Co-simulation with communication step size control in an FMI compatible master algorithm

Tom Schierz* Martin Arnold* Christoph Clauß#

* Martin Luther University Halle-Wittenberg
NWF II - Institute of Mathematics
D - 06099 Halle (Saale), Germany
martin.arnold@mathematik.uni-halle.de
tom.schierz@mathematik.uni-halle.de

# Fraunhofer Institute for Integrated Circuits IIS
Design Automation Division EAS
Zeunerstr. 38
D - 01069 Dresden, Germany
christoph.clauss@eas.iis.fraunhofer.de

Abstract

Complex multi-disciplinary models in system dynamics are typically composed of subsystems. This modular structure of the model reflects the modular structure of complex engineering systems. In industrial applications, the individual subsystems are often modeled separately in different mono-disciplinary simulation tools. The Functional Mock-Up Interface (FMI) provides an interface standard for coupling physical models from different domains and addresses problems like export and import of model components in industrial simulation tools (FMI for Model Exchange) and the standardization of co-simulation interfaces in nonlinear system dynamics (FMI for Co-Simulation), see [8]. In November 2011, the third β-version of FMI for Model Exchange and Co-Simulation v2.0 was released [13] that supports advanced numerical techniques in co-simulation like higher order extrapolation and interpolation of subsystem inputs, step size control including step rejection and Jacobian based linearly implicit stabilization techniques. Well known industrial simulation tools for applied dynamics support Version 1.0 of this standard and plan to support the forthcoming Version 2.0 in the near future, see the "Tools" tab of website [8] for up-to-date information. The renewed interest in algorithmic and numerical aspects of co-simulation inspired some new investigations on error estimation and stabilization techniques in FMI for Model Exchange and Co-Simulation v2.0 compatible co-simulation environments. The present paper extends recent results from [3] on reliable error estimation and communication step size control in the framework of FMI for Model Exchange and Co-Simulation v2.0. Based on a strict mathematical analysis, we study the asymptotic behaviour of the local error and two error estimates that may be used to adapt the communication step size automatically to the changing solution behaviour during time integration. These theoretical results are illustrated by numerical tests for a (linear) quarter car model and provide a basis for future investigations with more complex coupled engineering systems.

Keywords: FMI; error estimation; step size control

1 Introduction

Co-simulation is a rather general approach to the simulation of coupled technical systems and coupled physical phenomena in engineering with focus on instationary (time-dependent) problems. From the mathematical viewpoint, co-simulation results in a class of time integration methods for coupled systems which are described by time dependent ordinary differential equations (ODE) or differential algebraic equations (DAE). In that context, we consider $r \geq 2$ coupled subsystems in nonlinear state-space form

$$\left.\begin{aligned} \dot{\mathbf{x}}_i &= \mathbf{f}_i(t,\mathbf{x}_i,\mathbf{u}_i), \\ \mathbf{y}_i &= \mathbf{g}_i(t,\mathbf{x}_i,\mathbf{u}_i), \end{aligned}\right\} \quad i = 1,\dots,r, \quad t \in [t_{\text{start}}, t_{\text{stop}}] \tag{1a}$$

with the state vectors $\mathbf{x}_i$, inputs $\mathbf{u}_i$ and outputs $\mathbf{y}_i$. The subsystems are coupled by input-output relations

$$\mathbf{u}_i = \mathbf{c}_i(\mathbf{y}_1,\dots,\mathbf{y}_{i-1},\mathbf{y}_{i+1},\dots,\mathbf{y}_r), \quad (i = 1,\dots,r), \tag{1b}$$

see [12]. Summarizing all components in vector form, we get a coupled system in the more compact form

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(t,\mathbf{x},\mathbf{u}), \\ \mathbf{y} &= \mathbf{g}(t,\mathbf{x},\mathbf{u}), \quad \mathbf{u} = \mathbf{c}(\mathbf{y}), \end{aligned} \tag{2}$$

with $\mathbf{x} = (\mathbf{x}_1^T,\dots,\mathbf{x}_r^T)^T, \dots$

The simulation time interval is split by a grid of *communication points* $t_{\text{start}} = T_0 < T_1 < \ldots < T_N = t_{\text{stop}}$, where the data exchange between the subsystems is restricted to these discrete points. In each *communication step* $T_n \to T_n + H_n =: T_{n+1}$, $(n = 0, \ldots, N-1)$, with *communication step size* H_n, the subsystems are solved separately and independently from each other.

Since the update of the inputs $\mathbf{u}$ is restricted to the time discrete communication points, these terms have to be approximated by signal extrapolation for the current communication step $T_n \to T_{n+1}$

$$\bar{\mathbf{u}}(t) \approx \mathbf{u}(t), \quad t \in [T_n, T_{n+1}],$$

where the approximation $\bar{\mathbf{u}}(t)$, is based on information $\mathbf{u}(T_{n-\iota})$, $(\iota = 0, \ldots, k)$ from $k+1$ previous communication points.

This leads to the new coupled problem for $t \in [T_n, T_n + H_n]$

$$\begin{aligned} \dot{\bar{\mathbf{x}}}(t) &= \mathbf{f}(t, \bar{\mathbf{x}}, \bar{\mathbf{u}}), \\ \bar{\mathbf{y}}(t) &= \mathbf{g}(t, \bar{\mathbf{x}}, \bar{\mathbf{u}}). \end{aligned} \tag{3}$$

The coupled system (3) is composed of subsystems that are solved separately from each other, where we get the numerical solution $\bar{\mathbf{x}}(t) \approx \mathbf{x}(t)$ and $\bar{\mathbf{y}}(t) \approx \mathbf{y}(t)$, $(t \in [T_n, T_n + H_n])$. Typically, different time integration methods and / or different (*micro*) step sizes are used in the individual subsystems resulting in a multi-method and / or multi-rate method for the coupled system.

Following Jackson [10], the time integration methods to solve the coupled system (3) are called *modular* to underline the modular structure of the approach.

In practical applications, usually constant communication step sizes $H_n := H$ with $n = 0, \ldots, N-1$ are used, since the simulation results and the behavior of the underlying integration methods can reliably be controlled. Further gains in efficiency and robustness of numerical co-simulation techniques are expected by variable communication step sizes $H_n := T_{n+1} - T_n$ that are adapted automatically to the solution behavior (communication step size control).

2 Estimation of the local error

For a reliable communication step size control an error estimate **EST** for the local error of the numerical solution is needed and is compared to user defined error bounds (tolerances), since for coupled systems (2) without algebraic loops the global error is bounded by a multiple of the maximum local error [3].

