Co-simulation: State of the art

Cláudio Gomes (claudio.gomes@uantwerp.be)
Casper Thule (casper.thule@eng.au.dk) David Broman (dbro@kth.se)
Peter Gorm Larsen (pgl@eng.au.dk)
Hans Vangheluwe (hans.vangheluwe@uantwerp.be)

February 1, 2017

Contents

Abstract

It is essential to find new ways of enabling experts in different disciplines to collaborate more efficient in the development of ever more complex systems, under increasing market pressures. One possible solution for this challenge is to use a heterogeneous model-based approach where different teams can produce their conventional models and carry out their usual mono-disciplinary analysis, but in addition, the different models can be coupled for simulation (co-simulation), allowing the study of the global behavior of the system. Due to its potential, co-simulation is being studied in many different disciplines but with limited sharing of findings. Our aim with this work is to summarize, bridge, and enhance future research in this multidisciplinary area.

We provide an overview of co-simulation approaches, research challenges, and research opportunities, together with a detailed taxonomy with different aspects of the state of the art of co-simulation and classification for the past five years.

The main research needs identified are: finding generic approaches for modular, stable and accurate coupling of simulation units; and expressing the adaptations required to ensure that the coupling is correct.

1 Introduction

1.1 Motivation

Truly complex engineered systems that integrate physical, software and network aspects, are emerging [153, 185]. Due to external pressure, the development of these systems has to be concurrent and distributed, that is, divided between different teams and/or external suppliers, each in their own domain and each with their own tools. Each participant develops a partial solution to a constituent system, that needs to be integrated with all the other partial solutions. The later in the process the integration is done, the less optimal it is [228].

Innovative and truly optimal multi-disciplinary solutions can only be achieved through a holistic development process [237] where the partial solutions developed independently are integrated sooner and more frequently. Furthermore, the traditional activities carried out at the partial solution level —such as requirements compliance check, or design space exploration— can be repeated at the global level, and salient properties spanning multiple constituent systems can be studied.

Modeling and simulation can improve the development of the partial solutions (e.g., see Friedman and Ghidella [92]), but falls short in fostering this holistic development process [34]. To understand why, one has to observe that: *a*) models of each partial solution cannot be exchanged or integrated easily, because these are likely developed by different specialized tools; and *b*) externally supplied models have Intellectual Property (IP), hence cannot be cheaply disclosed to system integrators.

1.2 Co-simulation

Co-simulation is proposed as a solution to overcome these important challenges. It consists of the theory and techniques to enable global simulation of a coupled system via the composition of simulators. Each simulator is a *black box* mock-up of a constituent system, developed and provided by the team that is responsible for that system. This allows each team/supplier to work on its part of the problem with its own tools without having the coupled system in mind, and eases the relationship between system integrators and suppliers: the latter provide virtual mock-ups of the solutions they are selling without revealing their IP; the former can perform early conformance checks and evaluate different designs from multiple competing suppliers.

An alternative to co-simulation is co-modelling, were models are described in a unified language, and then simulated. There are advantages to this approach but each domain has its own particularities when it comes to simulation (e.g., see [63, 167, 246]) making it impractical to find a language and simulation algorithm that fits all.

As part of the systematic review that led to the current document (see section 6.1 for details), we took note of the approaches to co-simulation and the publications in applications of co-simulation. The approaches to co-simulation shaped the taxonomy in section 6.2 and the applications of co-simulation shows that in the last five years, co-simulation has been applied in many different engineering domains, as fig. 1 shows. In concrete, the publications are:

Automotive - [4, 26, 28, 29, 39, 44, 69, 74, 82, 132, 156, 168, 213, 237, 258, 264, 266]

Electricity Production and Distribution - [3, 5, 33, 78, 95, 96, 111, 120, 139, 157–159, 163, 205, 224, 243, 248, 255, 267]

HVAC - [73, 88, 114, 188, 192, 251]

IC and SoC Design - [209]

Maritime - [194, 195]

Robotics - [143, 198, 199, 265]

A closer look at the publications shows, however, that the average reported co-simulation scenario includes only two simulators, each a mock-up of a constituent system from a different domain. While this gives evidence that co-simulation enhances the development multi-domain systems, it is not yet up-to-par with the scale of Cyber-Physical Systems (CPSs). The unexplored potential is recognized in a number of completed and ongoing projects that address co-simulation (MOD-

ELISAR[1], DESTECS[2], INTO-CPS[3], ACOSAR[4], ACoRTA[5]), and is one of the reasons why the Functional Mock-up Interface (FMI) Standard was created.

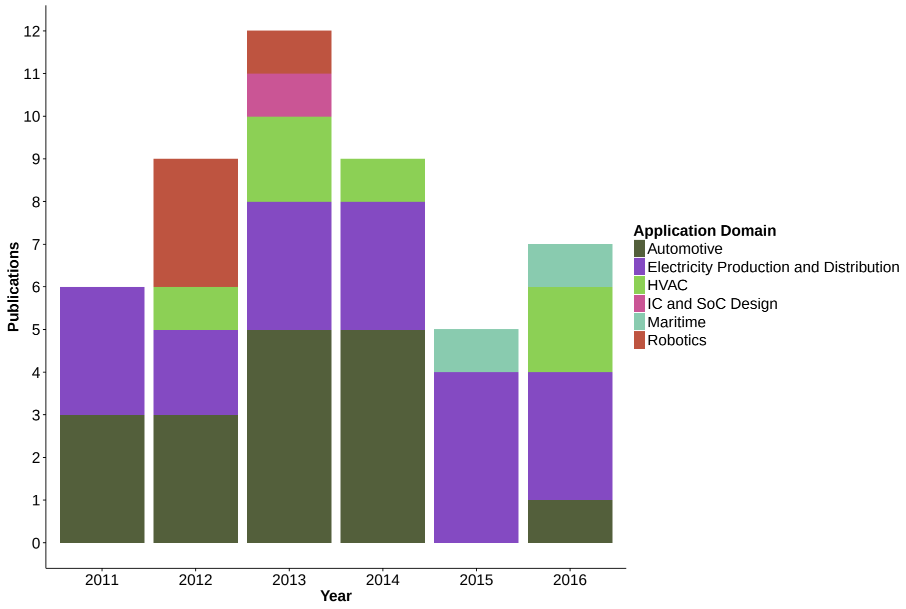


Figure 1: Research publications of co-simulation applications over the past five years.

