Extension of the Model-based Pre-step Stabilization Method for Non-iterative Co-simulation
Including Direct-feedthrough

Simon Genser and Martin Benedikt
Virtual Vehicle Research Center, Inffeldgasse 21a, Graz, Austria

Keywords: Pre-step Stabilization, Co-simulation, Weak Coupling, Interface Jacobians.

Abstract: For integration of subsystems co-simulation tools and master algorithms have to cope with the problem of restricted access to subsystem information. Especially for efficient handling of inherent properties like stiffness and direct feedthrough of industrial co-simulations additional subsystem information is mandatory. As provision of dedicated partial derivatives, especially regarding subsystem states, is complex and computationally expensive, recently a novel pre-step co-simulation approach for handling stiff system simulations based on formal subsystem transformations was proposed. The approach follows the idea of the Nearly Energy Preserving Coupling Element (NEPCE) scheme where exclusively subsystem inputs are modified for compensation of the co-simulation discretization error. This paper outlines a generalization of the presented pre-step co-simulation method for handling stiffness and direct feedthrough of subsystems. The generalized pre-step co-simulation algorithm is examined along a theoretical 2 DoF benchmark co-simulation example indicating outstanding co-simulation performance.

1 INTRODUCTION

Nowadays, usage of numerical simulation for design, analysis and engineering of technical systems is common practice in industry and a lot of different modelling approaches and tailored numerical solvers are available. When it comes to holistic system considerations typically subsystem simulations from different domains, departments or companies have to be integrated and co-simulation is the preferred solution (Arnold, 2006). This way, subsystems are handled as black-boxes with defined interfaces for the control and the exchange of inputs and outputs and higher-level integration schemes are necessary for synchronization purposes, i.e. master algorithms (Bastian et al., 2011). As in classical numerical simulation the evaluation of the right-hand side of the governing subsystem equations is crucial for efficient treatment of overall stiff co-simulations and related Jacobi matrices are typically used by the master algorithm when linear subsystem behaviour can be assumed. Accessing subsystem model information itself has to be supported by the simulation tools via enhanced co-simulation capabilities or by implementation of the Functional Mock up Interface, a standard for model exchange and integration for data exchange and control of subsystem simulations. Especially FMI 2.0 supports the exchange of partial derivatives of subsystem models.

Utilization of the right-hand side evaluations of subsystems – partial derivatives of subsystem models - by master algorithms is beneficial for improving overall co-simulation performance and, on the other hand, mandatory for handling dedicated properties and characteristics of the co-simulation at hand. As co-simulation is typically applied in cases where it is impossible to port and assemble the individual subsystem simulations into a standalone, multi-purpose simulation model and its related tool, due to missing modelling features or tailored solvers, rather complex subsystems have to be integrated. These subsystems most often possess properties like stiffness and direct feedthrough.

Similar to classical numerical simulation stiffness is addressed by implicit schemes or at least explicit ones with implicit behavior, e.g. linearly-implicit approaches (Rosenbrock W-methods) (Cellier and Kofman, 2010). Direct feedthrough represents a subsystem property which influences overall system stability. Assuming the assembled overall system is stable (Kuebler and Schiehlen, 2000), direct feedthroughs of individual subsystems introduce proportional input-output dependencies of subsystems, affecting performance[1] of the master algorithms too. Some approaches are published (Arnold, 2006; Sicklinger et al., 2013; Sadjina et al., 2016; Viel, 2014) to cope with direct feedthrough. In contrast, recently, another non-iterative approach was proposed based on formal subsystem transformations and the utilization of subsystems Interface Jacobians exclusively, referred to as NECPE-II (Genser and Benedikt, 2018), in order to reduce the computational burden on subsystem side. Within this paper the proposed algorithm is generalized for handling of subsystems direct feedthroughs. The paper is organized as follows: Section 2 contains the detailed derivation of the pre-step stabilization algorithm. In Section 3 the improved performance of the presented algorithm is illustrated by a 2 DoF benchmark co-simulation example.

2 PRE-STEP ALGORITHM

The aim of this paper is to generalize the model-based pre-step stabilization method (NEPCE-II) for the non-iterative co-simulation (Genser and Benedikt, 2018) in order to handle additionally direct feedthroughs of subsystems.

2.1 System Description

The state-space representation (Dorf and Bishop, 2017) of subsystem S_i:

$$\dot{x}_i(t) = A_i \cdot x_i(t) + B_i \cdot u_i(t) \tag{1}$$

$$y_i(t) = C_i \cdot x_i(t) + D_i \cdot u_i(t) \tag{2}$$

is a suitable option to include direct feedthrough. The condition:

$$D_i \neq 0,$$

for least an $i = 1, \dots, N$ assures the presence of a direct feedthrough, whereby N denotes the number of subsystems included in the overall co-simulation. Compared to the output-based system description in in (Genser and Benedikt, 2018), the herein used one (3) has an additional term, presented by $\dot{u}_i$:

$$\dot{y}_i = S_i(y_i, u_i, \dot{u}_i). \tag{3}$$

The proposed pre-step coupling algorithm is based on the linearization of (3) around $(0,0,0)$:

$$\dot{y}_i \approx S_i(0,0,0) + \frac{\partial S_i}{\partial y} \cdot y_i + \frac{\partial S_i}{\partial u} \cdot u_i + \frac{\partial S_i}{\partial \dot{u}} \cdot \dot{u}_i.$$

[1]Note: In the context of co-simulation, handling of direct feedthroughs is different to handling differential algebraic equations, as direct feedthroughs are not stating constraints on subsystem states.

