Co-simulation with variable approximation order: order control algorithm for solver coupling approaches

J. Kraft · B. Schweizer

Received: 24 September 2021 / Accepted: 14 February 2022 / Published online: 30 May 2022
© The Author(s) 2022

Abstract
Considering co-simulation and solver coupling approaches, the coupling variables have to be approximated within a macro-time step (communication-time step), e.g., by using extrapolation/interpolation polynomials. Usually, the approximation order is assumed to be fixed. The efficiency and accuracy of a co-simulation may, however, be increased by using a variable approximation order. Therefore, a technique to control the integration order is required. Here, an order control algorithm for co-simulation and solver coupling methods is presented. The order controller is incorporated into the control algorithm for the macro-step size so that co-simulations with variable integration order and variable macro-step size can be carried out. Different numerical examples are presented, which illustrate the applicability and benefit of the proposed order control strategy. This contribution mainly focuses on mechanical systems. The presented techniques may, however, also be applied to nonmechanical dynamical systems.

Keywords Co-simulation · Solver coupling · Order control algorithm · Macro-step size controller

1 Introduction
Co-simulation terms a variety of numerical techniques, which can be applied to couple different solvers in time domain. A classical field of application for co-simulation methods concerns the simulation of multiphysical systems, see, e.g., [13, 16, 20, 49, 69]. In this case, each physical subdomain is modeled and simulated with its own (specialized) subsystem solver, whereas each subsystem may be considered as a black-box system. The subsystem solvers are numerically coupled by an appropriate co-simulation method. In literature, many interesting multidisciplinary problems have been solved incorporating co-simulation techniques, e.g., fluid/structure interaction problems [1, 43], analysis of coupled finite-element/multibody models [2, 3, 48], multidisciplinary simulations in the framework of vehicle dynamics [11, 36, 45] or coupled problems including particle models [14, 33, 70, 76].

Besides this classical area of application, co-simulation may also be applied advantageously to parallelize monodisciplinary dynamical systems, see, for instance, [4, 10, 31, 32, 39, 75].

✉ B. Schweizer

1 Department of Mechanical Engineering, Institute of Applied Dynamics, Technical University of Darmstadt, Otto-Berndt-Straße 2, 64287 Darmstadt, Germany
Therefore, the global system is split into a certain number of subsystems, which are connected and simulated by a proper co-simulation approach. As a consequence, the overall computation time may be reduced massively.

A very straightforward consideration can be useful to demonstrate the possible computational benefit, which may be achieved by a co-simulation-based parallelization approach. We consider therefore a multibody system with \( n \) degrees of freedom, which is decomposed into \( r \) equally sized subsystems. The decomposed system is assumed to be integrated with an implicit predictor/corrector co-simulation method, see Sect. 3. Note that for implicit co-simulation approaches, an interface-Jacobian is required, which may, however, be calculated very easily in parallel with the predictor/corrector subsystem integrations so that the computation of the interface-Jacobian will not (markedly) increase the overall co-simulation time. Regarding an implicit co-simulation approach, the main computational effort is produced by the repetition of the macro-step for carrying out the corrector iteration over the macro-interval. Now, we assume that an \( O(n^\alpha) \)-algorithm is used for solving the equations of motion, where \( \alpha \) is usually a value between 1 and 3. Consequently, the simulation time for the monolithic model is given by

\[
T_{\text{Mono}} = \text{constant} \cdot n^\alpha.
\]

Applying an implicit co-simulation scheme and assuming an average number of \( l \) corrector iterations within each macro-step, the simulation time of the co-simulation may roughly be estimated by

\[
T_{\text{Cosim}} = (1 + l) \cdot \text{constant} \cdot \left( \frac{1}{r} \right)^\alpha + \text{Overhead} \quad \text{(one predictor-step and } l \text{ corrector-steps)}.
\]

Practical simulations show (see [40]) that the number of corrector steps \( l \) is frequently rather small (\( l = 2 \) or \( l = 3 \)). Hence, if the overall model is decomposed into a sufficiently large number of subsystems, the reduction of computation time may become very significant. It should be noted that the estimation formula for \( T_{\text{Cosim}} \) will only yield reasonable results if the (average) integration step-size of the monolithic model equals the (average) integration step-size of the subsystems, see Sect. 6.6. The possible speed-up compared to a monolithic simulation may be improved even more, if the co-simulation is accomplished with a variable macro-step size (macro-step size controller). A further improvement may be achieved if a variable integration order is used (i.e., variable approximation order for the coupling variables). Applying an explicit co-simulation method, the possible speed-up may become even larger, since \( l = 0 \) for explicit co-simulations. It should be pointed out that for the case that \( r \) is chosen too large, the overhead will become the dominant part of the computation time so that the co-simulation will get inefficient [29, 40].

Besides the directly obvious speed-up due to the decomposition into subsystems, which can be solved independently and in parallel within each macro-step, an improvement with respect to the simulation time may also be achieved due to the multi-rate effect. If the global model can be split into a small-sized stiff subsystem and a large-scaled non-stiff subsystem, only the smaller stiff subsystem has to be integrated with small subsystem integration-step sizes, while the larger non-stiff subsystem can be integrated very efficiently and stable with larger subsystem integration-step sizes.

**Remark** Parallelization of a dynamical system can be realized in different ways. Considering a monolithic dynamical model, which is discretized by an (implicit) time-integration scheme (e.g., BDF method, Runge–Kutta method), an algebraic system of equations has to be solved in each time step. Usually, parallelization is accomplished by solving the algebraic system of equations in parallel, i.e., the equation solver is parallelized. In contrast to this classical type of parallelization, a further parallelization can be achieved by making use of a co-simulation approach, as mentioned above. Hence, parallelization may be realized on two levels, namely

i) by a decomposition of the overall model into subsystems, which are integrated in parallel within each macro-time step and
by applying the classical type of parallelization, i.e., by using parallelized subsystem solvers for solving the algebraic subsystem equations.

Here, we omit a detailed literature review on co-simulation and solver coupling techniques. The interested reader is referred to literature, see, e.g., [17, 18, 22, 56].

Within a co-simulation approach, a macro-time grid (also called communication-time grid) has to be defined, i.e., macro-time points $T_i$ ($i = 0, 1, 2, \ldots$) have to be specified, where the subsystems exchange the coupling variables. Between the macro-time points, the coupling variables are approximated, e.g., by means of extrapolation/interpolation polynomials of degree $k$. In the simplest case, an equidistant macro-time grid is used and the order of the approximation polynomials for the coupling variables is assumed to be constant. Using a co-simulation with constant order and constant macro-step size may, however, be rather inefficient. It has been shown in literature that the efficiency, accuracy, and stability of a co-simulation implementation may significantly be improved by incorporating a macro-step size controller, see [6, 40, 51, 53, 54].

In the current manuscript, the possible improvement of a co-simulation with variable integration order is investigated. For realizing a variable integration order, a strategy is required, which determines the optimal integration order. Therefore, the local truncation error within the macro-step from $T_N$ to $T_{N+1} = T_N + H_N$ ($H_N$ denotes the current macro-step size) is estimated for different polynomial degrees $k$. The optimal polynomial order minimizes the estimated local error. Applying an appropriate order control algorithm, co-simulations with variable integration order can be realized. In this work, only co-simulation implementations with variable macro-step size are considered. It should therefore be stressed that the control algorithm for the integration order and the macro-step size control algorithm cannot be considered independently, since both influence each other. As a consequence, order and step-size have to be controlled simultaneously.

In the paper at hand, mechanical co-simulation models with an arbitrary number of subsystems are investigated. The developed order control algorithm may, however, also be used for nonmechanical systems. Considering mechanical systems, the coupling of the subsystems can be realized either by constitutive laws (coupling by applied-forces) [1, 3, 9, 19, 21, 48, 58, 60] or by algebraic constraint equations (coupling by reaction forces) [5, 23, 30, 39, 55, 57, 59, 61, 62, 72, 73, 75]. In this work, we restrict ourselves to the case of applied-force coupling approaches (explicit and implicit). Furthermore, we only consider parallel co-simulation implementations of Jacobi type. Sequential integration techniques (Gauss–Seidel type) [26, 27] are not applied, since we focus on fully parallelizable models. To approximate the coupling variables between two macro-time points, Lagrange polynomials are used here. Other approximation approaches – acceleration based techniques [34, 35], interface-model-based approaches [47], context-based approximation polynomials [7], methods applying a least square fit [42], etc. – are not considered in the current manuscript.

The new contributions of this manuscript can be summarized as follows:

- An algorithm for controlling the order of the approximation polynomials for co-simulation and solving coupling methods is presented.
- The developed order controller is imbedded into the macro-step size control algorithm.
- Efficiency, accuracy, and stability of a co-simulation implementation with variable integration order and variable macro-step size is analyzed with different numerical examples.
- The large potential of parallelizing dynamical models with the help of co-simulation is demonstrated.
The manuscript is organized as follows: In Sect. 2, the basic idea of co-simulation and the relevant definitions are briefly summarized. The classical explicit and implicit co-simulation methods are shortly recapitulated in Sect. 3. The different error estimators, which are used in the current work for controlling the macro-step size, are collected in Sect. 4. The order control strategy and the combined algorithm for controlling the integration order and the macro-step size simultaneously are presented in Sect. 5. Numerical examples are given in Sect. 6. The paper is concluded in Sect. 7. Calculation of the interface-Jacobian, which is required for the implicit co-simulation approach, is treated in Appendix A. Some additional simulation results for Example 3 in Sect. 6.3 are collected in Appendix B.

2 Co-simulation model

2.1 General case: arbitrary mechanical system decomposed into subsystems

To demonstrate the basic strategy behind a co-simulation approach, a general mechanical system is considered, which is mathematically defined by the following first-order DAE system (differential-algebraic system of equations)

\[ B(t, z) \dot{z} = f(t, z). \]  

The vector \( z \) contains the position variables \( q \), the velocity variables \( v \), and also the Lagrange multipliers \( \lambda \); \( B \) represents a symmetric square matrix, which is not invertible in the general case, where Eq. (1) represents a DAE. For classical constrained multibody systems [12], for instance, the matrix \( B \) is given by

\[
\begin{pmatrix}
I & 0 & 0 \\
0 & M & 0 \\
0 & 0 & 0
\end{pmatrix},
\]

where \( I \) denotes the identity matrix and \( M \) the mass matrix. In the ODE case, \( B \) is invertible.

Next, the overall system is decomposed into \( r \) subsystems. To describe the coupling between the subsystems, appropriate coupling variables \( u \) have to be defined. Throughout this manuscript, we assume that the subsystems are connected by applied forces/torques (applied-force coupling) so that the coupling between the subsystems is described by constitutive laws. Constraint coupling, where the subsystems are connected by algebraic constraint equations and corresponding reaction forces (Lagrange multipliers) is not considered here. In the framework of a co-simulation approach, macro-time points \( T_i \) (\( i = 0, 1, 2, \ldots \)) have to be defined. Macro-time points are discrete, not necessarily equidistant time points. With the macro-time points, the macro-step size \( H_N = T_{N+1} - T_N \) is defined. If a constant macro-step size \( H = T_{N+1} - T_N = \text{const.} \) is used, \( H \) has to be specified by the user so that the co-simulation runs stable and provides accurate results. Applying an appropriate error estimator, the current macro-step size \( H_N \) is determined by the macro-step size controller so that a user-defined error criterion is fulfilled. Between the macro-time points, the subsystems are integrated independently, since the subsystems only exchange information at the macro-time points. To carry out the subsystem integrations between the macro-time points, the coupling variables have to be approximated. In this work, Lagrange polynomials are used for approximating the coupling variables.

The states and multipliers of an arbitrary subsystem \( s \in \{1, \ldots, r\} \) are arranged in the vector \( z^s = (q^T, v^T, \lambda^T)^T \). Note that the number of the subsystem is indicated by the superscript \( s \). The equations of motion of subsystem \( s \) read

\[ B^s(t, z^s) \dot{z}^s = f^s(t, z). \]
The coupling variables $u$, which here represent input variables of the subsystems, can be defined implicitly by the coupling functions $g_c := u - \varphi(t, z, u) = 0$. It should be mentioned that the subsystem coupling is, one the one hand, defined by the constitutive laws of the coupling forces/torques and, on the other hand, by the decomposition technique. Note that basically three different decomposition techniques may be distinguished, namely force/force-decomposition, force/displacement-decomposition, and displacement/displacement-decomposition, see [58, 61, 75] for more details. It should further be stressed that for applied-force coupling approaches, the coupling variables are only functions of $q$ and $v$, i.e., the coupling functions may be expressed as $g_c := u - \varphi(t, q, v, u) = 0$. Making use of the coupling variables, the equations of motion of subsystem $s$ can be rewritten as

$$B^s(t, z^s)\dot{z}^s = f^{co,s}(t, z^s, u).$$

Collecting the right-hand sides of all subsystem equations into the vector $f^{co}(t, z, u) = (f^{co,1}(t, z^1, u)^T, \ldots, f^{co,r}(t, z^r, u)^T)^T$, the decomposed co-simulation system can be formulated as

$$B(t, z)\dot{z} = f^{co}(t, z, u),$$

$$g_c(t, z, u) = 0$$

with $z = (z^1, \ldots, z^r)^T$. Note that in the framework of a co-simulation approach, the coupling conditions are only enforced at the macro-time points $T_N$. Between the macro-time points, approximated coupling variables are used. Further details and examples concerning the decomposition of an overall system into subsystems can, for instance, be found in [29, 40]. It should be emphasized again that co-simulation and the here investigated strategies for controlling the macro-step size and approximation order are not limited to mechanical systems.

### 2.2 Special case: system of coupled multibody models

In the subsequent sections, we focus on the special case that the subsystems are general constrained multibody systems. Then, the equations of motion (3) for an arbitrary subsystem $s$ can be written as

$$\dot{x}^s = H^s(x^s)v^s,$$

$$M^s(t, x^s)\dot{v}^s = f^s(t, x^s, v^s) - H^s(x^s)G^s(t, x^s)\lambda^s + f^{c,s}(t, x^s, v^s, u),$$

$$0 = g^s(t, x^s).$$

The matrix $H^s$ defines the relationship between the (generalized) velocity variables $v^s$ and the (generalized) position variables $x^s$; $M^s$ denotes the symmetric mass matrix, and $f^s$ the vector of the applied and Coriolis forces. The algebraic constraint equations of the subsystem are collected in the vector $g^s(t, x^s) = 0$. The term $H^s(x^s)G^s(t, x^s)\lambda^s$ with the constraint Jacobian $G^s = \frac{\partial g^s}{\partial x}$ and the vector of Lagrange multipliers $\lambda^s$ represents the constraint forces/torques generated by the subsystem constraint equations. The coupling forces/torques of subsystem $s$ are arranged in the vector $f^{c,s}$. A comparison with the general case of Sect. 2.1 shows that one gets for the special case of a system of coupled multibody models the relationships $B^s(t, z) = \text{blockdiag}(I, M^s, 0)$ and $f^{co,s} = \begin{pmatrix} H^s v^s \\ f^s - H^s G^s \lambda^s + f^{c,s} \end{pmatrix}$. 

Springer
2.2.1 Example: two multibody systems coupled by a flexible bushing element

As an example, two arbitrary unconstrained mechanical subsystems are considered. It is assumed that the subsystems are mechanically coupled by a linear viscoelastic bushing element (flexible at point joint) [58], see Fig. 1. More precisely, point $C_i$ of body $i$ (center of mass $S_i$, mass $m_i$, inertia tensor $J_i$) belonging to subsystem 1 is connected to marker $C_j$ of body $j$ (center of mass $S_j$, mass $m_j$, inertia tensor $J_j$) related to subsystem 2 by a linear three-dimensional spring/damper element.

The coupling points $C_i, C_j$ are represented by the body fixed vectors $r_{C_i}, r_{C_j}$. Applying absolute coordinates, position and orientation of an arbitrary rigid body is defined by the 3 coordinates of the center of mass $r_{S_q}$ and by 3 rotation parameters arranged in the vector $\gamma_q \in \mathbb{R}^3$ (e.g., 3 Bryant angles). The coordinates of the angular velocity $\omega_q$ are associated with the time derivative of the rotation parameters by the general expression $\dot{\gamma}_q = H(\gamma_q)\omega_q$, with $H \in \mathbb{R}^{3 \times 3}$. The equations of motion (Newton–Euler equations) for the coupling bodies $i$ and $j$ are given by

**Coupling body $i$ (subsystem 1):**

\[
\begin{align*}
\dot{r}_S^i &= v_S^i, \\
m_i \dot{v}_S^i &= f_i(t, x^1, v^1) + f_{c,i}^1, \\
\dot{\gamma}_i &= H(\gamma_i)\omega_i, \\
J_i \dot{\omega}_i + \omega_i \times J_i \omega_i &= \tau_i(t, x^1, v^1) + \tau_{c,i}^1.
\end{align*}
\] (6)

**Coupling body $j$ (subsystem 2):**

\[
\begin{align*}
\dot{r}_S^j &= v_S^j, \\
m_j \dot{v}_S^j &= f_j(t, x^2, v^2) + f_{c,2}^j, \\
\dot{\gamma}_j &= H(\gamma_j)\omega_j, \\
J_j \dot{\omega}_j + \omega_j \times J_j \omega_j &= \tau_j(t, x^2, v^2) + \tau_{c,2}^j.
\end{align*}
\] (7)
In the above equations, $f_i$ and $f_j$ represent the resultant applied forces, which are acting on the two coupling bodies. The resultant applied torques are denoted by $\tau_i$ and $\tau_j$.

