A Stable but Explicit Cosimulation Coupling Method

Thilo Moshagen

Hochschule Wismar, Fakultät Technik, Bereich Maschinenbau

Abstract. The term *co-simulation* denotes the coupling of some simulation tools for dynamical systems into one big system by having them exchange data at points of a fixed time grid and extrapolating the received data into the interval, while none of the steps is repeated for iteration. From the global perspective, the simulation thus has a strong explicit component. Frequently, among the data passed across subsystem boundaries there are flows of conserved quantities, and as there is no iteration of steps, system-wide balances may not be fulfilled: the system is not solved as one monolithic equation system. If these *balance errors* accumulate, simulation results become inaccurate. Balance correction methods which compensate these errors by adding corrections for the balances to the signal in the next coupling time step have been considered in past research. But establishing the balance of one quantity *a posteriori* due to the time delay in general cannot establish the balances of quantities that depend on the exchanged quantities, usually energy. In most applications from physics, the balance of energy is equivalent to stability. In this paper, a method is presented which allows users to choose the quantity that should be balanced to be that energy, and to accurately balance it. This establishes also numerical stability for many classes of stable problems.

Co-simulation, coupled problems, simulator coupling, explicit coupling, stability, convergence, balance correction

1 Introduction

With the rise of simulation software for technical systems emerged the desire to couple those simulations in order to take into account the influence the systems exercise onto each other. In other words, these systems are now viewed as subsystems which form one big system.

One now wants to simulate this large system, using the subsystems' simulator software and coupling it by sharing data. What used to be a parameter when the systems were calculated separately is given now by a state variable of the other subsystem, reading:

$$S_1 : \dot{x}_1 = f_1(x_1,x_2,z_1,z_2) \tag{1}$$
$$0 = g_1(x_1,x_2,z_1,z_2) \tag{2}$$
$$S_2 : \dot{x}_2 = f_2(x_1,x_2,z_1,z_2) \tag{3}$$
$$0 = g_2(x_1,x_2,z_1,z_2). \tag{4}$$

Here, the (x_1,x_2) are the differential, the (z_1,z_2) are the algebraic states. The setting generalizes to n subsystems in a straightforward way, and it includes parabolic partial differential equations. We require that the derivatives $d_{z_i}g_i$ have full rank. Such each of the S_i is an index-1 differential-algebraic system if the $(x_{k\neq i},z_{k\neq i})$ are seen as parameters of it. The influence of x_2,z_2 in a split setting is therefore modeled by parameters u_{12} in S_1 and x_1,z_1 as parameters u_{21}:

$$S_1 : \dot{x}_1 = f_1(x_1,z_1,u_{12}) \tag{5}$$
$$0 = g_1(x_1,z_1,u_{12}) \tag{6}$$
$$S_2 : \dot{x}_2 = f_2(x_2,z_2,u_{21}) \tag{7}$$
$$0 = g_2(x_2,z_2,u_{21}). \tag{8}$$

When coupled, the u_{ij} are determined by the coupling conditions

$$0 = h_{21}(x_1,z_1,u_{21}) \tag{9}$$
$$0 = h_{12}(x_2,z_2,u_{12}) \tag{10}$$

that have to be fulfilled, and exchanged at fixed time nodes T_k. Between them, the u_{ij} are extrapolated. To establish coupling, the h_{ik} must be solvable with respect to the u_{ik}. The $d_{z_i}g_i$ have full row rank, too. Such, the differential-algebraic system given by Equations (5) - (10) is again of index 1.

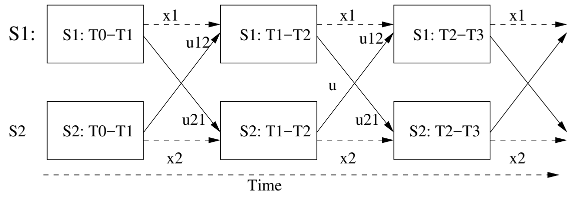

Figure 1: Explicit co-simulation scheme.

This description of the setting is widespread [2].

