Methodology

Simulation

Simulation: Transactions of the Society for Modeling and Simulation International
1–14
© The Author(s) 2022
DOI: 10.1177/00375497221097128
journals.sagepub.com/home/sim
SAGE SCS

Numerically robust co-simulation using transmission line modeling and the Functional Mock-up Interface

Robert Braun[1] and Dag Fritzson[2]

Abstract
Modeling and simulation are important tools for efficient product development. There is a growing need for collaboration, interdisciplinary simulation, and re-usability of simulation models. This usually requires simulation tools to be coupled together for co-simulation. However, the usefulness of co-simulation is often limited by poor performance and numerical instability. Achieving stability is especially hard for stiff mechanical couplings. A suitable method is to use transmission line modeling (TLM), which separates submodels using physically motivated time delays. The most established standard for tool coupling today is the Functional Mock-up Interface (FMI). Two example models in one dimension and three dimensions are used to demonstrate how the next version of FMI for co-simulation can be used in conjunction with TLM. The stability properties of TLM are also proven by numerical analysis. Results show that numerical stability can be ensured without compromising on performance. With the current FMI standard, this requires tailor-made models and custom solutions for the interpolation of input variables. Without using custom solutions, variables must be exchanged using sampled communication and extrapolation. In this case, stability properties can be improved by reducing communication step size. However, it is shown that stability cannot be achieved even when using unacceptably small communication steps. This motivates the need for the next version of FMI to include an intermediate update mode, where variables can be interchanged in between communication points. It is suggested that the FMI standard should be extended with optional callback functions for providing intermediate output variables and requesting intermediate input variables.

Keywords
Functional Mock-up Interface, transmission line modeling, co-simulation, interpolation

1. Introduction

Model-based development of cyber-physical systems is a complex task. System-level modeling typically involves submodels from a diversity of different domains and disciplines. In addition, it also requires collaboration between various organizations and teams. This is usually addressed by using co-simulation, where submodels from different simulation tools or using different solver algorithms are coupled into an overall system model.[1]

The ability to connect simulation models from different tools offers several benefits. Each simulation tool is usually tailor-made for a certain problem or domain. With co-simulation, each part of a larger cyber-physical system can be modeled in its most suitable tool. In addition to providing more accurate results, this also enables multi-domain simulations and facilitates collaboration between organizations. Furthermore, distributed modeling can improve performance, modularity, and maintainability in model-based product development.

One challenge in co-simulation is to maintain numerical stability while dividing the equation system across separated solvers.[2] When using weak coupling methods, two dependent parts of a system can never be solved separately without introducing some delay to the interchanged variables. Strong coupling methods are able to avoid these delays,[3] but such methods are not supported by all tools or submodels because they require the capability of saving and restoring states. This is often difficult to implement in

[1]Linköping University, Sweden
[2]AB SKF, Sweden (Retired)

Corresponding author:
Robert Braun, Linköping University, 581 83 Linköping, Sweden.
Email: robert.braun@liu.se

mature tools. If left unresolved, delayed variables can affect both stability and accuracy. Stability problems can arise from sampled communication with too low sampling frequency or from sampled communication between models with different time constants, which may result in aliasing effects.[4]

Connecting simulation tools is facilitated by standardized interfaces, which reduce the need for writing custom code and using tool-dependent solutions. The most established standard for co-simulation today is the Functional Mock-up Interface (FMI).[5] In order to achieve stable co-simulation, it is important that such standards provide tools for ensuring numerical stability.

This paper analyzes FMI-based co-simulation using the transmission line modeling (TLM) technique for achieving stable tool coupling. The benefits of TLM are motivated by a numerical assessment of its stability properties. Limitations of FMI 2.0 are identified in order to motivate new features for the new FMI version 3.0.

1.1. Related work

The experiments presented in this work all use fixed communication step size. An alternative approach is to use adaptive communication step size.[6] This involves error estimation techniques and algorithms for reducing step size until the error is below a specified tolerance. When the estimated error is small, step size can be increased to improve performance. Algorithms for step size control can be either *conservative* or *optimistic*. Conservative algorithms always reject a step when the error tolerance is exceeded and repeat it with a smaller step size. Optimistic algorithms, on the contrary, allow the tolerance to be exceeded and continue with a smaller step size at the next step.

Another method is to rely on *relaxation* techniques,[7] which iteratively produce more accurate approximations based on an initial estimate of the resulting variables.

A drawback of both conservative step size control and relaxation techniques is that they rely on *rollback mechanisms*, meaning the capability of the simulation tool to undo computations and reset to a previous state. This feature is not supported by all simulation tools and is usually not possible for real-time simulations. Optimistic step size control, where the last step is not repeated when reducing the step size, does not require this. However, this may not be sufficient for sensitive models.[8]

Instability can be caused by aliasing effects as a result of sampled communication patterns. Such problems are particularly likely to occur when connecting submodels with a large divergence in time constants. If this is the case, anti-aliasing filters may need to be applied.[4,9] This requires submodels to provide intermediate output values, that is, not only the final value of the communication step, but also time-stamped values produced between the communication points. An anti-aliasing filter can then be applied in the master simulation tool. Although such a filter can improve stability, information that would be passed from one model to another is removed by the filter and stability and accuracy cannot be guaranteed. This method has similarities with the TLM approach in that it would benefit from similar extensions to the FMI standard that is described in this paper. Another similarity is that both methods provide good numerical stability while using fixed communication step size for data exchange. Consequently, it does not rely on rollback mechanisms and is suitable for real-time simulation.

Another method for improving performance without compromising on accuracy is suggested in Khaled et al.[10] By using context-aware extrapolation, communication step size can be increased while keeping integration errors within controlled bounds. This allows internal solvers to take longer integration steps. The technique is used in CHOPTrey, a forecasting framework for hybrid dynamic co-simulation.[11] However, this technique relies on slacked synchronization, which will always introduce some numerical errors and cannot provide guaranteed numerical stability for stiff connections.

A novel approximation technique based on dynamic decoupling and mixed-mode integration was presented in Papadopoulos and Leva.[12] Complex monolithic models are partitioned into weakly coupled submodels, suitable for distributed simulation. This allows increased simulation performance for large and complex models. Contrary to this, the work on this paper is more focused on connecting existing models from different simulation tools.

A co-simulation framework for hybrid discrete-event simulations of cyber-physical systems using FMI was presented in Camus et al.[13] Heterogeneous tools are integrated using a middle-ware based on the discrete-event system specification. In contrast to the work presented in this paper, the focus is on connecting Functional Mock-up Units (FMUs) with discrete-event components. Numerical stability problems are not investigated.

A common approach for investigating the numerical properties of a co-simulation algorithm is to discretize a test model with a coupling algorithm and then analyze the spectral radius of the resulting recurrence equation. Li[14] gives a good overview of how this method can be used to analyze numerical stability and convergence behavior of co-simulation methods. It has also been used to investigate continuous approximation techniques,[15] implicit algorithms,[3] adaptive algorithms,[16] and in the INTO CPS project.[8]

2. Background

2.1. Transmission line modeling

A co-simulation is a simulation of coupled *simulation units*. Typically, an *orchestrator* is used to coordinate the simulation.[17] Orchestration algorithms may be either *iterative* or *non-iterative*, use either *fixed* or *variable* communication steps, and be either *sequential* (Gauss-Seidel) or *parallel* (Jacobi). *Transmission line modeling* (*TLM*) can be seen as a non-iterative fixed-step parallel orchestration algorithm. Non-iterative co-simulation will inevitably introduce time delays to the exchanged variables. If not dealt with, such delays will affect the stability and accuracy of the simulation. TLM addresses this by exploiting physically motivated time delays for separating simulation units in time. Such delays will not affect numerical stability or accuracy because they are part of the physical model of the system. This is based on the fact that all physical interactions have a finite propagation speed. TLM originates from the method of bi-lateral delay lines[18] and provides numerically stable solver coupling.[19] Figure 1 shows an overview of a one-dimensional TLM element. All physical elements have both some inductance (L) and some capacitance (C). This is, for example, represented by mass and flexibility for mechanical systems and by fluid inertia and compressibility for fluid systems. A wave traversing an element is described by the *wave equation* as shown in Equation (1):

$$\frac{\partial^2 f(t,q)}{\partial t^2} = a^2 \frac{\partial^2 f(t,q)}{\partial q^2}. \tag{1}$$