The coupling force $f^c$ represents a spring/damper force, which is proportional to the relative displacement and to the relative velocity of the coupling points $C_i$ and $C_j$; $f^c$ can implicitly be described by the three coupling conditions

$$g_c := f^c + C \cdot [(r_{Si} + r_{Ci}) - (r_{Sj} + r_{Cj})] + D \cdot \frac{d}{dt} [(r_{Si} + r_{Ci}) - (r_{Sj} + r_{Cj})] = 0,$$

where the diagonal matrix $C = \text{diag}(c_x, c_y, c_z)$ collects the three spring constants and the diagonal matrix $D = \text{diag}(d_x, d_y, d_z)$ the three damping coefficients. Note that the total time derivative with respect to the inertial system $K_0$ is denoted by $\frac{d}{dt}$. Hence, the coupling forces and torques, which are acting on the two coupling bodies, read

$$f^c_{i,1} = f^c, \quad \tau^c_{i,1} = r_{Ci} \times f^c,$n

$$f^c_{i,2} = -f^c, \quad \tau^c_{i,2} = -r_{Ci} \times f^c.$$

### 3 Classical explicit and implicit co-simulation schemes

For approximating the coupling variables $u$ within the macro-step $T_N \rightarrow T_{N+1}$, extrapolation/interpolation polynomials $p_{N+1}(t)$ of degree $k$ (local approximation order $k+1$, i.e., $O(\varepsilon^{k+1})$) are used. Since the subsystems only exchange information at the macro-time points, the coupling conditions are only enforced at the macro-time points $T_N$, i.e., $g_c(t, z, u) = 0$ is only fulfilled at the macro-time points. Between the macro-time points, the subsystems can be integrated independently.

Applying a co-simulation approach, system (1) is replaced by the weakly coupled co-simulation system

$$B(t, z) \dot{z} = f^c(t, z, p_{N+1}(t)).$$

The new variables $z_{N+1}$ are obtained by integrating Eq. (10) from $T_N$ to $T_{N+1}$ with the initial conditions $z_N$.

It should be noted that for an explicit co-simulation method, the extrapolated coupling variables $p_{N+1}(T_{N+1})$ will usually not satisfy the coupling equations at the new macro-time point $T_{N+1}$ so that $p_{N+1}(T_{N+1}) - \varphi(T_{N+1}, z_{N+1}, p_{N+1}(T_{N+1})) \neq 0$. Consistent coupling variables $u_{N+1}$ can, however, simply be calculated by an update step. Therefore, the coupling equations $u_{N+1} = \varphi(T_{N+1}, z_{N+1}, u_{N+1})$ have just to be solved for $u_{N+1}$. The update step will produce (small) jumps in the coupling variables, because $p_{N+1}(T_{N+1})$ is usually not equivalent with the updated coupling variables $u_{N+1}$.

### 3.1 Explicit co-simulation scheme

The classical explicit co-simulation approach can be described very easily by considering the general macro-time step from $T_N \rightarrow T_{N+1}$ [52, 60, 64, 69]. We assume that at the beginning of the macro-time step, the variables of the subsystems (subsystem states and subsystem
Fig. 2 Explicit co-simulation scheme \((k = 1)\): (a) approximation of coupling variable \(u_i\) with linear extrapolation polynomial; (b) update step

multipliers) as well as the coupling variables are known.

- In a first step, the vector collecting the extrapolation polynomials \(p_{NL+1}^{\text{ext}}(t)\) for the coupling variables \(u(t)\) in the time interval \([T_N, T_{N+1}]\) is generated by making use of the \(k + 1\) sampling points \((T_N, u_N), (T_{N-1}, u_{N-1}), \ldots, (T_{N-k}, u_{N-k})\).
- In a second step, Eq. \(\text{(10)}\) is integrated from \(T_N \rightarrow T_{N+1}\) using the extrapolation polynomials \(p_{N+1}^{\text{ext}}(t)\). This yields the variables \(z_{N+1}^{\text{ext}}\). In case of the explicit co-simulation scheme, the new variables at \(T_{N+1}\) are simply given by \(z_{N+1} := z_{N+1}^{\text{ext}}\).
- In a final step, updated coupling variables \(u_{N+1}\) are computed. Therefore, the variables \(z_{N+1}^{\text{ext}}\) are inserted into the coupling equations and \(u_{N+1} = \varphi(T_{N+1}, z_{N+1}, u_{N+1})\) is solved for \(u_{N+1}\).

Now, the next macro-time step \(T_{N+1} \rightarrow T_{N+2}\) can be accomplished. The integration scheme is sketched in Fig. 2.

Note that for practical reasons, problems may occur in conjunction with feed-through systems. From the practical point of view, it may be complicated to perform the update step, e.g., if commercial simulation tools are used with reduced solver access. In this case, \(u_{N+1} = \varphi(T_{N+1}, z_{N+1}, u_{N+1})\) is just replaced by \(u_{N+1} = \varphi(T_{N+1}, z_{N+1}, p_{N+1}(T_{N+1}))\). As a result, problems with the error estimator may arise (see [40] for more details). In the following, we therefore assume that updated coupling variables \(u_{N+1}\) are calculated.

### 3.2 Implicit co-simulation scheme

The classical implicit co-simulation scheme is based on a predictor/corrector approach, see, e.g., [58, 60]. Again, the general macro-time step from \(T_N\) to \(T_{N+1}\) is considered to explain the approach. The coupling variables \(u(t)\) are approximated by Lagrange polynomials of degree \(k\). At the beginning of the macro-time step, the subsystem variables and the coupling variables are assumed to be known.

**Predictor step:**

- Firstly, predictor polynomials \(p_{N+1}^{\text{pre}}(t)\) for the coupling variables \(u(t)\) in the time interval \([T_N, T_{N+1}]\) are generated by making use of the \(k + 1\) sampling points \((T_N, u_N), (T_{N-1}, u_{N-1}), \ldots, (T_{N-k}, u_{N-k})\).
- Secondly, Eq. \(\text{(10)}\) is integrated from \(T_N \rightarrow T_{N+1}\) with \(p_{N+1}^{\text{pre}}(t)\). This yields the variables \(z_{N+1}^{\text{pre}}\). It should be mentioned that the predictor step is identical with the explicit co-simulation step of Sect. 3.1 (except for the update step).
Corrector iteration:

- Let \( u_{N+1}^* \) denote arbitrary (initially unknown) coupling variables at the macro-time point \( T_{N+1} \). Using \( u_{N+1}^* \), interpolation polynomials \( p_{N+1}^* (t) \) for the coupling variables \( u(t) \) in the time interval \( [T_N, T_{N+1}] \) can formally be generated with the help of the \( k+1 \) sampling points \( (T_{N+1}, u_{N+1}^*), (T_N, u_N) \), \( \ldots \), \( (T_{N-k+1}, u_{N-k+1}) \). Note that \( u_N, \ldots, u_{N-k-1} \) term the coupling variables at the previous macro-time points, which are assumed to be known.

- A formal integration of the subsystems with the polynomials \( p_{N+1}^* (t) \) will yield a relationship of the form \( z_{N+1}^* (u_{N+1}^*) \).

- By inserting \( z_{N+1}^* (u_{N+1}^*) \) into the coupling equations \( g_c \), one formally gets the relationship \( g_c \left( T_{N+1}, z_{N+1}^* (u_{N+1}^*), u_{N+1}^* \right) \). Considering arbitrary coupling variables \( u_{N+1}^* \), the coupling equations are not fulfilled, i.e., \( g_c \left( T_{N+1}, z_{N+1}^* (u_{N+1}^*), u_{N+1}^* \right) \neq 0 \).

  To compute corrected coupling variables \( u_{N+1} \), which fulfill the coupling equations \( g_c \left( T_{N+1}, z_{N+1} (u_{N+1}), u_{N+1} \right) = 0 \), a Newton iteration has to be carried out. Therefore, the relationship \( g_c \left( T_{N+1}, z_{N+1}, u_{N+1} \right) = 0 \) is linearized by a Taylor series expansion.

- In the first corrector iteration step, the predictor point \( (T_{N+1}, z_{N+1}^* (u_{N+1}^*), u_{N+1}^*) \) is used as expansion point for the Taylor series expansion. Note that the predictor point may also be denoted as zeroth corrector point, i.e., \( (T_{N+1}, z_{N+1}^0 (u_{N+1}^0), u_{N+1}^0) \). The Taylor series expansion yields the linearized coupling equations

\[
\begin{align*}
g_c^{\text{linear}} \left( T_{N+1}, z_{N+1} (u_{N+1}), u_{N+1} \right) & := g_c \left( T_{N+1}, z_{N+1}^* (u_{N+1})^*, u_{N+1}^* \right) + \frac{d g_c}{d u_{N+1}^*} \bigg|_{u_{N+1}^*} \cdot \left( u_{N+1} - u_{N+1}^* \right) = 0. \tag{11}
\end{align*}
\]

By solving the linear system (11) for \( u_{N+1} \), corrected coupling variables \( u_{N+1}^{\text{cor}, 1} \) are obtained.

  To compute corrected subsystem variables \( z_{N+1}^{\text{cor}, 1} \), interpolation polynomials \( p_{N+1}^{\text{cor}, 1} (t) \) for the coupling variables \( u(t) \) are generated by using the \( k+1 \) sampling points \( (T_{N+1}, u_{N+1}^{\text{cor}, 1}), (T_N, u_N), \ldots, (T_{N-k+1}, u_{N-k+1}) \). Now, the subsystem integration from \( T_N \to T_{N+1} \) is repeated with the polynomials \( p_{N+1}^{\text{cor}, 1} (t) \), which yields the corrected subsystem variables \( z_{N+1}^{\text{cor}, 1} \).

- Considering the \( \alpha \)th corrector step, the \((\alpha-1)\)th corrector point \( (T_{N+1}, z_{N+1}^{\text{cor}, \alpha-1}, u_{N+1}^{\text{cor}, \alpha-1}) \) is used as expansion point for the Taylor series expansion and the above procedure is repeated.

- The procedure is repeated \( l \) times, until a convergence criterion is fulfilled. The final variables \( u_{N+1} := u_{N+1}^{\text{cor}, l} \) are the corrected variables.

- By setting \( z_{N+1} := z_{N+1}^{\text{cor}, l} \), the next macro-step from \( T_{N+1} \to T_{N+2} \) is accomplished.

The implicit co-simulation scheme is illustrated in Fig. 3.

Some remarks on the different possibilities to calculate/approximate the interface-Jacobian \( \frac{d g_c}{d u_{N+1}^*} \) can be found in Appendix A. In the following sections, basically two methods are used and compared, namely the calculation of the interface-Jacobian by a finite difference approach (Appendix A.1) and the approximate computation of the interface-Jacobian based on a low-order expansion technique (Appendix A.2).
4 Error estimators for the classical explicit and implicit co-simulation schemes

The efficiency and accuracy of a co-simulation may significantly be improved by using a macro-step size controller so that a nonequidistant communication-time grid can be realized. Therefore, error estimators are required, which can be used to estimate the numerical error produced by the co-simulation approach, i.e., the error introduced by the polynomial approximation of the coupling variables. In [40], three different error estimators for the classical explicit co-simulation approach (called ExLocal, ExMilne, ExSLocal) and two error estimators for the classical implicit co-simulation approach (called ImMilne, ImSLocal) have been derived. The basic idea for the construction of an error estimator, which estimates the local error generated in the macro-step \( T_N \rightarrow T_{N+1} \), is to compare two different solutions at \( T_{N+1} \): the state variables \( q_{N+1}, v_{N+1} \) (or, alternatively, the coupling variables \( u_{N+1} \)) of the actual co-simulation generated with the classical explicit/implicit approach are compared with reference variables \( \hat{q}_{N+1}, \hat{v}_{N+1} \) (or alternatively \( u_{N+1}^{ext}, u_{N+1}^{pre} \)). The differences \( q_{N+1} - \hat{q}_{N+1}, v_{N+1} - \hat{v}_{N+1} \) (or, alternatively, \( u_{N+1} - u_{N+1}^{ext}, u_{N+1} - u_{N+1}^{pre} \) for the explicit and \( u_{N+1} - u_{N+1}^{pre} \) for the implicit co-simulation) can be used to construct an error estimator. In this section, the main formulas for the five error estimators are shortly recapitulated. For a detailed description, we refer to [40]. It should be pointed out that the error estimators used here are only valid for continuous co-simulation systems. Considering discontinuous systems, systems with switches, etc., a modified error estimation strategy has to be applied.

In this work, the classical explicit and implicit co-simulation schemes are considered, see Sect. 3. Using extrapolation/interpolation polynomials of degree \( k \) (i.e., polynomial order \( k+1 \)), the coupling variables are approximated with \( O(H^{k+1}) \). Since the coupling variables and therefore the accelerations are approximated with \( O(H^{k+1}) \), the velocities \( v_{N+1} \) will converge with \( O(H^{k+2}) \) and the position variables \( q_{N+1} \) with order \( O(H^{k+3}) \) for both the explicit and implicit scheme. Hence, the following estimates hold:

\[
q_{N+1} = q(T_{N+1}) + O(H^{k+3}),
\]
\[
v_{N+1} = v(T_{N+1}) + O(H^{k+2}),
\]

where \( q(T_{N+1}) \) and \( v(T_{N+1}) \) denote the analytical solutions with \( q(T_N) = q_N \) and \( v(T_N) = v_N \). As can be seen, the integration order of the co-simulation may directly be controlled by controlling the polynomial degree of the approximation polynomials (see Sect. 5.2). Note that the error bounds of Eq. (12) are only valid for the case that the errors resulting from the numerical subsystem integrations can be neglected, i.e., strictly speaking for the case that the subsystems are integrated analytically.
4.1 Error estimator 1 for explicit co-simulation (ExLocal)

The first error estimator for the classical explicit co-simulation is based on a local extrapolation technique \[65, 71\] and is denoted by ExLocal. In order to calculate an error estimator for \( q_{N+1} \) and \( v_{N+1} \), comparative solutions \( \hat{q}_{N+1} \) and \( \hat{v}_{N+1} \) are needed. If a local extrapolation technique is applied for deriving an error estimator, two solutions with different convergence orders are compared. For this purpose, two parallel co-simulations are carried out from \( T_N \) to \( T_{N+1} \), whereby the comparative solution is calculated with the same initial conditions (\( \hat{z}_N = z_N \)):

- The first simulation (actual co-simulation) is accomplished with the extrapolation polynomials \( P_{N+1}^{ext} (t) = P_{N+1}^k (t; (T_N, u_N), \ldots, (T_{N-k}, u_{N-k})) \) of degree \( k \) (approximation order \( k + 1 \)). This integration yields the variables \( q_{N+1} \) and \( v_{N+1} \).
- The second simulation (comparative solution) is carried out with the extrapolation polynomials \( P_{N+1}^{ext} (t) = P_{N+1}^{k+1} (t; (T_N, u_N), \ldots, (T_{N-k}, u_{N-k}), (T_{N-k-1}, u_{N-k-1})) \) of degree \( k + 1 \). The second integration provides the variables \( \hat{q}_{N+1} \) and \( \hat{v}_{N+1} \).

The two co-simulations are illustrated in Fig. 4. Within the local extrapolation technique, the local error of the co-simulation in the macro-step from \( T_N \to T_{N+1} \) can very easily be estimated by the difference of the two solutions. Considering the first simulation, the local errors of the position variable \( q_i \) and the velocity variable \( v_i \) are defined by

\[
\begin{align*}
\varepsilon^\text{pos}_{i,N+1} &= |q_i,N+1 - q_i(T_{N+1})|, \\
\varepsilon^\text{vel}_{i,N+1} &= |v_i,N+1 - v_i(T_{N+1})|.
\end{align*}
\]

(13)

Substituting the analytical solutions \( q_i(T_{N+1}) \) and \( v_i(T_{N+1}) \) by \( \hat{q}_{i,N+1} = q_i(T_{N+1}) + \mathcal{O}(H^{k+4}) \) and \( \hat{v}_{i,N+1} = v_i(T_{N+1}) + \mathcal{O}(H^{k+3}) \), one gets

\[
\begin{align*}
\varepsilon^\text{pos}_{i,N+1} &= |q_i,N+1 - (\hat{q}_{i,N+1} + \mathcal{O}(H^{k+4}))| \\
&= |q_i,N+1 - \hat{q}_{i,N+1}| + \mathcal{O}(H^{k+4}), \\
\varepsilon^\text{vel}_{i,N+1} &= |v_i,N+1 - (\hat{v}_{i,N+1} + \mathcal{O}(H^{k+3}))| \\
&= |v_i,N+1 - \hat{v}_{i,N+1}| + \mathcal{O}(H^{k+3}).
\end{align*}
\]

(14)

Therefore, it is possible to use the expressions

\[
\begin{align*}
\varepsilon^\text{pos}_{i,N+1} &= |q_i,N+1 - \hat{q}_{i,N+1}|, \\
\varepsilon^\text{vel}_{i,N+1} &= |v_i,N+1 - \hat{v}_{i,N+1}|
\end{align*}
\]

(15)

as error estimators for the actual co-simulation. The estimators converge to the local errors \( \varepsilon^\text{pos}_{i,N+1} \) and \( \varepsilon^\text{vel}_{i,N+1} \) with order \( \mathcal{O}(H^{k+4}) \) and \( \mathcal{O}(H^{k+3}) \), respectively \[40\].

With the error estimators \( \varepsilon^\text{pos}_{i,N+1} \) and \( \varepsilon^\text{vel}_{i,N+1} \) according to Eq. (15), it is possible to monitor the local error of all state variables. To simplify the implementation, it would be possible to only monitor the state variables of the coupling bodies.