It is commonly said that the coupling is done by a *co-simulation scheme* if the u_{ij} are calculated from Equations (9) and (10) at exchange time nodes T_k and then passed on to S_2 and S_1, respectively. Of course, some extrapolation of u_{ij} into $[T_k,T_{k+1})$ is required. Considerable research has been done on coupling [6, 7, 5]. A lot of methods repeat the timestep after the calculation with an extrapolation that has been improved with respect to some objective. Thus, they are implicit, see e.g. [5]. For

ARGESIM Report 59 (ISBN 978-3-901608-93-3), p 49-55, DOI: 10.11128/arep.59.a59008

a convergence proof, see [2]. The convergence of explicit co-simulation methods for ODE and index one DAE not surprisingly as well improves with the extrapolation order of subsystems input [8, 2]. The situation here in simulator coupling mirrors the one in ODE solvers: The explicit solvers are quick in each step but not stable [12, 8], while the implicit ones require iterations within each step but usually ensure some stability. When used for stiff problems, the explicit schemes require such small stepwidths that the implicit schemes are finally cheaper. Also, implicit algorithms for coupled solvers require additional programming and storage. Therefore, the co-simulation scheme, where one just proceeds to the next timestep (Figure 1), is still popular.

So far, it has been common sense that the usual stability classifications like A - and B-Stability cannot be achieved with explicit algorithms [14]. A solution for these stability issues would be helpful in many applications and is the subject of this contribution.

It is important to note that all results and figures herein have been published before in [1]. This contribution is a highly condensed presentation of that content for the purpose of reaching the engineering community rather than novelty.

2 The lack of stability

2.1 Stability classifications

For readability, we present the concepts of stability classifications of methods.

Definition 2.1 (Stable points of ODE) *Let x^* be an equilibrium point of the ODE $\dot{x} = f(x)$ and $\phi^t x$ the solution for the initial value $x(t_0) = x$. Then x^* is*

- stable *if* $\forall \varepsilon > 0\ \exists \delta > 0 : \|x - x^*\| < \delta \Rightarrow \|\phi^t x - x^*\| < \varepsilon\ \forall t \in [t_0, T]$
- asymptotically stable *if* $\exists r > 0 : \|x - x^*\| < r \Rightarrow \lim_{t \longrightarrow \infty} \phi^t x = x^*$.

Definition 2.2 (Stable Point of Difference equation)

Let x^* be an equilibrium point of the $k-th$ order difference equation $x^{n+1} = f(x^n, ..., x^{n-k})$. Then x^* is classified as in Definition (2.1) where x is replaced by x_n and $\forall t \in [t_0, T]$ by $\forall n \in \{1, ..., N\}$ and furthermore $t \longrightarrow \infty$ by $n \longrightarrow \infty$.

Using these two definitions, stability classifications like zero-, A- or B-stability are defined: The respective stability of a method is the inheritance of the stability of an equilibrium point of a certain ODE class to the equilibrium point of the difference equation yielding from the application of the numerical scheme.

2.1.1 Stability, consistency and convergence

In this framework, zero stability of a numerical method means that the difference equation that one gets by applying the method to $\dot{x} = 0$ is stable. It is well-known that this is a necessary condition for convergence [13, 14]. But this condition is fulfilled by all one-step methods[1] as $x_{n+1} = x_n + 0$ is a stable equation. So unlike for multistep methods, there is no need here to examine zero-stability when one examines convergence of one-step methods. It frequently causes confusion that zero stability in the original paper [13] was labeled *stability* only, and with this nomenclature Lax's and Richtmyers' theorem is given in an equation-like form *stability + consistency = convergence*.

2.2 Stability

These results were confirmed numerically in [8, Sec.3.2] using the two-dimensional linear problem

$$\dot{x} = Ax, \tag{11}$$

which with

$$A = \begin{pmatrix} 0 & 1 \\ -\frac{c}{m} & 0 \end{pmatrix}, \qquad x = \begin{pmatrix} x \\ \dot{x} \end{pmatrix} \tag{12}$$

can be interpreted as linear spring-mass oscillator with mass m and spring constant c. This problem is the most simple problem possible that is linear and can be splitted. The original problem is marginally stable, so stable, as its spectrum is purely imaginary.