The TLM equations are derived from the plane wave solution to the wave equation (see Equation (2)):

$$F(s,q) = C_1 e^{\frac{sq}{a}} + C_2 e^{-\frac{sq}{a}}. \tag{2}$$

The relationship between force and effort is given by the Telegrapher's equation:

$$\frac{\partial e(t,q)}{\partial t} = -aZ_c \frac{\partial f(t,q)}{\partial q}. \tag{3}$$

Combining Equations (2) and (3) yields the solution for the effort variable:

$$E(s,q) = -Z_c\left[C_1 e^{\frac{sq}{a}} - C_2 e^{-\frac{sq}{a}}\right]. \tag{4}$$

Constants C_1 and C_2 can be obtained from the boundary conditions, as shown in Equation (5). L is the length of the element, defined as $L = Ta$:

$$\begin{cases} E(s,q=0) = E_1(s) \\ F(s,q=0) = F_1(s) \\ E(s,q=L) = E_2(s) \\ F(s,q=L) = F_2(s) \end{cases}. \tag{5}$$

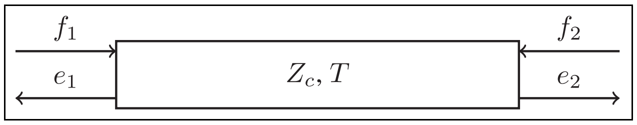

Figure 1. A one-dimensional TLM element with characteristic impedance Z_c and time delay T.

The latter two conditions also give the relationship between the variables on the two sides of the element:

$$E_2(s) = Z_c F_2(s) + (E_1(s) + Z_c F_1(s))e^{-sT}. \tag{6}$$

Exploiting symmetry and transforming to time domain yields the governing equations:

$$e_1(t) = e_2(t-T) + Z_c[f_1(t) + f_2(t-T)]. \tag{7}$$

$$e_2(t) = e_1(t-T) + Z_c[f_2(t) + f_1(t-T)]. \tag{8}$$

This can be further simplified by introducing *wave variables* according to Equations (9) and (10), representing all delayed information from the other end of the element:

$$c_1(t) = e_2(t-T) + Z_c f_2(t-T). \tag{9}$$

$$c_2(t) = e_1(t-T) + Z_c f_1(t-T). \tag{10}$$

This allows the TLM equations to be written in a more compact form, as shown in Equation (12):

$$e_1(t) = c_1(t) + Z_c f_1(t). \tag{11}$$

$$e_2(t) = c_2(t) + Z_c f_2(t). \tag{12}$$

Wave variables are exchanged between the interconnected simulation models with their specified time delay. The time delay (T) and characteristic impedance (Z_c) are computed from inductance L and capacitance C:

$$Z_c = \sqrt{LC}. \tag{13}$$

$$T = \sqrt{\frac{L}{C}}. \tag{14}$$

According to Equation (8), the effort variable on one side of the element does not depend on any variables from the other end at time t, because all variables from the other end are delayed. This gives a time frame of T, during which the solvers will be numerically isolated from each other. TLM has been used for multi-core simulation,[20,21] co-simulation,[22,23] and real-time simulation.[19,24] It has also been successfully implemented for multi-body mechanics.[25,26]

Mathematical properties of a TLM element can be analyzed by assuming zero flow on one end of the element ($f_2 = 0$). This will double the total time delay since the wave needs to travel both back and forth. For this reason, the time delay is divided by 2. With these modifications,

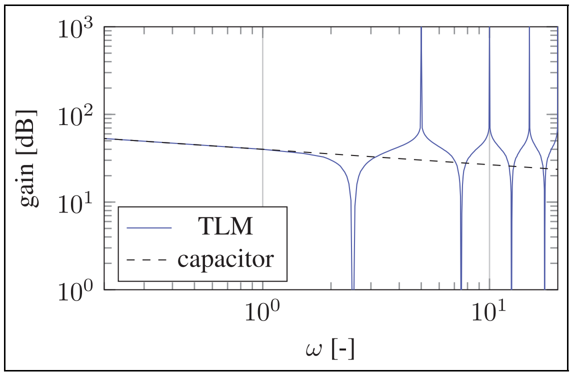

Figure 2. Bode diagram comparing a TLM connection with an ideal capacitor.

Equation (8) yields Equation (15). A closed-end TLM element gives a risk of standing waves and can be considered a worst-case scenario from a stability point of view:

$$e_1(t) = e_1(t - T) + Z_c[f_1(t) + f_1(t - T)]. \qquad (15)$$

The Laplace transform of Equation (15) gives the following:

$$E_1(s) = Z_c \frac{1 + e^{-Ts}}{1 - e^{-Ts}} F_1(s) = G_{\text{TLM}}(s) F_1(s). \qquad (16)$$

TLM elements are commonly introduced by replacing ideal capacitors, for example, fluid volumes or ideal springs. Hence, Figure 2 shows the Bode diagram for $G_{\text{TLM}}(s)$ compared to that of a pure capacitor. Both transfer functions use the same capacitance parameter. While the spring is acting as a perfect integrator between effort and flow, the TLM element exhibits harmonic oscillations. These are not numerical artifacts but represent the actual harmonics in the physical element. In conclusion, a TLM connection not only introduces a time frame for solver independence but also provides a more accurate representation of the real physical element than an ideal capacitor connection.

2.2. Functional Mock-up Interface

The FMI is an open tool-independent standard for co-simulation and model exchange.[5] Models are exchanged as compressed archives containing an XML specification file and compiled C-code. Such an archive is called an FMU. An FMU can support FMI for co-simulation (FMI CS), FMI for model exchange (FMI ME), or both. With FMI CS, the numerical solver is embedded inside the FMU. Interconnected FMUs exchange data on predefined communication points. FMI ME, on the contrary, requires a numerical solver in the tool that imports the FMU, hereafter denoted as "importing tool." Each FMU exposes its states and their time derivatives, that is the right-hand side of the system of ordinary differential equations (ODEs). The current version of FMI is 2.0. At the time of writing, version 3.0 is still under development.

Table 1. Continuity of output and input variables with four different FMI approaches.

Input / Output	Zero-order hold	First-order hold	intermediate
Sampled	FMI 2.0 CS	FMI 2.0 CS(CG)	FMI 2.0 CS(FG)
Intermediate			FMI ME/FMI 3.0 CS

FMI: Functional Mock-up Interface; CG: coarse-grained interpolation; FG: fine-grained interpolation.

2.3. Sampling techniques

Four different sampling methods have been investigated and compared, as shown in Table 1. With FMI ME, the data flow is controlled completely by the importing tool. Thus, both inputs and outputs can be read from or written to the FMU whenever needed. This enables a continuous data exchange.

When stepping from t_n to t_{n+1} with TLM, input variables are always available for $t \leqslant t_{n+1}$. This is a consequence of the limitation of $T_{\text{com}} < 0.5T$ and that maximum internal step sizes in the submodels are limited to T_{com}, as explained in Nakhimovski.[27] Even in the worst possible case, when one submodel begins its step directly before the other submodel is finished stepping, interpolated variables will then still be available during the entire step for the first submodel.

If FMI CS is used without input derivatives, inputs will be sampled according to a zero-order hold model. This often provides inadequate stability for dynamic simulations. If using TLM with zero-order hold, future information is not taken into account. Figure 3 shows the principle of zero-order hold inputs, and Figure 4 shows the communication pattern. As can be seen, both inputs and outputs are only updated before and after the communication step.