4.2 Error estimator 2 for explicit co-simulation (ExMilne)

The second error estimator for the classical explicit co-simulation approach is indicated by ExMilne and based on a Milne device approach \[41\]. Within this approach, two different
The matrix $A$ and $C_k$ are calculated with polynomials of degree $TN$. Solutions with the same convergence order but different leading error terms are compared. Again, two parallel co-simulations from $T_N$ to $T_{N+1}$ are executed with the same initial conditions ($\hat{z}_N = z_N$):

- The first simulation (actual co-simulation) is carried out with the extrapolation polynomials $p_{N+1}^{ex}(t) = P_{N+1}^k(t; (T_N, u_N), \ldots, (T_{N-k}, u_{N-k}))$, which yields the variables $q_{N+1}$ and $v_{N+1}$.
- The second simulation (comparative solution) with the interpolation polynomials $\hat{p}_{N+1}^{int}(t) = P_{N+1}^k(t; (T_N, u_N), \ldots, (T_{N-k}, u_{N-k}))$ provides the variables $\hat{q}_{N+1}$ and $\hat{v}_{N+1}$, where $\hat{u}_{N+1} = \hat{p}_{N+1}^{int}(T_{N+1})$ represent extrapolated coupling variables generated with the extrapolation polynomials $\hat{p}_{N+1}^{ext}(t) = P_{N+1}^{k+1}(t; (T_N, u_N), \ldots, (T_{N-k}, u_{N-k}), (T_{N-k-1}, u_{N-k-1}))$ of degree $k + 1$.

The two co-simulations are illustrated in Fig. 5 for the case $k = 1$. Note that $q_{N+1}$ and $\hat{q}_{N+1}$ are calculated with polynomials of degree $k$ and locally converge with $O(H^{k+3})$. In [40] (Appendix A.2), it is shown that the local errors of $q_{N+1}$ and $\hat{q}_{N+1}$ may be written as

$$q_{N+1} - q(T_{N+1}) = H_N^2 C_{pos,N+1}^k A_N^{co} \Delta p_{N+1} + O(H^{k+4}),$$

$$\hat{q}_{N+1} - q(T_{N+1}) = H_N^2 (C_{pos,N+1}^k - C_{pos,N+1}^{k+1}) A_N^{co} \Delta p_{N+1} + O(H^{k+4}),$$

(16)

where $H_N = T_{N+1} - T_N$ represents the current macro-step size. The error constants $C_{pos,N+1}^k$ and $C_{pos,N+1}^{k+1}$ can be expressed as special integrals of the Lagrange basis polynomials $L^k(t)$, namely

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

(17)

The matrix $A_N^{co}$ represents a special Jacobian matrix, which does not explicitly depend on the macro-step size $H_N$. The vector $\Delta p_{N+1}$ is also not explicitly depending on $H_N$ and may be calculated by the difference of two polynomials. A direct calculation of the leading error term $H_N^2 C_{pos,N+1}^{k+1} A_N^{co} \Delta p_{N+1}$ in Eq. (16) is not possible. The leading error term may, however, be determined in a straightforward manner with the help of the two solutions $q_{N+1}$ and $\hat{q}_{N+1}$. Subtracting Eq. (16a) from Eq. (16b) and multiplying by $C_{pos,N+1}^{k+1}/C_{pos,N+1}^k$ gives

$$q_{N+1} - q(T_{N+1}) - (\hat{q}_{N+1} - q(T_{N+1})) = H_N^2 C_{pos,N+1}^k A_N^{co} \Delta p_{N+1}$$

$$\Rightarrow \frac{C_{pos,N+1}^k}{C_{pos,N+1}^{k+1}} (q_{N+1} - \hat{q}_{N+1}) = H_N^2 C_{pos,N+1}^k A_N^{co} \Delta p_{N+1} + O(H^{k+4}).$$  

(18)
Obviously, the local error $\varepsilon_{i,N+1}^{\text{pos}} = |q_i,N+1 - q_i(T_{N+1})|$ of the actual co-simulation can be estimated by the error estimator

$$\varepsilon_{i,N+1}^{\text{pos}} = \frac{C_{k+1}^{\text{pos},N+1}}{C_{k}^{\text{pos},N+1}} |q_i,N+1 - \hat{q}_i,N+1|, \quad (19)$$

which converges to the local error $\varepsilon_{i,N+1}^{\text{pos}}$ with order $O(H_k^{k+4})$.

Also, an error estimator $\varepsilon_{i,N+1}^{\text{vel}}$ for the velocity variables can be derived, namely

$$\varepsilon_{i,N+1}^{\text{vel}} = \frac{C_{k+1}^{\text{vel},N+1}}{C_{k}^{\text{vel},N+1}} |v_i,N+1 - \hat{v}_i,N+1|,$$

with

$$C_{\text{vel},N+1}^{k} = \frac{1}{H_N} \int_{T_N}^{T_{N+1}} L_{i}^{k}(t) \, dt \quad \text{and} \quad L_{i}^{k}(t) = \prod_{j=0}^{k-1} \frac{t - T_{N-j}}{T_{N+1} - T_{N-j}}, \quad (20)$$

which approximates the local error $\varepsilon_{i,N+1}^{\text{vel}} = |v_i,N+1 - v_i(T_{N+1})|$ of the actual co-simulation with order $O(H_k^{k+1})$.

### 4.3 Error estimator 3 for the explicit co-simulation (ExSLocal)

The third error estimator for the classical explicit co-simulation approach is also constructed with the help of the local extrapolation technique and termed by ExSLocal. With the error estimators of Sects. 4.1 and 4.2, the local errors $\varepsilon_{i,N+1}^{\text{pos}} = |q_i,N+1 - q_i(T_{N+1})|$ and $\varepsilon_{i,N+1}^{\text{vel}} = |v_i,N+1 - v_i(T_{N+1})|$ of the state variables $q$ and $v$ can be estimated. In an alternative approach, the quality of the co-simulation is not directly measured by considering the local error of the state variables, but by regarding the local error of the coupling variables $u$. The coupling variables are generally defined by the implicit coupling conditions

$$u - \varphi(t, q, v, u) = 0$$

and may therefore simply be considered as a nonlinear function of $q$ and $v$. Instead of using $\varepsilon_{i,N+1}^{\text{pos}}$ and $\varepsilon_{i,N+1}^{\text{vel}}$ for measuring the quality of the co-simulation, the local error $|u_{i,N+1} - u_i(T_{N+1})|$ of the coupling variables may be considered.

Applying the classical explicit co-simulation scheme, the extrapolated coupling variables $p_{N+1}^{\text{ext}}(t) = P_{N+1}^{k}(t; (T_N, u_N), \ldots, (T_{N-k}, u_{N-k}))$ are used for integrating the subsystems. After the subsystem integration from $T_N$ to $T_{N+1}$ has been accomplished, updated coupling variables $u_{N+1}$ are computed at the new time point $T_{N+1}$. Therefore, the new state variables $q_{N+1}$, $v_{N+1}$ are inserted into the coupling equations and the relationship $u_{N+1} - \varphi(T_{N+1}, q_{N+1}, v_{N+1}, u_{N+1}) = 0$ is solved for $u_{N+1}$.

The predicted coupling variables $u_{N+1}^{\text{ext}} = P_{N+1}^{k}(T_{N+1}; (T_N, u_N), \ldots, (T_{N-k}, u_{N-k}))$ are usually different from the updated coupling variables $u_{N+1}$. The difference $u_{N+1}^{\text{ext}} - u_{N+1}$
may therefore be used to construct an error estimator based on the local extrapolation technique. It has been shown in [40] that the local error $e_{i,N+1}^{u} = |u_{i,N+1}^{ext} - u_i (T_{N+1})|$ of the predicted coupling variable $u_{i,N+1}^{ext}$ can be estimated with the help of the updated coupling variable $u_{i,N+1}$ according to

$$e_{i,N+1}^{u} = |u_{i,N+1}^{ext} - u_{i,N+1}|. \quad (21)$$

The estimator $e_{i,N+1}^{u}$ converges with order $O(H^{k+2})$ to the local error $e_{i,N+1}^{u}$ if consistent coupling variables $u_{i,N+1}$ are used. Hence, this estimator can only be applied, if updated coupling variables are calculated at $T_{N+1}$ by solving $u_{N+1} - \phi (T_{N+1}, q_{N+1}, v_{N+1}, u_{N+1}) = 0$ for $u_{N+1}$. Note that $e_{i,N+1}^{u}$ does not monitor the local error $|u_{i,N+1}^{ext} - u_i (T_{N+1})|$ of the updated coupling variables, but the local error $|u_{i,N+1}^{ext} - u_i (T_{N+1})|$ of the predicted coupling variables.

It should be mentioned that for systems without feed-through, consistent coupling variables can be calculated explicitly in a very straightforward manner with the help of the new variables $q_{N+1}$ and $v_{N+1}$ by simply evaluating $u_{N+1} = \phi (T_{N+1}, q_{N+1}, v_{N+1})$. For systems with feed-through, the update relationship $u_{N+1} = \phi (T_{N+1}, z_{N+1}, u_{N+1})$ is – for practical reasons – sometimes replaced by the simplified update $u_{N+1} = \phi (T_{N+1}, z_{N+1}, p_{N+1} (T_{N+1}))$. In this case, $e_{i,N+1}^{u}$ will in general not converge with $O(H^{k+2})$ to the local error $e_{i,N+1}^{u}$. As a consequence, Eq. (21) will not represent a mathematically sound error estimator for the predicted coupling variables.

### 4.4 Error estimator 4 for the implicit co-simulation (ImMilne)

The implicit co-simulation scheme according to Sect. 3.2 is based on a predictor/corrector approach. With the help of the Milne device technique, the difference between the predicted and corrected state variables can be used to construct an error estimator [40]. The local error of the corrected position variables $q_{i,N+1}$ can be estimated by

$$e_{i,N+1}^{pos} = \left(1 - \frac{C_{pos,N+1}^{k+1}}{C_{pos,N+1}^{k}}\right) \left|q_{i,N+1}^{pre} - q_{i,N+1}\right|. \quad (22)$$

where $C_{pos,N+1}^{k}$ are the error constants defined in Eq. (17). The estimator $e_{i,N+1}^{pos}$ converges to the local error $e_{i,N+1}^{pos} = \left|q_{i,N+1}^{pre} - q_{i} (T_{N+1})\right|$ with order $O(H^{k+4})$.

A similar approach for the velocity variables yields the error estimator

$$e_{i,N+1}^{vel} = \left(1 - \frac{C_{vel,N+1}^{k+1}}{C_{vel,N+1}^{k}}\right) \left|v_{i,N+1}^{pre} - v_{i,N+1}\right|. \quad (23)$$

where $C_{vel,N+1}^{k}$ denote the error constants of Eq. (20). The estimator $e_{i,N+1}^{vel}$ approximates the local error $e_{i,N+1}^{vel} = \left|v_{i,N+1}^{pre} - v_{i} (T_{N+1})\right|$ of the corrected velocity variables with order $O(H^{k+3})$.

### 4.5 Error estimator 5 for the implicit co-simulation (ImSLocal)

This error estimator, which is based on a local extrapolation approach, can be used to monitor the error of the predicted coupling variables $u_{i,N+1}^{pre}$, i.e., to estimate the local error $e_{i,N+1}^{u} = \left|u_{i,N+1}^{pre} - u_i (T_{N+1})\right|$. As shown in [40], the difference between the predicted
and corrected coupling variables, i.e.,

$$\varepsilon_{i,N+1}^u = \left| u_{i,N+1}^{pre} - u_{i,N+1} \right|,$$

converges with order $O\left(H^{k+2}\right)$ to the local error $\varepsilon_{i,N+1}^u$.

### 5 Macro-step size and order control algorithm

The here proposed algorithm for controlling the macro-step size and the integration order (i.e., the polynomial degree $k$ of the approximation polynomials for the coupling variables) is carried out in three steps. The suggested approach may be interpreted as a straightforward generalization of step size and order control strategies well-established in the field of classical time integration [8, 15, 67, 68]. In the first step, it is checked, whether the macro-step can be accepted or has to be repeated since the estimated error is too large (Sect. 5.1). For that purpose, the error estimators for the local error of Sect. 4 are used. The second step contains the algorithm for the selection of the optimal integration order (Sect. 5.2). In the third step, the optimal step size for the next macro-step is determined (Sect. 5.3).

#### 5.1 Step 1: Local error test

After the macro-step $T_N \rightarrow T_{N+1}$ with $T_{N+1} = T_N + H_N$ has been accomplished, the local error is estimated with the help of one of the entire five methods suggested in Sect. 4. The estimated error $\varepsilon_{N+1}$ – more precisely the weighted root mean square norm of $\varepsilon_{N+1}$ – has to pass the subsequent local error test

$$\|\varepsilon_{N+1}\|_{\text{WRMS}} = \sqrt{\frac{1}{n^*} \sum_{i=1}^{n^*} \left( \frac{\varepsilon_{i,N+1}}{W_{i,N+1}} \right)^2} \leq 1. \quad (25)$$

Note that $n^*$ designates the number of variables, which are used for controlling the macro-step size. Basically, three alternatives exist. First, all state variables of all subsystems may be considered in the sum of Eq. (25). Second, only the state variables of the coupling bodies are taken into account in the summation. Third, in connection with the error estimators of Sect. 4.2 and Sect. 4.5, the summation is carried out over the $n_c$ coupling variables (i.e., $n^* = n_c$). The weights $W_{i,N+1}$ depend on the user-defined relative and absolute error tolerances $\text{rtol}$ and $\text{atol}$, and also on the magnitude of the variables, which are used for the error estimation. Hence, essentially two cases have to be distinguished:

i) The error of the state variables is estimated ($\text{ExLocal}, \text{ExMilne}, \text{ImMilne}$);

ii) The error of the coupling variables is estimated ($\text{ExSLocal}, \text{ImSLocal}$).

In case i), the error analysis is carried out separately on position level and on velocity level. This is reasonable, since the position and velocity variables exhibit a different convergence behavior, see Sect. 4. Hence, in case i) the local error test

$$\max(\|\varepsilon_{N+1}\|_{\text{WRMS}}, \|\varepsilon_{N+1}\|_{\text{WRMS}}) \leq 1 \quad (26)$$

with the weights $W_{i,N+1}^{\text{pos}} = \text{atol}_i \cdot |q_{i,N+1}|$ and $W_{i,N+1}^{\text{vel}} = \text{atol}_i + \text{rtol} \cdot |v_{i,N+1}|$ is applied. Principally, all subsystem state variables can be used for the error estimation, i.e., the index $i$ in Eq. (25) runs over all bodies. Alternatively, only the states of the coupling
bodies are considered for the error estimation so that the index \( i \) only runs over the coupling bodies. It should be mentioned that the value \( atol \) can be chosen individually for each component, while \( rtol \) is identical for all components. It is clear that the proper choice of \( atol \) and \( rtol \) is problem-dependent and strongly affects the accuracy and the efficiency of the co-simulation.

In case ii), the local error test reads

\[
\|e^i_{N+1}\|_{WRMS} \leq 1, \tag{27}
\]

with the weights \( W^i_{i,N+1} = atol^i + rtol \cdot |u_{i,N+1}| \). The index \( i \) in Eq. (25) runs over all coupling variables.

Now, two possibilities have to be considered:

a) The local error test is passed, i.e., condition (26) or condition (27) is fulfilled. Then, the next macro-step \( T_{N+1} \rightarrow T_{N+2} \) with \( T_{N+2} = T_{N+1} + H_{N+1} \) can be carried out.

b) The local error test is not passed. In this case, a repetition of the current macro-step \( T_N \rightarrow \tilde{T}_{N+1} (\tilde{T}_{N+1} = T_N + \tilde{H}_N) \) with a reduced step size \( \tilde{H}_N \) has to be accomplished.

For a concise notation, both possibilities are combined. Therefore, the macro-step carried out next is denoted by \( T_M \rightarrow T_{M+1} (T_{M+1} = T_M + H_M) \), where

\[
T_M \rightarrow T_{M+1} := \begin{cases} 
T_{N+1} \rightarrow T_{N+2} & \text{for} \quad \|e_{N+1}\|_{WRMS} \leq 1, \\
T_N \rightarrow \tilde{T}_{N+1} & \text{for} \quad \|e_{N+1}\|_{WRMS} > 1. 
\end{cases} \tag{28}
\]

In short, the index \( M \) is set to \( M := N + 1 \) if the local error test is passed. If the local error test has failed, the index \( M \) is set to \( M := N \).

5.2 Step 2: optimal integration order and order control algorithm

The task consists of the selection of the optimal polynomial degree \( k \) for the approximation polynomials for the next macro-step \( T_M \rightarrow T_{M+1} \). Note again that the next macro-step is either a new macro-step (case a)) or a repetition of the current macro-step (case b)).

5.2.1 Order control algorithms for BDF solvers

a) Basic idea:

The order control algorithm applied here for calculating the optimal polynomial degree \( k \) for the approximation polynomials is inspired by well-established order control approaches used in connection with BDF solvers. Regarding variable-order BDF solvers, the optimal integration order is determined by considering the leading error term in the remainder of a Taylor series expansion. The remainder of the Taylor series is used for estimating the local truncation error (LTE). The local truncation error of an arbitrary time integration method of order \( k \) depends on the \((k+1)\)th derivative of the solution vector \( z \) and on the solver step size \( h \) [15, 67] and reads

\[
LTE(k) = c_{k+1} h^{k+1} z^{(k+1)} + O(h^{k+2}). \tag{29}
\]

Note that \( c_{k+1} \) terms the leading error constant, which generally depends on the numerical integration method. Considering concretely the time step from \( t_N \) to \( t_{N+1} = t_N + h \)
carried out with a BDF integrator of order $k$, the local truncation error reads $LTE(k) = \frac{1}{k+1} h^{k+1} z^{(k+1)}(t_{i+1}) + O(h^{k+2})$. Since $z^{(k+1)}(t_{i+1})$ is unknown at the beginning of the time step, the term $z^{(k+1)}(t_{i+1})$ is replaced by the $(k+1)^{\text{st}}$ derivative of a predictor polynomial $P_{k+1}^\text{pre}(t)$, i.e., $z^{(k+1)}(t_{i+1}) \approx \frac{d^{k+1}}{dt^{k+1}} P_{k+1}^\text{pre}(t)\big|_{t_{i+1}}$.