Written as a co-simulation problem, Problem (11) with (12) yields Table 1. In [8] and [12] it is shown that co-simulation schemes are not stable for linear problems, even not for stable subsystem solvers. The stability for linear problems replaces the notion of A-stability, as the one-component equation used there cannot be split. When treated with a co-simulation scheme (output of the spring is the force $f = -cx$, that of the mass is the velocity $v = \dot{x}$) the emerging (method-induced) ODE

$$\begin{cases} \dot{x}_1 = a_{1,1}x_1 + a_{1,2}\,\mathrm{Ext}(x_2) \\ \dot{x}_2 = a_{2,2}x_2 + a_{2,1}\,\mathrm{Ext}(x_1) \end{cases} . \tag{15}$$

is obviously unstable [8, Section 2.5].

Its numerical solution is shown in Figure 2, – there is no linear stability for general step sizes. This means $\|x\| \longrightarrow \infty$ for $t \longrightarrow \infty$. The energy of our system is $E = \frac{1}{2}mv^2 + \frac{1}{2}cs^2 = \langle x, \frac{1}{2}\,\mathrm{diag}(m,c)x \rangle = \langle x, x \rangle_{\frac{1}{2}\,\mathrm{diag}(m,c)} =$

[1] One-step methods can be written as $x_{n+1} = x_n + h\psi(x_n, t_n, h_n)$, and $\psi(x_n, t_n, 0) = f$, where f is the ODE's right hand side.

Spring		Mass
	System States	
$x_1 := s = x$		$x_2 := v = \dot{x}$
	Outputs	
$u_{21} := F = -cx$		$u_{12} := v = \dot{x}$
	Inputs	
u_{12}		u_{21}
	Equations	
$\dot{x_1} = \text{Ext}(u_{12}) = v$		$\dot{x_2} = -\frac{1}{m}\text{Ext}(u_{21}) = -\frac{F}{m}$

Spring		Mass
	System States	
$\dots$		$\dots$
	Outputs	
$u_{21} := (f, \dot{f}) = (-cx, -cv)$	(13)	$u_{12} := (v, a) = (\dot{x}, \dot{f}/m)$ (1
	Inputs	
u_{12}		u_{21}
	Equations	
$\vdots$		$\vdots$

Table 1: Standard Co-simulation schemes for the spring-mass system, top constant, down linear extrapolation. When there is no difference, dots have been used.