The FMI standard[6] defines the interfaces that enable modelling and simulation tools to cooperate in a simulation, while keeping the IP in the models protected. As of December 2016, 15 tools implement the standard[7] and many others are planning to support it.

The standard was originally created for the coupling of continuous systems, but has been used (or abused) to simulate hybrid systems. Extensions have been proposed in the literature to facilitate and standardize this simulation (e.g., zero step size transitions [36, 232], or step size prediction [46]), and we complement those by adding to the understanding of the the challenges arising in hybrid system co-simulation. Co-simulation is not a new concept and there are many approaches in the state of the art, in particular, discrete event based approaches, that have been studied and may shed light on how hybrid system co-simulation can be performed.

Contribution. We present a survey and a taxonomy, as an attempt to bridge, relate and classify the many co-simulation approaches in the state of the art.

[1]`https://itea3.org/project/modelisar.html`
[2]`http://www.destecs.org/`
[3]`http://into-cps.au.dk/`
[4]`https://itea3.org/project/acosar.html`
[5]`http://www.v2c2.at/research/ee-software/projects/acorta/`
[6]`https://www.fmi-standard.org`
[7]This number includes only the tools that have passed the crosscheck in `https://www.fmi-standard.org/tools`

1.3 Need for the Survey

Despite the growing interest in the benefits and scientific challenges of co-simulation, to the best of our knowledge, no existing survey attempts to cover the heterogeneous communities in which it is being studied. The lack of such a survey means that the same techniques are being proposed independently with limited sharing of findings. To give an example, the use of dead-reckoning models is a well known technique in discrete event co-simulation [152], but only very recently it was used in a continuous time co-simulation approach [221]. **Our hypothesis** is that bridging these different co-simulation approaches means that solutions and techniques can be exchanged, and a deeper understanding of hybrid system co-simulation can be attained.

Scientifically, co-simulation —multi-disciplinary by its very nature— mixes the following fields of research:

1. Numerical analysis – Accuracy and stability of the coupled system have to be studied [10, 51, 66, 105, 107, 109, 128, 197].

2. Differential Algebraic System Simulation – The composition of co-simulation units is, in the most general sense, made through algebraic constraints [214, 219, 229].

3. Hybrid Systems – co-simulation scenarios, in the most general sense, are hybrid systems [47, 154, 172–174, 184, 251, 263]

4. Optimization – the heterogeneous capabilities of co-simulation units pose interesting tradeoffs [46, 236].

5. Hierarchy – Systems of systems are hierarchical and the corresponding co-simulation scenarios should be hierarchical as well [96]. Compositionality properties of co-simulations becomes an interesting research challenge.

6. Formal Verification – The co-simulation orchestration algorithm, also known as the master, can be certified to be correct under certain assumptions about the co-simulation scenarios [46, 89, 99, 100, 179].

7. System Testing – Co-simulation can be used for exhaustively testing a set of black-box constituent systems, with a non-deterministic environment [150, 164].

8. Dynamic Structure Systems – Subsystems can have different dependencies depending on whom, and which level of abstraction, they interact with [21–23, 234].

9. Multi-paradigm Modeling – Subsystems can have have different models at different levels of abstraction [246]. The relationships between the multiple levels have to be known so that correct dynamic switching between levels abstraction can be made.

1.4 Outline

To help structure the characteristics of the simulators and how they interact, we distinguish two main approaches for co-simulation: Discrete Event (DE), described in section 3, and Continuous Time (CT), described in section 4. Both of these can be, and are, used for the co-simulation of continuous, discrete, or hybrid coupled systems. We call Hybrid co-simulation, described in section 5, a co-simulation approach that mixes the DE and CT approaches [8]. section 6 categorizes the features provided by co-simulation frameworks, and classifies the state of the art with that taxonomy. Finally, section 7 concludes this publication. The section below provides the terminology used in the rest of the survey.

2 Modeling, Simulation, and Co-simulation

2.1 Dynamical Systems – Models of Real Systems

In this section we present, in an informal manner, the concepts that will be used throughout the document.

A *dynamical system* is a model of a real system (for instance a physical system or a computer system) characterized by a state and a notion of evolution rules. The state is a set of point values in a state space. The evolution rules describe how the state evolves over an independent variable, usually time. For example, a traffic light, the real system, can be modeled by a dynamical system that, at any time, can be in one of four possible states (`red`, `yellow`, `green`, or `off`). The evolution rules may dictate that it changes from `red` to `green` after some time (e.g., 60 seconds). Another example is a mass-spring-damper, modeled by a set of first order Ordinary Differential Equations (ODEs). The equations describe how the state —position and velocity of the mass— changes continuously over the simulated time. In contrast with the traffic light system, where the state cannot take an infinite number of different values over a finite duration of simulated time, the state of the mass-spring-damper can.

The *behavior trace* is the set of trajectories followed by the state (and outputs) of a dynamical system. For example, a state trajectory x can be defined as a mapping between a time base T and the set of reals $\mathbb{R}$, that is, $x : T \to \mathbb{R}$. fig. 2 shows a possible behavior trace for each of the example systems described before. In this example, the time base is $\mathbb{R}$.

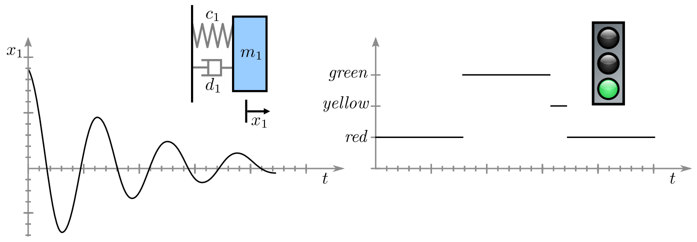


Figure 2: Examples of behavior traces.