In order to improve accuracy, it is possible to provide time derivatives of input variables. This method is denoted coarse-grained interpolation (CG). These derivatives can be estimated by using the difference between the values just before and just after the step:

$$\tilde{\dot{c}}(t) = \frac{c(t_{n+1}) - c(t_n)}{t_{n+1} - t_n}. \qquad (17)$$

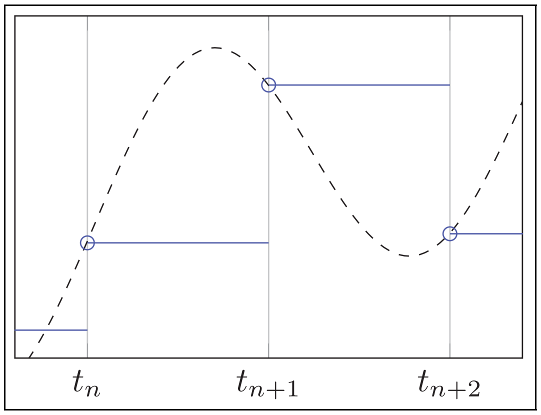

Figure 3. Sampling of input variables without interpolation (zero-order hold).

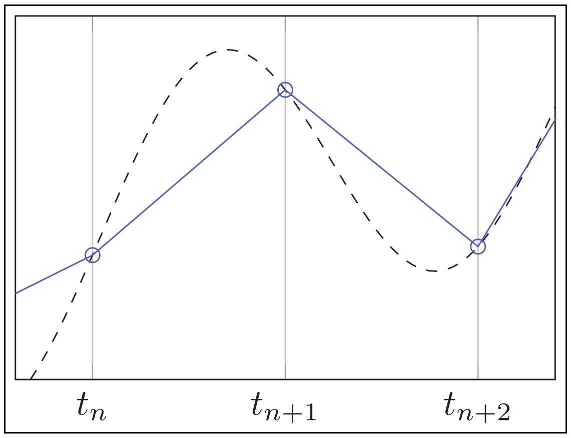

Figure 5. Sampling of input variables with coarse-grained interpolation (first-order hold).

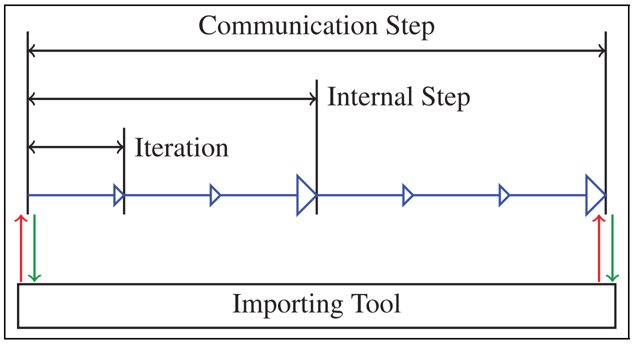

Figure 4. Communication pattern of FMI for co-simulation with zero-order hold. Vertical arrows represent reading inputs and writing output, respectively. Large unfilled arrows represent successful internal steps, while small unfilled arrows represent solver iterations.

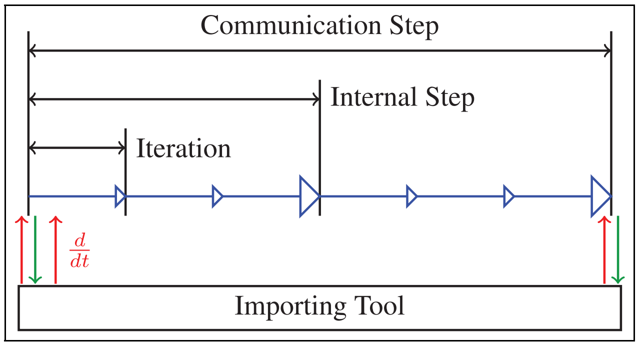

Figure 6. Communication pattern of FMI for co-simulation with coarse-grained interpolation. Vertical arrows represent reading inputs and writing output, respectively. Large unfilled arrows represent successful internal steps, while small unfilled arrows represent solver iterations.

$$\begin{gathered} c(t) = c(t_n) + \tilde{\dot{c}}(t)(t - t_n) \\ \text{where} \\ t_n < t < t_{n+1}. \end{gathered} \tag{18}$$

This requires that the FMU is capable of interpolating using input derivatives, which is not always the case. The first-order hold principle is shown in Figure 5, and the communication pattern is shown in Figure 6. Inputs and outputs are only updated before and after the communication step, but the input derivative is also provided. Hence, the FMU can use interpolation internally to estimate the intermediate values of the input variables.

Numerically stiff problems may require higher resolution of interpolation tables in order to ensure stability. This is addressed by providing the FMU with an entire interpolation table, denoted fine-grained interpolation (FG). A large number of variables and corresponding time stamps are written to the FMU using the `setReal()` API function. For no particular reason, the number of samples to be provided to the FMU at each communication step was set to 10. In this way, intermediate inputs will be available inside the FMU whenever its solver needs them. However, outputs can still only be provided after each communication step. Figure 7 shows the principle of fine-grained interpolation, and Figure 8 shows the communication pattern. Input variables can be updated whenever the internal solver needs them, that is, at every internal step and every solver iteration.

For comparison, the communication pattern when using the intermediate update mode in FMI 3.0 is shown in Figure 9. As for fine-grained interpolation, inputs can be obtained on every internal solver iteration. In addition, output values can also be sent back to the master tool after every successful internal step. Table 1 summarizes

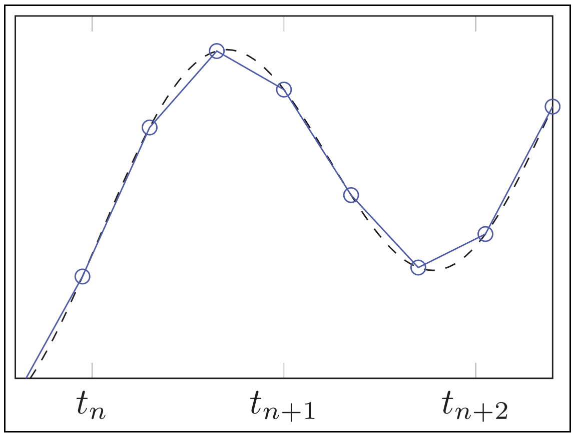

Figure 7. Sampling of input variables with fine-grained interpolation (continuous).

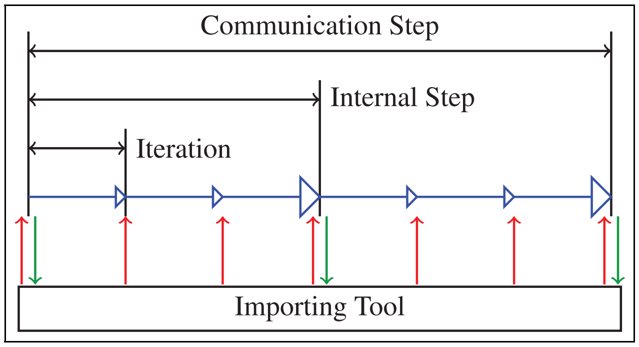

Figure 9. Communication pattern of FMI for co-simulation using intermediate update mode. Vertical arrows represent reading inputs and writing output, respectively. Large unfilled arrows represent successful internal steps, while small unfilled arrows represent solver iterations.

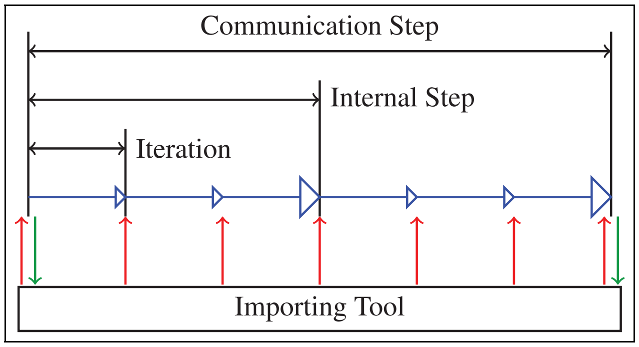

Figure 8. Communication pattern of FMI for co-simulation with fine-grained interpolation. Vertical arrows represent reading inputs and writing output, respectively. Large unfilled arrows represent successful internal steps, while small unfilled arrows represent solver iterations.