The integration order for the next solver step is determined with the aim to maximize the integration step size. Therefore, the local truncation error norms $\|LTE(k-1)\| \approx \|c_k h^k z^{(k)}\|$, $\|LTE(k)\| \approx \|c_k h^{k+1} z^{(k+1)}\|$, and $\|LTE(k+1)\| \approx \|c_{k+2} h^{k+2} z^{(k+2)}\|$ are considered. The integration order, which provides the lowest estimated truncation error and consequently allows the largest step size, is used for the next step [50].

If higher order integration methods are used ($k > 2$), the order selection algorithm is usually extended by a heuristic approach to detect numerical instabilities. The heuristic approach suggested in [68] is based on the idea that oscillations with high frequencies and growing amplitudes will arise, if the simulation gets unstable. As a consequence, the local truncation error will oscillate rapidly with increasing magnitudes, when solver instabilities occur.

\[b) \text{ Modified approach:}\]

A slightly modified approach for controlling the integration order, which is, for instance, implemented in the IDA solver of the SUNDIALS package [25], has been suggested in [8]. Note that IDA is a variable-order, variable-coefficient BDF solver for the solution of implicit differential-algebraic equation systems. The modified approach also considers the leading error term of the remainder of a Taylor series expansion. However, the leading error term is scaled to be independent of the error constant. Within the control algorithm proposed in [8], the three terms $\|h^{k-1} z^{(k-1)}\|$, $\|h^k z^{(k)}\|$, and $\|h^{k+1} z^{(k+1)}\|$ are compared. If these three terms do not form a decreasing sequence, i.e., if the condition $\|h^{k-1} z^{(k-1)}\| > \|h^k z^{(k)}\| > \|h^{k+1} z^{(k+1)}\|$ is not fulfilled, the integration order is decreased to $k - 1$. If the condition is fulfilled and if also $\|h^{k+1} z^{(k+1)}\| > \|h^{k+2} z^{(k+2)}\|$ is satisfied, the order may be increased to $k + 1$. Numerical studies have shown that applying the modified approach, the integration order is reduced sooner in case of numerical instabilities compared to the approach described in a).

5.2.2 Order control algorithm for co-simulation

The order control algorithm implemented here for controlling the integration order of the co-simulation may be regarded as a straightforward adaption of the order control strategy suggested in [8] for BDF solvers. Within each macro-step, extrapolation polynomials $P_{M+1}^j = P_{M+1}^j(t; (T_M, u_M), \ldots, (T_{M-j}, u_{M-j}))$ of different degrees $j = 1, \ldots, k + 1$ are considered. Then, corresponding scaled derivative norms according to $SDN(j) := \|H_N P_{M+1}^{j,(j)}\|$ are computed for the different polynomials, where $H_N$ is the current macro-step size. Note that the upper index in brackets denotes the differentiation with respect to time $t$, i.e., $P_{M+1}^{j,(j)} = \frac{d^j P_{M+1}^j}{dt^j}$. If the condition

$$\|H_N^1 P_{M+1}^{1,(1)}\| > \cdots > \|H_N^{k-1} P_{M+1}^{k-1,(k-1)}\| > \|H_N^k P_{M+1}^{k,(k)}\| > \|H_N^{k+1} P_{M+1}^{k+1,(k+1)}\|$$ (30)

is fulfilled, the co-simulation is continued with the current polynomial degree $k$. If the condition is not fulfilled, the polynomial degree is reduced to $k - 1$. If the condition (30) is
fulfilled and if \( k + 1 \) consecutive macro-time steps have been carried out successfully with degree \( k \), also the term \( \| H_N^{k+1} P_{M+1}^{k+2,(k+2)} \| \) is computed. The polynomial degree will be increased to \( k + 1 \), if \( \| H_N^{k+1} P_{M+1}^{k+2,(k+2)} \| > \| H_N^{k+2} P_{M+1}^{k+2,(k+2)} \| \) is fulfilled.

The complete order control algorithm implemented here is summarized in the subsequent scheme and in the flowchart of Fig. 6 and is mainly based on the order controller of the IDA solver [25]:

- Check, if polynomial order has to be decreased:
  - If \( k = 1 \) and \( SDN(1) \leq \frac{1}{2} SDN(2) \):
    * \( k_{\text{new}} := 0 \)
  - Else if \( k > 1 \) and \( \max(SDN(k), SDN(k-1)) \leq SDN(k+1) \):
    * \( k_{\text{new}} := k - 1 \)
  - Else:
    * \( k_{\text{new}} := k \)

- Additional condition based on the number of consecutive failed macro-steps \( N_{\text{fail}} \):
  - If \( N_{\text{fail}} > 5 \):
    * \( k_{\text{new}} := 0 \)
– Else if \( N_{\text{fail}} > 3 \):
  * \( k_{\text{new}} := \min (k_{\text{new}}, 1) \)
– Else:
  * \( k_{\text{new}} \) is not adjusted

- If the number of consecutive successful macro-steps with constant order \( k \) and without reducing the macro-step size fulfills \( N_{\text{acc}} > k \), then \( SDN (k+2) \) is computed and an adjustment of the order is considered:
  - If \( k = 0 \) and \( SDN (2) < \frac{1}{2} SDN (1) \):
    * \( k_{\text{new}} := 1 \)
  - Else if \( k > 0 \) and \( SDN (k) \leq \min (SDN (k+1), SDN (k+2)) \):
    * \( k_{\text{new}} := k - 1 \)
  - Else if \( k > 0 \) and \( SDN (k+2) < SDN (k+1) \):
    * \( k_{\text{new}} := k + 1 \)
  – Else:
    * \( k_{\text{new}} \) is not adjusted

5.3 Macro-step size controller

After the polynomial degree \( k := k_{\text{new}} \) for the next macro-step has been determined, the algorithm for the macro-step size selection is executed. The macro-step size controller computes a positive, real-valued scaling factor \( r^p \). This factor is used for calculating the new macro-step size \( H_M \) for the next macro-step \( T_M \rightarrow T_{M+1} \) (see Eq. (28)) by means of the simple relationship \( H_M = r^p \cdot H_N \). The calculation of the new macro-step size \( H_M \) is based on the estimated error \( \varepsilon_{N+1} \). The idea is to select the new macro-step size \( H_M \) in such a way that the resulting estimated error exactly fulfills the local error condition (25), i.e., \( \| \varepsilon_{M+1} \|_{WRMS} = 1 \).

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

1) The two consecutive macro-steps \( T_N \rightarrow T_{N+1} \) and \( T_M \rightarrow T_{M+1} \) are carried out with the same polynomial degree. In this case, the error estimator \( \varepsilon_{N+1} \) according to Sect. 4, which has been used for the local error test in Eq. (25), is also applied for calculating the new macro-step size \( H_M \).

2) After the macro-step \( T_N \rightarrow T_{N+1} \) has been accomplished, the polynomial degree is changed by the order control algorithm. In this second case, the error estimator \( \varepsilon_{N+1} \) cannot be used to calculate the step size for the next macro-step. Hence, the alternative \textit{a priori} error estimator

\[
\tilde{\varepsilon}_{M+1}^u = P_{M+1}^{k+1}(T_{M+1}) - P_{M+1}^k(T_{M+1})
\]

\[
= \frac{1}{(k+1)!} \prod_{j=1}^{k+1} (T_{M+1} - T_{M+1-j}) P_{M+1}^{k+1,(k+1)}
\]  \hspace{1cm} (31)

is used for the next macro-step \( T_M \rightarrow T_{M+1} \). The two vectors \( P_{M+1}^{k+1}(t) = P_{M+1}^{k+1}(t; (T_M, u_M), \ldots, (T_{M-k}, u_{M-k})) \) and \( P_{M+1}^k(t) = P_{M+1}^k(t; (T_M, u_M), \ldots, (T_{M-k}, u_{M-k})) \) collect approximation polynomials of degree \( k \) and \( k+1 \), respectively. Note again that the upper index in brackets denotes the differentiation with respect to the time, i.e., \( P_{M+1}^{k+1,(k+1)} \). It should be mentioned that \( \tilde{\varepsilon}_{M+1}^u \) simply estimates the local error of the predicted coupling variables.
**Remark 1** The a priori error estimator $\tilde{\varepsilon}_{M+1}^u$ is only used to determine the new macro-step size $H_M$. After the macro-step $T_M \rightarrow T_{M+1}$ has been accomplished with $H_M$, the error is again checked with the a posteriori error estimator $\varepsilon_{M+1}$ according to Sect. 4. Hence, acceptance of a macro-step, see Eq. (25), is always based on the a posteriori error estimator.

**Remark 2** The a priori estimator only monitors the error of the coupling variables. Using the error estimators $Ex_{Local}$, $Ex_{Milne}$, $Im_{Milne}$, however, the error of the state variables is monitored. As a consequence, the a priori estimator and the a posteriori estimator may be based on different variables.

Considering case 1), the scaling factor $r^\rho$ for the new (optimal) macro-step size $H_M$ is calculated by one of the following two relations:

\[
(\alpha) \quad r^{pos} = \left( SF \cdot \left\| \varepsilon_{N+1}^{pos} \right\|_{WRMS} \right)^{-\frac{1}{k+3}}, \quad r^{vel} = \left( SF \cdot \left\| \varepsilon_{N+1}^{vel} \right\|_{WRMS} \right)^{-\frac{1}{k+2}},
\]

\[
(\beta) \quad r^u = \left( SF \cdot \left\| \varepsilon_{N+1}^{u} \right\|_{WRMS} \right)^{-\frac{1}{k+1}}, \quad \tilde{r}^u = \left( SF \cdot \left\| \tilde{\varepsilon}_{M+1}^{u} \right\|_{WRMS} \right)^{-\frac{1}{k+1}}.
\]

Relation (\alpha) is used for the error estimators $Ex_{Local}$, $Ex_{Milne}$ and $Im_{Milne}$, which are based on state variables; (\beta) is applied in connection with the error estimators $Ex_{SLocal}$ and $Im_{SLocal}$, which are based on coupling variables.

Regarding case 2), the scaling factor $r^\rho$ for the new macro-step size $H_M$ is computed by

\[
\tilde{r}^u = \left( SF \cdot \left\| \tilde{\varepsilon}_{M+1}^{u} \right\|_{WRMS} \right)^{-\frac{1}{k+1}}.
\]

The exponents in Eqs. (32) and (33) refer to the convergence order of the co-simulation method, which is $k + 3$ on position level and $k + 2$ on velocity level for the here considered explicit and implicit co-simulation approaches. The coupling variables converge with order $k + 1$. A detailed derivation of Eqs. (32) and (33) can be found in [40]; $SF$ is a user-defined safety factor, e.g., $SF \in [2, 6]$. The safety factor has to be chosen carefully. If the safety factor $SF$ is chosen too small, the number of macro-step repetitions may increase significantly due to the increased number of failed local error tests. Choosing $SF$ too large may entail too small macro-step sizes. If both the error of the position variables and the error of the velocity variables are estimated ($Ex_{Local}$, $Ex_{Milne}$, $Im_{Milne}$), the scaling factor is here chosen conservatively by means of $r^\rho = \min \left( r^{pos}, r^{vel} \right)$.

Summarizing, the scaling factor $r^\rho$ is determined according to

\[
r^\rho = \begin{cases} 
\min \left( r^{pos}, r^{vel} \right) & \text{error estimator for state variables (ExLocal, ExMilne, ImMilne),} \\
\tilde{r}^u & \text{error estimator for coupling variables (ExSLocal, ImSLocal),} \\
r^u & \text{change of the order of the approximation polynomials.}
\end{cases}
\]

Finally, the macro-step size control algorithm is explained. Depending on the number $N_{fail}$ of consecutively failed macro-steps within the current step $T_N \rightarrow T_{N+1}$, the following limitations for the scaling factor $r^\rho$ are used (see also the flowchart in Fig. 7):

$$\text{min}$$
Co-simulation with variable approximation order

Fig. 7 Flowchart of the macro-step size controller

- $N_{\text{fail}} = 0$:
  
  (a) If $r^\rho \geq r_{\text{max}}^\rho$:
    perform next macro-step with $H_M := \frac{r^\rho}{r_{\text{max}}^\rho} \cdot H_N$ (e.g., $r_{\text{max}}^\rho = 2.0$)
    (restriction for maximum step-size increase).
  
  (b) If $r^\rho < 1$:
    perform next macro-step with $H_M := \min \{0.9, \max(r_{\text{min}}^\rho, r^\rho)\} \cdot H_N$ (e.g., $r_{\text{min}}^\rho = 0.5$).
  
  (c) If $1 \leq r^\rho < r_{\text{max}}^\rho$:
    perform next macro-step with $H_M := H_N$.

- $N_{\text{fail}} = 1$:
  repeat current macro-step with reduced step size $H_M := \min \{0.9, \max(0.25, 0.9 \cdot r^\rho)\} \cdot H_N$.

- $N_{\text{fail}} > 1$ or corrector failed to converge (implicit co-simulation):
  repeat current macro-step with reduced step size $H_M := 0.25 \cdot H_N$.

- If $H_M \leq H_{M,\text{min}} = 10^{-12}$:
  simulation is aborted and an error message is printed.

It should be noted that the error tolerances $atol_i$ and $rtol_i$ for the co-simulation as well as the safety factor $SF$ have to be specified by the user. Generally, they are problem-dependent.

The macro-step size control algorithm described above may be interpreted as an I-controller. Alternatively, a PI-controller may be used by incorporating the previous errors $\varepsilon_{\text{pos}}^{i,N}$ and $\varepsilon_{\text{vel}}^{i,N}$ [24].

6 Numerical examples

The co-simulation models considered here consist of the subsystems and the co-simulation interface. The communication between the co-simulation interface and the subsystems is
managed over the Message Passing Interface v3.1 (Intel MPI Library). Each subsystem is executed by an individual MPI-rank. If multiple integrations of the same subsystem within a macro-time step are required, e.g., to numerically compute the interface Jacobian by a finite difference approach or in connection with the calculation of the error estimator, the additional subsystem integrations are parallelized via OpenMP. The parallel threads are synchronized at the macro-time points.

A flowchart of the parallel implementation of the explicit co-simulation method with the macro-step size and order control algorithm is shown in Fig. 8. A corresponding flowchart of the implicit co-simulation approach is depicted in Fig. 9.

It should be mentioned that the co-simulations have been executed fully parallelized on a high-performance supercomputer [37] with Intel Xeon Platinum 9242 processors. Each subsystem integration has been accomplished on an individual core; the different cores are connected over an InfiniBand interconnect. Note that the computation times shown in the subsequent figures reflect the wall-clock time (including communication latency).