We refer to the time variable $t \in T$ by *simulated time* —or simply *time*, when no ambiguity exists— defined over a time base T (typical the real numbers $\mathbb{R}$), as opposed to the *wall-clock time* $\tau \in WcT$, which is the time that passes in the real world [94]. When computing the behavior trace of a dynamical system over an interval $[0, t]$ of simulated time, a computer takes τ units of

[8]Note that in this survey, we are focusing on timed formalisms (also called models of computation) and how they interact in a hybrid co-simulation environment. Other formalisms, with no or only logical notion of time (such as dataflow and synchronous reactive), are not discussed in this survey. For an overview of formalisms and models of computation, please see the book on Ptolemy II [200] and the following survey [45].

wall-clock time that depend on t. τ can therefore be used to measure the run-time performance of simulators. fig. 3a highlights different kinds of simulation, based on the relationship between τ and t. In *real-time simulation*, the relationship between t and τ is $t = \alpha\tau$, for a given $\alpha > 0$. In most cases $\alpha = 1$ is required, but making sure this is obeyed by the simulation algorithm is one of the main challenges in real-time simulation, and by extension, of co-simulation. In as-fast-as-possible —or *analytical*— simulation, the relationship between τ and t is not restricted. Simulation tools that offer interactive visualization allow the user to pause the simulation and/or set the relationship between τ and t.

Knowing when a dynamical system can be used to predict the behavior of a real system is crucial. The *experimental frame* describes, in an abstract way, a set of assumptions in which the behavior trace of the dynamical system can be compared with the one of the real system [24, 230, 245, 246, 262]. By real system we mean either an existing physical system, or a fictitious —to be engineered— one. *Validity* is then the difference between the behavior trace of the dynamical system and the behavior trace of the real system, measured under the assumptions specified by the experimental frame. This is what conveys predictive power to dynamical systems. For example, Hooke's law, in the mass-spring-damper system, can only be used to predict the reaction force of the spring for small deformations. For the traffic light dynamical system, the experimental frame includes the assumption that the transition from the red light to the green light is instantaneous. It is a valid assumption, provided that the executing platform in which the controller software runs, has enough computing power [76, 181, 247]. A model is invalid when, within the experimental frame assumptions, its behavior trace is so different than the one of the real system, that it cannot be used to predict properties of the real system.

In order to be practical, the behavior trace of a dynamical system has to highlight just the features of interest of the real system that are relevant for the tasks at hand [145]. In the traffic light model, the precise amount of wall-clock time a transition from red to green takes is unknown, but deemed small enough to be neglected. In the mass-spring-damper, Hooke's law was chosen because the maximum displacement of the mass will not be large when the context in which the system will be used is taken into account.

Finally, we consider only those dynamical systems for which it is possible to obtain its meaning, i.e. the behavior trace, even if only an approximation.

2.2 Simulators – Computing the Behavior Trace

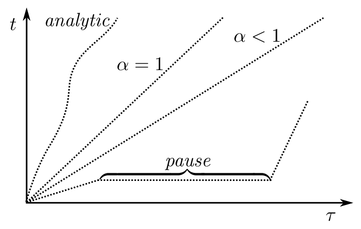
There are two generally accepted ways of obtaining the behavior trace of a dynamical system:

Translational Translate the dynamical system into another model, which can be readily used to obtain the behavior trace. Obtaining the analytical solution of the mass-spring-damper equations is an example of this approach. For instance, if the traffic light model is expressed in the Statechart formalism, it can be translated into a DEVS model, as done in Borland [40], which can be used to obtain the behavior trace.

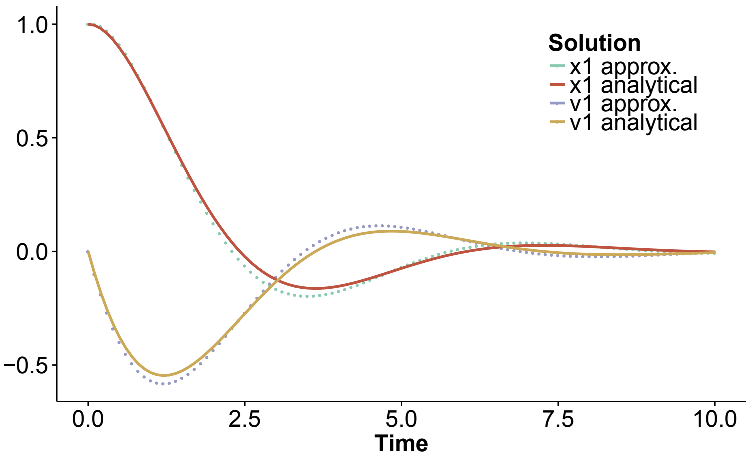
Operational Use of a solver – an algorithm that takes the dynamical system as input, and outputs a behavior trace. For the mass-spring-damper example, a numerical solver can be used to obtain an approximation of the behavior trace.

We focus on the latter.

A *simulator* is an algorithm that takes a dynamical system and computes its behavior trace. Running in a digital computer, it is often the case that a simulator will only be able to approximate that trace. Two aspects contribute to the error in these approximations: inability to calculate a



(a) Classification of time constraints in simulation. Based on [177, 239].



(b) Approximate behavior trace of the mass-spring-damper system computed by a Forward Euler solver. Parameters are: $m = 1$, $p = 1$, $s = 0$, $c_1 = 1$, $d_1 = 1$, and $F_e(t) = 0$. x is the displacement and v is the velocity. The dotted lines are the approximated behaviour traces and the solid lines are the analytical behaviour traces.

Figure 3

trajectory over a continuum, and the finite representation of infinitely small quantities. Simulators of discrete dynamical systems may also tolerate some inaccuracies in the behavior traces as well (e.g., if that brings a performance benefit). fig. 3b shows an example approximation of the behavior trace of the mass-spring-damper system, computed by the Forward Euler. The inaccuracies are clear when compared to the analytical trace.

In order to define what an *accurate simulator* is, or even be able to talk about error, we need to postulate that every dynamical system has an analytical behavior trace. The error can then be defined as the absolute difference between the behavior trace computed by a simulator and the analytical trace. An accurate simulator is one that produces traces that are very close[9] to the analytical behaviour. Even if it is not possible to obtain the analytical behavior of every dynamical system, there are theoretical results that allow simulators to control the error they make. These techniques are applied to co-simulation in section 4.3.4. For the mass-spring-damper, and linear ODEs in general, the analytical trace follows a known structure [48]. For the traffic light, and timed statemachine models in general, the analytical behavior trace can be obtained with a sequential solver, that respects the causality of events. In short, validity is a property of a dynamical system whereas accuracy is a property of a simulator [63]. It is perfectly possible to have an accurate behaviour trace of a model that is invalid, and vice versa [62]. For continuous time systems, the choice of an appropriate solver is important and should be made by domain experts [63, 167, 175].