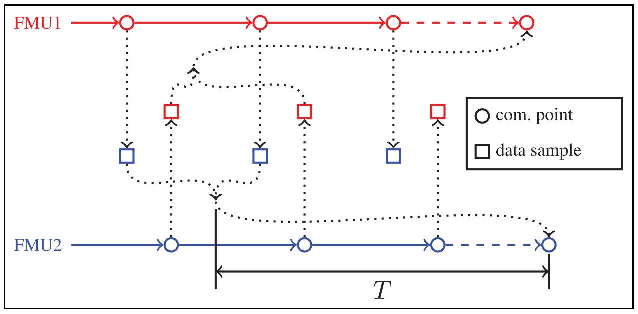

Figure 10. Asynchronous communication pattern with interpolation used by OMSimulator.

sampling techniques for input and output variables provided by each of the data transfer approaches.

2.4. OMSimulator

OMSimulator is an open-source master simulation tool for FMI-based co-simulation developed and maintained by the Open Source Modelica Consortium.[28] It supports both FMI 2.0 CS and FMI 2.0 ME, and FMUs can be connected using either causal connections or TLM. The TLM capabilities were originally developed at SKF[23] as part of the BEAST simulation tool.[29] TLM submodels communicate with the simulator using Transmission Control Protocol/Internet Protocol (TCP/IP) network sockets, which enables the use of distributed computers. OMSimulator has a C API and scripting support for Lua and Python. There are also graphical user interfaces in OpenModelica Connection Editor (OMEdit)[30] and Papyrus.[31] In addition to the demonstrator models in this paper, OMSimulator has been used for modeling cyber-physical aircraft systems.[32]

Interconnected TLM FMUs may not always use the same integration step size. Integration step sizes may also vary during simulation in case an FMU is using internal step size control. OMSimulator handles this by allowing asynchronous data exchange. Whenever a step in a submodel is complete, it sends time-stamped data to the master simulation tool. These data are inserted into the interpolation table of the other FMU. Whenever input data are required by an FMU, they are interpolated from this table. This approach is enabled by the use of physically motivated delays. Assuming the maximum integration step in each FMU is limited to half the TLM delay, data will always be available for the entire duration of each macro step. Figure 10 shows the communication pattern between two FMUs.

Choosing an interpolation method is a trade-off between accuracy and stability. Higher order interpolation may produce more accurate approximations. On the other hand, it can result in aliasing problems when connecting two models with a large deviation in time scales. While linear

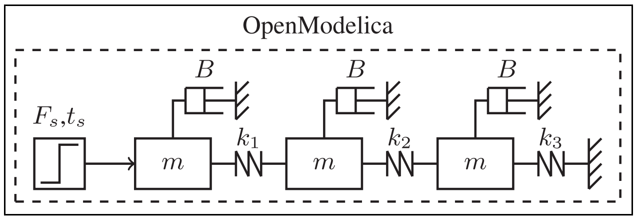

Figure 11. Monolithic reference model of 1D spring–mass demonstrator model.

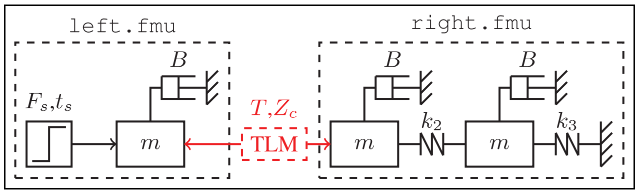

Figure 12. One-dimensional spring–mass demonstrator model consisting of two FMUs.

interpolation may be less accurate, it guarantees that the interpolated value is always between the two surrounding values, which reduces the risk of instability. One of the strengths of the TLM method is that it effectively allows both solvers to use interpolation. The same is true for strong coupling methods, but the TLM method requires no rollback. This results in a method with the same stability properties as strong coupling methods, as shown in the numerical stability analysis section, but without the performance penalty or the implementation complexity.

3. Case studies

3.1. 1D demonstrator model

A spring–mass system with three masses is used as a one-dimensional (1D) demonstrator model. A monolithic reference model is created in OpenModelica (see Figure 11). The composite model is created by replacing the leftmost spring with a TLM element (see Figure 12). Time delay and characteristic impedance are tuned so that the spring stiffness k_s will be the same as in the monolithic model according to Equation (19). This results in a parasitic inductance m_s determined by Equation (20). $Z_c = 0.4$ N s/m and $T = 4 \times 10^{-4}$ s thus give $k_s = 1000\,\text{N/m}$ and $m_s = 2 \times 10^{-4}$ kg:

$$k_s = \frac{Z_c}{T}. \tag{19}$$

$$m_s = Z_c \times T. \tag{20}$$

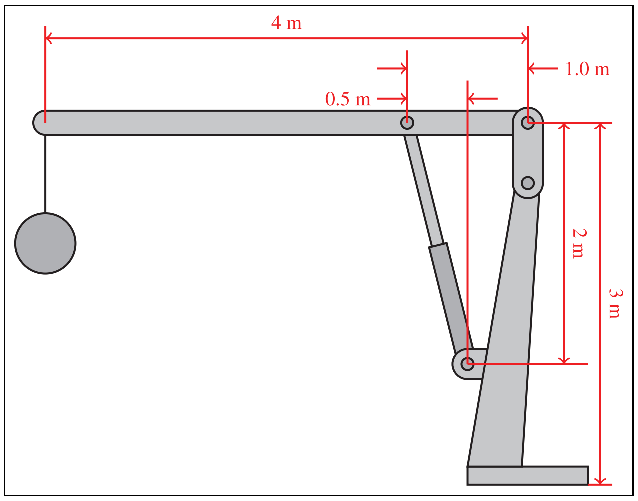

Figure 13. Geometry of the hydraulic crane demonstrator model.

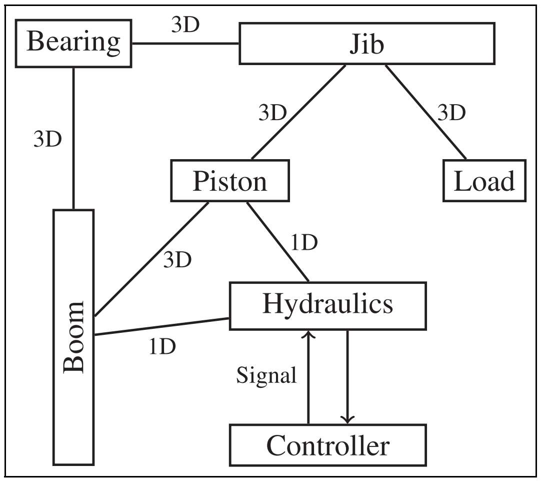

Figure 14. Structure of composite model and connection types with 1D and 3D TLM elements.

3.2. 3D demonstrator model

For analyzing the proposed methods under more realistic conditions, an industrial demonstrator for evaluation of cyber-physical systems including rolling bearings has been developed. An earlier prototype of the demonstrator was described in Braun et al.[33] The composite model represents a hydraulic crane as shown in Figure 13. Submodels include mechanical parts, a hydraulic circuit, and a controller. Figure 14 shows an overview of the interconnected submodels. The bearing is modeled in BEAST[29] and includes flexible bodies and contact mechanics. Other mechanical bodies are modeled as rigid bodies in Dymola.[34] They are subsequently exported to

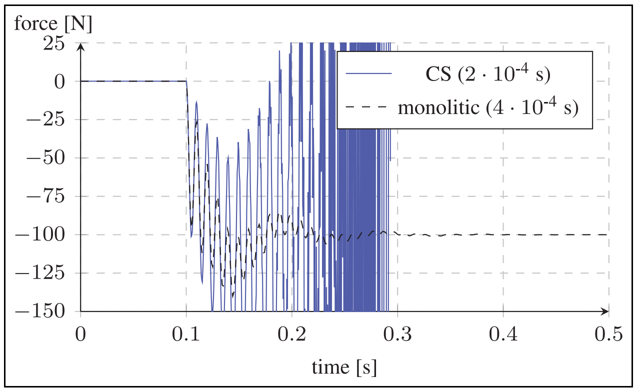

Figure 15. Force in 1D model with zero-order hold.

