-time point. Therefore, the computation time increases as the macro-step size is decreased. The optimal macro-step size is when the subsystem solver takes one micro-step per macro-step. When the macro-step size is reduced beyond this point, the computation time increases again.

To clarify the influence of the discontinuities in the coupling variables that result from a large macro-step size, Fig. 14 shows the average number of Newton iterations of the subsystem solvers for the co-simulation of model 1. If linear approximation polynomials for the coupling variables are used, the effect of the discontinuities on the computation time is stronger. The average number of Newton iterations for an explicit co-simulation with a macro-step size of 2.5e-5 is more than 3 times larger than for a co-simulation with a macro-step size of 7.5e-6.

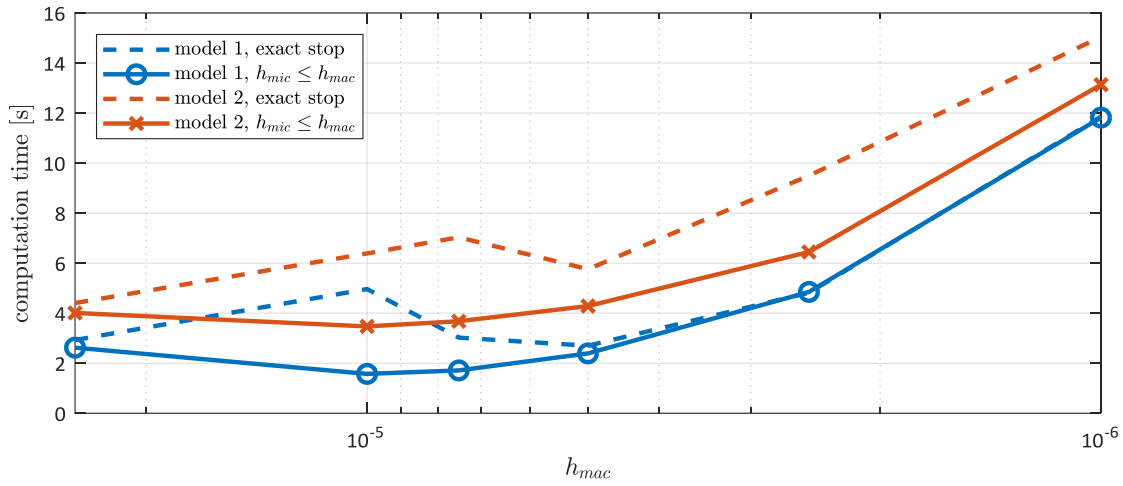

Fig. 13: Explicit co-simulation: computation time as a function of the macro-step size