[9]The meaning of "very close" depends on the numerical tolerance of the observer.

2.3 Simulation Units - Mock-ups of Reality

In strict terms, a simulator is not readily executable: it needs a dynamical system and the input trajectories to that dynamical system, before being able to compute the behavior trace. We use the term *simulation unit* for the composition of a simulator with a dynamical system. A simulation unit is a replacement of the real system, ready to take inputs and produce a behavior trace. In the FMI standard, the analogous term is the Functional Mock-up Unit (FMU) for co-simulation.

A *simulation* is the behavior trace obtained with a simulation unit. The correctness of a simulation unit is dictated by the correctness of the simulation, which depends on the accuracy of the simulator and the validity of the dynamical system.

2.4 Compositional Co-simulation

As described in section 1, it useful to obtain correct simulations of complex, not yet existing, systems. Since the constituent systems are developed independently by specialized teams/suppliers, simulation units can be produced for these. The simulation units can be coupled via their inputs-/outputs to produce a behavior trace of the coupled system. A *co-simulation*, a special kind of simulation, is the set of simulations computed by the coupled simulation units.

The simulation units can be independent black boxes, possibly running in different computers. Hence, an *orchestrator* is necessary to couple them. The orchestrator controls how the simulated time progresses in each simulation unit and moves data from outputs to inputs according to a co-simulation scenario. A *co-simulation scenario* is the information necessary to ensure that a correct co-simulation can be obtained. It includes how the inputs of each simulation unit are computed from other outputs, their experimental frames, etc. In the FMI Standard, the orchestrator is called a master algorithm.

Analogously to the simulator and simulation unit concepts, the composition of a specific orchestrator with a co-simulation scenario, yields a *co-simulation unit*, which is a special kind of simulation unit, and a substitute of the real coupled system. It follows that a co-simulation is the simulation trace computed by a co-simulation unit. This characterization enables hierarchical co-simulation scenarios, where co-simulation units are coupled.

The *main purpose of the orchestrator* is to obtain a correct behavior trace of the coupled system, assuming that each simulation unit in itself is correctly defined. We make this assumption because: (1) the simulation units are often well known parts of the system, and (2) they are developed by experienced and specialized teams/suppliers.

Co-simulation enables design decisions to be tried out in the model (*what-if* analysis), cheaply [10], early in the process, and possibly automatically [87, 104].

In this survey, we focus on the coupling of black box simulation units, where limited knowledge of the models and simulators is available. However, as will become clear in the sections below, the black box restriction has to be relaxed so that certain properties related to correctness can be ensured. Understanding what kind of information should be revealed and how IP can still be protected is an active area of research in co-simulation.

Most challenges in co-simulation are related to compositionality: if every simulation unit S_i in a co-simulation scenario satisfies property P, then the co-simulation unit, with a suitable orchestrator, must also satisfy P. The correctness is a property that should be compositional in co-simulation. Other properties include validity, or accuracy. It is an open research question to ensure that a

[10] Another aspect to consider is the balance between insights gained and resources spent [85].

co-simulator is compositional for a given set of properties. The following three sections provide an overview of the information and techniques being used throughout the state of the art, divided into three main approaches: discrete event (section 3), continuous time (section 4), and hybrid (section 5) co-simulation.

3 Discrete Event Based Co-simulation

The Discrete Event (DE) based co-simulation approach describes a family of orchestrators and characteristics of simulation units that are borrowed from the discrete event system simulation domain. We start with a description of DE systems, and then we extract the main concepts that characterize DE based co-simulation.

The traffic light is a good example of a DE system. It can be one of the possible modes: `red`, `yellow`, `green` or `off`. The off mode is often used by the police, which in some countries is characterized by a blinking yellow. Initially, the traffic light can be red. Then, after 60 seconds, it changes to green. Alternatively, before those 60 seconds pass, some external entity (e.g., a police officer) may trigger a change from red to off. The output of this system can be an event signaling its change to a new color. This example captures some of the essential characteristics of a DE dynamical system: *reactivity* – instant reaction to external stimuli (turning off by an external entity); and *transiency* – a DE system can change its state multiple times in the same simulated time point, and receive simultaneous stimuli. In the traffic light, *transiency* would happen if the light changes always after 0s (instead of 60s), or if the police officer would turn off and on the traffic light in the same instant.

These characteristics are embraced in DE based co-simulation, where the orchestrator acknowledges that simulation units can evolve the internal state and exchange values despite the fact that the simulated time is stopped.

3.1 DE Simulation units

A DE simulation unit is a black box that exhibits the characteristics of a DE dynamical system, but the dynamical system it stands for does not need to be a DE one. Furthermore, it is typical to assume that DE simulation units communicate with the environment via time-stamped *events*, as opposed to signals. This means that the outputs of simulation units can be absent at times where no event is produced.

We adapt the definition of DEVS[11] in [242] (originally proposed in [259]) to formally define a

[11]In the original DEVS definition, the initial state and the absent value in the output function are left implicit. Here we make them explicit, to be consistent with section 4. Note also that there are many other variants of discrete-event formalisms. For instance, DE in hardware description languages (VHDL and Verilog) and actor based systems (for instance the DE director in Ptolemy II [200]).

DE simulation unit S_i, where i denotes the reference of the unit:

$$\begin{aligned}
& S_i = \left\langle X_i, U_i, Y_i, \delta_i^{ext}, \delta_i^{int}, \lambda_i, ta_i, q_i(0) \right\rangle \\
& \delta_i^{ext} : Q_i \times U_i \to X_i \\
& \delta_i^{int} : X_i \to X_i \\
& \lambda_i : X_i \to Y_i \cup \{\phi\} \\
& ta_i : X_i \to \mathbb{R}_0^+ \cup \infty \\
& q_i(0) \in Q_i \\
& Q_i = \{(x, e) | x \in X_i \text{ and } 0 \leq e \leq ta_i(x)\}
\end{aligned} \tag{1}$$

where:

- X_i, U_i, and Y_i are the set of possible discrete states, input events, and output events, respectively;
- $\delta_i^{ext}(q_i, u_i) = x_i'$ is the external transition function that computes a new total state $(x_i', 0) \in Q_i$ based on the current total state q_i and an input event u_i;
- $\delta_i^{int}(x_i) = x_i'$ is the internal transition function that computes a new total state $(x_i', 0) \in Q_i$ when the current total state is $(x_i, ta_i(x_i)) \in Q_i$;
- e denotes the elapsed units of time since the last transition (internal or external);
- $\lambda_i(x_i) = y_i \in Y_i \cup \{\phi\}$ is the output event function, invoked right before an internal transition takes place and ϕ encodes an absent value;
- $ta_i(x_i) \in \mathbb{R}$ is the time advance function that indicates how much time passes until the next state change occurs, assuming that no external events arrive;
- $q_i(0)$ is the initial state;

The execution of a DEVS simulation unit is described informally as follows. Suppose that the simulation unit is at time $t_i \in \mathbb{R}_0^+$ and marks the current discrete state as x_i for $e \geq 0$ elapsed units of time. Since $e \leq ta_i(x_i)$, the total state is $(x_i, e) \in Q_i$. Let $tn = t_i + ta_i(x_i) - e$. If no input event happens until tn, then at time tn an output event is computed as $y_i := \lambda_i(x_i)$ and the new discrete state x_i is computed as $x_i := (\delta_i^{int}(x_i), 0)$. If, on the other hand, there is an event at time $ts < tn$, that is, u_i is not absent at that time, then the solver changes to state $x_i := (\delta_i^{ext}((x_i, e + ts - t_i), u_i), 0)$ instead.

In the above description, if two events happen at the same time, both are processed before the simulated time progresses. Due to the transiency and reactivity properties, the state and output trajectories of a DE simulation unit can only be well identified if the time base, traditionally the positive real numbers, includes a way to order simultaneous events, and simultaneous state changes. An example of such a time base is the notion of superdense time [154, 162, 165], where each time point is a pair $(t, n) \in \mathcal{T} \times \mathcal{N}$, with $\mathcal{T}$ typically being the positive real numbers and $\mathcal{N}$, called the index, is the natural numbers. In this time base, a state trajectory is a function $x_i : \mathcal{T} \times \mathcal{N} \to V_{x_i}$, where V_{x_i} is the set of values for the state, and an output/input trajectory is $u_i : \mathcal{T} \times \mathcal{N} \to V_{u_i} \cup \{\phi\}$. Simultaneous states and events can be formally represented with incrementing indexes. See [47] for an introduction.

Equations 2 and 3 show examples of simulation units.

A DE simulation unit is passive: it expects some external coordinator to set the inputs and call the transition functions. This passivity enables an easier composition of simulation units in a co-simulation, by means of a coordination algorithm, as will be shown later in section 3.2. Algorithm 1 shows a trivial orchestrator which computes the behavior trace of a single DE simulation unit, as specified in eq. (1), that has no inputs. Remarks:

- tl holds the time of the last transition;
- the initial elapsed time satisfies $0 \leq e \leq ta_i(x_i(0))$;

If Algorithm 1 is used to coordinate the execution of the traffic light simulation unit in eq. (2), then the resulting behavior trace is the piecewise constant traffic light state $x_1(t)$, together with the output events. The latter is represented as a trajectory $y_i(t)$ that is mostly undefined (or absent), except for the single points where an output is produced, according to ta_1.

Algorithm 1: Single autonomous DE simulation unit orchestration. Based on [242] and originally proposed in [259].

```
Data: A simulation unit $S_i = \left\langle X_i, \emptyset, Y_i, \delta_i^{ext}, \delta_i^{int}, \lambda_i, ta_i, (x_i(0), e_i) \right\rangle$.
$t_i := 0$ ;
$x_i := x_i(0)$ ;                                  // Initial discrete state
$tl := -e_i$ ;                                     // Account for initial elapsed time
while $true$ do
    $t_i := tl + ta_i(x_i)$ ;                      // Compute time of the next transition
    $y_i := \lambda_i(x_i)$ ;                      // Output
    $x_i := \delta_i^{int}(x_i)$ ;                 // Take internal transition
    $tl := t_i$ ;
end
```

3.2 DE Co-simulation Orchestration

DEVS simulation units communicate with their environment exclusively through inputs and outputs. DE co-simulation scenarios are comprised of multiple DE units S_i (eq. (1)) coupled through output to input connections, which map output events of one unit to external events in other unit.

Consider the following DE simulation units of a traffic light and a police office, respectively:

$$
\begin{aligned}
&S_1 = \left\langle X_1, U_1, Y_1, \delta_1^{ext}, \delta_1^{int}, \lambda_1, ta_1, q_1(0) \right\rangle \\
&X_1 = \{red, yellow, green, off\} \\
&U_1 = \{toAuto, toOff\} \\
&Y_1 = X_1 \\
&\delta_1^{ext}((x_1, e), u_1) = \begin{cases} off & \text{if } u_1 = toOff \\ red & \text{if } u_1 = toAuto \text{ and } x_1 = off \end{cases} \\
&\delta_1^{int}(x_1) = \begin{cases} green & \text{if } x_1 = red \\ yellow & \text{if } x_1 = green \\ red & \text{if } x_1 = yellow \end{cases} \\
&\lambda_1(x_1) = \begin{cases} green & \text{if } x_1 = red \\ yellow & \text{if } x_1 = green \\ red & \text{if } x_1 = yellow \end{cases} \\
&ta_1(x_1) = \begin{cases} 60 & \text{if } x_1 = red \\ 50 & \text{if } x_1 = green \\ 10 & \text{if } x_1 = yellow \\ \infty & \text{if } x_1 = off \end{cases} \\
&q_1(0) = (red, 0)
\end{aligned}
\tag{2}
$$

$$
\begin{aligned}
&S_2 = \left\langle X_2, U_2, Y_2, \delta_2^{ext}, \delta_2^{int}, \lambda_2, ta_2, q_2(0) \right\rangle \\
&X_2 = \{working, idle\} \\
&U_2 = \emptyset \\
&Y_2 = \{toWork, toIdle\} \\
&\delta_2^{int}(x_2) = \begin{cases} idle & \text{if } x_2 = working \\ working & \text{if } x_2 = idle \end{cases} \\
&\lambda_2(x_2) = \begin{cases} toIdle & \text{if } x_2 = working \\ toWork & \text{if } x_2 = idle \end{cases} \\
&ta_2(x_2) = \begin{cases} 200 & \text{if } x_2 = working \\ 100 & \text{if } x_2 = idle \end{cases} \\
&q_2(0) = (idle, 0)
\end{aligned}
\tag{3}
$$

With the following remarks:

- The current state of the model in the definition of δ_1^{ext} is $q_1 = (x_1, e)$ with e being the elapsed time since the last transition.
- The output event function λ_1 is executed *immediately before* the internal transition takes place. It must then publish the next state instead of the current.