Richardson extrapolation is a time-consuming but reliable way to estimate local errors in ODE and DAE time integration and considers two (small) consecutive communication steps of size $H_n = H_{n+1} = H$ from $T_n \to T_n + H =: T_{n+1}$ and $T_{n+1} \to T_n + 2H =: T_{n+2}$. To get an estimate for the local error, the solution of these two steps is compared with a second numerical solution that is computed for a (large) communication step $T_n \to T_n + 2H =: T_{n+2}$. Substantial savings of computing time result from an algorithmic modification of the Richardson extrapolation that is tailored to the FMI co-simulation framework. Both approaches will be studied theoretically (by an asymptotic error analysis) as well as practically (by numerical tests for a quarter car benchmark problem). For the theoretical analysis, we neglect the discretization errors in the subsystems that should be kept small in a practical implementation by appropriate settings of error tolerances.

To keep the notation compact, we restrict the theoretical analysis of the local error estimates to pure polynomial signal extrapolation in all subsystems [3]. Consider the polynomial $\bar{\mathbf{u}}(t)$, that interpolates the analytical solution $\mathbf{u}(t)$ of (2) at the $k+1$ equidistant previous communication points $T_{n-\iota}$, $(\iota = 0, \ldots, k)$. The approximation error of this interpolating polynomial $\bar{\mathbf{u}}(t)$ is for all $t \in [T_n, T_n + H]$ bounded by [7]

$$\begin{aligned} \bar{\mathbf{u}}(t) - \mathbf{u}(t) = &-\frac{\mathbf{u}^{(k+1)}(T_n)}{(k+1)!} \prod_{\iota=0}^{k} (t - T_{n-\iota}) \\ &+ \mathscr{O}(H^{k+2}). \end{aligned} \tag{4}$$

In the first (small) communication step $T_n \to T_n + H$ (first step of the error estimation by Richardson extrapolation) we have to solve system (3) starting from $\bar{\mathbf{x}}(T_n) = \mathbf{x}(T_n)$. Linearization shows that the error in the state and output vectors satisfy

$$\begin{aligned} \dot{\bar{\mathbf{x}}}(t) - \dot{\mathbf{x}}(t) &= \mathbf{A}_n(\bar{\mathbf{x}}(t) - \mathbf{x}(t)) + \mathbf{B}_n(\bar{\mathbf{u}}(t) - \mathbf{u}(t)) + \\ &\quad + \mathscr{O}(H^{k+2}), \\ \bar{\mathbf{y}}(t) - \mathbf{y}(t) &= \mathbf{C}_n(\bar{\mathbf{x}}(t) - \mathbf{x}(t)) + \mathbf{D}_n(\bar{\mathbf{u}}(t) - \mathbf{u}(t)) + \\ &\quad + \mathscr{O}(H^{k+2}), \end{aligned}$$

where the system matrices $\mathbf{A}_n$, $\mathbf{B}_n$, $\mathbf{C}_n$, $\mathbf{D}_n$ denote the Jacobians $\mathbf{f}_x$, $\mathbf{f}_u$, $\mathbf{g}_x$, $\mathbf{g}_u$ evaluated at $\mathbf{x} = \mathbf{x}(T_n)$, $\mathbf{u} = \mathbf{u}(T_n)$. Up to higher order terms, the error in the state vector is given by the solution of a linear time invariant system for $t \in [T_n, T_n + H_n]$ and may be expressed as

$$\begin{aligned} \bar{\mathbf{x}}(t) - \mathbf{x}(t) = &\exp(\mathbf{A}_n(t - T_n)) \overbrace{(\bar{\mathbf{x}}(T_n) - \mathbf{x}(T_n))}^{=0} \\ &+ \int_{T_n}^{t} \exp(\mathbf{A}_n(t - s)) \mathbf{B}_n(\bar{\mathbf{u}}(s) - \mathbf{u}(s)) \mathrm{d}s \\ &+ \mathscr{O}(H^{k+3}). \end{aligned} \tag{5}$$

With (4) and (5) and the substitution $\sigma := (s - T_n)/H$, $H\mathrm{d}\sigma = \mathrm{d}s$ the error of the output vector is

$$\begin{aligned}\bar{\mathbf{y}}(T_{n+1}) - \mathbf{y}(T_{n+1}) = &\\ &\underbrace{-\mathbf{C}_n\mathbf{B}_n \frac{\mathbf{u}^{(k+1)}(T_n)}{(k+1)!}\int_0^1 \prod_{\iota=0}^{k}(\sigma+\iota)\mathrm{d}\sigma\, H^{k+2}}_{=:\,\gamma_k} \\ &\underbrace{-\mathbf{D}_n\mathbf{u}^{(k+1)}(T_n)}_{=:\,\delta_k} H^{k+1} + \mathscr{O}(c_D H^{k+2} + H^{k+3})\end{aligned}$$

with constant c_D, which is set to $c_D = 0$ if $\partial\mathbf{g}/\partial\mathbf{u} \equiv 0$ and $c_D = 1$ otherwise.

In the next (small) communication step $T_n + H \to T_n + 2H$ the input function is substituted by a polynomial $\bar{\bar{\mathbf{u}}}(t)$ that interpolates $\mathbf{c}(\bar{\mathbf{y}})$ at $t = T_n + H$ and $\mathbf{u}(t)$ at $t = T_{n-\iota}$, $(\iota = 0,\dots,k-1)$.

The error in the output vector $\bar{\bar{\mathbf{y}}}(T_{n+2})$ is then given by

$$\begin{aligned}\bar{\bar{\mathbf{y}}}(T_{n+2}) - \mathbf{y}(T_{n+2}) = &\\ &- 2\gamma_k H^{k+2} \\ &- \Big(\delta_k + (k+1)\mathbf{D}_n\mathbf{L}_n\delta_k\Big)H^{k+1} \\ &+ \mathscr{O}(c_D H^{k+2} + H^{k+3})\end{aligned} \tag{6}$$

with $\mathbf{L}_n = \partial\mathbf{c}/\partial\mathbf{y}(T_n)$.

For error estimation by Richardson extrapolation, we consider in time interval $[T_n, T_n + 2H]$ a second numerical solution for the output vector $\tilde{\mathbf{y}}(T_{n+2})$ and input function $\tilde{\mathbf{u}}(t)$ that is defined by the interpolation polynomial for data points $(T_{n-2\iota}, \mathbf{u}(T_{n-2\iota}))$, $(\iota = 0,\dots,k)$. Analogously to the first step we get the error

$$\begin{aligned}\tilde{\mathbf{y}}(T_{n+2}) - \mathbf{y}(T_{n+2}) = -\gamma_k(2H)^{k+2} - \delta_k(2H)^{k+1} &\\ + \mathscr{O}(c_D H^{k+2} + H^{k+3}).&\end{aligned} \tag{7}$$

In ODE and DAE time integration, the comparison of the numerical results for a double-step with (small) step size H and a single (large) step with step size $2H$ allows to estimate the leading term of the local error [9]. For modular time integration, this error estimate is given by [11]

$$\mathbf{EST}_{\mathrm{Rich}} := \frac{\bar{\bar{\mathbf{y}}}(T_{n+2}) - \tilde{\mathbf{y}}(T_{n+2})}{(1 - 2^{k+1})}.$$