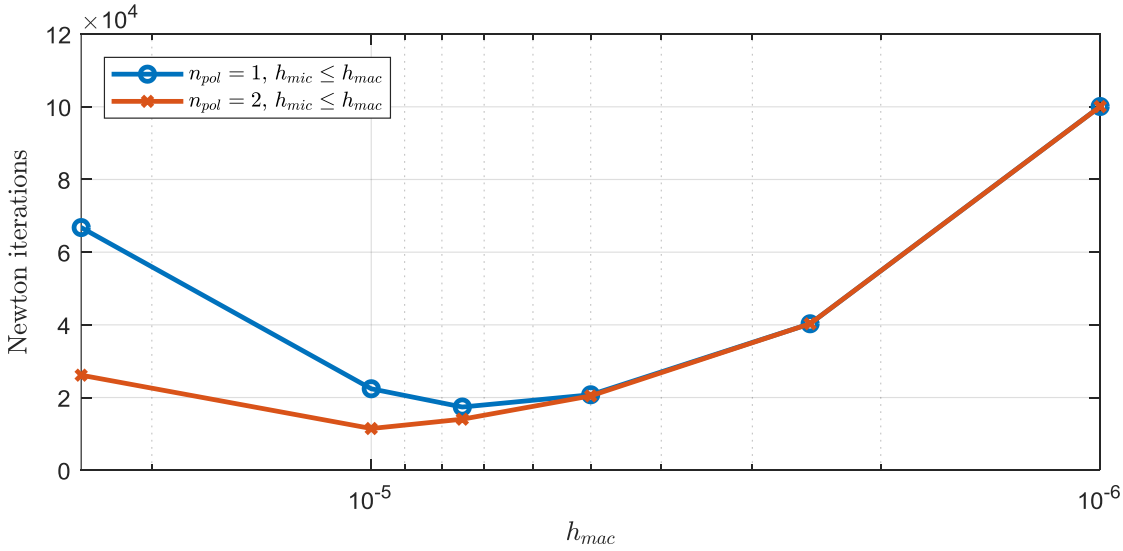

Fig. 14: Effect of the discontinuities in the coupling variables on the number of Newton iterations of the subsystem solver

4.2.4 Macro-Step Size Controller

Fig. 15 shows the number of macro-steps of an explicit co-simulation with quadratic approximation polynomials. The co-simulation with a constant macro-step size is carried out with $h_{mac} = 1.0e\text{-}5$. The error tolerances of the macro-step size controller are chosen in such a way that the global numerical error of both simulations is of the same order of magnitude. The usage of the macro-step size controller reduces the number of macro-steps for both models significantly. For model 2 the number of macro-steps is reduced by about 25% and for model 1 it reduced by more than 50%. Especially for co-simulations where the dominating factor of the computation time is the data transfer between the subsystems, the macro-step size controller may reduce the computation time significantly.

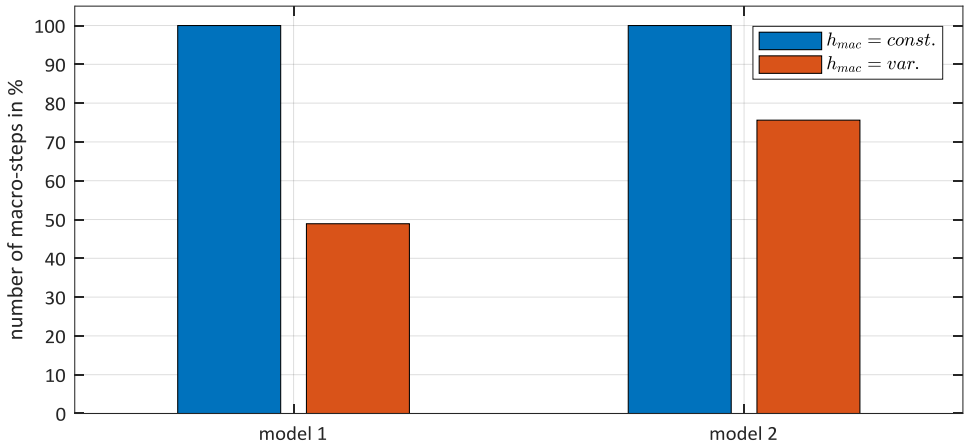

Fig. 15: Explicit co-simulation with macro-step size controller: number of macro-steps

5 Conclusions

The analysis and the reduction of the computation time using a parallel implementation of a Jacobi-type co-simulation is a challenging task because it is influenced by many factors. Besides the computational aspects like data transfer and thread synchronization in parallel computing, also the effect of the particular co-simulation method on the subsystem solvers plays an important role.

In this manuscript, two co-simulation methods, namely an explicit and a semi-implicit method have been utilized to parallelize the computation of a multibody system. The convergence behavior has been analyzed in detail. Although both methods have the same convergence order, the errors of the semi-implicit co-simulation approach are significantly smaller than for the explicit approach, especially for larger macro-step sizes. The semi-implicit method shows also a better numerical stability. The main drawback of the semi-implicit co-simulation method is that each macro-step has to be repeated and in addition, partial derivatives with respected to the coupling variables have to be computed. For the explicit co-simulation method, a macro-step size controller based on an error estimator has been implemented. It has been shown that the number of macro-steps can be reduced significantly without increasing the numerical error with the help of a macro-step size controller.