Without loss of generality one can assume that $S_i(0,0,0) = 0$, which leads to the transformed linearized output-based subsystem description:

$$\dot{y}_i \approx \frac{\partial S_i}{\partial y} \cdot y_i + \frac{\partial S_i}{\partial u} \cdot u_i + \frac{\partial S_i}{\partial \dot{u}} \cdot \dot{u}_i. \tag{4}$$

As complete partial-derivatives are typically difficult to be accessed, in most cases a linear approximation of the subsystems is used. Especially this is specified by the Functional Mock up Interface 2.0 (Blochwitz, 2012). Therefore a connection between this representation an the utilized output-based description (4) is desirable. The derivation of this transformation starts with the time derivative of (2):

$$\dot{y}_i = C_i \cdot \dot{x}_i + D_i \cdot \dot{u}_i.$$

Substituting (1) leads to:

$$\dot{y}_i = C_i \cdot (A_i \cdot x_i + B_i \cdot u_i) + D_i \cdot \dot{u}_i. \tag{5}$$

Rearranging (2) and inserting into (5) results in:

$$\dot{y}_i = C_i \cdot \left(A_i \cdot C_i^{-1} \cdot (y_i - D_i \cdot u_i) + B_i \cdot u_i\right) + D_i \cdot \dot{u}_i.$$

From algebraic rearrangements follows:

$$\begin{aligned} \dot{y}_i = {} & \left[C_i \cdot A_i \cdot C_i^{-1}\right] \cdot y_i + \dots \\ & \dots \left[C_i \cdot B_i - C_i \cdot A_i \cdot C_i^{-1} \cdot D_i\right] \cdot u_i + D_i \cdot \dot{u}_i. \end{aligned}$$

Comparing this equation to (4) directly turns out the related transformation rules[2]:

$$\begin{aligned} \frac{\partial S_i}{\partial y} &= C_i \cdot A_i \cdot C_i^{-1}, \\ \frac{\partial S_i}{\partial u} &= C_i \cdot B_i - C_i \cdot A_i \cdot C_i^{-1} \cdot D_i, \\ \frac{\partial S_i}{\partial \dot{u}} &= D_i. \end{aligned}$$

In case of black-box subsystems, system identification methods are used to approximate (4) directly. There are different system identification methods available. The herein preferred ones, due to their ability to identify direct feedthrough, are the so-called subspace methods (Overschee and Moor, 1996; Trnka, 2005), especially the Multivariable Output-Error State Space (MOESP) algorithm, see (Katayama, 2005).

2.2 Derivation of the Algorithm

To keep the notation in the following derivation of the algorithm as simple as possible the following assumptions are made:

[2]For generalization purposes the term C_i^{-1} represents the pseudo-inverse.

- the macro-step size ΔT and the micro-step size δT are fixed for the whole computation time and for all subsystems (i.e. $\Delta T_i^k = \Delta T$, $\delta T_i^k = \delta T$);
- the overall co-simulation consists of two fully coupled subsystems (i.e. $u_1 = y_2$, $u_2 = y_1$ at the coupling instants and $N = 2$);
- the input and output signals u, y are scalars for both subsystems;
- the output y describes a time continuous signal.

As outlined in (Genser and Benedikt, 2018) also the generalized algorithm contains three main steps as illustrated in Figure 1:

1. approximate the exact, monolithic output $\tilde{y}$ utilizing the global *Error Differential Equation*;
2. perform a global model-based extrapolation $\hat{y}$ of the output;
3. based on the extrapolation $\hat{y}$ perform a local optimization of the input u for every subsystem individually.

From an abstract point of view these steps basically comply with the ones in (Genser and Benedikt, 2018) but differs in important details. To outline the differences the complete derivation of the main steps is given in the subsections below.

2.2.1 Step 1: Solving the Error Differential Equation

To approximate the exact, monolithic solution $\tilde{y}_i$ over the last macro-step $[T^{k-1}, T^k]$, see Step 1 in Figure 1, of a subsystem the so-called Error Differential Equation is solved and its derivation is based on the subsystem description (3) and two important cases:
Ideal coupling of the two subsystems:

$$\dot{\tilde{y}}_1 = S_1(\tilde{y}_1, \tilde{y}_2, \dot{\tilde{y}}_2) \tag{6}$$

$$\dot{\tilde{y}}_2 = S_2(\tilde{y}_2, \tilde{y}_1, \dot{\tilde{y}}_1) \tag{7}$$

Coupling by co-simulation:

$$\dot{y}_1 = S_1(y_1, u_1, \dot{u}_1) \tag{8}$$

$$\dot{y}_2 = S_2(y_2, u_2, \dot{u}_2) \tag{9}$$

Comparing (6) with (8):

$$\dot{y}_1 - S_1(y_1, u_1, \dot{u}_1) = \dot{\tilde{y}}_1 - S_1(\tilde{y}_1, \tilde{y}_2, \dot{\tilde{y}}_2),$$

rearranging and the definition of $\delta_i := \tilde{y}_i - y_i$, see Figure 1, leads to:

$$S_1(\tilde{y}_1, \tilde{y}_2, \dot{\tilde{y}}_2) - S_1(y_1, u_1, \dot{u}_1) = \underbrace{\dot{\tilde{y}}_1 - \dot{y}_1}_{\dot{\delta}_1}. \tag{10}$$

Rearranging and substituting $\varepsilon_1 := y_2 - u_1$ and $\varepsilon_2 := y_1 - u_2$, see Figure 1, into (10) leads to:

$$\begin{aligned}\dot{\delta}_1 =& S_1(u_2 + \varepsilon_2 + \delta_1, y_2 + \delta_2, \dot{y}_2 + \dot{\delta}_2) - \ldots \\ &\ldots S_1(u_2 + \varepsilon_2, y_2 - \varepsilon_1, \dot{y}_2 - \dot{\varepsilon}_1)\end{aligned}$$

Evaluating[3] the equation in T^k leads to:

$$\begin{aligned}\dot{\delta}_1 =& S_1(u_2^k + \varepsilon_2 + \delta_1, y_2^k + \delta_2, \dot{y}_2^k + \dot{\delta}_2) - \ldots \\ &\ldots S_1(u_2^k + \varepsilon_2, y_2^k - \varepsilon_1, \dot{y}_2^k - \dot{\varepsilon}_1).\end{aligned}$$

From the spatial linearization based on the Taylor-series, with the center of the series in ($u_2^k := u_2(T^k)$, $y_2^k := y_2(T^k)$, $\dot{y}_2^k := \dot{y}_2(T^k)$) it follows

$$\begin{aligned}&S_1(u_2^k, y_2^k, \dot{y}_2^k) + \frac{\partial S_1}{\partial y} \cdot (\varepsilon_2 + \delta_1) + \frac{\partial S_1}{\partial u} \cdot \delta_2 + \ldots \\ &\ldots \frac{\partial S_1}{\partial \dot{u}} \cdot \dot{\delta}_2 - S_1(u_2^k, y_2^k, \dot{y}_2^k) - \frac{\partial S_1}{\partial y} \cdot (\varepsilon_2) + \ldots \\ &\ldots \frac{\partial S_1}{\partial u} \cdot \varepsilon_1 + \frac{\partial S_1}{\partial \dot{u}} \cdot \dot{\varepsilon}_1 \approx \dot{\delta}_1.\end{aligned}$$