The comparison of (6) and (7) shows

$$\begin{aligned}\bar{\bar{\mathbf{y}}}(T_{n+2}) - \mathbf{y}(T_{n+2}) = \mathbf{EST}_{\mathrm{Rich}} &\\ + \frac{(k+1)2^{k+1}}{(1-2^{k+1})}\mathbf{D}_n\mathbf{L}_n\delta_k H^{k+1} &\\ + \mathscr{O}(c_D H^{k+2} + H^{k+3}).&\end{aligned}$$

Here we can see, that in the context of co-simulation, Richardson extrapolation may give asymptotically wrong results if $\mathbf{D}_n\mathbf{L}_n\delta_k \neq 0$, i. e. , for coupled systems with direct feed-through in at least one subsystem. If there are no subsystems with direct feed-through ($\partial\mathbf{g}_j/\partial\mathbf{u}_j \equiv 0$, $(j = 1,\dots,r)$), $\mathbf{EST}_{\mathrm{Rich}}$ reproduces all components of the local error in the output variables correctly up to higher order terms.

In the ITEA2 project MODELISAR, several modifications of error estimate $\mathbf{EST}_{\mathrm{Rich}}$ were tested [15] to reduce the large extra effort for computing $\tilde{\mathbf{y}}$. Here we use $\bar{\mathbf{u}}(t) = \tilde{\mathbf{u}}(t)$ for $t \in [T_n, T_n + H]$ such that the intermediate results $\bar{\mathbf{x}}(T_{n+1})$ and $\tilde{\mathbf{x}}(T_{n+1})$ coincide. From the view point of numerical efficiency, it would be favorable to use one and the same approximation $\bar{\mathbf{u}}(t)$ of the input function $\mathbf{u}(t)$ for both numerical solutions in the first communication step $T_n \to T_n + H$ and to restrict the use of different input functions to the second communication step, i. e. , to $t \in [T_{n+1}, T_{n+2}]$.

In that way, co-simulation may proceed with a large communication step $T_n \to T_n + 2H$ of size $2H$ that is temporarily interrupted at $t = T_{n+1}$ to provide input data $\bar{\mathbf{y}}(T_{n+1})$ and $\mathbf{c}(\bar{\mathbf{y}}(T_{n+1}))$ for the second numerical solution to be used for error estimation. Alternatively, a small communication step $T_n \to T_n + H$ may be completed in the classical way and the two different numerical solutions on time interval $[T_{n+1}, T_{n+2}]$ are evaluated in parallel. With this second strategy, no subsystem solver has to go backward in time and the practical implementation might be simplified.

According to [3] the error estimate is then given by

$$\mathbf{EST}_{\mathrm{mod}} := \frac{\bar{\bar{\mathbf{y}}}(T_{n+2}) - \bar{\mathbf{y}}(T_{n+2})}{(1 - p_k)},$$

where p_k is given for constant $(k = 0)$, linear $(k = 1)$ and quadratic $(k = 2)$ extrapolation by

$$\begin{aligned}p_0 &= 2,\\ p_1 &= 14/5,\\ p_2 &= 32/9.\end{aligned}$$

To demonstrate the estimation of the local error and the communication step size control in co-simulation, we use a low dimensional linear benchmark problem. The *quarter car model* which goes along a road profile is defined by a one-dimensional oscillator with two mass points m_c and m_w for the chassis and the wheel. Both masses are coupled by a spring-damper element with spring constant k_c and damping constant d_c, see Figure 1. Moreover, the connection between wheel and road is represented by a spring-damper element with constants k_w and d_w. For typical parameter values (see below), the eigenfrequency of subsystem wheel is ten times larger than the one of subsystem chassis.

The displacements of the mass points are given by η_c, η_w and the profile of the road is defined by a time dependent function $z(t)$. For numerical tests we use the following parameter configuration:

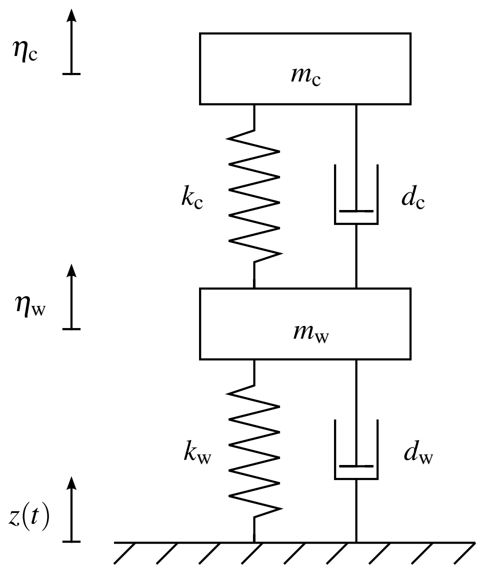

Figure 1: Quarter car model.

wheel mass	$m_\mathrm{w} = 40\,\mathrm{kg}$,
chassis mass	$m_\mathrm{c} = 400\,\mathrm{kg}$,
wheel spring	$k_\mathrm{w} = 150000\,\mathrm{Nm}^{-1}$,
chassis spring	$k_\mathrm{c} = 15000\,\mathrm{Nm}^{-1}$,
wheel damper	$d_\mathrm{w} = 0\,\mathrm{Nsm}^{-1}$,
chassis damper	$d_\mathrm{c} = 1000\,\mathrm{Nsm}^{-1}$,

initial values at $t = 0$

$$\eta_\mathrm{c}(0) = 0$$
$$\dot\eta_\mathrm{c}(0) = 0$$
$$\eta_\mathrm{w}(0) = 0$$
$$\dot\eta_\mathrm{w}(0) = 0$$

and excitation function

$$z(t) := \begin{cases} 0, & t < 0, \\ 0.1, & t \geq 0 \end{cases}$$

to get the system in motion. The equations of motion are given by

$$\begin{aligned} m_\mathrm{c}\ddot\eta_\mathrm{c}(t) &= k_\mathrm{c}(\eta_\mathrm{w}(t) - \eta_\mathrm{c}(t)) + d_\mathrm{c}(\dot\eta_\mathrm{w}(t) - \dot\eta_\mathrm{c}(t)), \\ m_\mathrm{w}\ddot\eta_\mathrm{w}(t) &= k_\mathrm{w}(z(t) - \eta_\mathrm{w}(t)) + d_\mathrm{w}(\dot z(t) - \dot\eta_\mathrm{w}(t)) \\ &\quad - k_\mathrm{c}(\eta_\mathrm{w}(t) - \eta_\mathrm{c}(t)) - d_\mathrm{c}(\dot\eta_\mathrm{w}(t) - \dot\eta_\mathrm{c}(t)). \end{aligned}$$