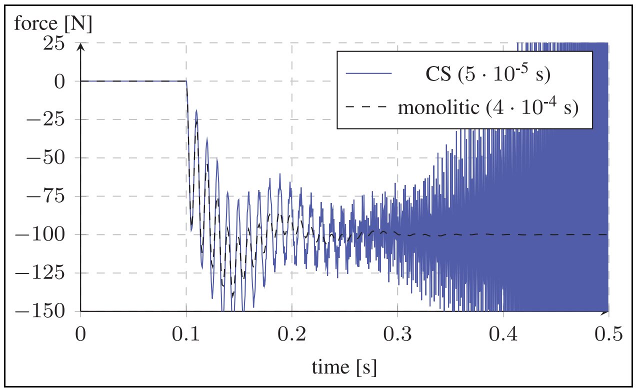

Figure 16. Force in 1D model with zero-order hold.

OMSimulator as FMUs, for either CS or ME. Finally, the hydraulic circuit and the controller are modeled in Hopsan,[35] a simulation tool for hydraulic and mechatronic systems developed at Linköping University. Hopsan is based on TLM technology and has previously been used for co-simulation.[22,36]

No single tool is capable of simulating all the parts of the crane model without simplifications. For this reason, it is not possible to create a monolithic reference model. This is the main reason why this study requires co-simulation. Consequently, FMI ME must be used as reference for the experiments with FMI CS. FMI ME provides the best possible results since it supports both intermediate outputs and intermediate inputs. All simulations on the crane model use the TLM method. Parameters for the translational and rotational connections are as follows:

$$\begin{aligned} T &= 1\times10^{-3}\ \text{s} & Z_{cr} &= 1\times10^{4}\ \text{N m s/rad},\\ Z_c &= 1\times10^{5}\ \text{N s/m} & k_r &= 1\times10^{7}\ \text{N m/rad},\\ k &= 1\times10^{8}\ \text{N/m} & J &= 10\ \text{kg m}^2,\\ m &= 1\times10^{2}\ \text{kg}. \end{aligned}$$

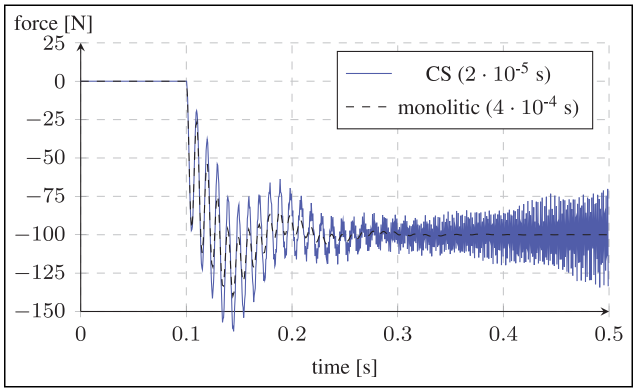

Figure 17. Force in 1D model with zero-order hold.

4. Results

Simulation results for both demonstrator models were compared to reference simulations. For the spring–mass–damper model, a monolithic reference model was used. The crane demonstrator uses a reference simulation with FMI 2.0 for model exchange, since it cannot be converted to a monolithic model. Both reference models provide stable results.

4.1. FMI 2.0 CS (zero-order hold)

Both models were first simulated with FMI CS without providing input derivatives. This results in a zero-order hold communication pattern. Step size is reduced iteratively to improve stability. Results for the spring–mass model are shown in Figures 15–17. Stability cannot be achieved even with $T_{\text{com}} = 2\times10^{-5}$ s, which is 2000 times smaller than the step size in the reference model.

Figures 18 and 19 show results for the crane model with different step sizes. Results are similar to those for the spring–mass model. Simulation becomes unstable even when using $T_{\text{com}} = 1\times10^{-6}$ s, a step size 1000 times smaller than what was used for the reference simulation.

4.2. FMI 2.0 CS (coarse-grained interpolation)

When providing FMUs with first-order time derivatives of the input variables, stability is greatly improved. Results for the spring–mass model are shown in Figures 20 and 21. When using half the step size compared to the monolithic reference model, early stages of instability can be seen. Reducing step size to 1×10^{-4} s gives only small and stationary oscillations.

The crane model is stable for the piston motion when using the same step size as the FMI ME reference model. However, the rotational motion becomes unstable after approximately 4.5 s of simulation. Figures 22 and 23 show the results.

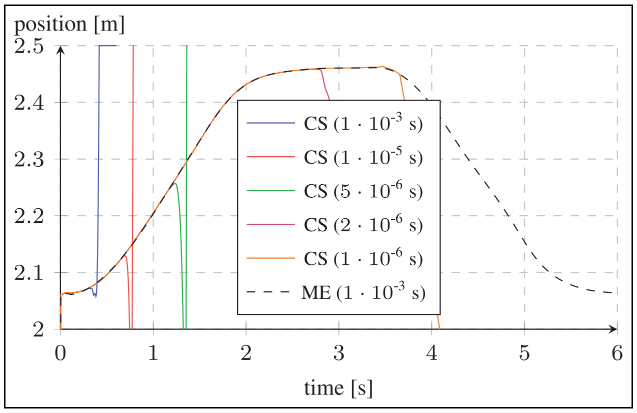

Figure 18. Piston position in 3D model with zero-order hold.

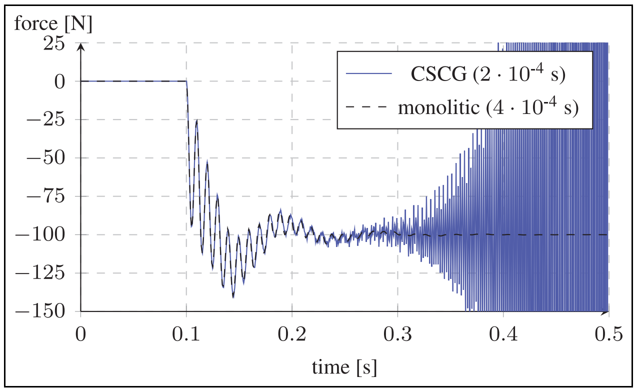

Figure 20. Force in 1D model with coarse-grained interpolation.

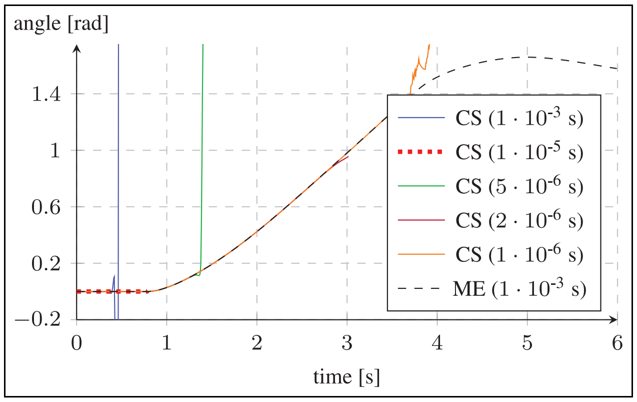

Figure 19. Motor angle in 3D model with zero-order hold.

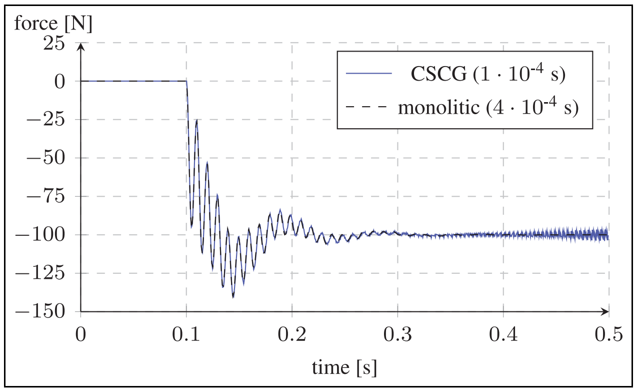

Figure 21. Force in 1D model with coarse-grained interpolation.

4.3. FMI 2.0 CS (fine-grained interpolation)

Fine-grained interpolation was implemented by providing the FMUs with 10 evenly distributed time-stamped entries of each input variable. FMUs use linear interpolation to internally compute input variables or the time instances where they are needed. Results for the spring–mass model are shown in Figure 24. When using the same step size as the monolithic reference model, no signs of instability can be observed. It can be concluded that 10 samples are sufficient for stabilizing the connection, even though a denser interpolation table could theoretically improve stability.