From algebraic rearrangements follows the final ordinary *Error Differential Equation*

$$\dot{\delta}_1 \approx \frac{\partial S_1}{\partial y} \cdot \delta_1 + \frac{\partial S_1}{\partial u} \cdot [\delta_2 + \varepsilon_1] + \frac{\partial S_1}{\partial \dot{u}} \cdot [\dot{\delta}_2 + \dot{\varepsilon}_1]. \tag{11}$$

Via symmetry for (7) and (9) it follows:

$$\dot{\delta}_2 \approx \frac{\partial S_2}{\partial y} \cdot \delta_2 + \frac{\partial S_2}{\partial u} \cdot [\delta_1 + \varepsilon_2] + \frac{\partial S_2}{\partial \dot{u}} \cdot [\dot{\delta}_1 + \dot{\varepsilon}_2]. \tag{12}$$

The fact that the two ordinary differential equations above are coupled motivates to write them in vector and matrix notation

$$\begin{aligned}\underbrace{\begin{pmatrix}\dot{\delta}_1 \\ \dot{\delta}_2\end{pmatrix}}_{=:\dot{\boldsymbol{\delta}}} =& \underbrace{\begin{pmatrix}\frac{\partial S_1}{\partial y} & \frac{\partial S_1}{\partial u} \\ \frac{\partial S_2}{\partial u} & \frac{\partial S_2}{\partial y}\end{pmatrix}}_{=:\tilde{A}} \cdot \underbrace{\begin{pmatrix}\delta_1 \\ \delta_2\end{pmatrix}}_{=:\boldsymbol{\delta}} + \underbrace{\begin{pmatrix}\frac{\partial S_1}{\partial u} & 0 \\ 0 & \frac{\partial S_2}{\partial u}\end{pmatrix}}_{=:\tilde{B}} \cdot \underbrace{\begin{pmatrix}\varepsilon_1 \\ \varepsilon_2\end{pmatrix}}_{=:\boldsymbol{\varepsilon}} \\ &+ \underbrace{\begin{pmatrix}0 & \frac{\partial S_1}{\partial \dot{u}} \\ \frac{\partial S_2}{\partial \dot{u}} & 0\end{pmatrix}}_{=:\tilde{C}} \cdot \underbrace{\begin{pmatrix}\dot{\delta}_1 \\ \dot{\delta}_2\end{pmatrix}}_{=:\dot{\boldsymbol{\delta}}} + \underbrace{\begin{pmatrix}\frac{\partial S_1}{\partial \dot{u}} & 0 \\ 0 & \frac{\partial S_2}{\partial \dot{u}}\end{pmatrix}}_{=:\tilde{D}} \cdot \underbrace{\begin{pmatrix}\dot{\varepsilon}_1 \\ \dot{\varepsilon}_2\end{pmatrix}}_{=:\dot{\boldsymbol{\varepsilon}}}.\end{aligned}$$

Rearranging this equation leads to

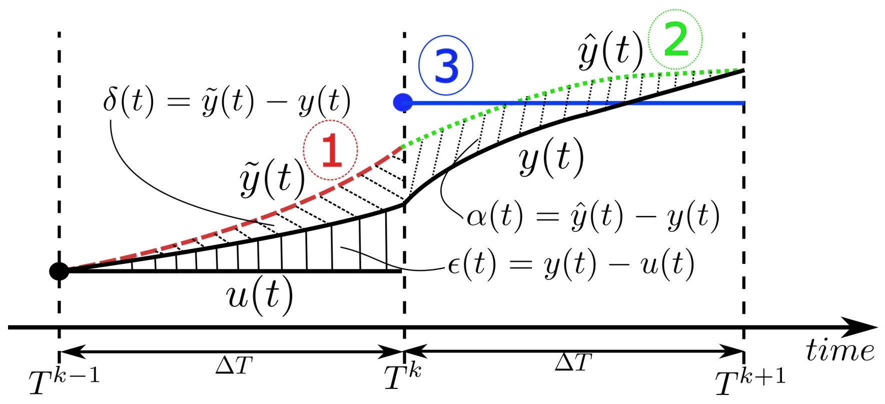
$$(I - \tilde{C}) \cdot \dot{\boldsymbol{\delta}} = \tilde{A} \cdot \boldsymbol{\delta} + \tilde{B} \cdot \boldsymbol{\varepsilon} + \tilde{D} \cdot \dot{\boldsymbol{\varepsilon}},$$

where I denotes an identity matrix from appropriate dimension. With the use of the pseudo inverse $(I - \tilde{C})^{-1}$ the final Error Differential Equation for results in

$$\dot{\boldsymbol{\delta}} = (I - \tilde{C})^{-1} \cdot \left[\tilde{A} \cdot \boldsymbol{\delta} + \tilde{B} \cdot \boldsymbol{\varepsilon} + \tilde{D} \cdot \dot{\boldsymbol{\varepsilon}}\right]. \tag{13}$$

Note: Due to the definition of δ and the assumption that y is continuous it follows that δ is continuous as well. This leads naturally to the initial conditions for δ, combining this with the equation above leads to a initial value problem for the computation of the exact output $\tilde{y}$ in $[T^{k-1}, T^k]$.

[3]For δ_i and ε_i the sub index k is omitted for sake of simplicity.



$\hat{y}(t)$
2
3
$\delta(t) = \tilde{y}(t) - y(t)$
1
$\tilde{y}(t)$
$y(t)$
$\alpha(t) = \hat{y}(t) - y(t)$
$\epsilon(t) = y(t) - u(t)$
$u(t)$
time
ΔT
ΔT
T^{k-1}
T^k
T^{k+1}

Figure 1: Illustration of the model-based pre-step stabilization algorithm (Genser and Benedikt, 2018).

