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

Model-based pre-step stabilization method for non-iterative co-simulation

Simon Genser[1] and Martin Benedikt[2]

[1] *VIRTUAL VEHICLE Research Center, Graz Austria, simon.genser@v2c2.at*
[2] *VIRTUAL VEHICLE Research Center, Graz Austria, martin.benedikt@v2c2.at*

ABSTRACT— *Complex subsystem integration typically renders to very stiff overall system simulations. Especially in terms of co-simulation handling of such stiff system simulations becomes difficult as restricted submodel information is accessible for the non-iterative higher-level solver — the challenge is to maximize co-simulation performance based on available information. Linearly-implicit coupling schemes has proven to fulfill the requirements for a large range of co-simulations by utilizing partial derivatives corresponding to subsystem states. In contrast, within this contribution a co-simulation algorithm is presented based on formal subsystem transformations and resulting Interface Jacobians exclusively. This approach enables the introduction of the so-called Error Differential Equation and a pre-step compensation of the introduced co-simulation discretization error in terms of energy-preservation. The outlined pre-step co-simulation algorithm is examined along a theoretical pendulum co-simulation example and compared to other approaches, demonstrating a significantly gained performance improvement.*

1 Introduction

Co-Simulation represents a special simulation discipline, where several subsystems are simulated independently over co-simulation time increments, so called macro time steps ΔT, and data is exchanged for synchronization purposes at dedicated points in time, so called communication points T. This approach is typically used for holistic system simulation, in cases where modeling or simulation of the overall system within a single simulation tool is impossible [1].

In contrast to the classical numerical simulation approach, in case of co-simulation, additional master algorithms, i.e. higher-level numerical solvers, are necessary to solve the overall system simulation [2]. Due to limited or restricted interfacing and co-simulation capabilities of the involved simulation tools iterative (implicit) co-simulation approaches are not applicable in general; those are typically used for the dedicated integration of a few (typically two or three) subsystem simulations. On the other hand, non-iterative co-simulation is common practice with its drawback of introducing a significant coupling error, i.e. co-simulation discretization error, on higher master algorithm level. Mitigation of this co-simulation discretization error is restricted due to limited interfacing and co-simulation capabilities of the involved simulation tools, e.g. no access to subsystem-internal system states or resetting of subsystem simulation increments. As co-simulation becomes more and more relevant in different industries, several approaches were recently developed in order to cope with this specific problem.

With respect to these obstacles a system-oriented approach (NEPCE) was presented in the past where modifications are applied exclusively to the subsystem inputs for compensation of the co-simulation discretization error over subsequent co-simulation time increment [3]. Recently, [4] proposed an extension to this approach where additional model information, i.e. output and input partial derivatives referred to as Interface-Jacobians, is additionally utilized for modification of the relevant subsystem inputs, especially for handling direct-feedthroughs of individual subsystems. The exchange and utilization of Interface-Jacobian matrices is motivated by the Functional

Mockup Interface standard (FMI 2.0) [5] and are exemplarily used by [6, 7]. This additional subsystem information enables master algorithms to cope with stability issues related to system simulation critical properties like stiffness or algebraic loops.

From a more detailed perspective it is common practice to apply non-iterative co-simulation schemes where inputs and corresponding outputs are exchanged at coupling time instances. Bidirectional dependencies of subsystems in connection with the time discrete exchange of coupling variables requires a solution to the resulting causality problem, which is typically solved by polynomial extrapolation of the coupling variables over the interval of the current co-simulation time increment. These approaches comply with explicit numerical schemes and also inherit there weaknesses. Particularly the communication overhead introduced in co-simulation motivates to enlarge the co-simulation time increments which are tightly dependent on the subsystem dynamics and thus, explicit numerical schemes for handling stiff systems comes into play. Instead of performing iterations over co-simulation time increments till convergence to the (unique) solution these explicit schemes (e.g. linearly-implicit schemes) are utilizing subsystem dynamics information. This way, the solution is determined based on holistic dynamical considerations of the co-simulated system at hand incorporating (dynamical) cross-couplings of subsystems, which enables the usage of significantly increased co-simulation time increments and leads to an improved stability behavior.

However, these classical approaches neglect the introduced co-simulation discretization error, which becomes especially important in case of large co-simulation time increments and energy considerations among subsystems. The problem originates in the kind of utilization of the system information, whereas typically the determined future quantity of the solution is modified at the end of the current simulation time increment, like a post-step approach. In contrast to the classical simulation where the system states are computed, in co-simulation exclusively the inputs to the subsystems are determined at coupling time instances, which may represent in some rare cases the subsystem internal states. In case of non-iterative co-simulation, as proposed by the NEPCE approach [3], the inputs have to be modified prior to the introduction of the co-simulation discretization error in order to avoid related co-simulation discretization errors in subsystem-internal states. The basic NEPCE algorithm proposed in [3] do not utilize subsystem information and thus only depends on past co-simulation discretization errors, which also leads to somehow delayed effects.

Meaning, currently existing approaches for mitigating the co-simulation discretization error in non-iterative co-simulation are applied exclusively after the current co-simulation time increment at the coupling time instant, i.e. post-step, and impact the overall system simulation in a delayed (!) manner. This may lead to stability issues or to a decreased accuracy of the overall co-simulation. Within this paper this approach is extend for usage of subsystem model information which enables a modification of the subsystem inputs prior to the co-simulation time increment as a kind of a novel model-based pre-step co-simulation algorithm.

The remainder is organized as follows: Section 2 introduces the model-based pre-step stabilization algorithm, containing the necessary subsystem description for the detailed derivation of the algorithm. In Section 3 the improved performance is illustrated by an stiff multibody system example, a pendulum coupled to an oscillator.

2 The Model-based Pre-Step Stabilization Algorithm

The main part of this contribution is the derivation of the model-based pre-step stabilization algorithm, which contains of three main steps:

1. estimation of the exact, monolithic output $\tilde{y}$ utilizing the *Error Differential Equation* for the last macro time step;
2. model-based extrapolation $\hat{y}$ of the exact output $\tilde{y}$, over the next macro time step, considering cross coupling effects (model-based);
3. optimize the subsystem inputs u for the next macro time step (pre-step), based on the extrapolation $\hat{y}$.

2.1 Subsystem Description

In this section the underlying assumptions and the required subsystem description are stated. For lack of simplicity the following assumptions are made:

- the macro time step size is fixed for the overall co-simulation (i.e. $\Delta T_i^k = \Delta T$);
- the input u and output y of every subsystem are scalar signals;
- the overall co-simulation consists of two fully coupled subsystems (i.e. $N = 2$ and $u_1 = y_2$, $u_2 = y_1$).