Figures 25 and 26 show the results for the crane model with fine-grained interpolation. No stability issues are observed with $T_{\text{com}} = 1 \times 10^{-3}$ s.

4.4. Numerical stability analysis

The stability properties of a coupled system can be determined by analyzing the spectral radius of the corresponding linear recurrence equation system. We will now illustrate these properties with an example model consisting of a linear two-mass oscillator as shown in Figure 27.

The two submodels have position coordinates x_1 and x_2, while m_1 and m_2 denote their masses, b_1 and b_2 the damping coefficients, and k_1 and k_2 the spring stiffnesses. For simplicity, the position coordinates are defined in opposite directions to make the equation system symmetrical. The two submodels are connected by a TLM element, which represents the spring between the two masses. This is actually a more accurate representation of a physical spring than just using a spring constant, since it includes not only the stiffness of the spring, but also its distributed inertia. Equations for the submodels are shown in Equations (21a) and (21b):

$$\begin{bmatrix} \dot{v}_1 \\ \dot{x}_1 \end{bmatrix} = \underbrace{\begin{bmatrix} -\frac{b_1}{m_1} & -\frac{k_1}{m_1} \\ 1 & 0 \end{bmatrix}}_{\mathbf{A_1}} \begin{bmatrix} v_1 \\ x_2 \end{bmatrix} + \underbrace{\begin{bmatrix} \frac{1}{m_1} \end{bmatrix}}_{\mathbf{B_1}} \begin{bmatrix} -F_{c,1} \\ 0 \end{bmatrix}. \qquad (21a)$$

$$\begin{bmatrix} \dot{v}_2 \\ \dot{x}_2 \end{bmatrix} = \underbrace{\begin{bmatrix} -\frac{b_2}{m_2} & -\frac{k_2}{m_2} \\ 1 & 0 \end{bmatrix}}_{\mathbf{A_2}} \begin{bmatrix} v_2 \\ x_2 \end{bmatrix} + \underbrace{\begin{bmatrix} \frac{1}{m_2} \end{bmatrix}}_{\mathbf{B_2}} \begin{bmatrix} -F_{c,2} \\ 0 \end{bmatrix}. \qquad (21b)$$

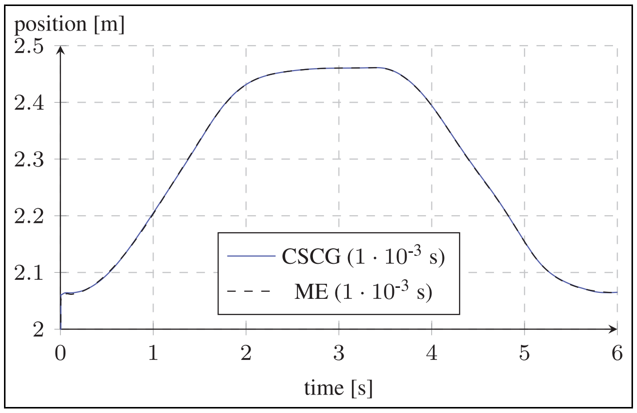

Figure 22. Piston position in 3D model with coarse-grained interpolation.

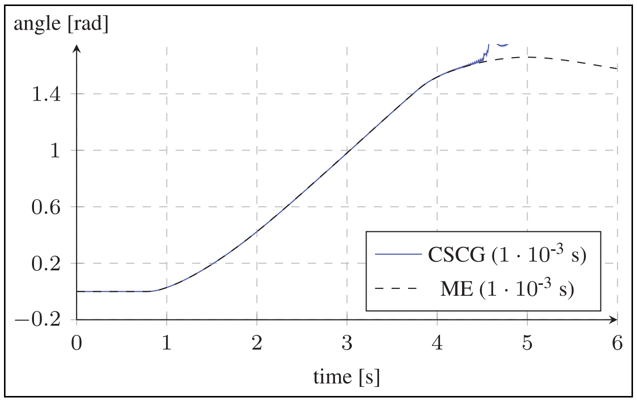

Figure 23. Motor angle in 3D model with coarse-grained interpolation.

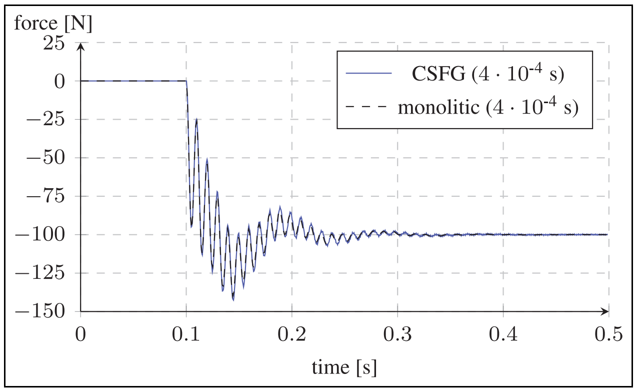

Figure 24. Force in 1D model with fine-grained interpolation.

Here, F_c represents the coupling force. It was shown in Li[14] that a spring–mass–damper system can be viewed as a mechanical representation of Dahlquist's test equation:

$$\dot{y}(t) = \Lambda \times y(t), \tag{22}$$

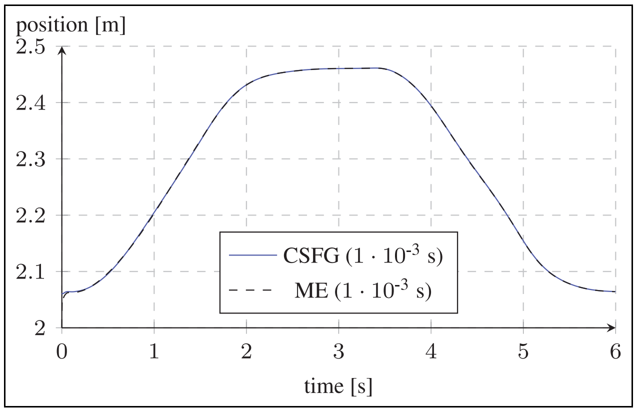

Figure 25. Piston position in 3D model with fine-grained interpolation.

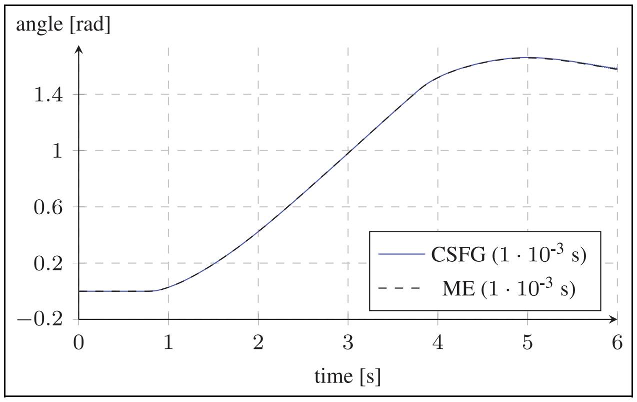

Figure 26. Motor angle in 3D model with fine-grained interpolation.