2.2.2 Step 2: Model-based Extrapolation

The model-based extrapolation of the overall system ensures the implicit behaviour of the algorithm. The idea is to predict the future progress of the exact solution $\hat{y}$ over the next macro-step ΔT, depicted as Step 2 in Figure 1. By the exact solution is the monolithic solution meant, i.e. all errors due to the weak coupling vanish. The differential equation for the extrapolation is based on the assumption of ideal coupling between the subsystems and the system description (4).
The derivation of the differential equation starts with

$$\dot{\hat{y}}_i \approx \frac{\partial S_i}{\partial y} \cdot \hat{y}_i + \frac{\partial S_i}{\partial u} \cdot u_i + \frac{\partial S_i}{\partial \dot{u}} \cdot \dot{u}_i \text{ for } i = 1,2. \quad (14)$$

Assuming ideal coupling between subsystems

$$\hat{y}_1 = u_2,$$
$$\hat{y}_2 = u_1$$

and inserting this in equation (14) leads to:

$$\underbrace{\begin{pmatrix}\dot{\hat{y}}_1 \\ \dot{\hat{y}}_2\end{pmatrix}}_{=:\dot{\hat{\mathbf{y}}}} = \underbrace{\begin{pmatrix}\frac{\partial S_1}{\partial y} & \frac{\partial S_1}{\partial u} \\ \frac{\partial S_2}{\partial u} & \frac{\partial S_2}{\partial y}\end{pmatrix}}_{=:\hat{A}} \cdot \underbrace{\begin{pmatrix}\hat{y}_1 \\ \hat{y}_2\end{pmatrix}}_{=:\hat{\mathbf{y}}} + \underbrace{\begin{pmatrix}0 & \frac{\partial S_1}{\partial \dot{u}} \\ \frac{\partial S_2}{\partial \dot{u}} & 0\end{pmatrix}}_{=:\hat{B}} \cdot \underbrace{\begin{pmatrix}\dot{\hat{y}}_1 \\ \dot{\hat{y}}_2\end{pmatrix}}_{=:\dot{\hat{\mathbf{y}}}}$$

Rearranging this equation and utilizing the pseudo-inverse $(I - \hat{B})^{-1}$ results in the final differential equation:

$$\dot{\hat{\mathbf{y}}} = (I - \hat{B})^{-1} \cdot \hat{A} \cdot \hat{\mathbf{y}}. \quad (15)$$

Note: The initial condition for (15) comes from the solution of the Error Differential Equation, i.e. $\hat{y}(T^k) = \tilde{y}(T^k)$ for the extrapolation over $[T^k, T^{k+1}]$, leading to a initial value problem for every macro time step.

2.2.3 Step 3: Pre-step Input Optimization

The local optimization of the input u_i represents the third main part of the presented algorithm, for illustration see Figure 1. Local means that this part can be computed for every subsystem individually and independent from the other subsystems, therefore the sub index i is omitted in this subsection. The optimization is based on the error:

$$\alpha := \hat{y} - y. \quad (16)$$

The idea is to choose the input u so that α is minimized over the next macro-step:

$$\min_{t \in [T^k, T^{k+1}]} |\alpha(t)| \quad (17)$$

There are different ways to solve the optimization problem (17), herein the focus lies on a discretized solution, based on the macro-step, i.e. (17) is transformed into:

$$|\alpha^{k+1}| \overset{!}{=} 0, \text{ with } \alpha^{k+1} := \alpha(T^{k+1}) \quad (18)$$

The discretization (18) of the optimization problem leads, with the definition (16), to

$$\hat{y}^{k+1} \overset{!}{=} y^{k+1}, \quad (19)$$

where $\hat{y}^{k+1} := \hat{y}(T^{k+1})$ and $y^{k+1} := y(T^{k+1})$. For explicitly solving this problem the dependency of the output y^{k+1} on the input u_k has to be exploited. To describe this relation a close look on the system description (4) and standard theory of ordinary differential equations is needed. The analytical solution of (4) is described through:

$$\begin{aligned} y(t) = e^{\frac{\partial S}{\partial y} \cdot t} \cdot y^0 + \int_{t_{start}}^{t} e^{\frac{\partial S}{\partial y} \cdot (t-\tau)} \cdot \frac{\partial S}{\partial u} \cdot u(\tau)\, d\tau + \dots \\ \dots \int_{t_{start}}^{t} e^{\frac{\partial S}{\partial y} \cdot (t-\tau)} \cdot \frac{\partial S}{\partial \dot{u}} \cdot \dot{u}(\tau)\, d\tau, \end{aligned} \quad (20)$$

where y^0 denotes the initial condition. The derivative $\dot{u}$ denotes the classical time derivative which is computed piecewise for every macro-step. Due to the appearance of $\dot{u}$ in (20) the choice of piecewise constant basic functions is a bad one because $\dot{u}$ would be zero all the time and so the second integral in (20) would be always neglected and this would lead to a worse approximation. That is the reason that piecewise linear basic functions are utilized to describe the input u in the following, i.e. the preferred approach for u is

$$u(t) = u_0^k + s^k \cdot t \text{ for } t \in [T^k, T^{k+1}], \tag{21}$$

where $u_0^k = y(T^k)$, with y representing the output from the coupled subsystem. The slope of u in the macro-step $[T^k, T^{k+1}]$ is denoted with s^k. Inserting this approach in (20) leads to

$$y(t) = e^{\frac{\partial S}{\partial y} \cdot t} \cdot y^0 + \sum_{l=0}^{k} \int_{T^l}^{T^{l+1}} e^{\frac{\partial S}{\partial y} \cdot (t-\tau)} \cdot \frac{\partial S}{\partial u} \cdot \ldots$$

$$\ldots (u_0^l + s^l \cdot \tau)\, d\tau + \sum_{l=0}^{k} \int_{T^l}^{T^{l+1}} e^{\frac{\partial S}{\partial y} \cdot (t-\tau)} \cdot \frac{\partial S}{\partial \dot{u}} \cdot s^l \, d\tau.$$