The basis of the subsystem description is the classical state-space representation [8]

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

$$y_i = C_i \cdot x_i \tag{2}$$

with the assumption that there is no direct-feedthrough in (2). The derivation of the required output-based subsystem description starts with the time derivative of (2)

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

Inserting (1) and applying the pseudo-inverse[1] C_i^{-1} results in

$$\dot{y}_i = C_i \cdot \left(A_i \cdot C_i^{-1} \cdot y_i + B_i \cdot u_i\right), \tag{3}$$

whereas the following subsystem description is motivated:

$$\dot{y}_i = S_i(y_i, u_i). \tag{4}$$

Linearisation of (4) leads to the required linear subsystem description:

$$\dot{y}_i = \frac{\partial S_i}{\partial y} \cdot y_i + \frac{\partial S_i}{\partial u} \cdot u_i. \tag{5}$$

Note: The linearisation is performed around a equilibrium point of S_i and therefore the constant part vanishes.
It should be mentioned that from this motivation transformation rules for the output-based system description (5) in the classical state-space representation (3) and vice-versa, are a bonus outcome:

$$\frac{\partial S_i}{\partial y} = C_i \cdot A_i \cdot C_i^{-1}, \tag{6}$$

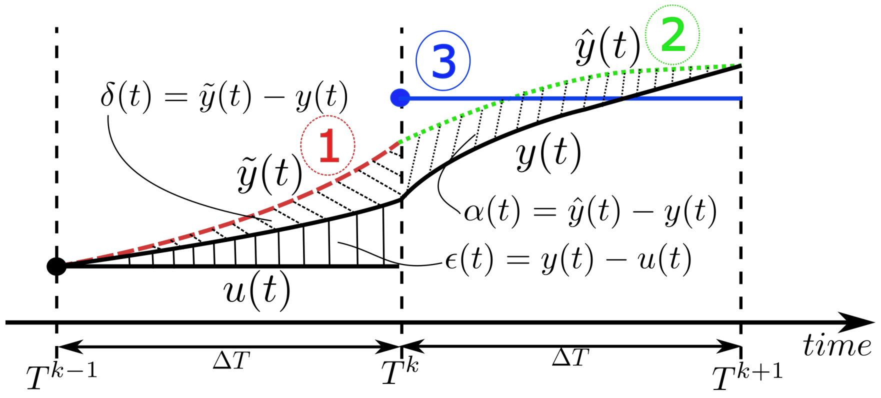
$$\frac{\partial S_i}{\partial u} = C_i \cdot B_i. \tag{7}$$

The utilization of the Functional Mock-up Interface 2.0 (FMI 2.0) [5] allows to access related system matrices, i.e. the partial derivatives, of a subsystem, therefore the transformation rules above are mandatory for exploiting the FMI 2.0 potentials.

[1]The utilization of the pseudo-inverse is due to generalization issues.

2.2 Derivation of the Main Steps

This section deals with the derivation of the three main steps of the pre-step algorithm, for schematic illustration see Figure 1. The derivation is outlined for the case that the current time instant is T^k, as denoted in Figure 1.



$\hat{y}(t)$
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
T^{k-1}
ΔT
T^k
ΔT
T^{k+1}

Fig. 1: Schematic illustration of the model-based pre-step stabilization algorithm, actual time is T^k

2.2.1 Step 1: Computing $\tilde{y}$ - Solving the Error Differential Equation

For the approximation of the exact, monolithic solution $\tilde{y}_i$, over the last macro time step $[T^{k-1}, T^k]$, the so called *Error Differential Equation* is required, illustrated as step 1 in Figure 1. The origin of the derivation is the subsystem description (4) and the cases:
ideal coupling (i.e. $u_1 = y_2$ and $u_2 = y_1$):

$$\dot{\tilde{y}}_1 = S_1(\tilde{y}_1, \tilde{y}_2), \tag{8}$$
$$\dot{\tilde{y}}_2 = S_2(\tilde{y}_2, \tilde{y}_1), \tag{9}$$

co-simulation coupling:

$$\dot{y}_1 = S_1(y_1, u_1), \tag{10}$$
$$\dot{y}_2 = S_2(y_2, u_2). \tag{11}$$

Comparing (8) with (10) leads to

$$\dot{y}_1 - S_1(y_1, u_1) = \dot{\tilde{y}}_1 - S_1(\tilde{y}_1, \tilde{y}_2). \tag{12}$$

To describe the deviation between the exact, monolithic output $\tilde{y}_i$ and the output computed by the subsystem y_i the δ-error is introduced as

$$\delta_i := \tilde{y}_i - y_i, \tag{13}$$

for illustration see Figure 1. Inserting δ_i in (12) leads to

$$S_1(\tilde{y}_1, \tilde{y}_2) - S_1(y_1, u_1) = \underbrace{\dot{\tilde{y}}_1 - \dot{y}_1}_{\dot{\delta}_1}. \tag{14}$$

The coupling errors ε_i are defined as

$$\varepsilon_1 := y_2 - u_1 \qquad \varepsilon_2 := y_1 - u_2, \tag{15}$$

whereas these errors arise from the solution of the causality problem of weak coupled problems by extrapolation of the input quantities. With the definitions of the δ-errors, it is possible to state

$$\tilde{y}_1 = \delta_1 + y_1 \qquad \tilde{y}_2 = \delta_2 + y_2. \tag{16}$$

Utilizing the ε-errors leads to

$$y_1 = u_2 + \varepsilon_2; \qquad u_1 = y_2 - \varepsilon_1; \qquad \tilde{y}_1 = u_2 + \varepsilon_2 + \delta_1. \tag{17}$$

Inserting (16) and (17) in (14) results in

$$\dot{\delta}_1 = S_1(u_2 + \varepsilon_2 + \delta_1, y_2 + \delta_2) - S_1(u_2 + \varepsilon_2, y_2 - \varepsilon_1). \tag{18}$$

Evaluating this equation at the time instant T^k leads to

$$\dot{\delta}_1 = S_1(u_2^k + \varepsilon_2 + \delta_1, y_2^k + \delta_2) - S_1(u_2^k + \varepsilon_2, y_2^k - \varepsilon_1), \tag{19}$$

where the index k is omitted for the δ- and ε-errors. Applying a spatial linearisation, around ($u_2^k := u_2(T^k)$, $y_2^k := y_2(T^k)$), on (19) leads to

$$S_1(u_2^k, y_2^k) + \frac{\partial S_1}{\partial y} \cdot (\varepsilon_2 + \delta_1) + \frac{\partial S_1}{\partial u} \cdot \delta_2 - S_1(u_2^k, y_2^k) - \frac{\partial S_1}{\partial y} \cdot \varepsilon_2 + \frac{\partial S_1}{\partial u} \cdot \varepsilon_1 = \dot{\delta}_1. \tag{20}$$

After cancelling out terms the final linear ordinary differential equation, the herein defined Error Differential Equation for S_1, renders to:

$$\dot{\delta}_1 = \frac{\partial S_1}{\partial y} \cdot \delta_1 + \frac{\partial S_1}{\partial u} \cdot [\delta_2 + \varepsilon_1]. \tag{21}$$

Analogously with (9) and (11) the Error Differential Equation for S_2 is:

$$\dot{\delta}_2 = \frac{\partial S_2}{\partial y} \cdot \delta_2 + \frac{\partial S_2}{\partial u} \cdot [\delta_1 + \varepsilon_2]. \tag{22}$$

The coupled structure of the Error Differential Equations for S_1 and S_2 is clearly visible in the following vector and matrix notation:

$$\underbrace{\begin{pmatrix} \dot{\delta}_1 \\ \dot{\delta}_2 \end{pmatrix}}_{=:\dot{\boldsymbol{\delta}}} = \underbrace{\begin{pmatrix} \frac{\partial S_1}{\partial y} & \frac{\partial S_1}{\partial u} \\ \frac{\partial S_2}{\partial u} & \frac{\partial S_2}{\partial y} \end{pmatrix}}_{=:\tilde{A}} \cdot \underbrace{\begin{pmatrix} \delta_1 \\ \delta_2 \end{pmatrix}}_{=:\boldsymbol{\delta}} + \underbrace{\begin{pmatrix} \frac{\partial S_1}{\partial u} & 0 \\ 0 & \frac{\partial S_2}{\partial u} \end{pmatrix}}_{=:\tilde{B}} \cdot \underbrace{\begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \end{pmatrix}}_{=:\boldsymbol{\epsilon}}. \tag{23}$$

So the global coupled Error Differential Equation is stated as:

$$\dot{\boldsymbol{\delta}} = \tilde{A} \cdot \boldsymbol{\delta} + \tilde{B} \cdot \boldsymbol{\epsilon}. \tag{24}$$