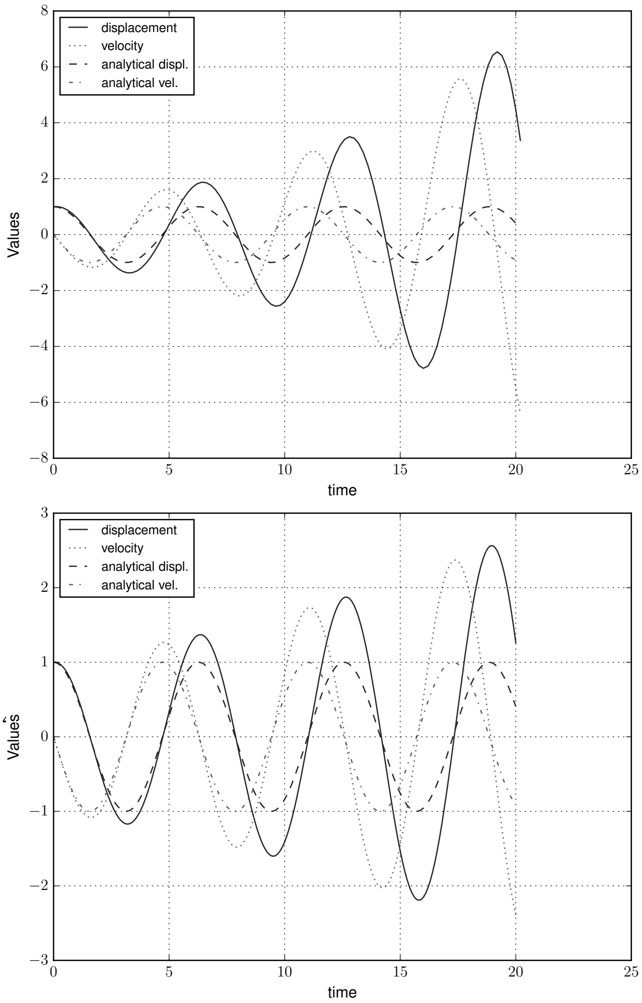

Figure 2: Simulation of the system (11)-(12) in the co-simulation scheme with constant extrapolation, varying the exchange step size H. Upper row, left: $H = 0.2$, right: $H = 0.1$. Previously published in [12].

$\|x\|_{\frac{1}{2}\text{diag}(m,c)}$, which is an equivalent norm, so lack of stability is equivalent to energy augmentation.

Using piecewise constant extrapolation of inputs, the force, as it is seen by the mass, is effectively shifted to later times: a value from time T_i is used for all future times $t \in (T_i, T_{i+1})$. The analogy with the reactive power and the real power of an electrical network is apparent. Work from oscillating systems with phase shift contains an integral over a constant and thus grows unbounded (Figure 2). Similar arguments hold for linear extrapolation co-simulation.

3 Enforcing balance by sharing the view on potential flow

3.1 The proposed method

The key feature to establish energy balance is exchanging the value of power and calculating the variable of interest from that power. Consider a co-simulation problem with subsystems S_1 and S_2 as given by Equations (5) - (10) with states x_1 and x_2 respectively and inputs u_{21} and u_{12}. We suggest the following procedure to enforce energy balance between subsystems S_1 and S_2:

1. At data exchange timepoint T_n the powers P_{ij} as the

flux of energy are calculated in both subsystems, using up-to-date input u^n_{ji}. In general $P_{ij} \neq P_{ji}$.
Applied to the $S_1 - S_2$ setting, P_{21} (the power calculated in S_1 for passing to S_2), is calculated using fresh u_{12}, the input into S_1. Now in the input vectors u_{ij} one component is replaced by P_{ij} and that new vector $u_{21,\text{bal}}$ is exchanged between the subsystems.
Applied to $S_1 - S_2$ setting, the value P_{21} replaces one component $(u_{21})_m$ of u_{21}, and respectively, P_{12} replaces $(u_{12})_n$ of u_{12}.
This means S_1's point of view about the power has been passed on to S_2 and vice versa.

2. Now both subsystems have the same information and thus the opportunity to draw the same conclusion on what energy exchange should be assumed. We denote this assumed energy exchange as

   $$\hat{P}_{12}(P_{21}, P_{12}) = -\hat{P}_{21}, \tag{16}$$

   a straightforward choice is $\hat{P}_{21} = (P_{12} - P_{21})/2 = -\hat{P}_{12}$, where now it is necessary to define flow directions: P_{ij} shall be negative if it leaves S_j, so it is counted with opposite sign in S_i.
   Again remember that P_{21} is the power calculated in S_1 for passing to S_2, calculated using u_{12}, the input into S_1. The former input $(u_{12})_n(t)$ now is calculated subject to

   $$P_{21}(x_1(t), u_{12\setminus n}, (u_{12})_n(t)) = \text{Ext}(\hat{P}_{12}). \tag{17}$$

   Analogously $(u_{21})_m(t)$ s.t. $P_{12}(x_2, u_{21\setminus m}, (u_{21})_m) = \text{Ext}(\hat{P}_{21})$ is calculated. The expression $12 \setminus k$ in subscript is to say that the k-th component of the vector is left out. For the unique inversion of P_{ij} it is required that the maps $(u_{ji})_k \longrightarrow P_{ij}(.,.,(u_{ji})_k)$ are strictly monotone.