The separation and summation of the integrals in the macro-steps is necessary due to the piecewise definition of u in (21). With $T^0 = t_{start}$ the starting time of the overall co-simulation is denoted and for the lack of simplicity ΔT and k are chosen so that $T^{k+1} = t$ holds. Rearranging the last equation results in

$$y(t) = e^{\frac{\partial S}{\partial y} \cdot t} \cdot y^0 + \sum_{l=0}^{k} u_0^l \cdot \int_{T^l}^{T^{l+1}} e^{\frac{\partial S}{\partial y} \cdot (t-\tau)} \cdot \frac{\partial S}{\partial u} \, d\tau + \ldots \tag{22}$$

$$\ldots \sum_{l=0}^{k} s^l \cdot \int_{T^l}^{T^{l+1}} e^{\frac{\partial S}{\partial y} \cdot (t-\tau)} \cdot \frac{\partial S}{\partial u} \cdot \tau \, d\tau + \ldots \tag{23}$$

$$\ldots \sum_{l=0}^{k} s^l \cdot \int_{T^l}^{T^{l+1}} e^{\frac{\partial S}{\partial y} \cdot (t-\tau)} \cdot \frac{\partial S}{\partial \dot{u}} \, d\tau, \tag{24}$$

with

$$\phi_A(t) := e^{\frac{\partial S}{\partial y} \cdot t},$$

$$\phi_{B_l}(t) := \int_{T^l}^{T^{l+1}} e^{\frac{\partial S}{\partial y} \cdot (t-\tau)} \cdot \frac{\partial S}{\partial u} \, d\tau,$$

$$\phi_{C_l}(t) := \int_{T^l}^{T^{l+1}} e^{\frac{\partial S}{\partial y} \cdot (t-\tau)} \cdot \frac{\partial S}{\partial u} \cdot \tau \, d\tau,$$

$$\phi_{D_l}(t) := \int_{T^l}^{T^{l+1}} e^{\frac{\partial S}{\partial y} \cdot (t-\tau)} \cdot \frac{\partial S}{\partial \dot{u}} \, d\tau,$$

it is possible to write the analytic solution of y as

$$y(t) = \phi_A(t) \cdot y^0 + \sum_{l=0}^{k} u_0^l \cdot \phi_{B_l}(t) + \ldots$$

$$\ldots s^l \cdot \left(\phi_{C_l}(t) + \phi_{D_l}(t) \right). \tag{25}$$

To fulfil (19) it is necessary to evaluate (25) in T^{k+1}, this leads to

$$y^{k+1} = \phi_A(T^{k+1}) \cdot y^0 + \sum_{l=0}^{k} u_0^l \cdot \phi_{B_l}(T^{k+1}) + \ldots$$

$$\ldots \sum_{l=0}^{k} s^l \cdot \left(\phi_{C_l}(T^{k+1}) + \phi_{D_l}(T^{k+1}) \right).$$

Recursive inserting of the equation above leads to

$$y^{k+1} = \phi_A(\Delta_T) \cdot y^k + u_0^k \cdot \phi_{B_k}(T^{k+1}) + \ldots$$

$$\ldots s^k \cdot \left(\phi_{C_k}(T^{k+1}) + \phi_{D_k}(T^{k+1}) \right). \tag{26}$$

Remark: $\phi_A(\Delta_T), \phi_{B_k}(T^{k+1}), \ldots, \phi_{D_k}(T^{k+1})$ denote matrices with constant coefficients. Similar to the transitions matrices for the standard state-space representation see (Dorf and Bishop, 2017).
Therefore (26) can be seen as an algebraic computation equation, the time variable and the sub index k are omitted, denoted as

$$y^{k+1} = \phi_A \cdot y^k + \phi_B \cdot u_0^k + s^k \cdot (\phi_C + \phi_D). \tag{27}$$

Inserting (27) into (19) results in

$$\hat{y}^{k+1} = \phi_A \cdot y^k + \phi_B \cdot u_0^k + s^k \cdot (\phi_C + \phi_D) \tag{28}$$

As it is the case, that y^k and u_0^k is known from the previous macro time increment and $\hat{y}^{k+1}$ is given due to the extrapolation, see Section 2.3.2, it is now possible to determine s^k through

$$s^k = (\phi_C + \phi_D)^{-1} \cdot \left(\hat{y}^{k+1} - \phi_A \cdot y^k + \phi_B \cdot u_0^k \right). \tag{29}$$

It should be mentioned that, this input optimization is computed locally and can therefore be easily performed in parallel.
Remark: Comparing this derivation herein with the basic algorithm in (Genser and Benedikt, 2018) leads to the conclusion that a direct feedthrough in a subsystem leads to additional terms in the subsystem description, see (4). These additional terms are the reason for the differences in the derivation of the three main steps. Setting the direct feedthrough terms to zero, the algorithm is exactly the same as in the one in (Genser and Benedikt, 2018), therefore the herein presented method is a true generalization of model-based pre-step stabilization. The generalization of this algorithm to a co-simulation consisting of more than two subsystems is straightforward, utilizing the coupling matrix L ($L \cdot y = u$) properly.

3 THEORETICAL EXAMPLE

To illustrate the improved performance of the herein presented algorithm a force-displacement coupled two-degrees of freedom oscillator has been chosen, see (Benedikt et al., 2013; Busch, 2012). The classical First-Order-Hold (FOH), the linearly-implicit stabilization (Arnold, 2011) and the energy-preserving approach (NEPCE) (Benedikt and Hofer, 2013) are utilized to compare the presented pre-step coupling algorithm with state of the art methods.

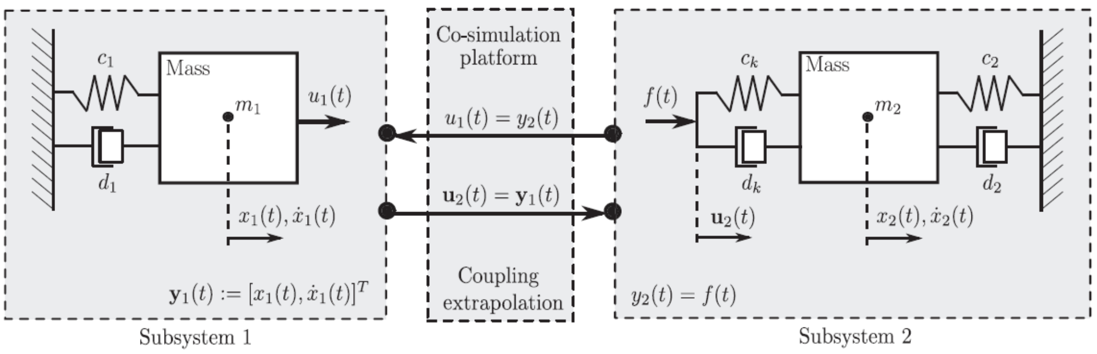


Figure 2: Co-simulation setup of a coupled 2-DOF oscillator, see (Benedikt et al., 2013).