In addition, a detailed computation time analysis of the co-simulation methods has been carried out. The effect of different parameters on the overall computation time has been studied. It was pointed out, that the choice of an appropriate macro-step size is a delicate matter because it affects all parts of the overall computation time. Depending on the particular model, the macro-step size may be chosen rather large to minimize the computational effort for data traffic and thread synchronization. For other models, for example our test model, it may be advantageous to choose a smaller macro-step size in order to reduce the discontinuities in the coupling variables and therefore reduce the subsystem computation time. Also, the macro-step size should be chosen large enough so that it does not limit the micro-step size. Assuming that each subsystem is attached to a fixed number of cores, the hardware utilization is affected by the macro-step size, too. A suitable restriction for the micro-step size is

$h_{mic} \leq h_{mac}$. The alternative, namely to stop the subsystem solver at each macro-time point exactly, is not practicable because of its negative effect on the subsystem computation time. Another important point is the choice of the number of subsystems. A proper number of subsystems is always a compromise between reducing the subsystem integration time on the one hand, and increasing overhead due to data traffic and thread synchronization on the other hand.

It has been shown, that the computation time of nonlinear multibody systems can be reduced significantly without increasing the numerical error by applying a parallel co-simulation.

References

[1] Ambrosio, J.; Pombo, J.; Pereira, M.; Antunes, P.; Mosca, A.: "A computational procedure for the dynamic analysis of the catenary-pantograph interaction in high-speed trains", *Journal of Theoretical and Applied Mechanics*, 50/3, pp. 681-699, Warsaw, 2012.

[2] M. Naya, J. Cuadrado, D. Dopico, U. Lugris. An Efficient Unified Method for the Combined Simulation of Multibody and Hydraulic Dynamics: Comparison with Simplified and Co-Integration Approaches. *Archive of Mechanical Engineering*, Vol. LVIII, pp. 223–243, 2011.

[3] Kübler, R.; Schiehlen, W.: "Two methods of simulator coupling", *Mathematical and Computer Modelling of Dynamical Systems*, Vol. 6, pp. 93-113, 2000.

[4] Arnold, M.: "Stability of sequential modular time integration methods for coupled multibody system models", *Journal of Computational and Nonlinear Dynamics*, Vol. 5, pp. 1-9, 2010.

[5] Bernhard Schweizer, Pu Li, and Daixing Lu. "Explicit and Implicit Cosimulation Methods: Stability and Convergence Analysis for Different Solver Coupling Approaches." *Journal of Computational and Nonlinear Dynamics* 10.5 (2015): 051007.

[6] Meyer, T.; Kraft, J.; Li, P.; Lu, D.; Schweizer, B.: "Error estimation approach for controlling the macro step-size for explicit co-simulation methods", *Proceedings of the 7th GACM Colloquium on Computational Mechanics for Young Scientists from Academia and Industry*, October 11-13, Stuttgart, Germany, 2017.

[7] Hindmarsh, Alan C and Brown, Peter N and Grant, Keith E and Lee, Steven L and Serban, Radu and Shumaker, Dan E and Woodward, Carol S, and A. Collier: "SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers", *ACM Transactions on Mathematical Software (TOMS),* Vol. 31, No. 3, 2005.

[8] Davis, Timothy A., and Ekanathan Palamadai Natarajan. "Algorithm 907: KLU, a direct sparse solver for circuit simulation problems." *ACM Transactions on Mathematical Software (TOMS)* 37.3 (2010): 36.

[9] Featherstone, Roy. "The calculation of robot dynamics using articulated-body inertias." *The International Journal of Robotics Research* 2.1 (1983): 13-30.