6.1 Example 1: Nonlinear two-mass oscillator

As a first example, we consider the nonlinear two-mass oscillator shown in Fig. 10 (masses $m_1$, $m_2$; spring constants $c_1$, $c_2$; damping coefficients $d_1$, $d_2$). The two masses are coupled
Co-simulation with variable approximation order

Fig. 9 Flowchart of the implicit co-simulation approach with the macro-step size controller ImMilne (interface Jacobian computed by a finite difference approach requiring additional subsystem integrations with perturbed approximation polynomials)

Fig. 10 Two coupled single-mass oscillators

by the nonlinear force $f^c = c_e \cdot (x_2 - x_1) + d_e \cdot (v_2 - v_1) + C_e \cdot (x_2 - x_1)^3 + D_e \cdot (v_2 - v_1)^3$. Furthermore, a penalty-based contact force $F_C = -10^{-2} \cdot e^{10^7 m^{-3} x_1}$ is assumed to act at $m_1$. 
Applying a force/force-decomposition approach, the coupled co-simulation system is defined by the following semi-explicit index-1 DAE system

**Subsystem 1:**
\[
\begin{align*}
\dot{x}_1 &= v_1, \\
\dot{v}_1 &= -\frac{c_1}{m_1} x_1 - \frac{d_1}{m_1} v_1 + \frac{F_C}{m_1} + \frac{f_c}{m_1} 
\end{align*}
\] (a)

**Subsystem 2:**
\[
\begin{align*}
\dot{x}_2 &= v_2, \\
\dot{v}_2 &= -\frac{c_2}{m_2} x_2 - \frac{d_2}{m_2} v_2 - \frac{f_c}{m_2} 
\end{align*}
\] (b)

**Coupling condition:**
\[
g_c := f_c - c_c \cdot (x_2 - x_1) - d_c \cdot (v_2 - v_1) - C_c \cdot (x_2 - x_1)^3 - D_c \cdot (v_2 - v_1)^3 = 0. 
\] (c)

The simulations have been carried out with the subsequent parameters: \(m_1 = 50 \text{ kg}, m_2 = 1 \text{ kg}, c_1 = 1.0E5 \frac{N}{m}, c_2 = 1.0E3 \frac{N}{m}, c_c = 1.0E5 \frac{N}{m}, d_1 = d_2 = d_c = 0, C_c = 1.0E5 \frac{N}{m^3}, \) and \(D_c = 1.0E2 - 2 \frac{N a^3}{m^5}.\) The initial conditions are: \(x_{1,0} = -2.0 \text{ m}, x_{2,0} = 0 \text{ m}, v_{1,0} = 100 \frac{m}{s}, \) and \(v_{2,0} = -200 \frac{m}{s}.\) Subsystem integration has been accomplished with a BDF integrator [25] (variable step-size, relative and absolute error tolerances \(atol_{x}^{BDF} = 10^{-9}, atol_{v}^{BDF} = 10^{-6}, rtof_{x}^{BDF} = 10^{-6}).\) The following parameters have been used for the macro-step size controller: \(SF = 6.0, atol_{x} = 10^{-8}, atol_{v} = 10^{-5}, rtof_{x} = 10^{-5}, atol_{f_{c}} = 10^{-1}, rtof_{f_{c}} = 10^{-3}, r_{\rho_{\text{min}}} = 0.5, \) and \(r_{\rho_{\text{max}}} = 2.0.\)

Figure 11 depicts the displacement \(x_i(t)\) and the corresponding velocity \(v_i(t)\) for the different error estimators of Sect. 4 in combination with the order control algorithm of Sect. 5. The reference solution has been generated with a monolithic model, which has been integrated with very small error tolerances. The macro-step sizes for the different error estimators are shown in Fig. 12; the degree \(k\) of the approximation polynomials is depicted in...
Fig. 12  Nonlinear two-mass oscillator: macro-step size $H$ for the different error estimators

Fig. 13  Nonlinear two-mass oscillator: polynomial degree $k$ of the approximation polynomials for the different error estimators

Fig. 14  Nonlinear two-mass oscillator: global errors and number of macro-steps for the error and order controlled co-simulations

Fig. 13. It can clearly be seen that the macro-step size and also the polynomial degree are reduced at the time points where the contacts occur. The macro-step size is reduced very
strongly with the simplified error estimators (ExSLocal and ImSLocal), since these two error estimators are based on the coupling force, which is large at the time points where the contacts occur. The total number of macro-steps and the resulting global error are illustrated in Fig. 14. Note that the scalar error norm used in Fig. 14(a) is based on the normalized root mean square error $e$ according to

$$
e_i = \sqrt{\frac{\sum_{j=1}^{N} (z_{i,j}^{ref} - z_{i,j})^2}{\sum_{j=1}^{N} (z_{i,j}^{ref} - z_{i}^{ref})^2}}$$

with $z_i^{ref} = \frac{1}{N} \sum_{j=1}^{N} z_{i,j}^{ref}$

$$\Rightarrow e = \|e\|_2 \text{ with } e = (e_1, \ldots, e_i, \ldots, e_{Nz})^T,$$

where $z^{ref} \in \mathbb{R}^{N \times N_z}$ denotes the reference solution; $N_z$ terms the number of state variables, and $N$ is the number of output points.

6.2 Example 2: Comparative study between explicit and implicit co-simulation approaches

A chain of $n_K$ masses is considered, see Fig. 15. The masses, described by the displacement coordinates $x_i$ and the velocities $v_i = \dot{x}_i$, are connected by nonlinear spring/damper elements. The spring/damper element depicted in Fig. 16, which connects the two masses $m_{i-1}$ and $m_i$, is defined by the constitutive equation

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

where $c_i, d_i, C_i, D_i$ characterize the linear and nonlinear stiffness and damping parameters. The chain model is used here because it is simple to scale with respect to the number of degrees of freedom. Also, the number of subsystems can be changed very easily. Therefore, the model is well suited to carry out reliable numerical studies in order to compare efficiency and accuracy of different co-simulation methods.

The model is separated into $r$ subsystems via force/force-decompositions. The $n_c = r - 1$ coupling conditions are defined by the constitutive equations of the coupling spring/damper
Fig. 16  Coupling force $\lambda_s$ between the last body of subsystem $s$ and the first body of subsystem $s+1$

elements. Regarding the two adjacent subsystems $s$ and $s+1$, the coupling condition $g_s$ is given by

$$g_s := \lambda_s - c_c \left( x_{s+1} - x_{sN.K,s} \right) - d_c \left( v_{s+1} - v_{sN.K,s} \right) - C_c \left( x_{s+1} - x_{sN,K,s} \right)^3$$

$$- D_c \left( v_{s+1} - v_{sN,K,s} \right)^3 = 0 \quad (s = 1, \ldots, n_c). \quad (38)$$

At special time points $t_{\alpha}$ ($t_{\alpha} = 0, 0.05, 0.1, \ldots$), impulse-shaped external forces $F_{Imp,j} = \frac{1}{2} \Delta F_j \left[ \tanh \left( \frac{t_{\alpha} - t_0}{\delta} \right) \right] - \tanh \left( \frac{t_{\alpha} + \Delta t}{\delta} \right)$ with $\delta = 1.0E - 5$ s and $\Delta t = 2.5E - 3$ s are applied at 30% of the masses (randomly selected). The force amplitude $\Delta F_j$ is chosen randomly in the interval $[-1.0E6, 5.0E6]$ N. Moreover, at 10% of the masses (randomly chosen), contact forces are applied by means of the penalty approach $F_{Con,i} = A e^{Bx_i}$ with $A = -1.0E - 2$ N and $B = 5.0E5$ m$^{-1}$.

A chain with 10,000 masses is considered here, which is decomposed into 191 subsystems. The simulations have been carried out with the subsequent parameters: $m_i = 1.0E - 1$ kg (varied in case study 3), $c_i = 1.0E7 \frac{N}{m}$ (varied in case study 2), $d_i = 1.0E4 \frac{Ns}{m}$ (varied in case study 1), $C_i = 1.0E14 \frac{Ns}{m^2}$, $D_i = 1.0E2 \frac{Ns^3}{m^3}$, simulation time $t_{\text{sim}} = 0.5$ s. Random initial conditions have been used ($x_{0,i} = [-1.0E - 3, 1.0E - 3]$ m and $v_{0,i} = [-1.0E1, 1.0E1] \frac{m}{s}$). The subsystem integrations have been accomplished with a variable step-size BDF integrator [25] with the solver settings $rtol^{BDF} = 1.0E - 6$, $atol^{BDF} = 1.0E - 9$, and $atol^{BDF} = 1.0E - 6$. The co-simulations have been carried out with the macro-step size and order control algorithm described in Sect. 5. Simulation results are presented for the two error estimators $ExSLocal$ and $ImSLocal$, see Sect. 4, with the subsequent macro-step size controller parameters: $SF = 6.0$, $atol_{m} = 10^{-1}$, $rtol_{m} = 10^{-3}$, $r_{\text{min}}^{\mu} = 0.5$, and $r_{\text{max}}^{\mu} = 2.0$.

6.2.1 Case study 1

The main intention of Example 2 is to compare the numerical performance of the explicit with the implicit co-simulation approach. Giving general statements is, of course, not possible since the performance of a co-simulation approach strongly depends on the system and the model parameters. For the considered model, our simulations have, however, shown that the damping coefficients are crucial and important parameters for comparing the efficiency of explicit and implicit methods. Within a first numerical study, the damping parameter $d_i$
of the spring/damper elements is therefore varied in the range \([1.0E3, 1.0E6]\) Ns/m, see Figs. 17–21.

Figure 17 shows displacement and velocity of mass 6891 for \(d_i = 1.0E4\) Ns/m simulated with the explicit co-simulation approach (variable co-simulation order; variable macro-step size with error estimator \(ExSLocal\)). Although the system is rather stiff and although steep gradients occur due to the impulse-shaped excitation, the co-simulation is stable and the results are accurate and close to the reference solution.

Figure 18 collects simulation results accomplished with the implicit co-simulation method (variable co-simulation order; variable macro-step size with error estimator \(ImSLocal\)). The same model parameters have been used as for the explicit co-simulations depicted in Fig. 17; only the damping parameter has been increased to \(d_i = 5.0E5\) Ns/m. Again, a good agreement between the co-simulation and the reference solution can be observed.

Figure 19 shows a direct comparison between the explicit (\(ExSLocal\)) and the implicit (\(ImSLocal\)) co-simulation approach for the case \(d_i = 1.0E4\) Ns/m. The plots illustrate the macro-step size as well as the polynomial degree of the approximation polynomials over the simulation time \(t\). As can be seen, the (average) macro-step size is (slightly) smaller for
The explicit approach. The order controller works well for both methods; the variation of the integration order is little larger for the explicit approach.

The same study for the case $d_i = 5.0E5$ Ns/m is collected in Fig. 20. Here, the (average) macro-step size is markedly smaller for the explicit approach indicating that the implicit co-simulation approach is more efficient for the higher-damped case with $d_i = 5.0E5$ Ns/m. Again, the order control algorithm works well for both co-simulation methods.

Figure 21 exhibits the computation time, the number of macro-steps and the global error of the explicit and implicit co-simulation approach as a function of the damping coefficient $d_i$. It can clearly be observed that the implicit co-simulation approach becomes more efficient for increasing values of $d_i$. The explicit co-simulation is faster for $d_i \leq 1.0E5$ Ns/m (Fig. 21(a)), although smaller macro-step sizes are needed in connection with the explicit approach (Fig. 21(b)). Note that the implicit approach requires a computationally expensive corrector iteration. As a consequence, the implicit approach will only be more efficient than the explicit approach, if the macro-step sizes for implicit method are significantly larger than for the explicit method. The resulting global error of the state variables of the coupling bod-
Fig. 21  Comparison between the explicit and the implicit co-simulation approach: computation time, number of macro steps, and global error as functions of the linear damping parameter $d_i$. 

ies is depicted in Fig. 21(c). As can be detected, the error of the implicit approach is usually little smaller, although both methods have been simulated with the same error tolerances for the macro-step size controller.

6.2.2 Case study 2

In a second numerical study, the performance of the explicit approach is compared with the performance of the implicit co-simulation method for different values of the stiffness coefficient $c_i$, see Fig. 22. The simulations have been carried out with the same parameters that have been used in the first case study. The damping coefficient has been set to $d_i = 1.0E4$ Ns/m. As can be seen, the explicit approach is always faster than the implicit method.

6.2.3 Case study 3

Within the third study, the explicit and implicit co-simulation methods are compared for different values of $m_i$, see Fig. 23. The same parameters have been used as in the first case study. The damping and stiffness coefficients have been set to $d_i = 1.0E4$ Ns/m and $c_i = 1.0E7$ N/m. The plots clearly indicate that the implicit approach is more efficient than the explicit method for small values of $m_i$ ($m_i \leq 8.0E - 3$ kg).
Co-simulation with variable approximation order

Fig. 23 Comparison of the explicit and the implicit co-simulation approach: computation time, number of macro steps and global error as a function of the mass \( m_i \)

Fig. 24 ExLocal approach: number of macro-steps and global error of the state variables of the coupling bodies for different error tolerances and different approximation orders

6.3 Example 3: comparative study between co-simulations with constant and variable integration order

Next, the performance of co-simulations including the order controller \((k = var.)\) is compared with the performance of co-simulations using a fixed polynomial degree for the approximation polynomials \((k = const.)\). The chain model of Example 2 is considered again and basically the same parameters are used. In Sect. 6.3, all explicit co-simulations are carried out with \( d_i = 1.0E4 \frac{N_s}{m} \) and all implicit co-simulations with \( d_i = 5.0E5 \frac{N_s}{m} \). The following macro-step size controller parameters have been used: \( SF = 6, \ atol_c = 10^{-3} \cdot rtol, \ atol_e = rtol, \ atol_h = 10^2 \cdot rtol, r^p_{\min} = 0.5, r^p_{\max} = 2.0 \). In the subsequent simulations, \( rtol \) is varied in order to show the influence of \( rtol \) on the global error and on the number of macro-steps.

Simulation results with the five different error estimators of Sect. 4 are depicted in Figs. 24–28. The plots show the total number of macro-steps and the resulting global error of the state variables of the coupling bodies. The following observations can be made:

- The order controller reduces the number of macro-steps for all considered co-simulation approaches and for all chosen error tolerances. An exception is only observed for the
ImMilne error estimator, where simulations with a fixed polynomial degree of $k = 1$ and $k = 2$ show a slightly better performance for certain error tolerances $rtol$.

- The simplified error estimators ($ExSLocal$ and $ImSLocal$) do not perform very well in combination with constant approximation polynomials ($k = 0$), because of the rather large discontinuities in the coupling variables at the macro-time points. Note that the corresponding curves are missing in the plots of Figs. 26 and 28.

- Considering the plots (a) of Figs. 24, 25, 26, it can be seen that for $k \geq 2$ the number of macro-steps is almost constant and not influenced by the error tolerance. Usually, one would expect that the number of macro-steps will decrease with increasing $rtol$. This unexpected behavior might be traced back to a stability problem [60]. Most likely, the co-simulations for $k \geq 2$ will get unstable for larger macro-step sizes. Therefore, the macro-step size controller reduces the macro-step size, until the system reaches a numerically stable region.

- Regarding the plots a) of Fig. 27 and Fig. 28, it can be observed that for $k \geq 3$ the number of macro-steps will not be reduced if the error tolerance $rtol$ is increased. This unexpected behavior might also be explained by the fact that the stability limit of the co-simulation is reached for $k \geq 3$. 
Co-simulation with variable approximation order

The resulting computation time (wall-clock time) of the considered co-simulation approaches are depicted in Figs. 29–33. The overall computation time of the fully parallelized co-simulation basically consists of four parts:

i) Subsystem integration time (the slowest subsystem integration process is relevant);

ii) Additional subsystem computations resulting from the co-simulation (save/reload of subsystem workspace in connection with macro-step repetitions);

iii) Special computations of the co-simulation interface, specific for the different co-simulation methods (e.g., macro-step size controller, computation of corrected coupling variables);

iv) Synchronization and data exchange between the subsystems and the co-simulation interface (MPI data traffic).

The four parts of the overall computation time are represented with different colors in Figs. 29–33.
Fig. 29 *ExLocal* approach: computation time for different error tolerances and different approximation orders

Fig. 30 *ExMilne* approach: computation time for different error tolerances and different approximation orders

Fig. 31 *ExSLocal* approach: computation time for different error tolerances and different approximation orders
6.4 Example 4: comparative study between finite difference gradients and low-order sensitivities

Next, the system size is increased by adding further masses to the chain. The model is decomposed into $r = 2399$ subsystems and simulated by the fully controlled, i.e., order and macro-step size controlled, $ExSLocal$ and $ImSLocal$ co-simulation approaches. The subsequent macro-step size controller parameters have been used: $SF = 6$, $atol_{i,c} = 10^{-1}$, $rtol_{j,c} = 10^{-3}$, $\rho_{\min} = 0.5$, $\rho_{\max} = 2$. Due to the increased number of subsystems, the number of coupling variables will also increase.

Basically, the interface-Jacobian required for implicit co-simulation approaches, see Eq. (11), can always be calculated numerically using a finite difference approach (FD), see Appendix A.1. The additional subsystem integrations with perturbed coupling variables, which are required for the finite difference approach, can be executed in parallel with the main subsystem integration (predictor/corrector integration). Due to the parallel integration, the overall computation time of the co-simulation will therefore not be (markedly) in-
increased. However, additional cores are required for carrying out the subsystem integrations with the perturbed approximation polynomials. The larger the number of coupling variables becomes, the larger the number of additional cores will be. Hence, for models with a large number of coupling variables, the FD approach for calculation the interface-Jacobian will get problematic or even impossible. In this case, the gradients can very easily be approximated by making use of a low-order approximation approach (low-order sensitivities, LOS), see [29]. For the considered model, for instance, $3 \cdot r - 2 + 1 = 7196$ cores are required in total if an FD approach is used to calculate the interface-Jacobian. With the low-order approximation approach, only $r + 1 = 2400$ cores are needed, which equals the number of required cores for a fully parallelized explicit co-simulation approach (ExSLocal). A description of the FD approach and a short summary of the LOS approach are collected in Appendix A.

In Example 4, the interface-Jacobian required for the implicit co-simulation approach (ImSLocal) is therefore calculated with the LOS approach. It should be mentioned that the interface-Jacobian for the examples in Sects. 6.1–6.3 have been calculated with the FD approach. For the example in Sect. 6.3, simulation results generated with the LOS technique are arranged in Appendix B so that implementations with FD and LOS may directly be compared.

Co-simulation results using the chain model of Sect. 6.2 with $n_K = \{5E4, 1E5, 5E5, 1E6\}$ masses are depicted in Fig. 34. The simulations have shown that the average macro-step size is slightly increased compared to the case $n_K = 1E4$ of Sect. 6.2. As a consequence, the computational overhead caused by the co-simulation interface (green) and also the overhead caused by the data traffic between the co-simulation interface and the subsystems (yellow) is slightly decreased compared to the case $n_K = 1E4$ in Sect. 6.2. The solver time per subsystem (blue) is increased, because the number of masses per subsystem is increased. Regarding the efficiency of the implementation, it can clearly be seen from Fig. 34 that a decomposition of the model into 2399 subsystems is inefficient for a model size with $n_K = 5E4$ and $n_K = 1E5$ masses, since the overhead is rather large compared to the overall computation time. If the number of masses increases, the overhead becomes lower compared to the subsystem integration time. This clearly indicates that a parallelization of a dynamical model by means of a co-simulation technique can very significantly reduce the overall simulation time. Increasing the number of masses even more ($n_K > 1E6$), the improvement would even be more apparent.