3.1 Two Degree-of-Freedom Oscillator

The preferred example represents a two degree-of-freedom (DoF) mechanical oscillator and is separated into two subsystems (Fig. 2), including a direct feedthrough in the second subsystem; the state-space representation of the subsystems is given by:

$$\dot{\boldsymbol{x}}_1 = A_1 \cdot \boldsymbol{x}_1 + B_1 \cdot u_1, \tag{30}$$

$$\boldsymbol{y}_1 = C_1 \cdot \boldsymbol{x}_1 \tag{31}$$

and

$$\dot{\boldsymbol{x}}_2 = A_2 \cdot \boldsymbol{x}_2 + B_2 \cdot \boldsymbol{u}_2, \tag{32}$$

$$y_2 = C_2 \cdot \boldsymbol{x}_2 + D_2 \cdot \boldsymbol{u}_2 \tag{33}$$

with

$$\boldsymbol{x}_1 = \begin{pmatrix} x_1 \\ \dot{x}_1 \end{pmatrix}, \boldsymbol{x}_2 = \begin{pmatrix} x_2 \\ \dot{x}_2 \end{pmatrix}, A_1 = \begin{pmatrix} 0 & 1 \\ -\frac{c_1}{m_1} & -\frac{d_1}{m_1} \end{pmatrix},$$

$$A_2 = \begin{pmatrix} 0 & 1 \\ -\frac{c_2+c_k}{m_2} & -\frac{d_2+d_k}{m_2} \end{pmatrix}, B_1 = \begin{pmatrix} 0 \\ \frac{1}{m_1} \end{pmatrix},$$

$$B_2 = \begin{pmatrix} 0 & 0 \\ \frac{k}{m_2} & \frac{d_k}{m_2} \end{pmatrix}, C_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$

$$C_2 = \begin{pmatrix} c_k & d_k \end{pmatrix}, D_2 = \begin{pmatrix} -c_k & -d_k \end{pmatrix}.$$

Choosing parameters like:

$$m_1 = 10\,kg,\ m_2 = 10\,kg,\ c_1 = 10^6\,Nm^{-1},$$
$$c_2 = 10^7\,Nm^{-1}, d_1 = 1\,Nsm^{-1},\ d_2 = 2\,Nsm^{-1},$$
$$c_k = 10^5\,Nm^{-1},\ d_k = 10^3\,Nsm^{-1},$$
$$t_{start} = 0,\ t_{end} = 0.5,$$

results in an overall stiffness:

$$\frac{\lambda_{max}}{\lambda_{min}} = \frac{-10^4}{-50} = 200.$$

The initial conditions are set to:

$$x_1^0 = 0.1, \quad \dot{x}_1^0 = 0, \quad x_2^0 = 0, \quad \dot{x}_2^0 = 0. \tag{34}$$

The constant macro-step size is varied for evaluation purposes within the interval $\Delta T \in [10^{-5}, 8 \cdot 10^{-3}]$; the fixed micro-step size is set to $\delta T = 10^{-5}$. The reference solution y_{ref} is determined individually by utilizing a monolithic simulation with $\Delta T = \delta T$. Therefore, the monolithic overall system is derived from the subsystem description (30)-(33) assuming ideal couplings $u_1 = y_2$ and $u_2 = y_1$:

$$\begin{pmatrix} \dot{\boldsymbol{x}}_1 \\ \dot{\boldsymbol{x}}_2 \end{pmatrix} = \begin{pmatrix} A_1 + B_1 \cdot D_2 \cdot C_1 & B_1 \cdot C_2 \\ B_2 \cdot C_1 & A_2 \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{pmatrix}.$$

The numerical reference solution is determined by using the explicit Euler method.

3.2 Performance Evaluation

For comparison purposes the following Figures 3 to 7 illustrate the force coupling variable $y_2(t)$ exchanged between the two subsystems.
In a first aspect the impact of utilizing piecewise linear (Fig. 3) or piecewise constant basis functions (Fig. 4) for the extrapolated inputs is examined. By comparing both figures piecewise constant basis functions leads to increased co-simulation discretization errors. The significant discontinuous steps in the force signal at coupling time instances are caused by the steps in the input signal and the existing direct feedthrough ($D_2 \neq 0$) in Subsystem 2. A second reason of using piecewise linear basis functions is motivated by the generalization of the pre-step stabilization method where the time derivative of the input $\dot{u}$ appears, e.g. see (4) and (13). As the usage of piecewise constant basis functions ($\dot{u} = 0$) will eliminate the generalized terms piecewise linear basis functions have been used for the following computations.
With respect to the classical NEPCE method, in contrast to piecewise linear functions (Fig. 5), piecewise constant functions (Fig. 6) indicate improved stability. In case of classical NEPCE applications piecewise constant functions are utilized for comparison.

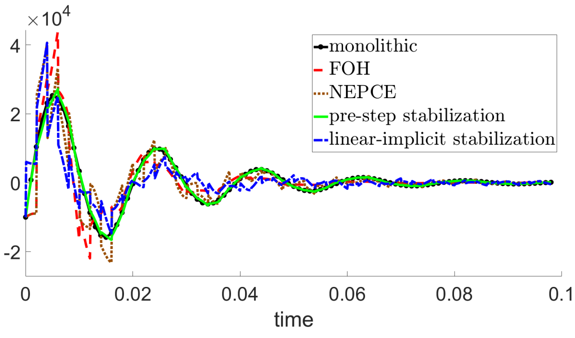


Figure 3: $\Delta T = 0.002$; output y_2 by FOH input extrapolation.