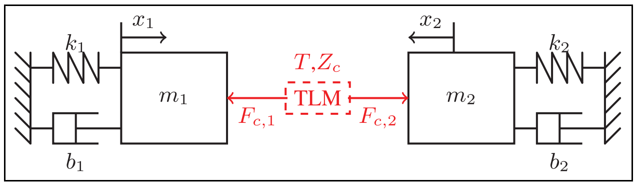

Figure 27. Linear two-degree-of-freedom oscillator with a TLM element.

where Λ is the eigenvalues of matrices A_1 and A_2:

$$\Lambda = -\frac{b}{2m} \pm \sqrt{\left(\frac{k}{m} - \frac{b}{2m}\right)^2} i. \tag{23}$$

Replacing F_c with the TLM boundary equation yields the equation system for the coupled problem:

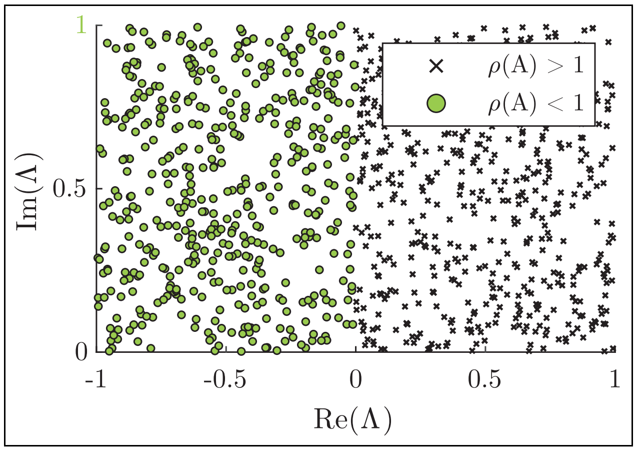

Figure 28. Spectral radius for the coupled system against the real and imaginary parts of the eigenvalues of the subsystems. Results confirm that TLM is A-stable.

$$\begin{cases} \dot{v}_1^n = -\frac{b_1}{m_1} v_1^n - \frac{k_1}{m_1} x_1^n - \frac{1}{m_1} F_{c,1}^n \\ \dot{x}_1^n = v_1^n \\ F_1^n = Z_c v_1^n + Z_c v_2^{n-1} + F_2^{n-1} \\ \dot{v}_2^n = -\frac{b_2}{m_1} v_2^n - \frac{k_2}{m_2} x_2^n - \frac{F_2^n}{m_2} \\ \dot{x}_2^n = v_2^n \\ F_{c,2}^n = Z_c v_2^n + Z_c v_1^{n-1} + F_1^{n-1} \end{cases} . \qquad (24)$$

The discretization delay between points n and $n-1$ equals the TLM time delay. The two subsystems consist of linear ordinary differential equations and can be integrated analytically. This is important in order to ensure that the stability properties of the coupled system are not affected by the numerical integration performed by each FMU. Applying analytical integration to Equation (24) yields a linear algebraic equation system described by the recurrence relation:

$$\mathbf{y}^n = \mathbf{A}\mathbf{y}^{n-1}, \qquad (25)$$

where $\mathbf{y} = [x_1 \quad v_1 \quad F_1 \quad x_2 \quad v_2 \quad F_2]^T$. According to theory, a recurrence relation is guaranteed to converge if the spectral radius of the matrix (the largest absolute value of its eigenvalues) is strictly less than 1:

$$\begin{cases} \lim_{k->\infty} \mathbf{A}^k = 0, & \rho(\mathbf{A}) < 1 \\ \lim_{k->\infty} \mathbf{A}^k = \infty, & \rho(\mathbf{A}) > 1 \end{cases} . \qquad (26)$$

The spectral radius of matrix **A** cannot be obtained analytically. Instead, the eigenvalues were computed numerically using a large number of randomized parameters with symmetric values for the two subsystems. Figure 28 shows the spectral radius plotted against the real and imaginary parts of the eigenvector of the subsystems. As can be seen, the spectral radius is smaller than 1 for the entire left-hand plane and larger than 1 for the entire right-hand plane. Hence, the TLM method is A-stable.

This conforms with previous research, which has shown that when a capacitance is replaced by a TLM element, the resulting equations are equal to the trapezoid rule of integration between flow and effort.[19,37] As a consequence, the stability region of a TLM coupling is the same as for the trapezoid rule.

5. TLM in FMI 3.0 CS

While the investigated interpolation techniques are shown to be effective, they require custom solutions and tailor-made FMUs. This could be avoided by allowing data transfer between the importing tool and an FMU between the regular communication points. If an FMU could request interpolated variables whenever they are needed or return variables to the master as soon as they are produced, all interpolations could be handled in the importing tool. This would enable numerically stable co-simulation with standard FMUs from an arbitrary simulation tool.

With the intermediate update mode planned for FMI 3.0, an FMU will be able to request intermediate input variables or expose intermediate outputs in between two communication points. This enables a master algorithm to run asynchronous co-simulation with TLM couplings between any FMUs, with the only requirement that each FMU supports intermediate update mode. The TLM boundary equations, for example, the calculation of effort variables, should preferably be handled by the importing tool. In this way, the only requirement on the FMUs is that they must take one effort variable as input and provide one flow variable as output. For the mechanical domain, this would equal an input force and an output speed. Such interfaces are already commonly used for, for example, force–force couplings, and many existing models can be adapted to these needs with minor efforts. This is an advantage when using co-simulation to connect existing models from different disciplines or organizations.

Fine-grained interpolation could be an alternative solution, but this requires the FMUs to handle interpolation internally. For this to be effective in the general case, it is not sufficient to rely on a fixed number of samples, as the density of the interpolation data may change between communication steps. FMI 3.0 does support array variables, but changing their length requires the FMU to enter *reconfiguration mode* to reallocate its memory. This would most likely reduce simulation performance significantly.

6. Conclusion

Co-simulation between different simulation tools can facilitate collaboration, improve modularity, and preserve investments. It also makes it possible to use the best suited tool for each part of a larger cyber-physical model. Efficient co-simulation requires standardized interfaces, low performance overheads, and numerically stable connections.

It is shown that FMI ME works well with TLM. FMI 2.0 for co-simulation without interpolation relies on zero-order hold sampling. With this approach, a stable connection can never be guaranteed. Stability issues arise not only due to discontinuous input variables, but also due to aliasing effects.

Coarse-grained interpolation using input derivatives provides zero-order continuity of input signals. This enables improved stability but is not able to fully stabilize the connection without reducing communication step size. While a reduction in communication step size can stabilize the connection, it also reduces simulation performance by effectively limiting the maximum integration step of the solvers in the FMUs. Moreover, it is difficult to determine in advance how small a step size that is needed to ensure stability during the entire simulation.

Fine-grained interpolation provides stable connections for both models without the need to reduce communication step size. Although output variables are sampled, the access to intermediate input variables is able to stabilize the connections. Ten samples for each communication step are sufficient in both examined models. Denser interpolation tables would provide more accurate results but are most likely only needed when connecting models with a larger deviation in time scales. A drawback with fine-grained interpolation is that a large number of variables need to be provided to the FMU at every communication step. It also requires fixed-size interpolation tables, regardless of the needs from the internal solver.

A more feasible solution would be to use intermediate variable access so that the FMU can provide intermediate output values or request intermediate input values during the step execution. Both interpolation and coupling equations can then be handled by the importing tool, which would allow connecting existing models without significant modifications. While it was shown that numerically stable asynchronous communication is possible even without the intermediate update mode, it makes the implementation significantly easier. In fact, it even eliminates the need for the communication steps, since the two solvers can exchange data whenever they need to in a truly asynchronous way.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work has been supported by Vinnova within the ITEA OpenCPS project.