For co-simulation this system is split into two subsystems (chassis and wheel) of the form

$$\begin{aligned} \dot{\mathbf{x}}_1 &= \mathbf{f}_1(\mathbf{x}_1, \mathbf{u}_1), & \dot{\mathbf{x}}_2 &= \mathbf{f}_2(\mathbf{x}_2, \mathbf{u}_2), \\ \mathbf{y}_1 &= \mathbf{g}_1(\mathbf{x}_1, \mathbf{u}_1), & \mathbf{y}_2 &= \mathbf{g}_2(\mathbf{x}_2, \mathbf{u}_2), \\ \mathbf{u}_1 &= \mathbf{y}_2, & \mathbf{u}_2 &= \mathbf{y}_1 \end{aligned}$$

with the state vectors $\mathbf{x}_1 = (\eta_\mathrm{c}, \dot\eta_\mathrm{c})^T$ and $\mathbf{x}_2 = (\eta_\mathrm{w}, \dot\eta_\mathrm{w})^T$, right hand sides of the subsystems $\mathbf{f}_1 = (\dot\eta_\mathrm{c}, \ddot\eta_\mathrm{c})^T$, $\mathbf{f}_2 = (\dot\eta_\mathrm{w}, \ddot\eta_\mathrm{w})^T$, inputs $\mathbf{u}_1$, $\mathbf{u}_2$ and outputs $\mathbf{y}_1$, $\mathbf{y}_2$. We consider a displacement-displacement and a displacement-force coupling [5]. For the displacement-displacement coupling the input and output vectors are given by

$$\begin{aligned} \mathbf{u}_2 &= \mathbf{y}_1 = \mathbf{x}_1, \\ \mathbf{u}_1 &= \mathbf{y}_2 = \mathbf{x}_2. \end{aligned}$$

In the case of displacement-force coupling, the output of the second subsystem (wheel) is defined by the coupling force of the spring-damper element between the two masses m_c and m_w which is also the input of the other subsystem (chassis):

$$\begin{aligned} \mathbf{u}_2 &= \mathbf{y}_1 = (\eta_\mathrm{c}, \dot\eta_\mathrm{c})^T, \\ \mathbf{u}_1 &= \mathbf{y}_2 = k_\mathrm{c}(\eta_\mathrm{w}(t) - \eta_\mathrm{c}(t)) + d_\mathrm{c}(\dot\eta_\mathrm{w}(t) - \dot\eta_\mathrm{c}(t)). \end{aligned}$$

In that case, the subsystem wheel has a direct feed-through of its input, $\partial\mathbf{g}_2/\partial\mathbf{u}_2 \neq \mathbf{0}$.

Figs. 2 and 3 show the local errors for the quarter car benchmark and illustrate that the new error estimate $\mathbf{EST}_\mathrm{mod}$ is as reliable as the classical estimate $\mathbf{EST}_\mathrm{Rich}$.

3 Variable communication step sizes

The communication step size control in co-simulation is an extension of the step size control in classical time integration methods for ODEs [9].

In the context of co-simulation, the error estimator $\mathbf{EST}$ should estimate in each consecutive communication step $T_n \to T_{n+1} \to T_{n+2}$ all errors in the slave outputs $\bar{\bar{\mathbf{y}}}_{n+2} := \bar{\bar{\mathbf{y}}}(T_{n+2})$, that result from the solution of (3) by a numerical time integration method with approximated slave inputs $\bar{\mathbf{u}}(t)$, $t \in [T_n, T_{n+1}]$ and $\bar{\bar{\mathbf{u}}}(t)$, $t \in [T_{n+1}, T_{n+2}]$. We consider the scalar error indicator

$$\mathrm{ERR} := \sqrt{\frac{1}{m}\sum_{j=1}^{m}\left(\frac{\mathrm{EST}_j}{\mathrm{ATOL}_j + \mathrm{RTOL}_j|\bar{\bar{\mathbf{y}}}^j_{n+2}|}\right)^2} \qquad (8)$$

with the vector

$$\bar{\bar{\mathbf{y}}}_{n+2} := (\bar{\bar{\mathbf{y}}}^T_{1,n+2}, \ldots, \bar{\bar{\mathbf{y}}}^T_{r,n+2})^T \in \mathbb{R}^m,$$

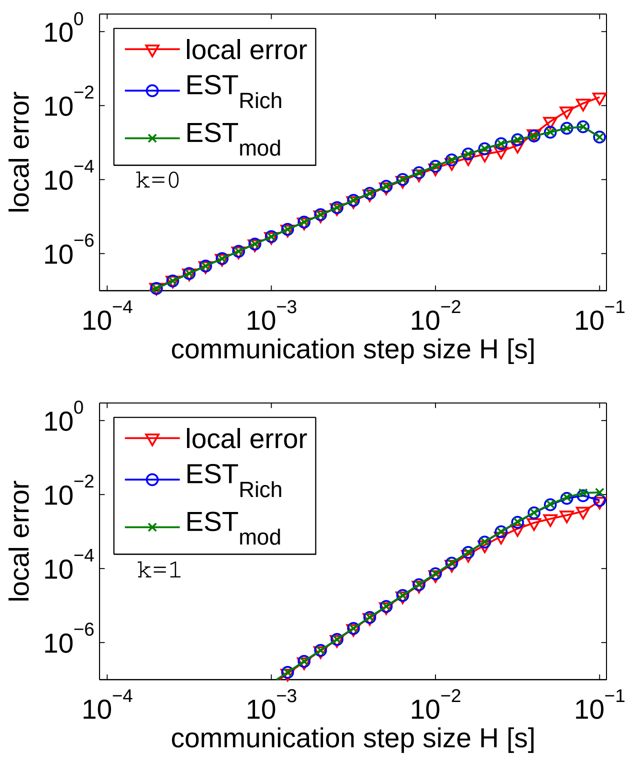

Figure 2: Benchmark Quarter car - co-simulation for a coupled system (2) without feed-through: Local error ("$\bigtriangledown$") and error estimates $\mathbf{EST}_{\mathrm{Rich}}$ ("$\circ$"), $\mathbf{EST}_{\mathrm{mod}}$ ("$\times$"). Constant ($k = 0$, upper plot) and linear extrapolation ($k = 1$, lower plot).