Fig. 34 ExSLocal approach (for $d_1 = 1.0E4$ Ns/m) and ImSLocal approach (for $d_1 = 5.0E5$ Ns/m) in connection with low-order sensitivities: number of macro-steps and computation time for different model sizes.

\[ \text{a) } \text{ExSLocal} \quad \text{b) } \text{ImSLocal (LOS-gradients)} \]

The simulations have shown that the average macro-step size is slightly increased compared to the case $n_K = 1E4$ of Sect. 6.2. As a consequence, the computational overhead caused by the co-simulation interface (green) and also the overhead caused by the data traffic between the co-simulation interface and the subsystems (yellow) is slightly decreased compared to the case $n_K = 1E4$ in Sect. 6.2. The solver time per subsystem (blue) is increased, because the number of masses per subsystem is increased. Regarding the efficiency of the implementation, it can clearly be seen from Fig. 34 that a decomposition of the model into 2399 subsystems is inefficient for a model size with $n_K = 5E4$ and $n_K = 1E5$ masses, since the overhead is rather large compared to the overall computation time. If the number of masses increases, the overhead becomes lower compared to the subsystem integration time. This clearly indicates that a parallelization of a dynamical model by means of a co-simulation technique can very significantly reduce the overall simulation time. Increasing the number of masses even more ($n_K > 1E6$), the improvement would even be more apparent.

\[ \text{c) } \text{ExSLocal} \quad \text{d) } \text{ImSLocal (LOS-gradients)} \]
6.5 Example 5: Study on the safety factor

The safety factor $SF$, see Eq. (32), is a parameter to tune the ratio between the total number of macro-steps and the number of failed macro-steps (failed local error tests). Again, the chain model of Sect. 6.2 is considered with the following parameters: $n_K = 10^5$ masses, $r = 100$ subsystems, $m_i = 5.0E-2$ kg, $c_i = 2.5E6$ N/m, $d_i = 2.5E1$ N/m, $C_i = 1.0E10$ N/m$^2$, $D_i = 1.0E-4$ N/m$^3$. External forces $F_{S,j} = \Delta F_{S,j} \left( \sin \Omega_{S,j} \right)^A_{s}$ with the amplitudes $\Delta F_{S,j}$ (randomly chosen in the interval $[-1.0E6, 1.0E7]$ N) and the frequencies $\Omega_{S,j}$ (randomly distributed in the interval $[5.0E2, 5.0E3]$ s$^{-1}$) and the parameter $A_s = 95$ are applied at 50% of the masses (randomly selected).

The model is computed by the explicit and the implicit co-simulation approach with quadratic approximation polynomials ($k = 2$). The macro-step size is controlled by the error estimators $ExMilne$ and $ImMilne$. The limits for the scaling factor $r^\rho$ between two consecutive macro-steps are chosen by $r^\rho_{\text{min}} = 0.5$, $r^\rho_{\text{max}} = 1.5$. The error tolerances for the macro-step size controller are set to $atol_x = 10^{-8}$, $atol_v = 10^{-4}$, and $rtol = 10^{-4}$. The resulting computation time as a function of the safety factor $SF$ is illustrated in Fig. 35.

For the chosen model parameters, the explicit co-simulation approach is more efficient than the implicit method: the $ExMilne$ approach (left y-axis, blue bars) is almost twice as fast as the $ImMilne$ method (right y-axis, red bars). The computation time of both co-simulation approaches can be reduced significantly by selecting an appropriate value for the safety factor $SF$. Choosing the safety factor in the range $SF \in [4,6]$ instead of using $SF = 1$, the computation time can approximately be reduced by 30%. The plot further indicates that the simulation time only slightly changes if the safety factor is varied in the range $SF \in [3,8]$. It should be emphasized that these findings are model dependent. Further studies, not presented here, indicate that a safety factor in the range $SF \in [2,6]$ might generally be an appropriate choice [28].

6.6 Example 6: Diverse case studies

6.6.1 Case study 1: monolithic simulation versus co-simulations with fixed/variable macro-step size

To compare the computation time of a co-simulation with the simulation time of the corresponding monolithic model, the oscillator chain of Sect. 6.2 is reconsidered. Therefore, the chain model with $n_K = 10^5$ masses is split into $r = 100$ equally-sized subsystems. The model parameters are: $m_i = 5.0E-5$ N/m$^2$, $c_i = 2.5E6$ N/m, $d_i = 2.5E-1$ N/m$^2$, $C_i = 1.0E10$ N/m$^3$, $D_i = 5.0E-5$ N/m$^3$. Impulse-shaped forces $F_{Imp,j} = \frac{1}{2} \Delta F_j \left[ \tan \left( \frac{t-t_0}{\delta} \right) - \tan \left( \frac{t-t_0+\Delta t}{\delta} \right) \right]$ with an amplitude of $\Delta F_j = (-1)^{j+1} \cdot 1.0E6$ N and the parameters
been chosen in such a way that the accuracy condition $\delta_2 = 0.01$ is computed. To obtain a reasonable comparison, the simulation parameters have been used: $SF = 6$,
\[\text{atol}_i = 10^{-3} \cdot \text{rtol}, \quad \text{atol}_c = \text{rtol}, \quad \text{atol}_b = 10^2 \cdot \text{rtol}, \quad r_{\min}^p = 0.5, \quad r_{\max}^p = 2.0.\]

The model is computed with the explicit and implicit co-simulation approaches, whereby three different cases are considered:

a) fixed order ($k = \text{const.}$) and fixed macro-step size ($H = \text{const.}$),
b) fixed order ($k = \text{const.}$) and variable macro-step size ($H = \text{var.}$),
c) variable order ($k = \text{var.}$) and variable macro-step size ($H = \text{var.}$).

The simulation results are compared with the results of a reference simulation (monolithic simulation with very tight error tolerances). In order to compare the simulation results, the global error $e^{\text{glob}}$ (normalized root mean square error) of the state variables of the coupling bodies is computed. To obtain a reasonable comparison, the simulation parameters have been chosen in such a way that the accuracy condition $e^{\text{glob}} \leq 2.50E - 3$ will be satisfied.

To achieve a global error of $e^{\text{glob}}_{\text{mono}} \approx 2.34E - 3$, a simulation time of $2.69E4$ s is required for the monolithic model (mean solver-step size: $\bar{h} = 1.34E - 6$ s).

**Case a):** $k = \text{const.}, \quad H = \text{const.}$

- To obtain results of the required accuracy with the implicit co-simulation method, a simulation with $k = 2$ and $H = 1.0E - 7$ s ($\text{Im2H}$) exhibits the best performance. With these parameters, the implicit co-simulation takes $2.0E4$ s with a global error of $e^{\text{impl}} \approx 1.07E - 3$.
- Using the explicit co-simulation approach, best results are achieved with the parameters $k = 1$ and $H = 2.5E - 8$ s ($\text{Ex1H}$). The computation time with the explicit approach is $1.89E4$ s yielding a global error of $e^{\text{expl}} \approx 7.94E - 4$.
- Remark. A comparison between the monolithic simulation ($\text{Mono}$) and the explicit co-simulation ($\text{Ex1H}$) shows that the simulation time is only slightly reduced ($\approx 30\%$) due to the co-simulation. The poor performance of the co-simulation can be explained as follows. The monolithic model contains a few impulse-shaped external forces, which require a small macro-step size in order to obtain a stable co-simulation. The monolithic model is, however, solved with a variable step-size solver so that the integration step-size is only reduced at the time points, where the impulse-shaped forces act. Hence, the mean integration step-size of the monolithic model is significantly larger than the mean integration step-size of the subsystem solvers. As a consequence, the formula in the introduction for estimating the possible speed-up by a co-simulation cannot be applied.

**Case b):** $k = \text{const.}, \quad H = \text{var.}$

- The best performance for the implicit approach has been achieved with $k = 5$ and $\text{rtol} = 10^{-4}$ ($\text{Im5Milne}$). With these parameters, the computation time is $3925$ s, the average macro-step size is $\bar{h} = 2.99E - 6$ s and the global error becomes $e^{\text{impl}} \approx 1.86E - 3$.
- The explicit approach performs best for $k = 4$ and $\text{rtol} = 10^{-6}$ (computation time $1014$ s, average macro-step size $\bar{h} = 9.594E - 7$ s, global error $e^{\text{expl}} \approx 3.95E - 5$).
- Remark. The formula for estimating the co-simulation time will now provide a more realistic estimation, since both the monolithic simulation and the co-simulation have similar
Co-simulation with variable approximation order

integration step-sizes. The real speed-up factor is \( \approx 26.5 \); the estimation formula provides \( T_{\text{Mono}} = 100 \cdot (T_{\text{Cosim}} - \text{Overhead}) \). Note that the scaling factor \( r \) is approximately 1 for the considered model.

**Case c):** \( k = \text{var.}, \, H = \text{var.} \)

- The fully controlled implicit co-simulation \( \text{ImMilne} \) with \( rtol = 10^{-4} \) takes 4154 s to obtain a resulting error of \( e_{\text{impl}}^{\text{glob}} \approx 9.00E - 4 \) (average macro-step size \( H = 3.03E - 6 \) s).
- Applying the simplified error estimator in combination with the implicit co-simulation method (\( \text{ImSLocal} \)) with \( rtol = 10^{-4} \), the computation is 3356 s with a global error of \( e_{\text{impl}}^{\text{glob}} \approx 9.36E - 4 \) (average macro-step size \( H = 5.08E - 6 \) s).
- The fully controlled explicit co-simulation \( \text{ExMilne} \) shows that the computation time decreases if the macro-step size is decreased. Best results with regard to the computation time are achieved with the rather tight error tolerances \( rtol = 10^{-6} \) (computation time 1187 s, global error \( e_{\text{expl}}^{\text{glob}} \approx 6.81E - 5 \), average macro-step size \( H = 1.10E - 6 \) s).
- For the fully controlled explicit co-simulation \( \text{ExLocal} \) with \( rtol = 10^{-6} \), similar results are obtained (computation time 1207 s, global error \( e_{\text{expl}}^{\text{glob}} \approx 7.49E - 5 \), average macro-step size \( H = 1.10E - 6 \) s).
- The fully controlled explicit co-simulation in combination with the simplified error estimator (\( \text{ExSLocal} \)) and an error tolerance of \( rtol = 10^{-6} \) takes 1265 s to obtain a global error of \( e_{\text{impl}}^{\text{glob}} \approx 3.43E - 4 \). The average macro-step size is \( H = 1.68E - 6 \) s.

The resulting computation times of the different approaches are summarized in Fig. 36.

Based on the above results, the following conclusions can be drawn:

- Co-simulations with a fixed macro-step size are rather inefficient for the considered model. Compared to the monolithic simulation, the computation time is only reduced by \( \approx 30\% \).
- Applying the macro-step size controller, efficiency of all co-simulation approaches may significantly be increased. A speed-up factor of \( \approx 26.5 \) compared to the monolithic computation may be achieved.
- For the considered model, the order control algorithm does not significantly reduce the computation time since the impulse-shaped forces only act at very few time points. At the time points, where the impulse-shaped forces act, the approximation order is – as expected – reduced. Between these time points, the order controller selects a larger integration order.
(\(k = 3 - 5\)). Therefore, the benefit of the order controller is rather low for the considered problem.

- The three error estimators for the explicit co-simulation approach exhibit a similar behavior.
- The two error estimators for the implicit co-simulation method yield comparable results.

### 6.6.2 Case study 2: Modified decomposition

The decomposition of the chain model used above may not be ideal because the coupling masses are directly affected by impulse-shaped forces. To reduce the computation time, it might be useful to apply a modified decomposition. Therefore, the chain model is partitioned into 101 subsystems in such a way that the first and the last subsystem contain 500 masses; the remaining 99 subsystems are assumed to be equally sized containing 1000 masses. The influence of the increased number of subsystems (101 instead of 100) on the simulation time is negligible. The essential point is that the impulse-shaped forces will not longer act directly at the coupling bodies, but inside the subsystems 2-100.

The results of the computations with the modified decomposition are collected in Fig. 37 (red bars). A comparison with the original decomposition of Sect. 6.6.1 shows that the computation time is (slightly) reduced with the modified decomposition. For the \(ExSLocal\) approach, for instance, the computation time is reduced by 14%.

### 6.6.3 Case study 3: Extended model and multirate effect

Finally, the oscillator chain is modified to demonstrate the multirate effect. Small subsystems with 25 masses are added at both ends of the oscillator chain. In the newly added small subsystems (subsystems 1 and 103), contact forces \(F_{Con,i} = A e^{Bxi}\) with \(A = -1.0E - 2\) N and \(B = 5.0E5\) mm\(^{-1}\) are assumed to act at three masses so that these two subsystems become very stiff. Subsystems 2-102 are identical with the model of Sect. 6.6.2 (modified decomposition).

Considering the monolithic simulation of the extended model, the stiff contacts strongly affect the computation time, since the solver-step size is reduced for the complete model.
However, applying a co-simulation approach, the integration-step size is selected individually by the step-size controllers of the subsystem solvers. Therefore, the solver-step sizes of subsystems 1 and 103 will be reduced due to the stiff contacts. The remaining subsystems 2-102 will be integrated with larger step sizes. Since the subsystems 1 and 103 are very small, the overall computation time of the co-simulation will not be increased significantly compared to the non-extended model.

Simulation results with the extended model (containing contacts in subsystems 1 and 103) are shown in Fig. 37 (yellow bars). As expected, the computation time of the monolithic simulation is significantly increased, while the computation time of the co-simulation approaches are not changed markedly. The number of macro-steps and the resulting global error are only slightly increased.

The speed-up factor of the explicit co-simulation ($ExMilne$) compared to the monolithic simulation is $\approx 400$ for the extended model with contacts (simulation time of the monolithic model: $5 \frac{1}{2}$ days; simulation time of the co-simulation model: 30 minutes). This very high speed-up factor has mainly two reasons: (i) dynamical parallelization of the model by means of the co-simulation and (ii) exploiting the multirate effect by an appropriate model decomposition.

7 Conclusions

An algorithm for controlling the degree of the polynomials, which are used for approximating the coupling variables within co-simulation and solver coupling approaches, has been presented. The order controller has been incorporated into the algorithm for controlling the macro-step size so that co-simulations with variable order and variable macro-step size could be realized. Different numerical examples have been presented, which confirm the functionality and the accurate operation of the combined order and step-size control algorithm. The numerical tests also clearly demonstrated the benefit, which can be achieved with a combined order and macro-step size control algorithm.

Using a constant (i.e., fixed) integration order, the question on the proper choice of the integration order arises. As a matter of fact, the optimal integration order is problem-dependent and may change during the co-simulation. Using a constant integration order can cause different difficulties:

- If the integration order is chosen too high, the co-simulation may get unstable or the macro-step size may be reduced significantly by the macro-step size controller.
- Choosing the integration order too small, the co-simulation may become inefficient and less accurate.
- Especially for highly nonlinear systems, for systems with impacts or for models where steep gradients occur, a simulation with a fixed integration order may be rather inefficient.

The main problems associated with the usage of a constant integration order – namely a trade-off between stability/accuracy and efficiency – will disappear, if an order controller is used.

The different numerical examples presented in Sect. 6 have shown that in most cases the efficiency and accuracy of a co-simulation could be (markedly) improved by incorporating an order control algorithm.

In our tests, failures of the order controller have not been detected, although rather stiff systems, systems with contacts and systems with steep, impulse-shaped external forces have
been considered. A rather stable and robust performance has been observed, which clearly recommends the usage of a variable integration order.

Considering implicit co-simulation methods, an interface-Jacobian is required for the corrector iteration, which is usually calculated by a finite difference (FD) approach. Applying an FD approach, a large number of additional cores might be necessary in order to calculate the interface-Jacobian in parallel with the predictor/corrector subsystem integrations. Alternatively, approximation formulas for the interface-Jacobian might be used (low-order sensitivities, LOS), which can be implemented in a very straightforward manner. Using the LOS approach, the additional parallel subsystem integrations required in connection with the FD approach can be omitted without a significant increase of the computation time. Depending on the problem, the usage of low-order sensitivities can (slightly) increase the co-simulation time, since the number of corrector iterations may (slightly) increase compared to the FD approach. Accuracy of the co-simulation is, however, not affected. In contrast, problems with ill-conditioning and numerical cancellation, which are often observed in association with FD approaches, will not occur if low-order sensitivities are applied.

Summarizing, the innovation of this work concerns the realization of variable-order co-simulations and the incorporation of the order control algorithm into the macro-step size controller. Using a combined order and macro-step size controller may significantly increase the numerical efficiency and accuracy of a coupled simulation. While co-simulation is frequently used to carry out multi-physical or multi-disciplinary simulations, co-simulation methods may also be applied to parallelize a mono-disciplinary system in time domain. To exploit the full potential of a co-simulation-based parallelized dynamical model, controlling the integration order and the macro-step size may be an essential point.

Concerning the future work, the subsequent topics may – amongst others – be interesting:

- Application of the presented approaches to co-simulation methods based on constraint coupling.
- Usage of co-simulation techniques to parallelize large-scaled finite element models in time domain.
- Development of error estimators for alternative explicit and implicit co-simulation methods.
- Error estimation and macro-step size control for systems with discontinuities.

Appendix A

This appendix contains additional information on the calculation of the interface-Jacobian, which is required for the implicit co-simulation scheme, see Sect. 3.2. Basically, the interface-Jacobian $\frac{dg}{du_{N+1}}$ can be computed in different ways, e.g., by using:

- analytical gradients,
- numerically calculated gradients based on a finite difference approach,
- approximated gradients,
- gradients computed with the help of automatic differentiation.

Here, the gradients are calculated either by a finite difference approach (Appendix A.1) or with the help of a low-order approximation technique (Appendix A.2).