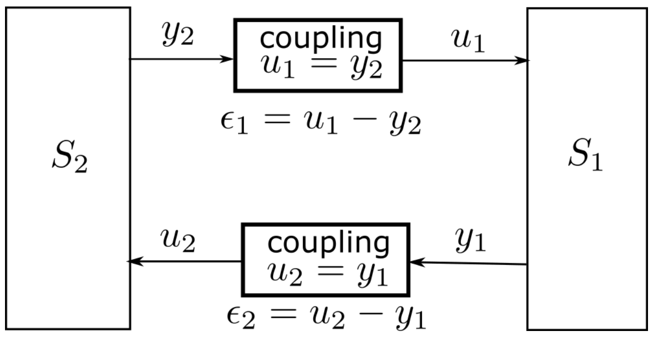


Fig. 2: Schematic illustration of two fully coupled subsystems, introducing the coupling error ε_i

The equation is denoted as global, i.e. it can not be solved by an individual subsystem independent from the others. The off-diagonal entries of $\tilde{A}$ are a crucial key to the improved performance because cross coupling effects between the individual subsystems are taken into account.
Remark: The initial conditions arises from the fact that y is continuous and therefore δ is continuous as well. This means that the computation of the exact, monolithic output $\tilde{y}_i$ is well defined through the differential equation (24), the initial conditions and definitions of the δ-error in (13), which can be rearranged to

$$\tilde{y}_i = \delta_i + y_i. \tag{25}$$

2.2.2 Step 2: Computing $\hat{y}$ - Model-based Extrapolation

The extrapolation $\hat{y}_i$ of the exact, monolithic $\tilde{y}_i$ output over the next macro time step $[T^k, T^{k+1}]$ is a mandatory step, see step 2 in Figure 1. Due to the exploitation of the partial derivatives of the subsystems, it is possible to take cross coupling effects between the subsystem in account. This is an essential key to the improved stability and accuracy of the algorithm, especially therefore it is called model-based stabilization. Comparing this to the idea of an implicit solver, one can interpret this stabilization algorithm as an implicit, non-iterative co-simulation approach.
The key idea behind the model-based extrapolation is the assumption that, the coupling over the next macro time step is ideal, i.e. $u_1(t) = \hat{y}_2(t)$ and $u_2(t) = \hat{y}_1(t)$ for all $t \in [T^k, T^{k+1}]$. Inserting this in

$$\dot{\hat{y}}_i \approx \frac{\partial S_i}{\partial y} \cdot \hat{y}_i + \frac{\partial S_i}{\partial u} \cdot u_i \text{ for } i = 1,2, \tag{26}$$

leads to:

$$\dot{\hat{y}}_1 \approx \frac{\partial S_1}{\partial y} \cdot \hat{y}_1 + \frac{\partial S_1}{\partial u} \cdot \hat{y}_2, \tag{27}$$

$$\dot{\hat{y}}_2 \approx \frac{\partial S_2}{\partial y} \cdot \hat{y}_2 + \frac{\partial S_2}{\partial u} \cdot \hat{y}_1. \tag{28}$$

The coupled structure of (27) and (28) is evident in:

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

The final global differential equation for the model-based extrapolation can be written as:

$$\dot{\hat{\boldsymbol{y}}} = \hat{A} \cdot \hat{\boldsymbol{y}}. \tag{29}$$

Due to the fact that $\hat{y}_i$ represents the extrapolation of $\tilde{y}_i$ over the next macro time step, the initial condition for (29) is:

$$\hat{y}_i(T^k) = \tilde{y}_i(T^k), \tag{30}$$

for the extrapolation over $[T^k, T^{k+1}]$. The value $\tilde{y}_i(T^k)$ is known from the solution of the Error Differential Equation in the previous step of the algorithm.

2.2.3 Step 3: Computing u - Pre-Step Input Optimization

The pre-step optimization of the input u for the next macro time step $[T^k, T^{k+1}]$, denoted as step 3 in Figure 1, is the third main step of the presented pre-step stabilization method. It is worth to mention that this optimization can be performed for every subsystem independently, therefore it is called a local computation and hence the sub index

i is neglected in this subsection. The key to a proper optimization is to define the right quantity to optimize, here the so called α-error

$$\alpha(t) := \hat{y}(t) - y(t), \tag{31}$$

is utilized, for illustration see Figure 1. The input optimization is based on the minimization of α over the next macro time step, i.e.

$$\min_{t \in [T^k, T^{k+1}]} |\alpha(t)| \tag{32}$$

is claimed. From a theoretical point of view there are different ways to solve optimization problems like (32). Herein a discretized solution, based on the macro time step size, is used, which leads to the following equation:

$$|\alpha^{k+1}| \overset{!}{=} 0, \tag{33}$$

with $\alpha^{k+1} := \alpha(T^{k+1})$. Future work will focus among others on other ways of solving (32), especially on variational approaches. Inserting the definition of the α-error (31) in (33) leads to

$$\hat{y}^{k+1} \overset{!}{=} y^{k+1}, \tag{34}$$

with $\hat{y}^{k+1} := \hat{y}(T^{k+1})$ and $y^{k+1} := y(T^{k+1})$. The connection between the input u and the output y is mandatory for choosing u in such a way that (34) is fulfilled. To keep the derivation of it as simple as possible piecewise constant basic functions[2] are chosen for the input u, i.e.

$$u(t) := u^k \text{ for } t \in [T^k, T^{k+1}]. \tag{35}$$

From the standard theory of ordinary differential equations [9] the analytical solution of a subsystem is described via

$$y(t) = e^{\frac{\partial S}{\partial y} \cdot t} \cdot y^0 + \int_{t_{start}}^{t} e^{\frac{\partial S}{\partial y} \cdot (t-\tau)} \cdot \frac{\partial S}{\partial u} \cdot u(\tau)\, d\tau \tag{36}$$

where y^0 denotes the initial condition for y, inserting the approach (35) in (36) results in

$$y(t) = e^{\frac{\partial S}{\partial y} \cdot t} \cdot y^0 + \sum_{l=0}^{k} \int_{T^l}^{T^{l+1}} e^{\frac{\partial S}{\partial y} \cdot (t-\tau)} \cdot \frac{\partial S}{\partial u} \cdot u^l\, d\tau \tag{37}$$

with $T^0 = t_{start}$ and ΔT and k are appropriate chosen, so that $T^{k+1} = t$ holds. The splitting of the integral is due to the piecewise definition of u in (35). With

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

it is possible to rewrite (37) in

$$y(t) = \phi_A(t) \cdot y^0 + \sum_{l=0}^{k} u^l \cdot \phi_{B_l}(t). \tag{38}$$

[2]This means that u is discontinuous but although the outputs are continuous signals by assumption this is not a problem due to the lack of a direct-feedthrough in the subsystems.