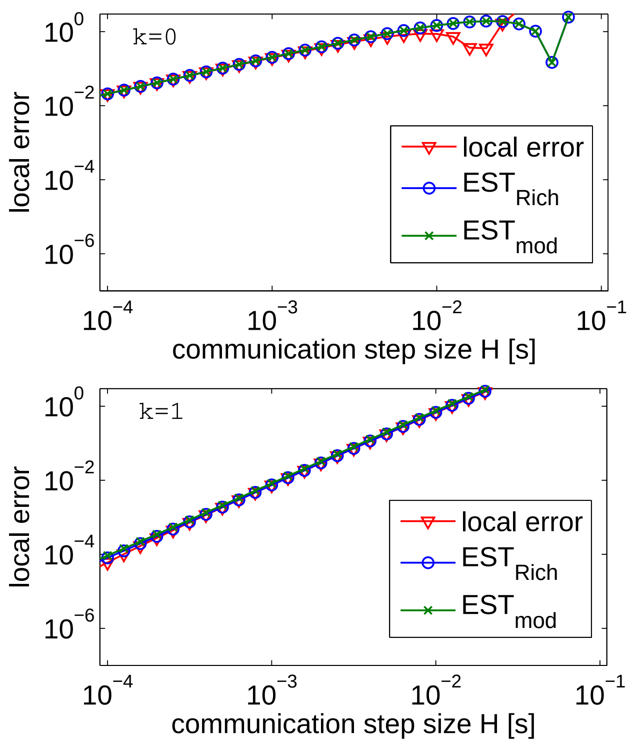

Figure 3: Benchmark Quarter car - co-simulation for a coupled system (2) with feed-through: Local error ("$\bigtriangledown$") and error estimates $\mathbf{EST}_{\mathrm{Rich}}$ ("$\circ$"), $\mathbf{EST}_{\mathrm{mod}}$ ("$\times$"). Constant ($k = 0$, upper plot) and linear extrapolation ($k = 1$, lower plot).

that contains all outputs of the $r \geq 2$ subsystems at $t = T_{n+2}$. The error indicator (8) shows whether the communication step size $H = H_n$ was sufficiently small to meet some user defined error bounds ATOL_j, RTOL_j or not. Analogously to the classical approach [9] a communication step is accepted, if $\mathrm{ERR} \leq 1$. When $\mathrm{ERR} > 1$, then the estimated error was too large and the communication step has to be repeated with a smaller step size to meet the accuracy requirements.

The ratio between the error indicator ERR and its optimal value 1.0 may be used to define a posteriori an "optimal" communication step size

$$H_{\mathrm{opt}} := \alpha H_n \left(\frac{1}{\mathrm{ERR}}\right)^{1/P}$$

with $P = k+1$ if there exist subsystems with direct feed-through, otherwise $P = k+2$, a safety factor $\alpha \in [0.8, 0.9]$ to reduce the risk of a rejection of the next communication step if the current step was accepted and k denoting the approximation order of the signal extrapolation for slave inputs $\bar{\mathbf{u}}(t)$. Note, that H_{opt} is always smaller than the current communication step size H_n if the error estimate **EST** exceeds the given tolerances ($\mathrm{ERR} > 1$).

In practical applications the step size is not allowed to increase nor to decrease too fast [9]. Therefore, parameters $\theta_{\min} \in [0.2, 0.5]$ and $\theta_{\max} \in [1.5, 5]$ are used to restrict the step size change. It is clear that choosing both parameters too small may increase the computational work. Moreover, choosing them too large can increase the risk of rejected steps. The resulting communication step size is given by

$$H_{\mathrm{opt}} = H_n \cdot \min\{\theta_{\max}, \max\{\theta_{\min}, \alpha \cdot (1/\mathrm{ERR})^{1/P}\}\}.$$

4 Generic implementation scheme in FMI 2.0

The Functional Mock-up Interface (FMI) for Model Exchange and Co-Simulation v2.0, see [2, 8], is a standard for the coupling and exchange of models and simulator coupling. A component implementing the FMI is called Functional Mock-up Unit (FMU). It consists of C-functions in source code or preprocessed binaries (like dll or shared object) and an XML description file, that contains all static information for calling the FMU [4]. The C-functions are called to set and get values in the FMU (FMI-functions `fmiSetReal` and `fmiGetReal`). If the FMU is used for simulator coupling it contains the full model and a simulator (slave). The slave is controlled by a function `fmiDoStep` [13] to process one communication step. With this FMI-function the computation of a communication step in a slave is started with the input parameters `currentCommunicationPoint` (current communication point T_n of the master), `communicationStepSize` (step size H_n of the communication step) and `noSetFMUStatePriorToCurrentPoint`. The Parameter `noSetFMUStatePriorToCurrentPoint` is `true` (`fmiTrue`) if the FMI-function `fmiSetFMUState` will no longer be called for time instants prior to `currentCommunicationPoint`. This is an important information for a slave, that is able to reject communication steps, since the FMU states have to be restored to the last accepted communication point in that case. With the flag `noSetFMUStatePriorToCurrentPoint` the slave can use this information to flush a result buffer. For the Richardson extrapolation based step size control the master has to repeat the simulation from the last accepted communication point, that means it has to save and restore the FMU states to this point for the computation of the reference solution to estimate the error or if a step was rejected because the step size was too large. `fmiGetFMUState` makes a copy of the internal FMU state and returns a pointer to this copy and `fmiSetFMUState` copies the content of the previously copied `FMUstate` back and uses it as current new FMU state.

A generic implementation scheme for a double step $T_n \to T_{n+1} \to T_{n+2}$ with Richardson extrapolation based communication step size control is given by:

(A) $t_{\text{curr}} = T_n$. Get slave states $slave_{i,n}$, $(i = 1,\ldots,r)$, calling `fmiGetFMUState` for all r slave FMUs.

(B) Define subsystem inputs $\tilde{\mathbf{u}}_i(t)$ providing derivatives $\mathrm{d}^l\tilde{\mathbf{u}}_i/\mathrm{d}t^l(T_n)$, $(l = 0,\ldots,k)$ by `fmiSetRealInputDerivatives`.

(C) Perform large communication step $T_n \to T_n + 2H = T_{n+2}$ calling `fmiDoStep` for all r slave FMUs and get outputs $\tilde{\mathbf{y}}(T_{n+2})$ by `fmiGetRealOutputDerivatives`.

(D) Reset all slave FMUs to state $slave_{i,n}$, $(i = 1,\ldots,r)$, calling `fmiSetFMUState`.

(E) Define subsystem inputs $\bar{\mathbf{u}}_i(t)$ providing derivatives $\mathrm{d}^l\bar{\mathbf{u}}_i/\mathrm{d}t^l(T_n)$, $(l = 0,\ldots,k)$ by `fmiSetRealInputDerivatives`.

(F) Perform first small communication step $T_n \to T_n + H = T_{n+1}$ calling `fmiDoStep` for all r slave FMUs and get outputs $\bar{\mathbf{y}}(T_{n+1})$ by `fmiGetRealOutputDerivatives`.

(G) Evaluate $\bar{\mathbf{u}}(T_{n+1}) = \mathbf{c}(\bar{\mathbf{y}}(T_{n+1}))$ and define subsystem inputs $\bar{\mathbf{u}}_i(t)$ providing derivatives $\mathrm{d}^l\bar{\mathbf{u}}_i/\mathrm{d}t^l(T_{n+1})$, $(l = 0,\ldots,k)$ by `fmiSetRealInputDerivatives`.