To explain the two approaches, we consider the equations of motion of an arbitrary subsystem $s$. For a concise representation, the subsystem is assumed to be a multibody system.
For other mechanical or nonmechanical subsystems, a very similar strategy may be applied. The equations of motion of the multibody subsystem $s$ are given by

\[
\dot{x}^s = B^s(x^s)v^s, \\
M^s(t,x^s)\ddot{v}^s = f^s(t,x^s,v^s) - B^{sT}(x^s)G^{sT}(t,x^s)\lambda^s
\]

\[
+ f^{c,s}(t,x^s,v^s,p_{N+1}^s(t)),
\]

\[
g^s(t,x^s) = 0
\]

with \( \{x^s,v^s\} \in \mathbb{R}^n, \lambda^s \in \mathbb{R}^m \), see Sect. 2.2. To keep the notation simple, the subsystem index $s$ is omitted in the following (i.e., $x^s \rightarrow x$, etc.). The coupling variables are approximated with the polynomials $p_{N+1}^s(t) = p_{N+1}^s(t; (T_{N+1}, u_{N+1}^s), (T_N, u_N), \ldots, (T_{N-k+1}, u_{N-k+1})) = \sum_{i=N-k+1}^N u_i L_i(t) + u_{N+1}^s L_{N+1}(t)$, where $L_i(t)$ represent the Lagrange basis polynomials.

The two approaches for calculating the interface-Jacobian are explained by regarding the $\alpha$th corrector step, i.e., the calculation of the interface-Jacobian $\frac{dg_c}{du_{N+1}}|_{u_{N+1}^{cor,a-1}}$ is considered. To compute the interface-Jacobian, it is useful to rewrite the term $\frac{dg_c}{du_{N+1}}$. The total derivative of $g_c(T_{N+1}, z_{N+1}^s(u_{N+1}^s), u_{N+1}^s) = u_{N+1}^s - \varphi(T_{N+1}, z_{N+1}^s(u_{N+1}^s), u_{N+1}^s)$ with respect to $u_{N+1}^s$ can be written as

\[
\frac{dg_c}{du_{N+1}}|_{u_{N+1}^{cor,a-1}} = \left( \frac{\partial g_c}{\partial u_{N+1}^s} + \frac{\partial g_c}{\partial z_{N+1}^s} \cdot \frac{\partial z_{N+1}^s}{\partial u_{N+1}^s} \right)|_{u_{N+1}^{cor,a-1}}
\]

\[
= \left( I - \frac{\partial \varphi}{\partial u_{N+1}^s} \cdot \frac{\partial \varphi}{\partial z_{N+1}^s} \cdot \frac{\partial z_{N+1}^s}{\partial u_{N+1}^s} \right)|_{u_{N+1}^{cor,a-1}}
\]

where $I$ is the identity matrix. Since the coupling functions $g_c := u - \varphi(t,z,u) = 0$ or, more precisely, the functions $g_c := u - \varphi(t,x,v,u) = 0$ are assumed to be explicitly known, the terms $\frac{\partial \varphi}{\partial u_{N+1}^s}$ and $\frac{\partial \varphi}{\partial z_{N+1}^s}$ can be expressed analytically. Note that for systems without feed-through, the coupling equations simply read $g_c := u - \varphi(t,x,v) = 0$ so that the term $\frac{\partial \varphi}{\partial z_{N+1}^s}$ equals zero. The main task to compute the interface-Jacobian is therefore the calculation of the term $\frac{\partial z_{N+1}^s}{\partial u_{N+1}^s}|_{u_{N+1}^{cor,a-1}}$. More precisely, the two expressions

\[
\frac{\partial x_{N+1}^s}{\partial u_{N+1}^s}|_{u_{N+1}^{cor,a-1}} \quad \text{and} \quad \frac{\partial v_{N+1}^s}{\partial u_{N+1}^s}|_{u_{N+1}^{cor,a-1}},
\]

which describe the sensitivities of the subsystem states with respect to the coupling variables, have to be determined.

### A.1 Numerical calculation of the interface-Jacobian using a finite difference approach (FD sensitivities/gradiente)

To numerically calculate the partial derivatives of Eq. (41) with the help of a finite difference approximation, additional subsystem integrations have to be carried out in parallel with the
The interface-Jacobians \( \frac{\partial x_{N+1}}{\partial u_{i,N+1}} \) and \( \frac{\partial x_{N+1}}{\partial u_{i,N+1}} \) represent the correction variables \( z_{i,N+1} \) and \( z_{i,N+1} \), where \( u_{i,N+1}^* = u_{N+1}^* \cdot e_i \) is an arbitrary input variable.

The \((\alpha - 1)\)th corrector step is accomplished with the interpolation polynomials

\[
P_{N+1}^{\text{cor},\alpha-1}(t) = P_{N+1}^k \left( t; \left( T_{N+1}, u_{N+1}^{\text{cor},\alpha-1}, (T_N, u_N), \ldots, (T_{N-k+1}, u_{N-k+1}) \right) \right),
\]

which yields the corrected variables \( z_{N+1}^{\text{cor},\alpha-1} \). To calculate the partial derivatives \( \frac{\partial x_{N+1}}{\partial u_{i,N+1}} \) and \( \frac{\partial x_{N+1}}{\partial u_{i,N+1}} \), additional subsystem integrations have to be carried out with perturbed interpolation polynomials. The perturbed corrector polynomials are given by

\[
P_{N+1}^{\text{cor},\alpha-1,\text{per}}(t) = P_{N+1}^k \left( t; \left( T_{N+1}, u_{N+1}^{\text{cor},\alpha-1} + \Delta u_{i,N+1}^* \right), (T_N, u_N), \ldots, (T_{N-k+1}, u_{N-k+1}) \right),
\]

where \( \Delta u_{i,N+1}^* \) denotes a small user-defined increment. Integrating the subsystem with the perturbed interpolation polynomials \( p_{N+1}^{\text{cor},\alpha-1,\text{per}}(t) \) provides the perturbed subsystem variables \( z_{N+1}^{\text{cor},\alpha-1,\text{per}} \).

The gradients may then simply be approximated by

\[
\begin{align*}
\frac{\partial x_{N+1}^*}{\partial u_{i,N+1}^*} & \approx \frac{x_{i,N+1}^{\text{cor},\alpha-1,\text{per}} - x_{i,N+1}^{\text{cor},\alpha-1}}{\Delta u_{i,N+1}^*}, \\
\frac{\partial v_{N+1}^*}{\partial u_{i,N+1}^*} & \approx \frac{\psi_{i,N+1}^{\text{cor},\alpha-1,\text{per}} - \psi_{i,N+1}^{\text{cor},\alpha-1}}{\Delta u_{i,N+1}^*}.
\end{align*}
\]

The interface-Jacobians \( \frac{\partial g}{\partial u_{i,N+1}} \) for the next corrector step is calculated in the same way.

The perturbation increment \( \Delta u_{i,N+1}^* \) has to be chosen very carefully. For the simulations in Sect. 6, the subsequent perturbation increments have been used

\[
\begin{align*}
\Delta u_{i,N+1}^* &= \max \left( |u_{i,N}^{\text{pre}} - u_{i,N}^{\text{cor},\alpha}|, \Delta u_{i,N+1}^\text{min} \right) \quad \text{for the predictor step}, \\
\Delta u_{i,N+1}^* &= \max \left( |u_{i,N+1}^{\text{cor},\alpha} - u_{i,N+1}^{\text{cor},\alpha-1}|, \Delta u_{i,N+1}^\text{min} \right) \quad \text{for the } \alpha \text{th corrector step},
\end{align*}
\]

where \( \Delta u_{i,N+1}^\text{min} \) is a user-defined (small) value; \( \Delta u_{i,N+1}^\text{min} \) has to be chosen appropriately with respect to the type of the coupling variable (position, velocity or force variable).

The corrector iteration is terminated if a convergence criterion is fulfilled. For this purpose, the scalar weighted root-mean-square norm

\[
\| u_{i,N+1}^{\text{cor},\alpha} - u_{i,N+1}^{\text{cor},\alpha-1} \|_{\text{WRMS}} = \sqrt{\frac{1}{n_c} \sum_{i=1}^{n_c} \left( \frac{u_{i,N+1}^{\text{cor},\alpha} - u_{i,N+1}^{\text{cor},\alpha-1}}{\text{atol}_i + \text{rtol} \cdot |u_{i,N+1}^{\text{cor},\alpha}|} \right)^2}
\]

is applied; \( \text{atol}_i \) and \( \text{rtol} \) term user-defined absolute and relative error tolerances, respectively. The corrector iteration is terminated if the following criterion is fulfilled:

\[
\| u_{i,N+1}^{\text{cor},\alpha} - u_{i,N+1}^{\text{cor},\alpha-1} \|_{\text{WRMS}} \cdot \frac{R_a}{1 - R_a} < \tau.
\]

The factor \( R_a = \| u_{i,N+1}^{\text{cor},\alpha} - u_{i,N+1}^{\text{cor},\alpha-2} \|_{\text{WRMS}} \) is a measure for the convergence rate [66] and \( \tau \) denotes a user-defined value (e.g., \( \tau = 0.3 \)).
A.2 Approximate calculation of the interface-Jacobian using low-order sensitivities

A straightforward approach for approximately calculating the coupling sensitivities \( \frac{\partial^2 \mathbf{x}_{N+1}}{\partial \mathbf{u}_{N+1}} = \mathbf{f}(\mathbf{u}_{N+1}) \) and \( \frac{\partial^2 \mathbf{v}_{N+1}}{\partial \mathbf{u}_{N+1}} = \mathbf{g}(\mathbf{u}_{N+1}) \) has been presented in [29]. These approximations, called low-order sensitivities (LOS) in the following, converge to the exact sensitivities for \( H \to 0 \). Concretely, the low-order sensitivities for the velocity variables converge with \( O(H^1) \) to the exact sensitivities and the low-order sensitivities for the position variables with \( O(H^2) \). Calculation of low-order sensitivities is rather straightforward and basically only requires the solution of a system of linear equations of the dimension of the subsystem. If the subsystems are ODEs – or, more precisely, if the coupling bodies are not connected by constraint equations to the rest of the subsystem – even closed analytical formulas may be derived for the low-order sensitivities.

Here, we just explain the application of the low-order sensitivity approach. For details, the reader is referred to [29]. In a first step, low-order sensitivities for the subsystem accelerations \( \dot{\mathbf{v}} \) (i.e., \( \dot{\mathbf{v}}^i \)) and for the subsystem multipliers \( \lambda \) (i.e., \( \lambda^i \)) are computed for the time-point \( T_{N+1} \). Therefore, the linear system of equations

\[
\begin{pmatrix}
\mathbf{M}(T_{N+1}, \mathbf{x}^{cor,a-1}(T_{N+1})) & \mathbf{B}^T(T_{N+1}, \mathbf{x}^{cor,a-1}(T_{N+1})) \\
\mathbf{G}(T_{N+1}, \mathbf{x}^{cor,a-1}(T_{N+1})) & 0
\end{pmatrix} \begin{pmatrix}
\frac{\partial \dot{\mathbf{v}}}{\partial \mathbf{u}_{N+1}}^{cor,a-1}(T_{N+1}) \\
\frac{\partial \lambda}{\partial \mathbf{u}_{N+1}}^{cor,a-1}(T_{N+1})
\end{pmatrix} = \begin{pmatrix}
\frac{\partial \mathbf{f}}{\partial \mathbf{p}_{N+1}}^{cor,a-1}(T_{N+1}), \mathbf{p}_{N+1}^{cor,a-1}(T_{N+1}) \\
0
\end{pmatrix} \cdot \mathbf{e}_i
\]

has to be solved for the expressions \( \frac{\partial \dot{\mathbf{v}}}{\partial \mathbf{u}_{N+1}}^{cor,a-1}(T_{N+1}) \) and \( \frac{\partial \lambda}{\partial \mathbf{u}_{N+1}}^{cor,a-1}(T_{N+1}) \). Note that \( \mathbf{M}(T_{N+1}, \mathbf{x}^{cor,a-1}(T_{N+1})) \), \( \mathbf{B}(\mathbf{x}^{cor,a-1}(T_{N+1})) \) and \( \mathbf{G}(T_{N+1}, \mathbf{x}^{cor,a}(T_{N+1})) \) denote subsystem matrices according to Eq. (39) to be evaluated at the macro-time point \( T_{N+1} \) with the known variables \( \mathbf{x}^{cor,a-1}(T_{N+1}) \) from the \( (\alpha - 1) \)th corrector step. The matrix \( \frac{\partial \mathbf{f}}{\partial \mathbf{p}_{N+1}}^{cor,a-1}(t), \mathbf{p}_{N+1}^{cor,a-1}(t) \) can simply be calculated with the help of the constitutive equations of the coupling forces/torques and the known variables \( \mathbf{x}^{cor,a-1}(T_{N+1}) \). Hence, after the \( (\alpha - 1) \)th corrector step has been carried out, \( \mathbf{A} \) and \( \mathbf{b} \) can be computed and Eq. (46) can be solved for the required sensitivities, i.e.,

\[
\begin{pmatrix}
\frac{\partial \dot{\mathbf{v}}}{\partial \mathbf{u}_{N+1}}^{cor,a-1}(T_{N+1}) \\
\frac{\partial \lambda}{\partial \mathbf{u}_{N+1}}^{cor,a-1}(T_{N+1})
\end{pmatrix} = \mathbf{A}^{-1} \cdot \mathbf{b}.
\]

With this result, the sensitivities for the accelerations can very easily be calculated in the complete macro-time interval \([T_N, T_{N+1}]\), namely

\[
\frac{\partial \ddot{\mathbf{v}}}{\partial \mathbf{u}_{N+1}}^{cor,a}(t) = \frac{\partial \ddot{\mathbf{v}}}{\partial \mathbf{u}_{N+1}}^{cor,a-1}(T_{N+1}) \cdot L_{N+1}(t).
\]
Fig. 38  *ImMilne* approach: number of macro-steps and global error of the state variables of the coupling bodies for different error tolerances and different approximation orders

\[
\frac{\partial v}{\partial u_{i,N+1}^{*,a-1}}(t) = \frac{\partial \dot{v}}{\partial u_{i,N+1}^{*,a-1}}(T_{N+1}) \cdot \int_T^{T_{N+1}} L_{N+1}(\overline{T})d\overline{T}. \quad (49)
\]

A further integration will then provide the sensitivities on position level

\[
\frac{\partial x}{\partial u_{i,N+1}^{*,a-1}}(t) = B(x_{\text{cor},a-1}(T_{N+1})) \int_T^{T_{N+1}} \frac{\partial v}{\partial u_{i,N+1}^{*,a-1}}(\overline{T})d\overline{T} \quad (B \neq I), \quad (50)
\]

Evaluation of Eqs. (49) and (50) at \(T_{N+1}\) yields the sensitivities \(\frac{\partial v}{\partial u_{i,N+1}^{*,a-1}}(T_{N+1})\) and \(\frac{\partial x}{\partial u_{i,N+1}^{*,a-1}}(T_{N+1})\) required in Eq. (40). Note that instead of \(M(T_{N+1}, x_{\text{cor},a-1}(T_{N+1}))\), \(B(x_{\text{cor},a-1}(T_{N+1}))\), and \(G(T_{N+1}, x_{\text{cor},a-2}(T_{N+1}))\), in a simplified approach also the matrices \(M(T_{N}, x_{\text{cor},a-2}(T_{N}))\), \(B(x_{\text{cor},a-2}(T_{N}))\), and \(G(T_{N}, x_{\text{cor},a-2}(T_{N}))\) might be used so that the sensitivities can be calculated in parallel with the \((a-1)\)th corrector step.