To model a scenario where the police officer interacts with a traffic light, the output events Y_2 have to be mapped into the external events U_1 of the traffic light simulation unit (eq. (2)). In this example, if $U_1 = \{toAuto, toOff\}$ are the external input events handled by the traffic light simulation unit, the mapping $Z_{2,1} : Y_2 \to U_1$ is defined by:

$$
Z_{2,1}(y_2) = \begin{cases} toAuto & \text{if } y_2 = toIdle \\ toOff & \text{if } y_2 = toWork \end{cases} \tag{4}
$$

This way, if the police officer changes to *working* state at time tn, then the output signal $y_2 := toWork$ will be translated by $Z_{2,1}$ into an input event $u_1 := toOff$ of the traffic light simulation unit.

Based on the idea of abstract simulation units [262], we formalize and illustrate the idea of a DE co-simulation scenario with reference cs as follows:

$$
\langle U_{cs}, Y_{cs}, D, \{S_d : d \in D\}, \{I_d : d \in D \cup \{cs\}\}, \{Z_{i,d} : d \in D \wedge i \in I_d\}, \text{Select}\rangle \tag{5}
$$

where:

- U_{cs} is the set of possible input events, external to the scenario;
- Y_{cs} is the set of possible output events from the scenario to the environment;
- D is an ordered set of simulation unit references;
- For each $d \in D$, S_d denotes a DE unit, as defined in eq. (1);
- For each $d \in D \cup \{cs\}$, $I_d \subseteq (D \setminus \{d\}) \cup \{cs\}$ is the set of simulation units that can influence simulation unit S_d, possibly including the environment external to the scenario (cs), but excluding itself (d);
- For each $i \in I_d$, $Z_{i,d}$ specifies the mapping of events: $$\begin{aligned} Z_{i,d} &: U_i \to U_d, \text{ if } i = cs \\ Z_{i,d} &: Y_i \to Y_d, \text{ if } d = cs \\ Z_{i,d} &: Y_i \to U_d, \text{ if } i \neq cs \text{ and } d \neq cs \end{aligned}$$
- Select : $2^D \to D$ is used to deterministically select one simulation unit, among multiple simulation units ready to produce output events simultaneously, i.e., when at time t, the set of simulation units $$IMM(t) = \{d | d \in D \wedge q_d(t) = (x_d, ta_d(x_d))\} \tag{6}$$ has more than one simulation unit reference. This function is restricted to select one from among the set $IMM(t)$, i.e., Select$(IMM(t)) \in IMM(t)$.

The following co-simulation scenario cs couples the traffic light simulation unit to the police officer simulation unit:

$$\begin{aligned} &\langle \emptyset, Y_{cs}, \{1,2\}, \{S_1, S_2\}, \{I_1, I_2, I_{cs}\}, \{Z_{2,1}, Z_{1,cs}\}, \text{Select} \rangle \\ &Y_{cs} = Y_1 \\ &I_1 = \{2\} \\ &I_2 = \emptyset \\ &I_{cs} = \{1\} \\ &Z_{1,cs}(y_1) = y_1 \end{aligned} \tag{7}$$

where:
- S_1 is the traffic light simulation unit (eq. (2)) and S_2 the police officer simulation unit (eq. (3));
- Y_1 is the output of S_1;
- $Z_{2,1}$ is defined in eq. (4); and
- The omitted $Z_{i,d}$ functions map anything to absent (ϕ).

The Select function is particularly important to ensure that the co-simulation trace is unique. For example, consider the co-simulation scenario of eq. (7), and suppose that at time tn both simulation units are ready to output an event and perform an internal transition. Should the traffic light output the event and perform the internal transition first, or should it be the police office to do it first? In general, the order in which these output/transition actions are performed matters. The reason is that the way one simulation unit reacts to the other simulation unit's output may be different, depending on the internal state of the former. In the example co-simulation scenario, the end result is always the same but this is not the general case.

Algorithm 2 illustrates the orchestrator of an autonomous (without inputs) DE co-simulation scenario. It assumes that the co-simulation scenario does not expect external events, that is, all events that can affect the simulation units are produced by other simulation units in the same scenario. External output events are possible though. Remarks:
- t_{cs} holds the most recent time of the last transition in the scenario;
- e_d is the elapsed time of the current state $q_d = (x_d, e_d)$ of simulation unit S_d;

- tn is the time of the next transition in the scenario;
- i^* denotes the chosen imminent simulation unit;
- I_{cs} is the set of simulation units that can produce output events to the environment;
- y_{cs} is the output event signal of the scenario to the environment; and
- $\{d | d \in D \wedge i^* \in I_d\}$ holds the simulation units that S_{i^*} can influence.

Algorithm 2: Autonomous DE co-simulation scenario orchestration. Based on [242].

```
Data: A co-simulation scenario $cs = \langle \emptyset, Y_{cs}, D, \{S_d\}, \{I_d\}, \{Z_{i,d}\}, \text{Select} \rangle$.
$t_{cs} := 0$ ;
$x_i := x_i(0)$ for all $i \in D$ ;                  // Store initial discrete state for each unit
while $true$ do
    $ta_{cs} := \min_{d \in D} \{ta_d(x_d) - e_d\}$ ;       // Time until the next internal transition
    $tn := t_{cs} + ta_{cs}$ ;                              // Time of the next internal transition
    $i^* := \text{Select}(IMM(tn))$ ;                               // Get next unit to execute
    $y_{i^*} := \lambda_{i^*}(x_{i^*})$ ;
    $x_{i^*} := \delta^{int}_{i^*}(x_{i^*})$ ;                              // Store new discrete state
    $e_{i^*} := 0$ ;                              // Reset elapsed time for the executed unit
    if $i^* \in I_{cs}$ then
        $y_{cs} := Z_{i^*,cs}(y_{i^*})$ ;                          // Compute output of the scenario
    end
    for $d \in \{d | d \in D \wedge i^* \in I_d\}$ do
        $u_d := Z_{i^*,d}(y_{i^*})$ ;  // Trigger internal units that are influenced by unit $i^*$
        $x_d := \delta^{ext}_d((x_d, e_d + ta_{cs}), u_d)$ ;
        $e_d := 0$ ;
    end
    for $d \in \{d | d \in D \wedge i^* \notin I_d\}$ do
        $e_d := e_d + ta_{cs}$ ;            // Update the elapsed time of the remaining units
    end
    $t_{cs} := tn$ ;                                                        // Advance time
end
```

fig. 4 shows the behavior trace of the co-simulation scenario in eq. (7).