Evaluation of this equation in T^{k+1} as it is required in (34), leads to

$$y^{k+1} = \phi_A(T^{k+1}) \cdot y^0 + \sum_{l=0}^{k} u^l \cdot \phi_{B_l}(T^{k+1}). \tag{39}$$

Recursive inserting of this equation, leads to

$$y^{k+1} = \phi_A(\Delta T) \cdot y^k + u^k \cdot \phi_{B_k}(T^{k+1}). \tag{40}$$

$\phi_A(\Delta T)$ and $\phi_{B_k}(T^{k+1})$ denote matrices with constant coefficients and therefore the time variable and the sub index k is neglected, this results in

$$y^{k+1} = \phi_A \cdot y^k + u^k \cdot \phi_B. \tag{41}$$

In this equation the connection between the input and the output of a subsystem is clearly visible. From inserting (41) in (34) follows

$$\hat{y}^{k+1} = \phi_A \cdot y^k + u^k \cdot \phi_B, \tag{42}$$

algebraic rearranging and utilization of the pseudo-inverse ϕ_B^{-1} leads to

$$u^k = \phi_B^{-1} \cdot \left(\hat{y}^{k+1} - \phi_A \cdot y^k \right). \tag{43}$$

Due to the fact that y^k is known from the previous macro time step and $\hat{y}^{k+1}$ is computed in the previous step of the algorithm, see therefore Section 2.2.2, the optimal input u^k for the next macro time step $[T^k, T^{k+1}]$ can be determined.
Note: If the number of inputs is higher than the number of outputs per subsystem, extra conditions are needed to determine the inputs in satisfying manner. Otherwise, if there are more outputs in a subsystem than inputs a least square solution is preferred, which is automatically ensured by utilizing the pseudo-inverse.

2.3 Workflow

This subsection summarizes the algorithm and explains in detail its workflow. There are five steps which have to be performed, see therefore the workflow in Figure 3:

1. accessing subsystem partial derivatives;
2. assembling of global matrices;
3. global approximation of the exact solution;
4. global model-based extrapolation;
5. local input optimization.

The assumptions[3] from Section 2.1. are extended with:

- the local inputs u_1 and u_2 are piecewise constant functions (ZOH);
- the macro step size ΔT is equal to the micro step size δT;
- the explicit Euler is used for numerically solving the ordinary differential equations.

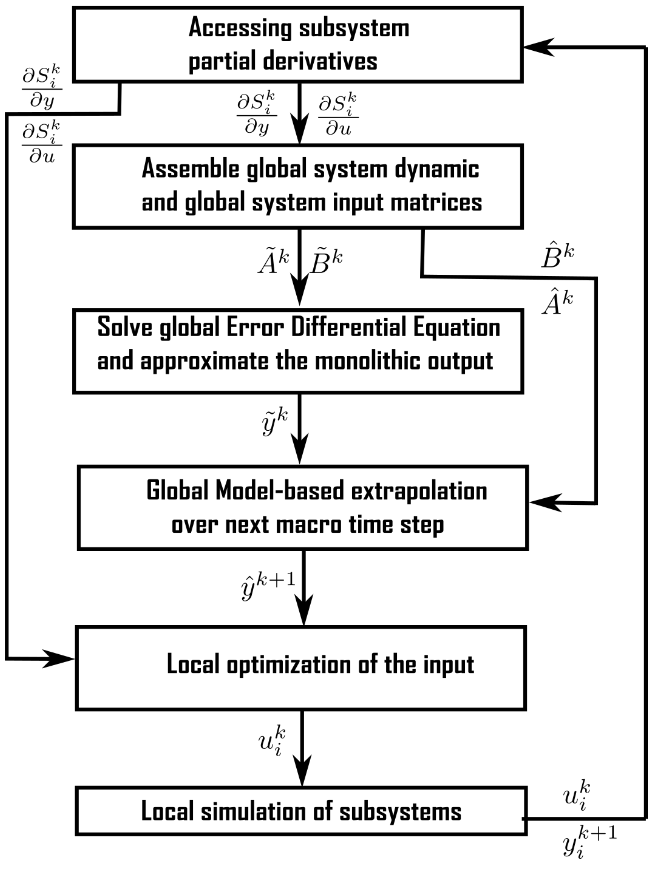

Accessing subsystem partial derivatives
$\frac{\partial S_i^k}{\partial y}$
$\frac{\partial S_i^k}{\partial u}$
Assemble global system dynamic and global system input matrices
$\tilde{A}^k$ $\tilde{B}^k$
$\hat{B}^k$
$\hat{A}^k$
Solve global Error Differential Equation and approximate the monolithic output
$\tilde{y}^k$
Global Model-based extrapolation over next macro time step
$\hat{y}^{k+1}$
Local optimization of the input
u_i^k
Local simulation of subsystems
u_i^k
y_i^{k+1}

Fig. 3: workflow diagram of the model-based pre-step stabilization algorithm

For the following description the actual time is T^k and therefore the objective of the algorithm is to compute the optimized inputs u_1^k and u_2^k. The quantities $y_i^k,\dots,y_i^1$, $\tilde{y}_i^{k-1},\dots,\tilde{y}_i^1$ and $u_i^{k-1},\dots,u_i^1$ for $i=1,2$ are available from the previous computational steps. For illustration of these quantities see Figure 4.

Accessing the subsystem partial derivatives (SID), for S_1 and S_2 can be performed independently from each other and is therefore local, i.e.

$$[\frac{\partial S_i^k}{\partial u},\frac{\partial S_i^k}{\partial y}] = SID(u_i^{k-1},\dots,u_i^1,y_i^k,\dots,y_i^1) \tag{44}$$

for $i=1,2$. With knowledge of the structure of the co-simulation, i.e. utilizing the coupling matrix[4], it is possible to assemble the global matrices:

$$\underbrace{\begin{pmatrix} \frac{\partial S_1^k}{\partial y} & \frac{\partial S_1^k}{\partial u} \\ \frac{\partial S_2^k}{\partial u} & \frac{\partial S_2^k}{\partial y} \end{pmatrix}}_{=\tilde{A}},\underbrace{\begin{pmatrix} \frac{\partial S_1^k}{\partial u} & 0 \\ 0 & \frac{\partial S_2^k}{\partial u} \end{pmatrix}}_{=\tilde{B}},\underbrace{\begin{pmatrix} \frac{\partial S_1^k}{\partial y} & \frac{\partial S_1^k}{\partial u} \\ \frac{\partial S_2^k}{\partial u} & \frac{\partial S_2^k}{\partial y} \end{pmatrix}}_{=\hat{A}}. \tag{45}$$

With $\tilde{A}$ and $\tilde{B}$ it is possible to formulate the Error Differential Equation (24):

$$\dot{\boldsymbol{\delta}} = \tilde{A}\cdot\boldsymbol{\delta} + \tilde{B}\cdot\boldsymbol{\epsilon} \tag{46}$$

[3]These assumptions are necessary to keep the notation simple, the generalization is straightforward.

[4]The coupling matrix L ($L\cdot y = u$) is here declared as $L = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$.

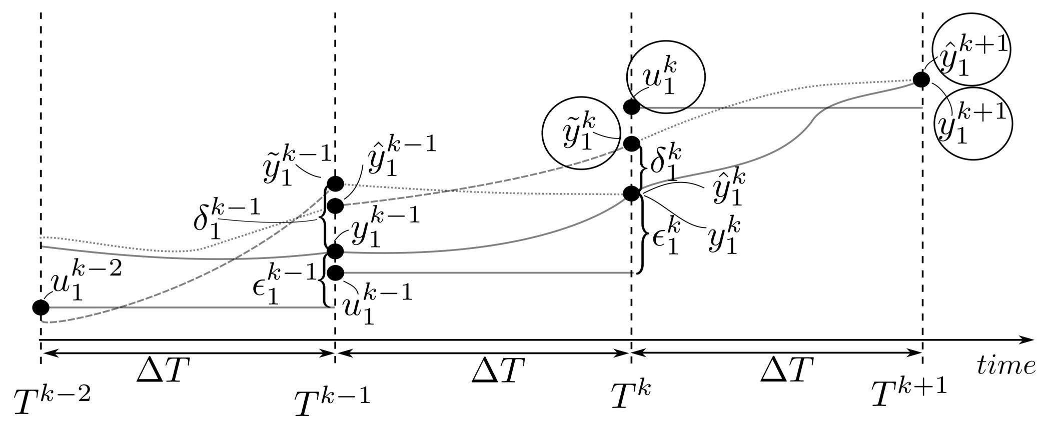


Fig. 4: Illustration of the discretized algorithm for subsystem S_1, the circled quantities are the ones, which have to be computed at the actual time T^k