(H) Perform second small communication step $T_{n+1} \to T_{n+1} + H = T_{n+2}$ calling `fmiDoStep` for all r slave FMUs and get outputs $\bar{\bar{\mathbf{y}}}(T_{n+2})$ by `fmiGetRealOutputDerivatives`.

(I) Evaluate error estimate $\mathbf{EST}_{\text{Rich}}$, error indicator ERR and optimal communication step size H_{opt}.

The function `fmiDoStep` returns `fmiOK` if the communication step was computed successfully until its end. `fmiDiscard` is returned if the slave computed successfully only a subinterval of the communication step, which may occur if the communication step size is too large and the simulation should be repeated with a smaller one. `fmiError` will be returned by `fmiDoStep` if the simulation of the communication step could not be carried out at all.

The capabilities of a slave are described in the XML file. A capability flag `canHandleVariableCommunicationStepSize` set to `true` indicates, that a slave is able to accept variable communication steps, which is important to implement a master with communication step size control. The complete interface description can be found in [13].

5 Numerical test for the FMI compatible master

For developing and testing new master algorithms, a master prototype was implemented by Fraunhofer IIS/EAS and Halle University [4, 6]. This master supports basic functions of FMI for Co-Simulation version 1.0, see [14], and has implemented several sophisticated master algorithms like communication step size control and the rejection of communication steps.

To use the Fraunhofer master with the given FMUs the compiled C-code of the master has to be linked to the slaves binaries, which may even be C-code that

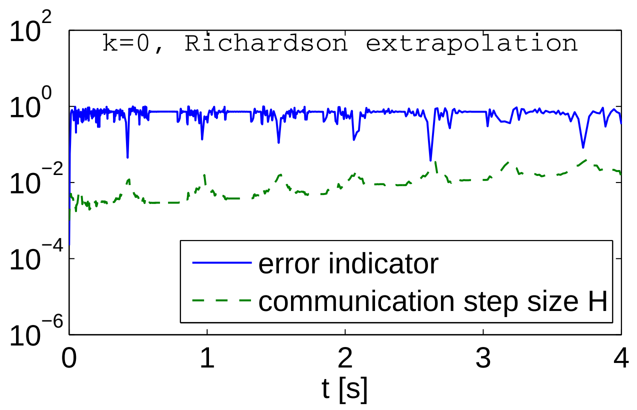

Figure 4: Benchmark Quarter car, displacement-displacement coupling. Error indicator and communication step size of the simulation with Richardson extrapolation based step size control with the Fraunhofer master.

is compiled with the master. The resulting executable consists of the master and the slaves.

For the numerical tests with the Fraunhofer master, the quarter car benchmark with the two slaves "chassis" and "wheel" is implemented in Dymola. We apply the communication step size control strategy from Section 3 with error estimates $\mathbf{EST}_{\mathrm{Rich}}$ and $\mathbf{EST}_{\mathrm{mod}}$ and study the influence of the order k of signal extrapolation and of the coupling strategy (displacement-displacement vs. displacement-force). From a practical viewpoint constant ($k = 0$), linear ($k = 1$) and quadratic ($k = 2$) signal extrapolation are most interesting since higher order extrapolations increase the risk of numerical instability, see also [1]. In all numerical tests, the error tolerances for slave FMUs are chosen two orders of magnitude smaller than the master tolerances ATOL_j, RTOL_j in (8).

Since the current implementation of the Fraunhofer master is limited to constant extrapolation with error estimation by Richardson extrapolation, only these numerical results are depicted in Figure 4 for the displacement-displacement coupling.

Extending these results, we will also consider the numerical simulations in a MATLAB-based test environment for the verification of the theoretical analysis. The numerical test for variable communication step sizes with error estimates $\mathbf{EST}_{\mathrm{Rich}}$ and $\mathbf{EST}_{\mathrm{mod}}$ for constant ($k = 0$) and linear ($k = 1$) extrapolation is depicted for the two coupling strategies in Figures 5 and 6 within the simulation time interval $t \in [0, 4]$. The simulation results for Richardson extrapolation and its modification are identical for $k = 0$, where the small differences in Table 1 are caused by implementation

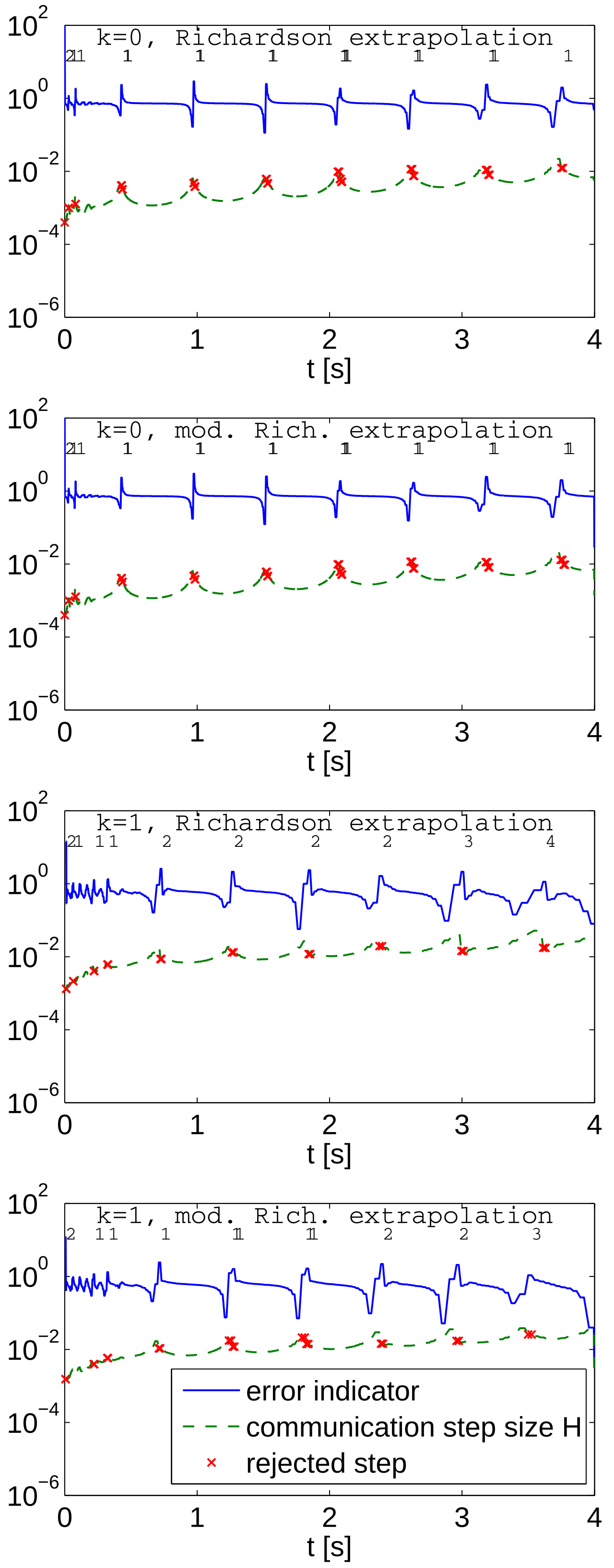

Figure 5: Benchmark Quarter car, displacement-displacement coupling. Number of step rejections, error indicator and communication step size of the simulation with error estimates $\mathbf{EST}_{\mathrm{Rich}}$ and $\mathbf{EST}_{\mathrm{mod}}$.