Algorithm 2 is similar to Algorithm 1:

- The time advance of the scenario ta_{cs} corresponds to the time advance of a single simulation unit.
- The output produced by the state transition is analogous to the λ function of a single simulation unit.
- The output and state transition of child S_{i^*}, together with the external transitions of the simulation units influenced by S_{i^*}, are analogous to the internal transition of a single simulation unit.

It is natural then that a co-simulation scenario cs as specified in eq. (5), can be made to behave as a single DE simulation unit S_{cs}. Intuitively, the state of S_{cs} is the set product of the total states of each child DE unit; ta_{cs} is the minimum time until one of the DE units executes an internal transition; the internal transition of S_{cs} gets the output event of the imminent unit, executes the external transitions of all the affected units, updates the elapsed time of all unaffected units, and computes the next state of the imminent unit; the external transition of S_{cs} gets an event from the environment, executes the external transition of all the affected units, and updates the elapsed time of all the unaffected units [262]. Formally:

$$
\begin{aligned}
& S_{cs} = \left\langle X_{cs}, U_{cs}, Y_{cs}, \delta_{cs}^{ext}, \delta_{cs}^{int}, \lambda_{cs}, ta_{cs}, q_{cs}(0) \right\rangle \\
& X_{cs} = \times_{d \in D} Q_d \\
& q_{cs}(0) = (\times_{d \in D} q_i(0), \min_{d \in D} e_d) \\
& ta_{cs}((\ldots,(x_d,e_d),\ldots)) = \min_{d \in D} \{ ta_d(x_d) - e_d \} \\
& i^* = \text{Select}(IMM(t)) \\
& \lambda_{cs}(x_{cs}) = \begin{cases} Z_{i^*,cs}(y_{i^*}(tn)) & \text{if } i^* \in I_{cs} \\ \phi & \text{otherwise} \end{cases} \\
& \delta_{cs}^{int}(x_{cs}) = (\ldots,(x'_d,e'_d),\ldots), \text{ for all } d \in D, \text{ where:} \\
& \quad x_{cs} = (\ldots,(x_d,e_d),\ldots) \\
& \quad (x'_d,e'_d) = \begin{cases} (\delta_d^{int}(x_d),0) \text{ if } i^* = d \\ (\delta_d^{ext}((x_d, e_d + ta_{cs}(x_{cs})), Z_{i^*,d}(\lambda_{i^*}(x_{i^*})),0) \text{ if } i^* \in I_d \\ (x_d, e_d + ta_{cs}(x_{cs})) \text{ otherwise} \end{cases} \\
& \delta_{cs}^{ext}((x_{cs},e_{cs}),u_{cs}) = (\ldots,(x'_d,e'_d),\ldots), \text{ for all } d \in D, \text{ where:} \\
& \quad x_{cs} = (\ldots,(x_d,e_d),\ldots) \\
& \quad (x'_d,e'_d) = \begin{cases} (\delta_d^{ext}((x_d, e_d + e_{cs}), Z_{cs,d}(u_{cs})),0) \text{ if } cs \in I_d \\ (x_d, e_d + e_{cs}) \text{ otherwise} \end{cases}
\end{aligned}
\tag{8}
$$

Remarks:
It is the Cartesian product of the total state of each child simulation unit that makes the discrete state of the co-simulation unit;
The elapsed times of each child simulation unit are managed solely by the co-simulation unit, whenever there is a transition (internal or external);
The external transition functions of each child are executed with the mapping of the events produced by the current state of the imminent child, and not the next one computed by $(\delta_d^{int}(x_d), 0)$;
An internal transition of a child simulation unit may cause an output event to the environment of the co-simulation unit, if the child is connected to the output of the co-simulation unit.

The same internal transition causes not only a change in the child discrete state, but also, due to its output event, may cause external transitions in other child simulation units. This is not a recursive nor iterative process: at most one external transition will occur in all the affected child simulation units; if any of the affected simulation units becomes ready for an internal transition, it waits for the next internal transition invoked from the coordinator of the co-simulation unit;

The resulting co-simulation unit S_{cs} behaves exactly as a DE unit specified in eq. (1). It can thus be executed with Algorithm 1 (in case of no inputs), or composed with other units in hierarchical co-simulation scenarios. Hierarchical co-simulation scenarios can elegantly correspond to real hierarchical systems, a natural way to deal with their complexity [138].

In summary, DE based co-simulation exhibits the following characteristics:

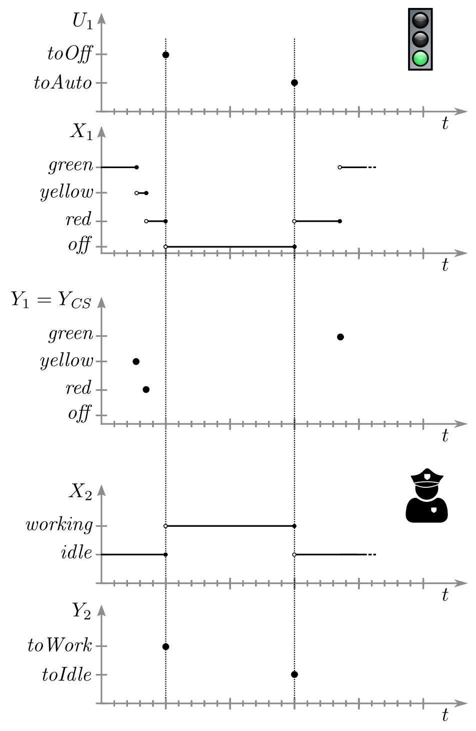


Figure 4: Example co-simulation trace of the traffic light and police officer scenario.

reactivity: A DE simulation unit (analogously, a DE co-simulation unit) has to process an event at the moment it occurs.

transiency: In both Algorithm 2 and in a DE co-simulation unit, the time advance ta_{cs} to the next imminent child internal transition can be zero for successive iterations, so an orchestrator has to be able to tolerate the fact that simulated time may not advance for several iterations.

predictable step sizes: In a DE co-simulation scenario without inputs, the orchestrator, as shown in Algorithm 2, can always predict the next simulated time step. In a scenario with inputs, if the environment provides the time of the next event, then the next simulated time step can be predicted too. For this to be possible, black box DE simulation units have to be able to inform the orchestrator what their time advance is, not a trivial task for DE units that simulate continuous systems whose future behavior trace, especially reacting to future inputs, is not easily predicted without actually computing it.