and approximate the exact, monolithic output $\tilde{y}$, where $\boldsymbol{\delta} := (\delta_1, \delta_2)^T$ and $\boldsymbol{\epsilon} := (\varepsilon_1, \varepsilon_2)^T$. The discretizied[5] equation, with explicit Euler schema, results in

$$\boldsymbol{\delta}^k = \boldsymbol{\delta}^{k-1} + \Delta T \cdot \left(\tilde{A}^k \cdot \boldsymbol{\delta}^{k-1} + \tilde{B}^k \cdot \boldsymbol{\epsilon}^{k-1} \right), \tag{47}$$

therefore $\boldsymbol{\delta}^{k-1}$ and $\boldsymbol{\epsilon}^{k-1}$ are necessary to solve the equation. From the previous computation step $\boldsymbol{\delta}^{k-1}$ is known. $\boldsymbol{\epsilon}^{k-1}$ is defined in (15) as

$$\varepsilon_2^{k-1} = y_1^{k-1} - u_2^{k-1} \qquad \varepsilon_1^{k-1} = y_2^{k-1} - u_1^{k-1}. \tag{48}$$

So they are computable at time instant T^k, as it is depicted in Figure 4. So $\boldsymbol{\delta}^k$ can be computed, which leads to the approximated, exact solution

$$\tilde{y}_i^k = \delta_i^k + y_i^k \tag{49}$$

for $i = 1, 2$. In the next step the global model-based extrapolation of $\tilde{y}_i^k$ over the next macro time step is performed, i.e. the computation of $\hat{y}_i^{k+1}$ for $i = 1, 2$. With the global assembled matrix $\hat{A}$ and (29) the future progress is approximated with

$$\underbrace{\begin{pmatrix} \dot{\hat{y}}_1 \\ \dot{\hat{y}}_2 \end{pmatrix}}_{=\dot{\hat{\boldsymbol{y}}}} = \underbrace{\begin{pmatrix} \frac{\partial S_1^k}{\partial y} & \frac{\partial S_1^k}{\partial u} \\ \frac{\partial S_2^k}{\partial u} & \frac{\partial S_2^k}{\partial y} \end{pmatrix}}_{=:\hat{A}^k} \cdot \underbrace{\begin{pmatrix} \hat{y}_1 \\ \hat{y}_2 \end{pmatrix}}_{=\hat{\boldsymbol{y}}}.$$

The numerical solution[5] is computed with the explicit Euler method with

$$\hat{\boldsymbol{y}}^{k+1} = \tilde{\boldsymbol{y}}^k + \Delta T \cdot \hat{A}^k \cdot \tilde{\boldsymbol{y}}^k. \tag{50}$$

Here it should be denoted that $\hat{\boldsymbol{y}}^k$ is replaced by $\tilde{\boldsymbol{y}}^k$, as it is denoted in Figure 4 and stated in (30). That is the reason for the discontinuity of $\hat{y}$ at every communication point, but this is crucial for the improved stability of the algorithm, especially in the case of stiff systems.
The last step is to locally optimize the input u_i^k for $i = 1, 2$. For piecewise constant functions the optimization is outperformed with (43)

$$u_i^k = \phi_{\frac{\partial S_i^k}{\partial u}}^{-1} \cdot \left(\hat{y}_i^{k+1} - \phi_{\frac{\partial S_i^k}{\partial y}} \cdot y_i^k \right). \tag{51}$$

[5]For a better numerical solution an additional step-size for the discretization of the differential equation can be utlized, typical chosen clearly smaller than the macro-step size.

The computation of $\phi_{\partial S_i^k \setminus \partial u}$ and $\phi_{\partial S_i^k \setminus \partial y}$ can be done with suited, classical numerical methods, see [8]. The usage of the pseudo-inverse is preferred due to generalization issues. With u_i^k the subsystem S_i is possible to simulate the next macro time step T^{k+1} and compute y_i^{k+1}. The computation is finished and the next macro time steps starts with the system identification at the time instant T^{k+1}.
Remark: To improve the computational effort the system identification can be fixed for some macro time steps and can only be updated every n-th step, see [10]. A second strategy to decrease the effort would be to only update the subsystem matrices if there is a sufficient big change in the dynamics of the systems.

3 Theoretical Stiff Multibody System Example

For demonstration of the improved performance of the herein presented algorithm, a recently analyzed non-linear multibody system example, a mathematical pendulum coupled to an oscillator, representing a stiff system has been chosen see [1] and Figure 5.

3.1 Pendulum coupled to an Oscillator

The model-based pre-step stabilization requires subsystems which are described via first order, linear differential equations. Therefore the following second order, non-linear differential equations have to be transformed in two first order, linear systems of differential equations. The second order differential equation

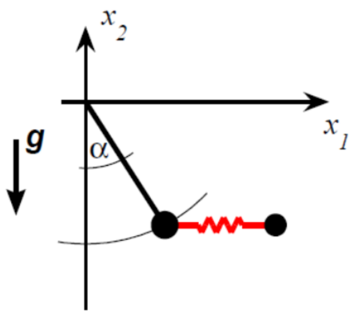


Fig. 5: Stiff multibody system example: mathematical pendulum coupled to an oscillator [1]

$$m_{pend} l \ddot{\alpha} = -m_{pend} g \sin(\alpha) + cos(\alpha) F_c(\alpha, \dot{\alpha}, x_1, \dot{x}_1) \tag{52}$$

describes the mathematical pendulum and

$$m_{osc} \ddot{x}_1 = -F_c(\alpha, \dot{\alpha}, x_1, \dot{x}_1) \tag{53}$$

describes the oscillator. The coupling between the pendulum and the oscillator is expressed with the force:

$$F_c = k \cdot (x_1 - l \sin(\alpha)) + d \cdot (\dot{x}_1 - l \dot{\alpha} \cos(\alpha)) \, . \tag{54}$$

The pendulum represents subsystem 1 and the oscillator subsystem 2. So it is necessary to transform (52) into a first order system, this is done by the substitution $\overline{x}_1 = \dot{x}_1$, see therefore standard ode-theory [9], which leads to

$$S_1 : \begin{cases} \dot{\alpha} & = \overline{\alpha}, \\ \dot{\overline{\alpha}} & = \ddot{\alpha} = -\frac{g}{l} \sin(\alpha) + \frac{\cos(\alpha)}{m_{pend} l} F_c(\alpha, \dot{\alpha}, x_1, \dot{x}_1). \end{cases}$$

So it is possible to write the non-linear subsystem S_1 as

$$\begin{pmatrix} \dot{\alpha} \\ \dot{\overline{\alpha}} \end{pmatrix} = \begin{pmatrix} \overline{\alpha} \\ -\frac{F_c(\alpha, \dot{\alpha}, x_1, \dot{x}_1)}{m_{osc}} \end{pmatrix} =: S_1 \left(\underbrace{\begin{bmatrix} \alpha \\ \overline{\alpha} \end{bmatrix}}_{=: \boldsymbol{y_1}}, \underbrace{\begin{bmatrix} x_1 \\ \overline{x}_1 \end{bmatrix}}_{=: \boldsymbol{u_1}} \right) = S_1 \left(\boldsymbol{y_1}, \boldsymbol{u_1} \right) \tag{55}$$

where $\boldsymbol{y_1}$ denotes the output and $\boldsymbol{u_1}$ the input of S_1. The analogue substitution for the subsystem S_2 leads to

$$S_2 : \begin{cases} \dot{x}_1 & = \overline{x}_1, \\ \dot{\overline{x}}_1 & = \ddot{x}_1 = -\frac{F_c(\alpha, \dot{\alpha}, x_1, \dot{x}_1)}{m_{osc}}. \end{cases}$$