Appendix B

In Sect. 6.3, the interface-Jacobian for the implicit co-simulation approaches (*ImMilne* and *ImSLocal*) has been calculated with a finite difference (FD) approach. In this appendix, the same simulations are considered; the interface-Jacobian is, however, generated with the low-order sensitivities (LOS) of Appendix A.2. As can be seen in Figs. 38, 39, 40, 41, the basic behavior with the low-order sensitivities is quite similar to the simulation results with FD gradients. For the considered model and parameters, the performance with the FD gradients is little better. Applying the LOS approach, the number of corrector iterations is usually little larger compared with the FD implementation so that the co-simulation time may slightly be increased in association with the LOS approximation. Using FD gradients, numerical problems (ill-conditioning, numerical cancellation) may be observed if the perturbation increments are chosen inappropriately or if the macro-step size becomes very small. Then, the usage of LOS may even be more efficient than applying FD gradients [29].
Co-simulation with variable approximation order

Fig. 39  *ImSLocal* approach: number of macro-steps and global error of the state variables of the coupling bodies for different error tolerances and different approximation orders

Fig. 40  *ImMilne* approach: computation time for different error tolerances and different approximation orders

Fig. 41  *ImSLocal* approach: computation time for different error tolerances and different approximation orders
### Nomenclature

| Symbol | Description |
|--------|-------------|
| $atol$ | absolute error tolerance of the macro-step size controller |
| $atol^{BDF}$ | absolute error tolerance of the subsystem solver |
| $B$ | symmetric square matrix (DAE-system) |
| $C^k$ | error constant |
| $C_i$ | nonlinear stiffness coefficient of spring/damper element $i$ |
| $c_i$ | linear stiffness coefficient of spring/damper element $i$ |
| $D_i$ | nonlinear damping coefficient of spring/damper element $i$ |
| $d_i$ | linear damping coefficient of spring/damper element $i$ |
| $e$ | local error |
| $f^c$ | coupling force |
| $G$ | constraint Jacobian |
| $g_c$ | vector of coupling conditions |
| $h$ | solver-step size |
| $HN$ | macro-step size |
| $J_i$ | inertia tensor of body $i$ |
| $k$ | polynomial degree of approximation polynomials |
| $l$ | number of corrector iterations |
| $L^k$ | Lagrange basis polynomial of degree $k$ |
| $M$ | mass matrix |
| $m_i$ | mass of body $i$ |
| $n_c$ | number of coupling variables |
| $n_K$ | number of coupling bodies |
| $P_{N+1}$ | approximation polynomials |
| $P^k$ | vector of interpolation polynomials of degree $k$ |
| $q$ | vector of position variables |
| $r$ | number of subsystems |
| $r^p$ | scaling factor for the macro-step size |
| $rtol$ | relative error tolerance of the macro-step size controller |
| $rtol^{BDF}$ | relative error tolerance of the subsystem solver |
| $S_i$ | center of mass of body $i$ |
| $SF$ | safety factor of the macro-step size controller |
| $s$ | subsystem index |
| $SDN$ | scaled derivative norm |
| $T_{Cosim}$ | computation time of the co-simulation model |
| $T_{Mono}$ | computation time of the monolithic simulation |
| $T_N$ | macro-time point $N$ |
| $u$ | vector of coupling variables |
| $v$ | vector of velocity variables |
| $W_i$ | weighting factor |
| $z$ | vector of state variables |
| $\alpha$ | index of corrector iteration |
| $\varepsilon$ | estimated local error |
| $\lambda_c$ | coupling force |
| $\phi(t, z, u)$ | vector of coupling functions |
| $\omega$ | vector of angular velocities |

**Funding Note** Open Access funding enabled and organized by Projekt DEAL.
References

1. Alioli, M., Morandini, M., Masarati, P.: Coupled multibody-fluid dynamics simulation of flapping wings. In: Proceedings of the ASME IDETC/CIE 2013, August 4–7 Portland, Oregon, USA (2013). DETC2013-12198
2. Ambrosio, J., Pombo, J., Rauter, F., Pereira, M.: A memory based communication in the co-simulation of multibody and finite element codes for pantograph-catenary interaction simulation. In: Bottasso, C.L. (ed.) Multibody Dynamics: Computational Methods and Applications, pp. 231–252. Springer, Berlin (2009)
3. 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. J. Theor. Appl. Mech. 50(3), 681–699 (2012)
4. Anderson, K., Duan, S.: A hybrid parallelizable low-order algorithm for dynamics of multi-rigid-body systems: part I, chain systems. Math. Comput. Model. 30(9–10), 193–215 (1999)
5. Arnold, M.: Stability of sequential modular time integration methods for coupled multibody system models. J. Comput. Nonlinear Dyn. 5, 1–9 (2010)
6. Arnold, M., Clauss, C., Schierz, T.: Error analysis and error estimates for co-simulation in FMI for model exchange and co-simulation in V2.0. Arch. Mech. Eng. 60(1), 75–94 (2013)
7. Ben Khaled-El Feki, A., Duval, L., Faure, C., Simon, D., Ben Gaid, M.: CHOPtrey: contextual online polynomial extrapolation for enhanced multi-core co-simulation of complex systems. Simulation 93(3), 185–200 (2017)
8. Brenan, K.E., Campbell, S.L., Petzold, L.R.: Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations, vol. 14. SIAM, Philadelphia (1996)
9. Busch, M.: Continuous approximation techniques for co-simulation methods: analysis of numerical stability and local error. ZAMM. Z. Angew. Math. Mech. (2016). https://doi.org/10.1002/zamm.201500196
10. Cuadrado, J., Cardenal, J., Morer, P., Bayo, E.: Intelligent simulation of multibody dynamics: space-state and descriptor methods in sequential and parallel computing environments. Multibody Syst. Dyn. 4, 55–73 (2000)
11. Datar, M., Stanciulescu, I., Negrut, D.: A co-simulation environment for high-fidelity virtual prototyping of vehicle systems. Int. J. Veh. Syst. Model. Test. 7, 54–72 (2012)
12. De Jalon, J.G., Bayo, E.: Kinematic and Dynamic Simulation of Multibody Systems: The Real-Time Challenge. Springer, Berlin (2012)
13. D’Silva, S., Sundaram, P., Ambrosio, J.: Co-Simulation Platform for Diagnostic Development of a Controlled Chassis System. SAE Technical Paper 2006-01-1058 (2006). https://doi.org/10.4271/2006-01-1058
14. Eberhard, P., Gaugel, T., Heisel, U., Storchak, M.: A discrete element material model used in a co-simulated charpy impact test and for heat transfer. In: Proceedings 1st Int. Conference on Process Machine Interactions, Hannover, Germany, 3–4 September, 2008 (2008)
15. Eich-Soolnner, E., Fuehrer, C.: Numerical Methods in Multibody Dynamics. Vieweg+Teubner, Wiesbaden (1998)
16. Fancello, M., Morandini, M., Masarati, P.: Helicopter rotor sailing by non-smooth dynamics co-simulation. Arch. Mech. Eng. 61(2), 253–268 (2014). https://doi.org/10.2478/meceng-2014-0015
17. Gomes, C.: Property preservation in co-simulation. Doctoral dissertation, University of Antwerp (2019)
18. Gomes, C., Thule, C., Broman, D., Larsen, P.G., Vangheluwe, H.: Co-simulation: a survey. ACM Comput. Surv. 51(3), 1–33 (2018). https://doi.org/10.1145/3179993
19. Gonzalez, F., Gonzalez, M., Cuadrado, J.: Weak coupling of multibody dynamics and block diagram simulation tools. In: Proceedings of the ASME 2009 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, IDETC/CIE 2009, San Diego, California, USA, August 30–September 2, 2009 (2009)
Gonzalez, F., Gonzalez, M., Mikkola, A.: Efficient coupling of multibody software with numerical computing environments and block diagram simulators. Multibody Syst. Dyn. 24(3), 237–253 (2010). https://doi.org/10.1007/s11044-010-9199-6

Gonzalez, F., Naya, M.A., Luaces, A., Gonzalez, M.: On the effect of multirate co-simulation techniques in the efficiency and accuracy of multibody system dynamics. Multibody Syst. Dyn. 25(4), 461–483 (2011). https://doi.org/10.1007/s11044-010-9234-7

Gonzalez, F., Arbatani, S., Mohtat, A., Kövecses, J.: Energy-leak monitoring and correction to enhance stability in the co-simulation of mechanical systems. Mech. Mach. Theory 131, 172–188 (2019). https://doi.org/10.1016/j.mechmachtheory.2018.09.007

Gu, B., Asada, H.H.: Co-simulation of algebraically coupled dynamic subsystems without disclosure of proprietary subsystem models. J. Dyn. Syst. Meas. Control 126, 1–13 (2004). https://doi.org/10.1115/1.1648307

Gustafsson, K., Lundh, M., Söderlind, G.: A PI step size control for the numerical solution of ordinary differential equations. BIT Numer. Math. 28(2), 270–287 (1988)

Hindmarsh, A.C., Brown, P.N., Grant, K.E., Lee, S.L., Serban, R., Shumaker, D.E., Woodward, C.S.: SUNDIALS: suite of nonlinear and differential/algebraic equation solvers. ACM Trans. Math. Softw. 31(3), 363–396 (2005)

Holzinger, F., Benedikt, M.: Optimal trigger sequence for non-iterative co-simulation. In: Proceedings of the 9th International Conference on Simulation and Modeling Methodologies, Technologies and Applications, pp. 80–87 (2019). SCITEPRESS-Science and Technology Publications, Ltd.

Holzinger, F.R., Benedikt, M.: Hierarchical coupling approach utilizing multi-objective optimization for non-iterative co-simulation. In: Proceedings of the 13th International Modelica Conference, Regensburg, Germany, March 4–6, 2019 vol. 157 (2019). Linköping University Electronic Press

Kraft, J.: Efficient Parallelization of Multibody Systems Incorporating Co-Simulation Techniques. Technical University Darmstadt (2021). https://doi.org/10.26083/tpurrints-00019123

Kraft, J., Klimmek, S., Schweizer, B.: Co-simulation and solver-coupling: efficient calculation of interface-Jacobian and coupling sensitivities/gradients. J. Comput. Nonlinear Dynam. (2021). https://doi.org/10.1115/1.4051823. CND-21-1063

Kühler, R., Schiehlen, W.: Two methods of simulator coupling. Math. Comput. Model. Dyn. Syst. 6, 93–113 (2000)

Lacoursière, C.: A parallel block iterative method for interactive contacting rigid multibody simulations on multicore PCs. Applied Parallel Computing. State of the Art in Scientific Computing, pp. 956–965 (2007)

Lacoursière, C., Nordfeldt, F., Linde, M.: A partitioning method for parallelization of large systems in realtime. In: Proceedings of the 3rd Joint International Conference on Multibody System Dynamics and the 7th Asian Conference on Multibody Dynamics, IMSD 2014, ACMD 2014, Bexco, Busan, Korea, June 30–July 3 (2014)

Lehnart, A., Fleissner, F., Eberhard, P.: Using SPH in a co-simulation approach to simulate sloshing in tank vehicles. In: Proceedings of SPHERIC4, Nantes, France, 27–29 May (2009)

Li, P., Lu, D., Schmoll, R., Schweizer, B.: Explicit co-simulation approach with improved numerical stability. In: IUTAM Symposium on Solver-Coupling and Co-Simulation, pp. 153–201. Springer, Berlin (2019)

Li, P., Yuan, Q., Lu, D., Meyer, T., Schweizer, B.: Improved explicit co-simulation methods incorporating relaxation techniques. Arch. Appl. Mech., 1–30 (2019)

Liao, Y.G., Du, H.I.: Cosimulation of multi-body-based vehicle dynamics and an electric power steering control system. Proc. Inst. Mech. Eng. K, J. Multibody Dyn. 215, 141–151 (2001)

Lichtenberg High Performance Computer of TU Darmstadt, https://www.hhlr.tu-darmstadt.de/hhlr/index.en.jsp

Malczyk, P., Fraczek, J.: Evaluation of parallel efficiency in modeling of mechanisms using commercial multibody solvers. Arch. Mech. Eng. LVI(3), 237–249 (2009)

Meyer, T., Li, P., Lu, D., Schweizer, B.: Implicit co-simulation method for constraint coupling with improved stability behavior. Multibody Syst. Dyn. 44(2), 135–161 (2018). https://doi.org/10.1007/s11044-018-9632-9

Meyer, T., Kraft, J., Schweizer, B.: Co-simulation; error estimation and macro-step size control. J. Comput. Nonlinear Dyn. 16(4), 041002 (2021). https://doi.org/10.1115/1.4048944

Milne, W.E.: Numerical integration of ordinary differential equations. Am. Math. Mon. 33(9), 455–460 (1926). https://doi.org/10.2307/2299609

https://www.mscsoftware.com. ADAMS manual

Naya, M., Cuadrado, J., Dopico, D., Lugris, U.: An efficient unified method for the combined simulation of multibody and hydraulic dynamics: comparison with simplified and co-integration approaches. Arch. Mech. Eng. LVIII, 223–243 (2011)
44. Negrut, D., Tasora, A., Mazhar, H., Heyn, T., Hahn, P.: Leveraging parallel computing in multibody dynamics. Multibody Syst. Dyn. 27, 95–117 (2012). https://doi.org/10.1007/s11044-011-9262-y
45. Negrut, D., Melanz, D., Mazhar, H., Lamb, D., Jayakumar, P.: Investigating through simulation the mobility of light tracked vehicles operating on discrete granular terrain. SAE Int. J. Passeng. Cars, Mech. Syst. 6, 369 (2013). https://doi.org/10.4271/2013-01-1191
46. Negrut, D., Serban, R., Mazhar, H., Heyn, T.: Parallel computing in multibody system dynamics: why, when and how. J. Comput. Nonlinear Dyn. 9(4), 041007 (2014). https://doi.org/10.1115/1.4027313
47. Peiret, A., Gonzalez, F., Kövecses, J., Teichmann, M.: Multibody system dynamics interface modelling for stable multirate co-simulation of multiphysics systems. Mech. Mach. Theory 127, 52–72 (2018). https://doi.org/10.1016/j.mechmachtheory.2018.04.016
48. Pombo, J., Ambrosio, J.: Multiple pantograph interaction with catenaries in high-speed trains. J. Comput. Nonlinear Dyn. 7(4), 041008 (2012)
49. Quaranta, G., Masarati, P., Mantegazza, P.: Multibody analysis of controlled aeroelastic systems on parallel computers. Multibody Syst. Dyn. 8(1), 71–102 (2002). https://doi.org/10.1023/A:1015894729968
50. Radhakrishnan, K., Hindmarsh, A.C.: Description and use of LSODE, the Livermore solver for ordinary differential equations. NASA Ref. Publ. 1327 (1993)
51. Rahikainen, J., Gonzalez, F., Naya, M.A.: An automated methodology to select functional co-simulation configurations. Multibody Syst. Dyn. 48(1), 79–103 (2019)
52. Rodriguez, B., Gonzalez, F., Naya, M.A., Cuadrado, J.: A test framework for the co-simulation of electric powertrains and vehicle dynamics. In: Proceedings of the ECCOMAS Thematic Conference on Multibody Dynamics, Duisburg, Germany (2019)
53. Sadjina, S., Pedersen, E.: Energy conservation and coupling error reduction in non-iterative co-simulations. Eng. Comput., 1–9 (2016)
54. Sadjina, S., Kyllingstad, L.T., Skjong, S., Pedersen, E.: Energy conservation and power bonds in co-simulations: non-iterative adaptive step size control and error estimation. Eng. Comput. 33, 607–620 (2017)
55. Schneider, F., Burger, M., Arnold, M., Simeon, B.: A new approach for force-displacement co-simulation using kinematic coupling constraints. J. Appl. Math. Mech. (2017). https://doi.org/10.1002/zamm.201500129
56. Schweizer, B.: In: Proceedings of the IUTAM Symposium on Solver-Coupling and Co-Simulation 2017, Darmstadt, Germany, September 18–20, 2017. Springer, Berlin (2019)
57. Schweizer, B., Li, P., Lu, D.: Stabilized index-2 co-simulation approach for solver coupling with algebraic constraints. J. Appl. Math. Mech. (2014). https://doi.org/10.1002/zamm.201300191
58. Schweizer, B., Lu, D.: Semi-implicit co-simulation approach for solver coupling. Arch. Appl. Mech. 84(12), 1739–1769 (2014). https://doi.org/10.1007/s00419-014-0883-5
59. Schweizer, B., Li, P., Lu, D.: Explicit and implicit co-simulation methods: stability and convergence analysis for different solver coupling approaches. J. Comput. Nonlinear Dyn. (2014). https://doi.org/10.1115/1.4028503
60. Schweizer, B., Li, P., Lu, D.: Implicit co-simulation methods: stability and convergence analysis for solver coupling with algebraic constraints. J. Appl. Math. Mech. (2015). https://doi.org/10.1002/zamm.201400087
61. Schweizer, B., Lu, D., Li, P.: Co-simulation method for solver coupling with algebraic constraints incorporating relaxation techniques. Multibody Syst. Dyn. (2015). https://doi.org/10.1007/s11044-015-9464-9
62. Serban, R., Melanz, D., Li, A., Stanciulescu, I., Jayakumar, P., Negrut, D.: A GPU-based preconditioned Newton–Krylov solver for flexible multibody dynamics. Int. J. Numer. Methods Eng. 102(9), 1585–1604 (2015)
63. Serban, R., Olsen, N., Negrut, D., Recuero, A., Jayakumar, P.: A co-simulation framework for high-performance, high-fidelity simulation of ground vehicle-terrain interaction. In: Proceedings of the NATO AVT-265 Specialists Meeting, Vilnius, Lithuania (2017)
64. Shampine, L.F.: Local extrapolation in the solution of ordinary differential equations. Math. Comput. 27(121), 91–97 (1973)
65. Shampine, L.F.: Implementation of implicit formulas for the solution of odes. SIAM J. Sci. Stat. Comput. 1(1), 103–118 (1980)
66. Shampine, L.F., Gordon, M.K.: Computer Solution of Ordinary Differential Equations: The Initial Value Problem. Freeman, New York (1975)
67. Skelboe, S.: The control of order and steplength for backward differential methods. BIT Numer. Math. 17(1), 91–107 (1977)
69. Solcia, T., Masarati, P.: Efficient multirate simulation of complex multibody systems based on free software. In: Proceedings of the ASME IDETC/CIE 2011, August 28–31, 2011, Washington, DC, USA. DETC2011-47306

70. Spreng, F., Eberhard, P., Fleissner, F.: An approach for the coupled simulation of machining processes using multibody system and smoothed particle hydrodynamics algorithms. Theor. Appl. Mech. Lett. 3(1), 8–013005 (2013)

71. Stetter, H.J., Weinmüller, E.: On the error control in ODE solvers with local extrapolation. Computing 27(2), 169–177 (1981)

72. Tomulik, P., Fraczek, J.: Simulation of multibody systems with the use of coupling techniques: a case study. Multibody Syst. Dyn. 25(2), 145–165 (2011)

73. Tseng, F., Hulbert, G.: Network-distributed multibody dynamics simulation-gluing algorithm. In: Ambrósio, J., Schiehlen, W. (eds.) Advances in Computational Multibody Dynamics. IDMEC/IST Lisbon, Portugal, pp. 521–540 (1999)

74. Valasek, M., Mraz, L.: Massive parallelization of multibody system simulation. Acta Polytech. 52(6) (2012)

75. Wang, J., Ma, Z.D., Hulbert, G.: A gluing algorithm for distributed simulation of multibody systems. Nonlinear Dyn. 34, 159–188 (2003)

76. Zhang, R., Zhang, H., Zanoni, A., Wang, Q., Masarati, P.: A tight coupling scheme for smooth/non-smooth multibody co-simulation of a particle damper. Mech. Mach. Theory 161, 104181 (2021). https://doi.org/10.1016/j.mechmachtheory.2020.104181

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.