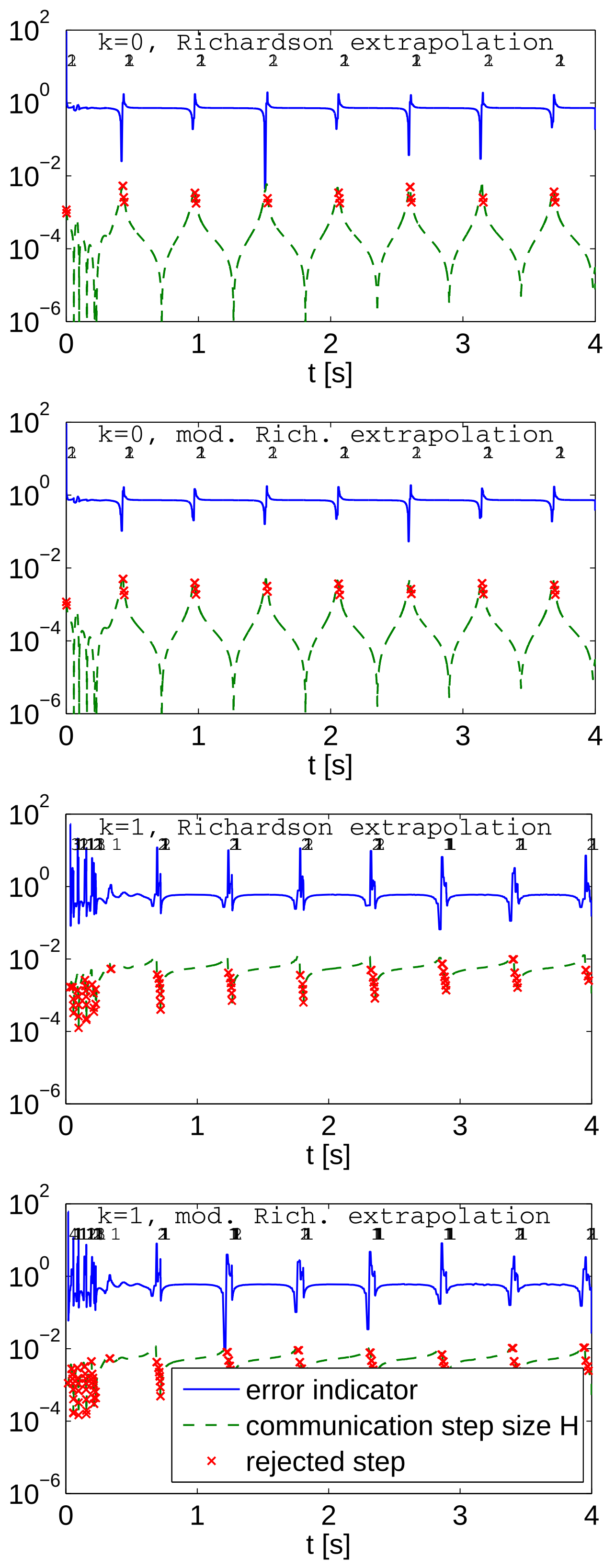

Figure 6: Benchmark Quarter car, displacement-force coupling. Number of step rejections, error indicator and communication step size of the simulation with error estimates $\mathbf{EST}_{\mathrm{Rich}}$ and $\mathbf{EST}_{\mathrm{mod}}$.

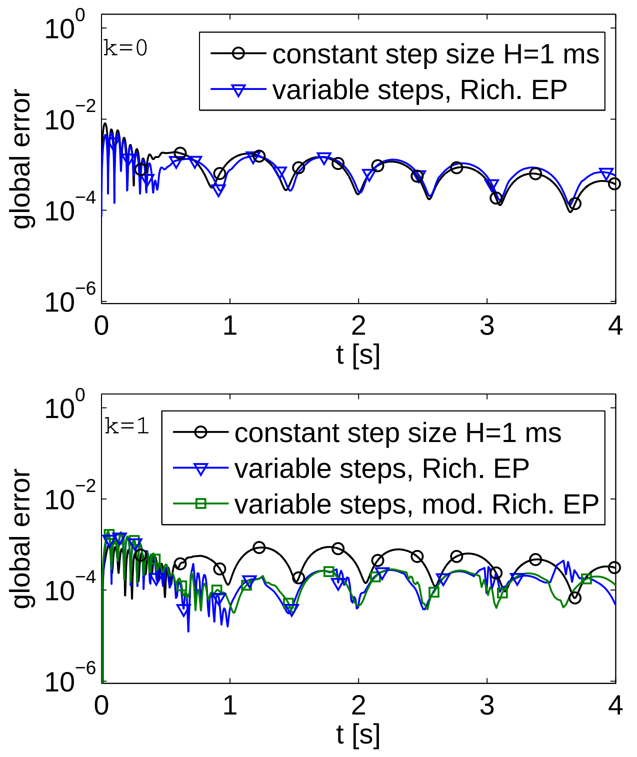

Figure 7: Benchmark Quarter car, displacement-displacement coupling. Global error of the simulation with constant step sizes compared to variable steps based on error estimates $\mathbf{EST}_{\mathrm{Rich}}$ and $\mathbf{EST}_{\mathrm{mod}}$.

issues, since the first (small) Richardson step is saved and therefore the inputs at $t = T_n + H$ have to be interpolated which causes problems if in the big step $T_n \rightarrow T_{n+2}$ only one micro step is taken.