   As $\text{Ext}(\hat{P}_{12}) = -\,\text{Ext}(\hat{P}_{21})$, now it is established that the inputs of S_1 and S_2 are consistent in terms of energy conservation for all t.

3.2 Example

To apply the scheme given in Section 3.1 above to a spring-mass system (12), replacing the standard co-simulation scheme from Table 1, one first calculates the energies of the systems parts, powers acting on subsystems boundaries, and their derivatives. As $P_i = \dot{W}_i$, $P_i < 0$ indicates that energy leaves S_i.

Spring	Mass
Energy	
$W = \int -f\,ds$	$W = \int f\,ds$
$= \int -fv\,dt$	$= \int mav\,dt$
Power	
$P = \dot{W}$	$P = \dot{W}$
$= -fv = cxv$	$= mav = fv$
Derivative of Power	
$\dot{P} = c(v^2 + sa)$	$\dot{P} = m(a^2 + v\dot{a})$
	$= m(a^2 + v\frac{\dot{f}}{m})$

The derivative of force $\dot{f}$ is available as output of spring, as it is usually needed for linearly extrapolating the input. With this, the scheme yields Table 2.

4 Stability of power balanced schemes

As discussed in Section 2.2 and shown in [8], stability for linear systems of a partly explicite scheme is not given. This section shall relate energy conservation of our method to stability. The class of problems under consideration are all stable *gradient flow problems*

$$\dot{x} = -M\nabla_x \mathscr{P}^T, \tag{18}$$

which is a huge class, containing entropy driven and energy conserving problems. The *mobility Matrix M* determines the systems stability - it is positive definite if the system is dissipative and skew if energy conserving. This behavior must be inherited to the ODE that is induced by our splitting method. We give an outline of the arguments:

1. Switch to gradient flow view. In this, inserting (18) into the time derivative of the respective potential $\mathscr{P}(x)$ yields

   $$\dot{\mathscr{P}}(x) = \langle \nabla_x \mathscr{P}(x), \dot{x} \rangle = \left\langle \nabla_x \mathscr{P}(x), -M\nabla_x \mathscr{P}(x)^T \right\rangle \tag{19}$$

   with the scalar product $\langle .,. \rangle$.

2. Introduce split system

Spring	Mass
System States	
$x_1 := s = x$	$x_2 := v = \dot{x}$
Outputs	
$(u_{21,\text{Std}})_1 := f = -cx$	$(u_{12,\text{Std}})_1 := v = \dot{x}$
$(u_{21,\text{Std}})_2 := \dot{f} = -cv$	$(u_{12,\text{Std}})_2 := \dot{v} = f/m$
(intermediately exchanging $u_{ij,\text{Std}}$)	
$(u_{21})_1 = P(x_1, u_{12})$ $= cxv = cx_1(u_{12})_1$	$(u_{12})_1 = P(x_2, u_{21})$ $= fv = (u_{21})_1 x_2$
$(u_{21})_2 = \dot{P}(x_1, u_{12})$ $= c(v^2 + xa)$ $= c((u_{12})_1^2 + x_1(u_{12})_2)$	$(u_{12})_2 = \dot{P}(x_2, u_{21})$ $= m(a^2 + v\frac{\dot{f}}{m})$ $= m\left(\frac{(u_{21})_1}{m}^2 + x_2\frac{(u_{21})_2}{m}\right)$
Inputs	
$(u_{12})_1 := \hat{P}$	$(u_{21})_1 := -\hat{P}$
$(u_{12})_2 := \hat{\dot{P}}$	$(u_{21})_2 := -\hat{\dot{P}}$
Input variables of standard method u_{std} depending on Power	
$v = \frac{\text{Ext}(\hat{P})}{cs} = \frac{\text{Ext}(u_{12})_1}{cx_1}$	$f = -\frac{\text{Ext}(\hat{P})}{v} = \frac{\text{Ext}(u_{21})_1}{x_2}$
Equations	
$\dot{x}_1 = v$	$\dot{x}_2 = \frac{f}{m}$