ORCID iD

Robert Braun https://orcid.org/0000-0002-7480-1922

References

1. Hafner I and Popper N. On the terminology and structuring of co-simulation methods. In: *Proceedings of the 8th international workshop on equation-based object-oriented modeling languages and tools (EOOLT'17)*, Weßling, 1 December 2017, pp. 67–76. New York: Association for Computing Machinery (ACM).
2. Kübler R and Schiehlen W. Modular simulation in multibody system dynamics. *Multibody Syst Dyn* 2000; 4: 107–127.
3. Schweizer B, Li P, Lu D, et al. Stabilized implicit co-simulation methods: solver coupling based on constitutive laws. *Arch Appl Mech* 2015; 85: 1559–1594.
4. Benedikt M, Watzenig D and Hofer A. Modelling and analysis of the non-iterative coupling process for co-simulation. *Math Comp Model Dyn* 2013; 19: 451–470.
5. Blochwitz T, Otter M, Arnold M, et al. The Functional Mockup Interface for tool independent exchange of simulation models. In: *Proceedings of the 8th international Modelica conference 2011*, Dresden, 20–22 March 2011, pp. 105–114. Como: Linköping University Electronic Press.
6. Schierz T, Arnold M and Clauß C. Co-simulation with communication step size control in an FMI compatible master algorithm. In: *Proceedings of the 9th international Modelica conference*, Munich, 3–5 September 2012, pp. 205–214. Como: Linköping University Electronic Press.
7. Schweizer B, Lu D and Li P. Co-simulation method for solver coupling with algebraic constraints incorporating relaxation techniques. *Multibody Syst Dyn* 2016; 36: 1–36.
8. Gomes C, Thule C, Lausdahl K, et al. Demo: stabilization technique in INTO-CPS. In: Mazzara M, Ober I and Salaün G (eds) *Software technologies: applications and foundations*. Cham: Springer International Publishing, 2018, pp. 45–51.
9. Drenth E. Method and system for control and co-simulation of physical systems. Patent application 15/232,261, USA, 2017.
10. Khaled AB, Duval L, Gaid MB, et al. Context-based polynomial extrapolation and slackened synchronization for fast multi-core simulation using FMI. In: *Proceedings of the 10th international Modelica conference*, Lund, 10–12 March 2014, pp. 225–234. Como: Linköping University Electronic Press.
11. Khaled-El Feki AB, Duval L, Faure C, et al. CHOPtrey: contextual online polynomial extrapolation for enhanced multi-core co-simulation of complex systems. *Simulation* 2017; 93: 185–200.
12. Papadopoulos AV and Leva A. Automating efficiency-targeted approximations in modelling and simulation tools: dynamic decoupling and mixed-mode integration. *Simulation* 2014; 90: 1158–1176.
13. Camus B, Paris T, Vaubourg J, et al. Co-simulation of cyber-physical systems using a DEVS wrapping strategy in the MECSYCO middleware. *Simulation* 2018; 94: 1099–1127.

14. Li P. *On the numerical stability of co-simulation methods*. PhD Thesis, Technische Universität Darmstadt, Darmstadt, 2017.
15. Busch M. Continuous approximation techniques for co-simulation methods: analysis of numerical stability and local error. *ZAMM: Z Angew Math Me* 2016; 96: 1061–1081.
16. Gomes C, Legat B, Jungers RM, et al. Stable adaptive co-simulation: a switched systems approach. In: Schweizer B (ed.) *IUTAM symposium on solver-coupling and co-simulation*. Cham: Springer International Publishing, 2019, pp. 81–97.
17. Gomes C and Vangheluwe H. Co-simulation of continuous systems: a hands-on approach. In: *Proceedings of the 2019 winter simulation conference (WSC)*, National Harbor, MD, 8–11 December 2019, pp. 1469–1481. New York: IEEE.
18. Auslander DM. Distributed system simulation with bilateral delay-line models. *J Basic Eng: T ASME* 1968; 90: 195–200.
19. Krus P. Robust modelling using bi-lateral delay lines for real time and faster than real time system simulation. In: *Proceedings of the ASME international design engineering technical conferences and computers and information in engineering conference 2009 (DETC'2009)*, San Diego, CA, 30 August–2 September 2009, vol. 2, pp. 131–138. New York: American Society of Mechanical Engineers (ASME).
20. Burton JD, Edge KA and Burrows CR. Modeling requirements for the parallel simulation of hydraulic systems. *J Dyn Syst: T ASME* 1994; 116: 137–145.
21. Braun R and Krus P. Multi-threaded distributed system simulations using the transmission line element method. *Simulation* 2016; 92: 921–930.
22. Braun R, Ericson L and Krus P. Full vehicle simulation of forwarder with semi-active suspension using co-simulation. In: *Proceedings of the ASME/BATH 2015 symposium on fluid power and motion control*, Chicago, IL, 12–14 October 2015. New York: American Society of Mechanical Engineers (ASME).
23. Siemers A, Fritzson D and Nakhimovski I. General meta-model based co-simulations applied to mechanical systems. *Simul Model Pract Th* 2009; 17: 612–624.
24. Braun R and Krus P. Multi-threaded real-time simulations of fluid power systems using transmission line elements. In: *Proceedings of the 8th international fluid power conference (IFK)*, Dresden, 26–28 March 2012.
25. Krus P. Modeling of mechanical systems using rigid bodies and transmission line joints. *J Dyn Syst: T ASME* 1999; 121: 606–611.
26. Fritzson D, Ståhl J and Nakhimovski I. Transmission line co-simulation of rolling bearing applications. In: *Proceedings of the 48th Scandinavian conference on simulation and modeling*, Göteborg, 30–31 October 2007, pp. 24–39. LiU Electronic Press.
27. Nakhimovski I. *Contributions to the modeling and simulation of mechanical systems with detailed contact analyses*. PhD Thesis, Programming Environment Laboratory (PELAB), The Institute of Technology, Linköping University, Linköping, 2006.
28. Ochel L, Braun R, Thiele B, et al. OMSimulator—integrated FMI and TLM-based co-simulation with composite model editing and SSP. In: *Proceedings of the 13th international Modelica conference*, Regensburg, 4–6 March 2019, pp. 69–78. Como: Linköping University Electronic Press.
29. Fritzson D, Stacke LE and Anders J. Dynamic simulation — building knowledge in product development. *Evolution* 2014; 1, https://evolution.skf.com/dynamic-simulation-building-knowledge-in-product-development/
30. Mengist A, Asghar A, Pop A, et al. An open-source graphical composite modeling editor and simulation tool based on FMI and TLM co-simulation. In: *Proceedings of the 11th international Modelica conference*, Versailles, 21–23 September 2015, pp. 181–188. Como: Linköping University Electronic Press.
31. Tavella JP, Caujolle M, Tan C, et al. Toward an hybrid co-simulation with the FMI-CS standard. Research report, 7 April 2016. ResearchGate and authors' instutition websites.
32. Hällqvist R, Schminder J, Eek M, et al. A novel FMI and TLM-based desktop simulator for detailed studies of thermal pilot comfort. In: *Proceedings of the 31st congress of the international council of the aeronautical sciences*, Belo Horizonte, Brazil, 9–14 September 2018, pp. 1–6. New York: IEEE.
33. Braun R, Asghar A, Pop A, et al. An open-source framework for efficient co-simulation of fluid power systems. In: *Proceedings of the 15th Scandinavian international conference on fluid power*, Linköpings Universitet, Linköping, 7–9 June 2017, pp. 393–400. Como: Linköping University Electronic Press.
34. Brück D, Elmqvist H, Mattsson SE, et al. Dymola for multi-engineering modeling and simulation. In: *Proceedings of the Modelica*, Oberpfaffenhofen, 18–19 March 2002.
35. Braun R, Nordin P, Ericson L, et al. Hopsan: an open-source tool for rapid modelling and simulation of fluid and mechatronic systems. In: *Proceedings of the BATH/ASME 2020 symposium on fluid power and motion control*, 9–11 September 2020. New York: American Society of Mechanical Engineers (ASME).
36. Braun R and Krus P. Tool-independent distributed simulations using transmission line elements and the Functional Mock-up Interface. In: *Proceedings of the SIMS 54th conference*, Bergen, 16–18 October 2013, vol. 24, pp. 149–154. Vienna: Arbeitgemeinschaft Simulation News (ARGESIM).
37. Johns PB and O'Brien MA. Use of the transmission-line modelling (TLM) method to solve non-linear lumped networks. *J I Electron Rad Eng* 1980; 50: 59–70.

Author biographies

Robert Braun is a Senior Lecturer at the Division of Fluid and Mechatronic Systems at Linköping University, Sweden. His research targets co-simulation and real-time simulation with a strong focus on standardization and numerical stability.

Dag Fritzson is a Retired Senior Researcher from SKF Research Center in Gothenburg and a visiting Professor at Linköping University, Sweden. His research includes parallel simulation, co-simulation, and modeling and simulation of contact mechanics.

Appendix 1

Notation

a	wave propagation speed
C	capacitance
c	wave variable
C_1, C_2	constants
e_1, e_2	effort variable
f, f_1, f_2	flow variable
J	moment of inertia
K	mechanical stiffness
L	inductance
M	mechanical inertia
q_1, q_2	displacement variable
t	simulation time
T	TLM time delay
T_{com}	communication step size
Z_c	characteristic impedance