Rewriting it in vector notation is resulting in

$$\begin{pmatrix}\dot{x}_1\\ \dot{\overline{x}}_1\end{pmatrix} = \begin{pmatrix}\overline{x}_1\\ -\frac{F_c(\alpha,\dot{\alpha},x_1,\dot{x}_1)}{m_{osc}}\end{pmatrix} =: S_2\left(\underbrace{\begin{bmatrix}x_1\\ \overline{x}_1\end{bmatrix}}_{=:\boldsymbol{y_2}}, \underbrace{\begin{bmatrix}\alpha\\ \overline{\alpha}\end{bmatrix}}_{=:\boldsymbol{u_2}}\right) = S_2\left(\boldsymbol{y_2},\boldsymbol{u_2}\right) \tag{56}$$

Note: Comparing (56) with (55) immediately leads to the coupling condition $\boldsymbol{u_1} = \boldsymbol{y_2}$ and $\boldsymbol{u_2} = \boldsymbol{y_1}$.
The following linearization of S_1 and S_2 is mandatory to utilize the model-based pre-step stabilization

$$\dot{\boldsymbol{y}}_{\boldsymbol{1}} = S_1\left(\boldsymbol{y_1},\boldsymbol{u_1}\right) \approx S_1\left(\boldsymbol{y_{10}},\boldsymbol{u_{10}}\right) + \frac{\partial S_1}{\partial y}\cdot\left(\boldsymbol{y_1}-\boldsymbol{y_{10}}\right) + \frac{\partial S_1}{\partial u}\cdot\left(\boldsymbol{u_1}-\boldsymbol{u_{10}}\right), \tag{57}$$

describes the first-order Taylor approximation with their origin in $(\boldsymbol{y_{10}},\boldsymbol{u_{10}})$. The center of the series is chosen as the equilibrium point, i.e. $S_1\left(\boldsymbol{y_{10}},\boldsymbol{u_{10}}\right) = 0$. Here it holds $\boldsymbol{y_{10}} = (0,0)^T$ and $\boldsymbol{u_{10}} = (0,0)^T$. This leads to

$$\dot{\boldsymbol{y}}_{\boldsymbol{1}} = S_1\left(\boldsymbol{y_1},\boldsymbol{u_1}\right) \approx \frac{\partial S_1}{\partial y}\cdot\boldsymbol{y_1} + \frac{\partial S_1}{\partial u}\cdot\boldsymbol{u_1}. \tag{58}$$

Analogously the linearisation of S_2 is

$$\dot{\boldsymbol{y}}_{\boldsymbol{2}} = S_2\left(\boldsymbol{y_2},\boldsymbol{u_2}\right) \approx \frac{\partial S_2}{\partial y}\cdot\boldsymbol{y_2} + \frac{\partial S_2}{\partial u}\cdot\boldsymbol{u_2}. \tag{59}$$

The analytic computation[6] of the partial derivatives $\frac{\partial S_1}{\partial y}$ and $\frac{\partial S_1}{\partial u}$ and utilization of the definition of F_c in (54) results in

$$\frac{\partial S_1}{\partial y} = \begin{pmatrix} 0 & 1\\ -g\cos(\alpha)/l + \{\cos(\alpha)\cdot[-kl\cos(\alpha)+\dots & -dl\cos(\alpha)^2/(m_{pend}l)\\ \dots dl\overline{\alpha}\sin(\alpha)] - \sin(\alpha)F_c\}/(m_{pend}l) & \end{pmatrix},$$
$$\frac{\partial S_1}{\partial u} = \begin{pmatrix} 0 & 0\\ k\cos(\alpha)/(m_{pend}l) & d\cos(\alpha)/(m_{pend}l)\end{pmatrix}.$$

and the partial derivatives for S_2 are stated as

$$\frac{\partial S_2}{\partial y} = \begin{pmatrix} 0 & 1\\ -k/m_{osc} & -d/m_{osc}\end{pmatrix},$$
$$\frac{\partial S_2}{\partial u} = \begin{pmatrix} 0 & 0\\ [kl\cos(\alpha) - dl\overline{\alpha}\sin(\alpha)]/m_{osc} & dl\cos(\alpha)/m_{osc}\end{pmatrix}.$$

For the following computations the parameters are chosen like in [1]

$$l = 1\,m,\ g = 9.81\,m/s^2,\ m_{pend} = m_{osc} = 1\,kg,\ \alpha(0) = 5^\circ,\ x_1(0) = 0.1\,m + 1\,m\cdot\sin(\alpha(0)), \tag{60}$$
$$\dot{\alpha}(0) = 0\,rad/s,\ \dot{x}_1(0) = 0\,m/s,\ k = 10^3\,N/m,\ d = 10\,Ns/m. \tag{61}$$

Due to this parameters the pendulum-oscillator coupling is a stiff problem, with a stiffness of

$$\frac{\Re(|\lambda_{max}|)}{\Re(|\lambda_{min}|)} = \frac{9.96}{0.0002} = 49\,800. \tag{62}$$

Here $\Re(\cdot)$ denotes the real value of a complex number and λ_{max} respectively λ_{min} stands for the maximum respectively the minimum of the eigenvalues of the monolithic coupled problem (65).
For the calculation of the errors a reference solution $y_{i_{ref}}$ is needed, therefore the linearized, monolithic system is

[6]To keep the error due to the linearisation as small as possible the linearisation is updated at every communication point.

utilized. The monolithic system results from the assumption that the coupling is ideal, i.e. $\boldsymbol{u_1} = \boldsymbol{y_2}$ and $\boldsymbol{u_2} = \boldsymbol{y_1}$. From (58) and (59) follows

$$\dot{\boldsymbol{y}}_{\boldsymbol{1}ref} = \frac{\partial S_1}{\partial y} \cdot \boldsymbol{y}_{\boldsymbol{1}ref} + \frac{\partial S_1}{\partial u} \cdot \boldsymbol{y}_{\boldsymbol{2}ref}, \tag{63}$$

$$\dot{\boldsymbol{y}}_{\boldsymbol{2}ref} = \frac{\partial S_2}{\partial y} \cdot \boldsymbol{y}_{\boldsymbol{2}ref} + \frac{\partial S_2}{\partial u} \cdot \boldsymbol{y}_{\boldsymbol{1}ref}. \tag{64}$$

This results in the monolithic 4-th order autonomous system

$$\begin{pmatrix} \dot{\boldsymbol{y}}_{\boldsymbol{1}ref} \\ \dot{\boldsymbol{y}}_{\boldsymbol{2}ref} \end{pmatrix} = \begin{pmatrix} \frac{\partial S_1}{\partial y} & \frac{\partial S_1}{\partial u} \\ \frac{\partial S_2}{\partial u} & \frac{\partial S_2}{\partial y} \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{y}_{\boldsymbol{1}ref} \\ \boldsymbol{y}_{\boldsymbol{2}ref} \end{pmatrix}. \tag{65}$$

The numerical computation of the reference solution is performed with the explicit Euler method with macro time step size is equal to the micro time step size, i.e. ΔT=δT. The utilized error is

$$err = \max_{t \in [t_{start}, t_{end}]} \sum_{i=1}^{2} |\boldsymbol{y}_{\boldsymbol{1}ref}(i) - \boldsymbol{y_1}(i)| + |\boldsymbol{y}_{\boldsymbol{2}ref}(i) - \boldsymbol{y_2}(i)|. \tag{66}$$

Remark: For the numerical solution of the Error Differential Equation a linear-implicit solver [11] with the micro step size δT is utilized. The model-based extrapolation is solved with the same numerical solver utilizing the macro step size ΔT, to keep the computational effort low. To improve the stability and accuracy of the pre-step method more accurate numerical solvers or smaller step sizes can be utilized.