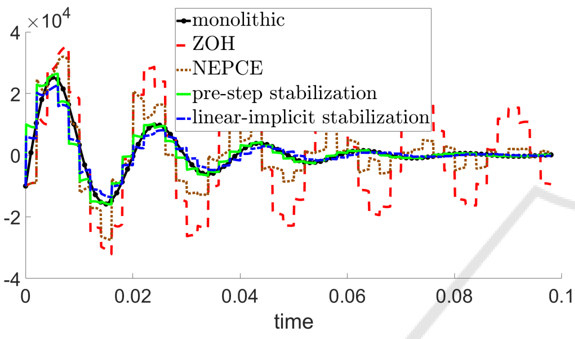


Figure 4: $\Delta T = 0.002$; output y_2 by ZOH input extrapolation.

In Figure 5 y_2 is illustrated for a macro step-size of $\Delta T = 0.00305$, which indicates the stability border of the FOH coupling approach. Figure 6 illustrates results for $\Delta T = 0.0037$ and indicates the stability border of the NEPCE coupling (Benedikt and Hofer, 2013). The stability limit of the linearly-implicit stabilization (Arnold, 2011) is experimentally deteremined at a macro-step-size of $\Delta T = 0.008$, see Fig. 7. By considering all cases, Fig. 5 to 7, the herein presented pre-step stabilization method outperforms state of the art algorithms, also in the case of eventually existing direct feedthroughs in subsystems.
For pointing out the significantly improved stability behavior of the proposed pre-step stabilization algorithm in detail, a logarithmic error plot, depending on the macro step-size ΔT is depicted in Figure 8. For defining a meaningful error measure, the following relative error is introduced:

$$err := \sum_i \left(\frac{1}{norm_i} \cdot \max_{t \in [t_{start}, t_{end}]} |y_i^{ref}(t) - y_i(t)| \right), \qquad (35)$$

with

$$norm_i := \max_{t \in [t_{start}, t_{end}]} [y_i^{ref}(t)] - \min_{t \in [t_{start}, t_{end}]} [y_i^{ref}(t)].$$

As the different outputs may vary in their magnitudes the constant $norm_i$, describing the ranges of the outputs y_i, is introduced for normalization purposes in

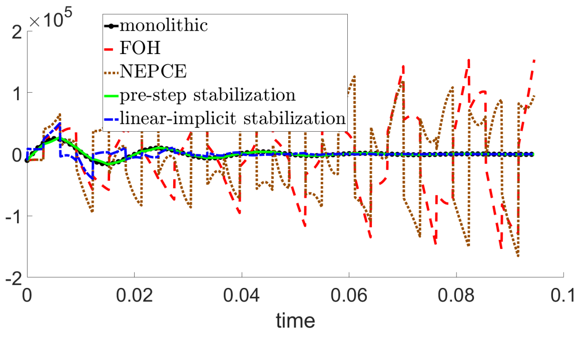


Figure 5: Output y_2; stability border $\Delta T = 0.00305$ for FOH coupling; NEPCE solution with FOH.

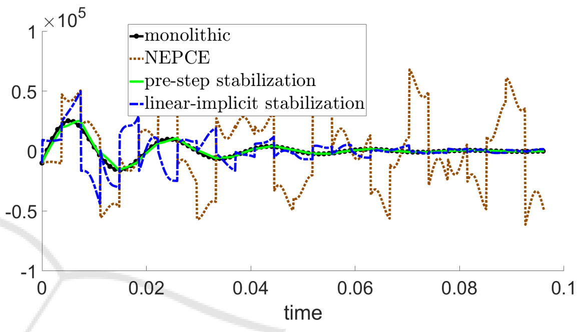


Figure 6: Output y_2; stability border $\Delta T = 0.0037$ for NEPCE coupling (Benedikt and Hofer, 2013); NEPCE solution with ZOH.

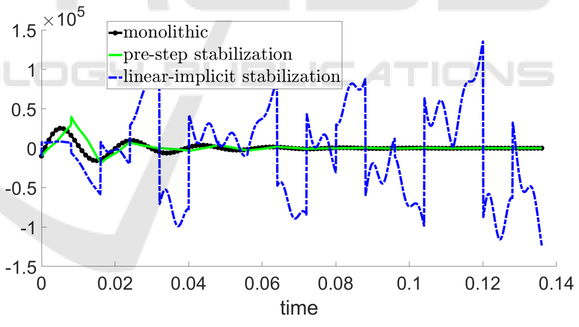


Figure 7: Output y_2; stability border $\Delta T = 0.008$ for linear implicit stabilization coupling (Arnold, 2011).

(35). It is worthy to mention that in the computation of the error depicted in Figure 8 all three different outputs $(\mathbf{y}_1, y_2)$ of the subsystems are considered. As one is interested to analyze the stability of the different coupling algorithms the simulation time is extended to $t_{end} = 0.5\ s$, so instabilities which are growing over the time are higher weighted. Figure 8 turns out following conclusions:

- an improved stability of the pre-stabilization approach using piecewise linear functions (FOH) is pointed out within the interval of $\Delta T \in [10^{-4}, 10^{-2}]$ compared to classical FOH, NEPCE and the linearly-implicit couplings.

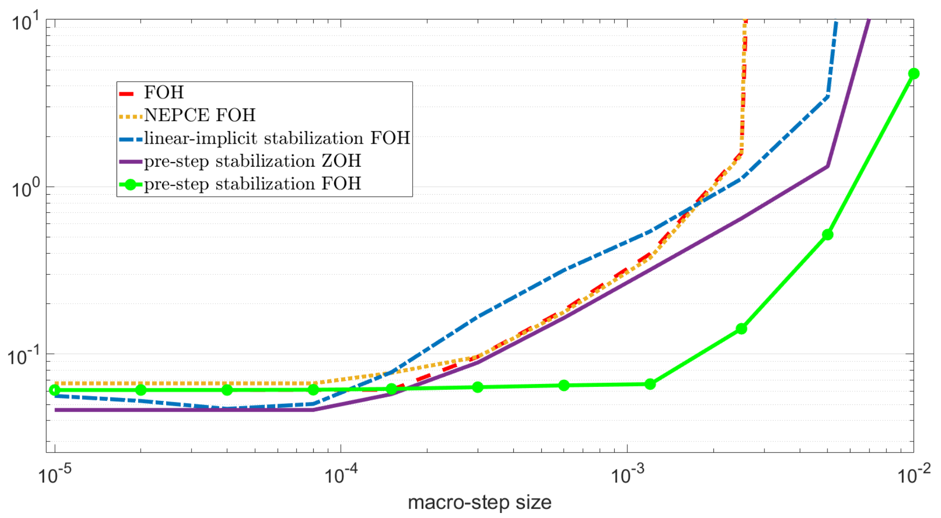


Figure 8: Logarithmic error plot.