Table 2: Method form Section 3.1 applied to the spring-mass system

- Identify coupling contributions
- Characterize potential conservation/dissipation properties (see below)

3. See method as decoupling ODE – Insert calculation of inputs from power into original equations
4. Relate decoupled ODEs stability properties to stability of original systems
   - Show that negotiated exchange conserves $\dot{\mathscr{P}} \leq 0$. It can be shown and there are straightforward arguments that there is no unphysical power production when sharing subsystems agree on the exchanged energy
   - Use Lyapunov's direct method on the decoupled system.
   - Additionally, one can argue that maximum stable stepwidth for dissipative systems is augmented (method is closer to B-stability than extrapolation of inputs method).
5. If such stable subsystems ODEs are solved with methods preserving that stability, overall solution will be stable.

Items (2) and also (4) need closer consideration. The split systems potential production $\dot{\mathscr{P}}(x)$ according to Eq. (19) in subsystem-wise block matrix form reads

$$\dot{\mathscr{P}}(x) = P_k + P_l + \ldots \tag{20}$$

$$= \begin{pmatrix} \cdot \\ (\nabla_x \mathscr{P}(x))_{I_k} \\ \cdot \\ (\nabla_x \mathscr{P}(x))_{I_l} \\ \cdot \end{pmatrix} \cdot \ldots \tag{21}$$

$$\begin{pmatrix} & & * & & \\ \ldots & -(M)_{I_k,I_k} & \ldots & -(M)_{I_k,I_l} & \ldots \\ & & * & & \\ \ldots & -(M)_{I_l,I_k} & \ldots & -(M)_{I_l,I_l} & \ldots \\ & & * & & \end{pmatrix} \begin{pmatrix} \cdot \\ (\nabla_x \mathscr{P}(x))_{I_k} \\ \cdot \\ (\nabla_x \mathscr{P}(x))_{I_l} \\ \cdot \end{pmatrix} \tag{22}$$

$$= \underbrace{\left\langle \nabla_{x_{I_k}} \mathscr{P}(x), -M_{I_k,I_k} \nabla_{x_{I_k}} \mathscr{P}(x)^T \right\rangle}_{P_{kk}} \tag{23}$$

$$+ \underbrace{\left\langle \nabla_{x_{I_k}} \mathscr{P}(x), -M_{I_k,I_l} \nabla_{x_{I_l}} \mathscr{P}(x)^T \right\rangle}_{P_{kl}}$$

$$+ \underbrace{\left\langle \nabla_{x_{I_l}} \mathscr{P}(x), -M_{I_l,I_l} \nabla_{x_{I_l}} \mathscr{P}(x)^T \right\rangle}_{P_{ll}} \tag{24}$$

$$+ \underbrace{\left\langle \nabla_{x_{I_l}} \mathscr{P}(x), -M_{I_l,I_k} \nabla_{x_{I_k}} \mathscr{P}(x)^T \right\rangle}_{P_{lk}} + \ldots,$$

we identify

$$P_{kl} := \left\langle (\nabla_x \mathscr{P}(x))_{I_k}, -(M)_{I_k,I_l} (\nabla_x \mathscr{P}(x)^T)_{I_l} \right\rangle \tag{25}$$

as the *potential production* in S_k by S_l's variables, or power acting from subsystem l onto subsystem k. Item (4) now means that those eliminate in the suggested scheme, as the exchanging subsystems agree on their value. So, there is no contribution to $\dot{\mathscr{P}}$ by the extrapolation during coupling.

Theorem 4.1 *For a Lyapunov stable (asymptotically stable) gradient flow initial value problem (IVP), the IVP resulting from the energy balancing method as described in Section 3.1 is also stable (asymptotically stable).*

5 Discussion, conclusion and future work

The suggested method overcomes the decade-old issue of stability in coupled simulation for a huge class of problems.

Moreover, the method has a clear interpretation in physics: the enforcement of the power balance in systems interactions. It can therefore be implemented by anyone with understanding of the systems they want to couple, without deep knowledge of numerical analysis. For simulations in industrial research and development, the new method enables stable calculations with big timesteps and few programming effort and is thus a big step forward.