3.2 Performance Evaluation

To illustrate the improved performance of the model-based pre-step stabilization it is compared to the classical Zero-Order Hold (ZOH) coupling, the Nearly Energy Preserving Coupling Element (NEPCE) method [3] and the NEPCE combined with an Anti-Aliasing Filter [12]. Figure 6 and 7 shows:

1. the improved stability of the model based pre-step stabilization method for $\Delta T > 5 \cdot 10^{-4}$;
2. for $\Delta T < 5 \cdot 10^{-4}$ all coupling algorithms converge to the same error boundary;
3. the model based pre-step stabilization method reaches this error boundary with $\Delta T = 6 \cdot 10^{-3}$, which is about 20 times higher compared to all other methods.

Due to the error estimation[7] [1]

$$err \leq C_0 \cdot (\delta T_1^{p_1} + \delta T_2^{p_2} + \Delta T) \tag{67}$$

and the fact that the micro-step size δT is fixed, the existence of an asymptotic border of the error is nothing surprising. The fact that the asymptotic error, i.e. the error for $\Delta T \to 0$, is, in at least an approximate sense, for $\delta T = 5 \cdot 10^{-5}$, Figure 7, the half of the error as for $\delta T = 10^{-4}$, Figure 6, for all coupling methods is a strong hint for the validity of the error estimation (67) for decreasing δT. Due to the faster convergence, to the asymptotic error, of the pre-step stabilization method, the validity of the error estimation is more precise.
For Zero-Order Hold coupling the error border is reached with $\Delta T = 2 \cdot 10^{-4}$. The asymptotic error border is reached with $\Delta T = 4 \cdot 10^{-4}$ by utilization of the NEPCE coupling. For NEPCE in combination with the Anti-Aliasing Filter the border is by approximately $\Delta T = 10^{-4}$, but the stability is higher compared to the classical NEPCE. For the model based pre-step stabilization the asymptotic border is by $\Delta T = 6 \cdot 10^{-3}$. This means that with the use of the herein invented pre-step stabilization method one can choose the macro step size approximately 15 times higher compared to NEPCE coupling or 30 times compared to ZOH coupling. This increased macro time step size leads to a decrease in the communication effort in the overall co-simulation. This lead to a wider class of

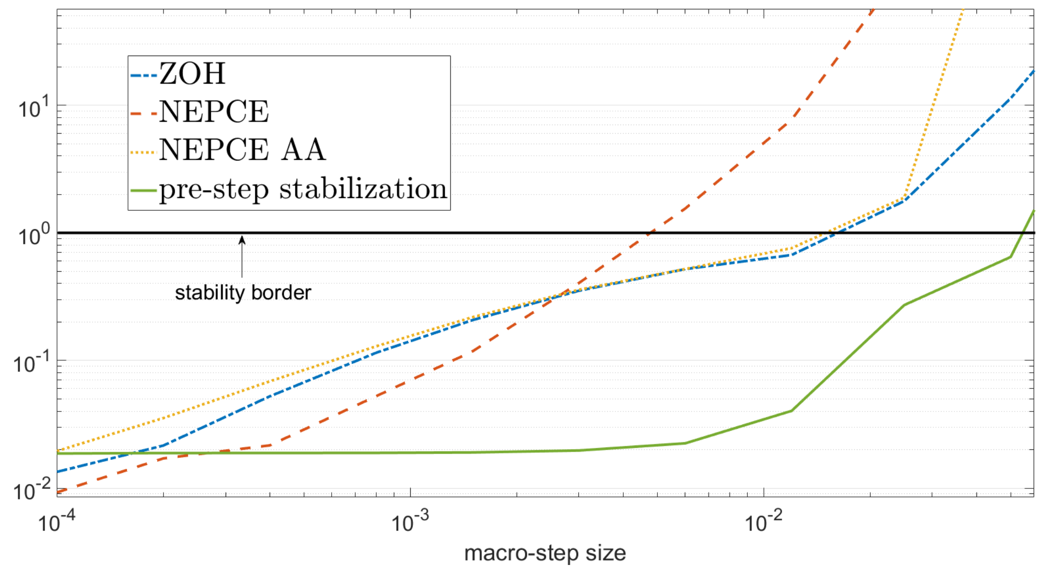


Fig. 6: Logarithmic error plot for varying macro-step size and $\delta T = 10^{-4}$; $t_{end} = 1s$

problems which will be suitable for real time simulations.

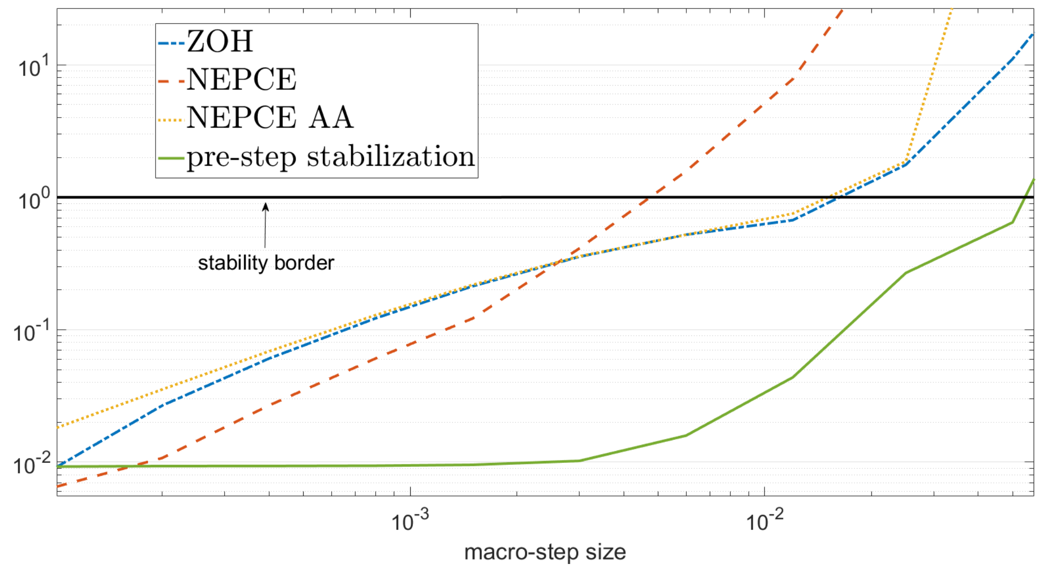


Fig. 7: Logarithmic error plot for varying macro-step size and $\delta T = 5 \cdot 10^{-5}$; $t_{end} = 1s$

Figure 8 depicts the angle velocity $\dot{\alpha}$ of the pendulum, computed with $\Delta T = 0.005$ and $\delta T = 10^{-4}$, here the improved accuracy of the invented pre-step stabilization is obvious. One can also see that the solution with the NEPCE coupling is instable and comparing this to the logarithmic error plots leads to the conclusion that an error over approximately 1 implies that the method is unstable. This leads to the conclusion that for this stiff multibody system example the model-based pre-step stabilization is stable up to an macro step size of $\Delta T = 0.04$.

[7]The error estimation holds for sufficiently small ΔT and piecewise constant extrapolation of the inputs. Here p_i denotes the convergence order of the local numerical solver for the subsystem S_i, e.g. $p_i = 1$ for an explicit Euler schema.