The global error of the numerical solution with step size control is very well controlled to a mean value of 10^{-4}, see Figure 7, which corresponds to the predefined error bounds $\mathrm{ATOL}_j = \mathrm{RTOL}_j = 10^{-4}$ for displacement-displacement coupling (in the case of displacement-force coupling we use $\mathrm{ATOL}_j = \mathrm{RTOL}_j = 10^{-3}$). In the transient phase $t \in [0, 0.5]$ very small communication steps are chosen and at later simulation time, the algorithm uses larger communication steps than in the beginning, since the subsystems behave slower and the distance between two communication points for an update of the subsystem inputs is increased. In time intervals, where the larger mass of the chassis has a strong influence on the wheel by changing the direction of motion, this is also triggered by the communication step size control resulting in a reduction of the step size such that the error bounds are met (see slow oscillation of the communication step size and also the rejected steps in Figure 5, where the communication steps are repeated with smaller step size). In the transient phase, we can see that the error is greater than the pre-defined error bound of 10^{-4}. In this phase smaller steps should be taken, which is triggered correctly by the step size control algorithm. If we compare the simulation results with communication step size control with $\mathbf{EST}_{\mathrm{Rich}}$ and $\mathbf{EST}_{\mathrm{mod}}$ to the simulation with constant communication step size $H = 10^{-3}$ (with a micro tolerance in the subsystems of 10^{-6}) in Figure 7 and Table 1, we can see, that the global error for constant extrapolation is decreasing during the simulation because the step size is not adapted to the solution behaviour. The accuracy is raised, if higher order extrapolations are used. We can also compare the computing times and see that the master algorithm with communication step size control is much faster resulting in a high efficiency (nearly the same mean global error in the simulation time interval compared to constant step sizes with constant extrapolation), even with Richardson extrapolation, where in every communication step the simulation is performed at least twice. Using the modification $\mathbf{EST}_{\mathrm{mod}}$ or a higher order of extrapolation ($k \geq 1$) further improves the simulation results and the computing time. This is a nice example for the advantage of controlling the step size compared to the brute force approach of using always constant communication step sizes.

Furthermore, we observe in Table 1 if we use a micro tolerance of 10^{-8} in the subsystems for the simulation with constant step sizes $H = 10^{-3}$ that the computation time is growing. Moreover, the accuracy of the simulation is improved, since the influence of the discretization error of the solution of the subsystems is reduced. The simulation with step size control with `micTOL=1e-6` on the other hand is robust and reliable and controls the error of the simulation results to the predefined tolerance of `macTOL=1e-4`, even if the influence of the discretization error of the subsystems is not neglected.

6 Conclusions

We have discussed the error estimation in co-simulation by classical Richardson extrapolation and by a modified algorithm for a reliable communication step size control based on an extension of the step size control of classical time integration. The local error of the simulation is estimated very well by these strategies.

The communication step size control was applied to a benchmark problem from vehicle dynamics which was implemented in a master prototype that is compatible to FMI for Co-Simulation v1.0. We have seen that communication step size control is possible, reliable and can improve the performance of the master algorithm significantly, especially the computing time and accuracy.

Table 1: Benchmark Quarter car - Simulation results for the displacement-displacement coupling.

	k	error	time [s]	`steps`	`rej`
$H = 1\,\mathrm{ms}$,	0	8.922E-004	33.335	4000	0
`micTOL=1e-6`	1	4.094E-004	29.295	4000	0
	2	4.138E-004	31.022	4000	0
$H = 1\,\mathrm{ms}$,	0	5.301E-004	51.280	4000	0
`micTOL=1e-8`	1	6.049E-005	43.364	4000	0
	2	1.967E-005	43.073	4000	0
Rich. EP,	0	1.050E-003	28.361	1610	18
`micTOL=1e-6,`	1	3.197E-004	10.379	402	20
`macTOL=1e-4`	2	3.180E-004	7.050	210	20
Mod. Rich. EP,	0	1.061E-003	19.712	1612	19
`micTOL=1e-6,`	1	3.317E-004	7.057	413	16
`macTOL=1e-4`	2	2.735E-004	4.835	218	17

References

[1] M. Arnold. Stability of sequential modular time integration methods for coupled multibody system models. *J. Comput. Nonlinear Dynam.*, 5:031003, 2010.

[2] M. Arnold, T. Blochwitz, C. Clauß, T. Neidhold, T. Schierz, and S. Wolf. FMI-for-CoSimulation. In *The International Journal of Multiphysics. Special Edition: Multiphysics Simulations. Advanced Methods for Industrial Engineering. Selected contributions from 1st Fraunhofer Multiphysics Conference*, pages 345–356, Brentwood, Essex, UK, 2011. Multi-Science Publishing Co. Ltd.

[3] M. Arnold, C. Clauß, and T. Schierz. Numerical aspects of FMI for Model Exchange and Co-Simulation v2.0. In P. Eberhard and P. Ziegler, editors, *Proc. of The 2nd Joint International Conference on Multibody System Dynamics, Stuttgart, Germany, May 29 - June 1, 2012*, ISBN 978-3-927618-32-9, 2012.

[4] J. Bastian, C. Clauß, S. Wolf, and P. Schneider. Master for Co-Simulation Using FMI. In C. Clauß, editor, *Modelica Association, Linköping: 8th International Modelica Confer-*

ence 2011 : Dresden, Germany, 20-22 March 2011, Dresden: Fraunhofer IIS / EAS, 2011.

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

[6] C. Clauß, M. Arnold, T. Schierz, and J. Bastian. Master zur Simulatorkopplung via FMI. In X. Liu-Henke, editor, *Tagungsband der ASIM/GI-Fachgruppen STS und GMMS, Wolfenbüttel, 23.02.-24.02.2012*, Ostfalia Hochschule für Angewandte Wissenschaften, Wolfenbüttel, 2012.

[7] P. Deuflhard and A. Hohmann. *Numerical Analysis in Modern Scientific Computing: An Introduction*. Number 43 in Texts in Applied Mathematics. Springer, 2nd edition, 2003.

[8] FMI. The functional mockup interface. `http://www.functional-mockup-interface.org/`.

[9] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations. I. Nonstiff Problems*. Springer–Verlag, Berlin Heidelberg New York, 2nd edition, 1993.

[10] K. Jackson. A survey of parallel numerical methods for initial value problems for ordinary differential equations. *IEEE Transactions on Magnetics*, 27:3792–3797, 1991.

[11] R. Kübler. *Modulare Modellierung und Simulation mechatronischer Systeme*. Fortschritt-Berichte VDI Reihe 20, Nr. 327. VDI–Verlag GmbH, Düsseldorf, 2000.

[12] R. Kübler and W. Schiehlen. Two methods of simulator coupling. *Mathematical and Computer Modelling of Dynamical Systems*, 6:93–113, 2000.

[13] Modelisar. Functional Mock-up Interface for Model Exchange and Co-Simulation v2.0 Beta 3. `http://www.functional-mockup-interface.org/specifications/FMI_for_ModelExchange_and_CoSimulation_v2.0_Beta3.pdf`, November 2011.

[14] Modelisar. Functional Mock-up Interface for Co-Simulation. `http://www.functional-mockup-interface.org/specifications/FMI_for_CoSimulation_v1.0.pdf`, October 2010.

[15] H. Olsson. Private communication within FMI 2.0 development, June 2011.