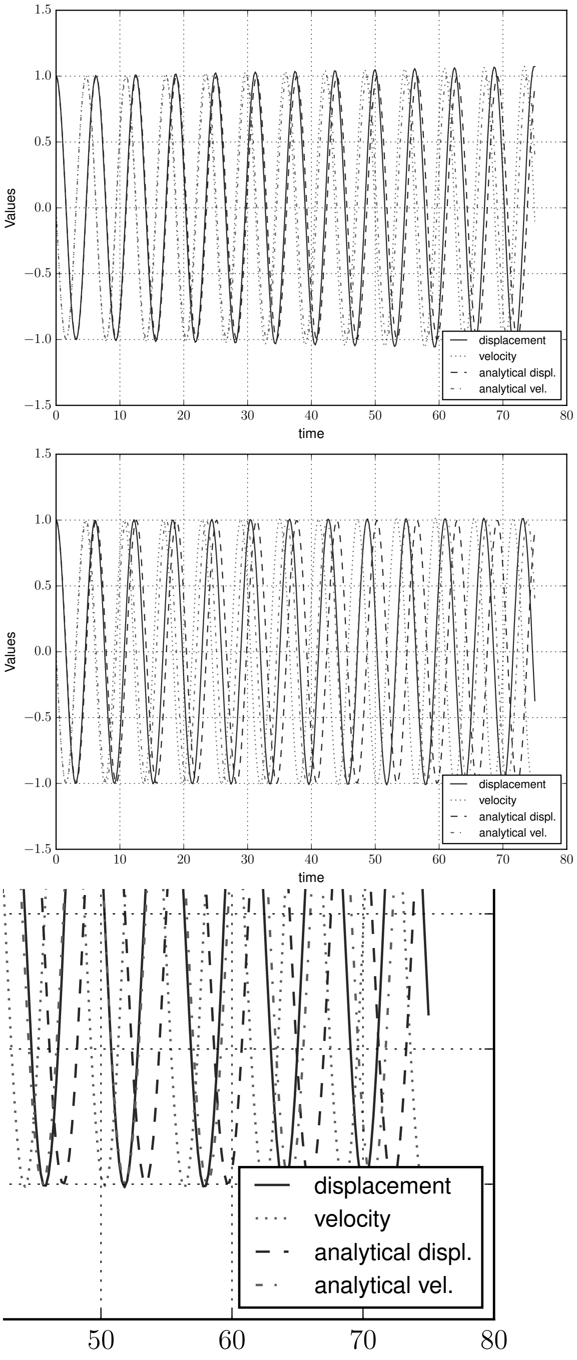

Figure 3: Stability of cosimulation schemes applied to spring-mass system: Top: Linear extrapolation, middle and bottom: Power balanced scheme. The image on bottem is an amplification of the lower right corner of the image above to confirm that there is no energy gain. $T_{\text{end}} = 75$, exchange stepwidth $H = 0.2$, subsystems refinement decisions left to subsystems solvers, stable *vode* used on subsystems.

References

[1] Moshagen, Thilo, *On Meeting Energy Balance Errors in Co-Simulations*, Mathematical and Computer Modelling of Dynamical Systems, 25, pp 139-166, Publisher: Taylor & Francis, 2019, Available at doi:10.1080/13873954.2019.1595667.

[2] M. Arnold and M. Günther, *Preconditioned Dynamic Iteration for Coupled Differential-Algebraic Systems*, BIT Numerical Mathematics 41 (2001), pp. 1–25, Available at http://dx.doi.org/10.1023/A3A1021909032551.

[3] M. Arnold, C. Clauss, and T. Schierz, *Error Analysis and Error Estimates for Co-Simulation in FMI for Model Exchange and Co-Simulation v2.0*, Progress in Differential-Algebraic Equations 60.1 (2013), pp. 75–94, Available at doi:10.2478/meceng-2013-0005.