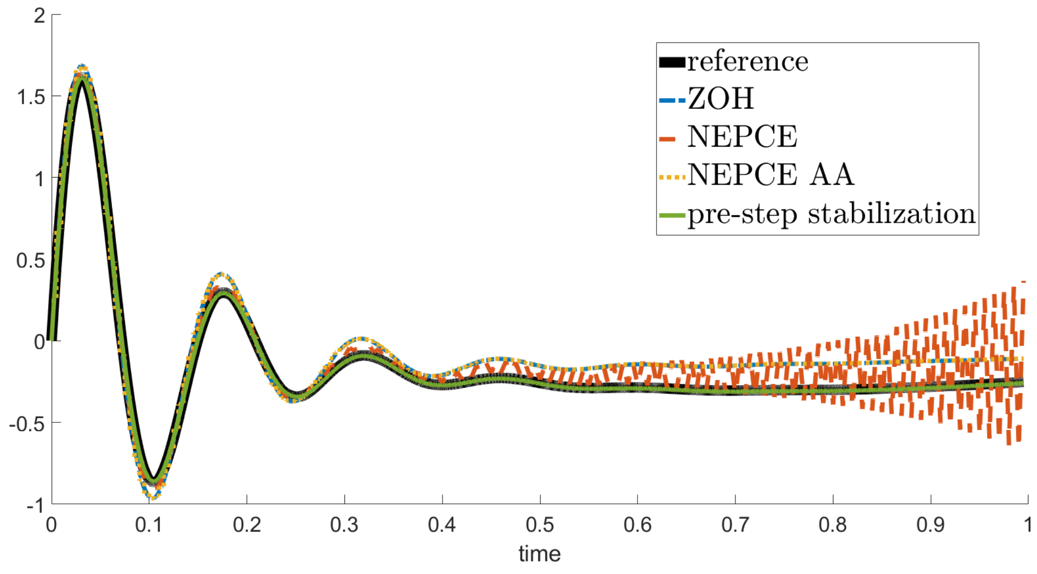


Fig. 8: The angle velocity $\dot{\alpha}$ of the pendulum for $\Delta T = 0.005$ and $\delta T = 10^{-4}$

In Figure 9 the angle velocity $\dot{\alpha}$ is depicted with $\Delta T = 0.06$ and $\delta T = 10^{-4}$, so it should be unstable compared to Figure 6. But it doesnt look typical unstable, the first thing that leap to ones eyes are the bumps in the output of the pre-step stabilization solution. But where is their origin?
If one compares the length of the bumps to the macro-step size one recognizes that they coincide. This phenomenon origins from the discretized choice of the input in (33). The optimization problem (32) is only solved for the communication steps T^{k+1} and not for the interval between them. This is the reason for the seemingly unstable behaviour of the model-based pre-step stabilization for large macro-step sizes. This motivates to solve the optimization problem (32) in such a way that the interval between two communication points is also taken into account, e.g. in a variational way i.e. $\int_{T^k}^{T^{k+1}} |\alpha(t)| dt \to 0$, which will be in focus of the future work.

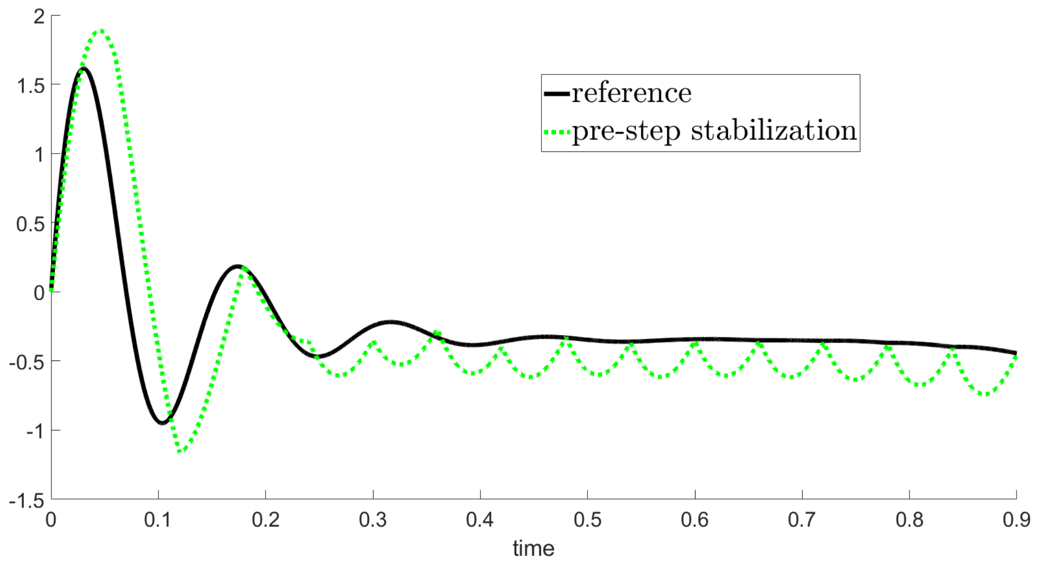


Fig. 9: The angle velocity $\dot{\alpha}$ of the pendulum for $\Delta T = 0.06$ and $\delta T = 10^{-4}$

4 Conclusions and Future Work

For efficient handling of stiff systems model-based numerical schemes have to be applied. In contrast to existing approaches, the proposed algorithm is based on a 1^{st} order approximation of the exact, monolithic solution and, in addition, performs an optimization-based pre-step correction, resulting in a significant performance improvement in terms of accuracy and stability. Enlargement of the macro step-size ($\approx$ 20 times) yields overall co-simulation improvement, especially the communicational effort is reduced.
Future work focusses on identification of sensitivities and analysing the different options to optimize the input. An additional focus lies on the generalization of the method for the case of subsystem with direct-feedthrough and industrial applications.

Patent Remark

The presented work describes a novel coupling approach for co-simulation of distributed components. Protected by a pending European patent [13] the outlined schemes are supported by the co-simulation platform $Model.CONNECT^{TM}$ [14] 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

[1] M. Arnold, “Multi-rate time integration for large scale multibody system models,” IUTAM Symposium on Multiscale Problems in Multibody System Contacts, 2006.

[2] M. Busch, *Zur Effizienten Kopplung von Simulationsprogrammen*. PhD thesis, University Kassel, 2012.

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

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

[5] T. Blochwitz, M. Otter, A. J., A. M., and C. C., “Functional mockup interface 2.0: The standard for tool independent exchange of simulation models,” 9th International Modelica Conference Munich, 2012.

[6] S. Sicklinger, B. V., and E. B., “Interface-jacobian based co-simulation,” NAFEMS World Congress 2013, 2013.

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

[8] R. C. Dorf and R. H. Bishop, *Modern Control Systems*. USA: Pearson, 13th ed., 2017.

[9] G. Teschl, *Ordinary Differential Equations and Dynamical Systems*. Vienna, Austria: American Mathematical Society, 1st ed., 2012.

[10] M. Arnold, “Numerical stabilization of co-simulation techniques, the ODE case,” Martin Luther University Halle-Wittenberg NWF II-Institute of Mathematics, 2011.

[11] F. Cellier and E. Kofman, *Continuous System Simulation*. New York: Springer, 1st ed., 2010.

[12] M. Benedikt and E. Drenth, “Relaxing sti system integration by smoothing techniques for non-iterative co-simulation.,” IUTAM Symposium on Co-Simulation and Solver-Coupling, 2018 (accepted).

[13] M. Benedikt and S. Genser, “Pre-step co-simulation method and device,” 2018.

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