- a significantly improved stability of the pre-stabilization approach using piecewise linear functions (FOH) is confirmed within the interval of $\Delta T \in [10^{-4}, 10^{-2}]$, compared to the pre-stabilization approach using piecewise constant functions (ZOH).
- compared to FOH, NEPCE coupling and the linearly-implicit stabilization the pre-step stabilization approaches (ZOH and FOH) are stable for macro-step sizes about 10^{-2}, as depicted in Figure 7 for $\Delta T = 0.008$.
- the analyzed coupling algorithms behave similar within the interval of $\Delta T \in [10^{-5}, 10^{-4}]$ where the error is dominated by the fixed micro-step size, cf. error estimation in (Arnold, 2006).
- the difference between the pre-step stabilization with ZOH and FOH within the interval of $\Delta T \in [10^{-5}, 10^{-4}]$ is noticeable; to be investigated in future work.

4 CONCLUSIONS

Direct feedthroughs of subsystems degrade co-simulation performance in general. This paper presents an extension to a recently proposed pre-step coupling algorithm for incorporation of optionally existing direct feedthroughs of subsystems. Due to the extended system description the time derivative of corresponding subsystem inputs appears, to be handled additionally by the master algorithm. In contrast to existing state of the art approaches, examination of the algorithm along a typical used co-simulation benchmark example turns out an outstanding performance in terms of stability and accuracy. This significant performance improvement is justified by the pre-step compensation of the co-simulation discretization error based on the approximated exact solution, i.e. the inputs to subsystems are modified according subsystem properties like stiffness and direct feedthrough prior to the execution of the co-simulation time increment.

5 FUTURE WORK

The current version of the pre-step co-simulation algorithm is able to cope with subsystem properties like stiffness and direct feedthrough. An extension of the algorithm is necessary for efficiently handling subsystems described by differential algebraic equation (DAEs), where inputs affect subsystem-internal algebraic constraints and thus introduce so-called drift effects. Furthermore, this contribution focusses on subsystems with a relative degree of zero and one. Future research has to investigate the importance of the relative degree in the context of co-simulation. Finally, the proposed pre-step co-simulation algorithm will be implemented prototypically into the co-simulation platform *Model.CONNECT*^{TM} from AVL to be applied to realistic industrial use-cases for an extended performance evaluation.

REMARK PATENT

The presented work describes an extension to a novel coupling approach for co-simulation of distributed dynamical subsystems. Protected by a pending European patent (Benedikt and Genser, 2018) the outlined schemes are supported by the co-simulation platform $Model.CONNECT^{TM}$ (AVL, 2018) from AVL.

ACKNOWLEDGEMENTS

The publication was written at VIRTUAL VEHICLE Research Center in Graz, Austria. The authors would like to acknowledge the financial support of the COMET K2 Competence Centers for Excellent Technologies Programme of the Federal Ministry for Transport, Innovation and Technology (bmvit), the Federal Ministry for Digital, Business and Enterprise (bmdw), the Austrian Research Promotion Agency (FFG), the Province of Styria and the Styrian Business Promotion Agency (SFG).

REFERENCES

Arnold, M. (2006). Multi-rate time integration for large scale multibody system models. IUTAM Symposium on Multiscale Problems in Multibody System Contacts.

Arnold, M. (2011). Numerical stabilization of co-simulation techniques, the ODE case. Martin Luther University Halle-Wittenberg NWF II-Institute of Mathematics.

AVL (2018). Model.connect^{TM}, the neutral model integration and co-simulation platform connecting virtual and real components. http://www.avl.com/-/model-connect-. [Online; accessed 31.01.2018].

Bastian, J., Glau, C., Wolf, S., and Schneider, P. (2011). Master for co-simulation using fmi. 8th Modelica Conference, Dresden, Germany,.

Benedikt, M. and Genser, S. (2018). Pre-step co-simulation method and device (filed).

Benedikt, M. and Hofer, A. (2013). Guidelines for the application of a coupling method for non-iterative co-simulation. IEEE, 8th Congress on Modelling and Simulation (EUROSIM), Cardiff Wales.

Benedikt, M., Watzenig, D., and Hofer, A. (2013). Modelling and analysis of the noniterative coupling process for cosimulation. TaylorFrancis.

Blochwitz, T. e. a. (2012). Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. 9th International Modelica Conference Munich.

Busch, M. (2012). *Zur Effizienten Kopplung von Simulationsprogrammen*. PhD thesis, University Kassel.

Cellier, F. and Kofman, E. (2010). *Continuous System Simulation*. Springer, New York, 1st edition.

Dorf, R. C. and Bishop, R. H. (2017). *Modern Control Systems*. Pearson, USA, 13th edition.

Genser, S. and Benedikt, M. (2018). Model-based pre-step stabilization method for non-iterative co-simulation. IMSD, Tecnico Lisboa.

Katayama, T. (2005). *Subspace Methods for System Identification*. Springer-Verlag London Limited, London, 1st edition.

Kuebler, R. and Schiehlen, W. (2000). Modular simulation in multibody system dynamics. Kluwer Academic Publishers.

Overschee, P. V. and Moor, B. D. (1996). *Subspace Identification for Linear Systems*. Kluwer Academic Publishers, London, 1st edition.

Sadjina, S., Kyllingstad, L., Skjong, S., and Pedersen, E. (2016). Energy conservation and power bonds in co-simulations: non-iterative adaptive step size control and error estimation. arXiv preprint arXiv:1602.06434.

Sicklinger, S., V., B., and B., E. (2013). Interface-jacobian based co-simulation. NAFEMS World Congress 2013.

Trnka, P. (2005). Subspace identification methods.

Viel, A. (2014). Implementing stabilized co-simulation of strongly coupled systems using the functional mock-up interface 2.0. 10th International Modelica Conference,Lund, Sweden.