[4] M. Arnold, C. Bausch, T. Blochwitz, C. Clauss, M. Monteiro, T. Neidhold, J.V. Peetz, and S. Wolf, *Functional Mock-up Interface for Co-Simulation* (2010), Available at https://svn.modelica.org/fmi/branches/public/specifications/v1.0/FMI_for_CoSimulation_v1.0.pdf.

[5] M. Busch, *Zur effizienten Kopplung von Simulationsprogrammen*, Ph.D. diss., Universität Kassel, 2012, Available at `http://books.google.de/books?id=0qBpXp-f2gQC`.

[6] R. Kübler and W. Schiehlen, *Modular Simulation in Multibody System Dynamics*, Multibody System Dynamics 4 (2000), pp. 107–127, Available at `http://dx.doi.org/10.1023/A:1009810318420`.

[7] S.B.E. Elmqvist, *Interface Jacobian-based Co-Simulation*, International Journal for Numerical Methods in Engineering 98 (2014), pp. 418–444, Available at `http://dx.doi.org/10.1002/nme.4637`.

[8] T. Moshagen, *Convergence of explicitly coupled Simulation Tools (Co-Simulations)*, Journal of Numerical Mathematics (2018), p. 27.

[9] R. Kossel, *Hybride Simulation thermischer Systeme am Beispiel eines Reisebusses*, Ph.D. diss., Techn. Univ. Braunschweig, 2012.

[10] D. Scharff, C. Kaiser, W. Tegethoff, and M. Huhn, *Ein einfaches Verfahren zur Bilanzkor-rektur in Kosimulationsumgebungen*, in *SIMVEC - Berechnung, Simulation und Erprobung im Fahrzeugbau*. 2012.

[11] M. Wells J.; Hasan and C. Lucas, *Predictive Hold with Error Correction Techniques that Maintain Signal Continuity in Co-Simulation Environments*, SAE Int. J. Aerosp (2012), pp. 481–493.

[12] D. Scharff, T. Moshagen, and J. Vondřejc, *Treating Smoothness and Balance during Data Exchange in Explicit Simulator Coupling or Cosimulation* (2017), p. 30, Available at `https://arxiv.org/abs/1703.05522`.

[13] P.D. Lax and R.D. Richtmyer, *Survey of the stability of linear finite difference equations*, Communications on Pure and Applied Mathematics 9 (56), pp. 267–293, Available at `https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.3160090206`.

[14] P. Deuflhard and F.A. Bornemann, *Numerische Mathematik II*, de Gruyter, 1994.

[15] *Scipy Integration and ODEs Web Documentation*. Available at `https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.ode.html`.

[16] K. Brenan, S. Campbell, and L. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, Society for Industrial and Applied Mathematics, 1995, Available at `http://epubs.siam.org/doi/abs/10.1137/1.9781611971224`.

[17] T. Moshagen, *Diffuse Grenzflächen thermodynamisch scharf - ein voll physikalisch eingebettetes Multiphasenfeldmodell*, Ph.D. diss., Universität Bremen, 2011, Available at `http://nbn-resolving.de/urn:nbn:de:gbv:46-00101865-17`, Online-Ressource (PDF: 208 S., 22,5 MB).

[18] A.V.D. Schaft, *Port-Hamiltonian Systems: an introductory Survey* (2006). Available at `http://www.icm2006.org/proceedings/Vol_III/contents/ICM_Vol_3_65.pdf`.

[19] A. van der Schaft and B.M. Maschke, *Port-hamiltonian Systems: a Theory for Modeling, Simulation and Control of Complex Physical Systems* (2003). Available at `http://www-lar.deis.unibo.it/euron-geoplex-sumsch/files/lectures_1/Van%20Der%20Schaft/VDSchaft_01_PCHS.pdf`.

[20] J. Dormand and P. Prince, *A Family of embedded Runge-Kutta Formulae*, Journal of Computational and Applied Mathematics 6 (1980), pp. 19 – 26, Available at `http://www.sciencedirect.com/science/article/pii/0771050X80900133`.