In the next sub-section, the main challenges in DE based co-simulation, and the requirements (or capabilities) their solutions impose in DE simulation units, are made explicit.

3.3 Challenges

3.3.1 Causality

For the sake of simplicity, Algorithm 2 is sequential. In a hierarchical co-simulation unit, the imminent simulation unit (closest to performing an internal transition) will be the one to execute, thus inducing that there is a global order in the events that are exchanged. This global order avoids causality violations but is too pessimistic. If an event $y_1(t_1)$ causes another event —by changing the internal state of some other simulation unit, which in turn changes its next output event— $y_2(t_2)$, then $t_1 \leq t_2$, which is ok. However, the converse is not true: $t_1 \leq t_2$ does not necessarily imply that $y_1(t_1)$ has caused $y_2(t_2)$, which means that simulation unit S_2 could execute before —in the wall-clock time sense— $y_1(t_1)$ without violating causality, at least within a small window of simulated time. To see why, suppose that S_1 and S_2 do not influence each other in the scenario. Then $y_2(t_2)$ would happen anyway, regardless of $y_1(t_1)$ occurring or not. Moreover, the co-simulation scenario holds information —the dependencies $\{I_d\}$— that can be used to determine who influences what [64, 147].

A parallel optimistic orchestrator that takes $\{I_d\}$ into account is, in general, faster in the wall clock time sense, than a pessimistic, sequential one. However, most of these, the Time-warp algorithm [123, 124] being a well known example, require rollback capabilities of simulation units. This is because simulation units proceed to advance their own time optimistically, assuming that any other simulation units will not affect them, until they are proven wrong by receiving an event which occurs before their own internal time. When that happens, the simulation unit has to rollback to a state prior to the time of timestamp of the event that just arrived. This may in turn cause a cascade of rollbacks in other affected simulation units. Moreover, in parallel optimistic DE co-simulation, any of the units in the scenario needs (theoretically) to support multiple rollbacks and have enough memory to do so for an arbitrary distant point in the past [94]. This point in the past is limited in Time-warp by the Global Virtual Time (GVT). The GVT represents the minimum internal time of all simulation units. By definition, no event that is yet to be produced (in wall-clock time) can have a timestamp smaller than the GVT.

We make the distinction between multiple rollback, from single rollback capabilities. To support single rollback, a simulation unit needs to store only the last committed state, thereby saving memory.

Causality is a compositionality property: if each child simulation unit does not violate causality, then any orchestrator has to ensure that the causality is not violated when these units are coupled. Optimistic orchestration algorithms do so by requiring rollback capabilities from child simulation units, whereas pessimistic algorithms do so at the cost of performance.

3.3.2 Determinism and Confluence

Determinism is also a compositional property. The Select function, in the co-simulation scenario definition of eq. (5), is paramount to ensure the compositionality of deterministic behavior. This function is used to ensure that a unique behavior trace can be obtained when the co-simulation scenario is executed by Algorithm 2 or when it is turned into a co-simulation unit, as in eq. (8). The alternative to the Select function is to ensure that all possible interleavings of executions always lead to the same behavior trace – this is known as "confluence". Intuitively, if a co-simulation unit is compositional with respect to confluence, then it is also compositional with respect to determinism.

Proving confluence is hard in general black box DE co-simulation because it depends on how the child simulation units react to external events: potentially valuable IP. Parallel-DEVS [67] is an approach, which leaves the confluence property to be satisfied by the modeler.

3.3.3 Dynamic Structure

Until now, the dependencies $\{I_d\}$, in eq. (5), have been assumed to be fixed over time. From a performance perspective, a static sequence of dependencies may be too conservative, especially if used to ensure causality in optimistic parallel co-simulation. To see why, consider that in a large scale simulation, there is a simulation unit S_1 which may influence simulation unit S_2 but only under a very specific set of conditions, which may not be verified until a large amount of simulated time has passed. A pessimistic co-simulation unit assumes that S_1 may always affect S_2 and hence, tries to ensure that the simulated time of S_2 is always smaller than S_1, to minimize possible rollbacks. This incurs an unnecessary performance toll in the overall co-simulation because S_1 does not affect S_2 most of the time. This is where making I_2 dynamic can improve the performance of the co-simulation since the co-simulation unit will know that most of the time, S_1 does not affect S_2. Dynamic structure co-simulation allows for $\{I_d\}$ to change over time, depending on the behavior trace of the simulation units. It can be used to study self-organizing systems [21, 235].

3.3.4 Distribution

Co-simulation units whose child simulation units are geographically distributed are common [94]. Interesting solutions like computation allocation [180, 240], bridging the hierarchical encapsulation [241], and the use of dead-reckoning models [152], have been proposed to mitigate the additional communication cost. Moreover, security becomes important, and solutions such as [187] address it.

4 Continuous Time Based Co-simulation

In the continuous time (CT) based co-simulation approach, the orchestrators' and simulation units' behavior and assumptions are borrowed from the continuous time system simulation domain. We describe these below.

4.1 CT Simulation Units

A continuous time simulation unit is assumed to have a state that evolves continuously over time. It is easier to get the intuitive idea of this by considering a simulation unit of a continuous time dynamical system, such as a mass-spring-damper, depicted in the left hand side of fig. 5. The state is given by the displacement x_1 and velocity v_1 of the mass, and the evolution by:

$$\begin{aligned} \dot{x_1} &= v_1 \\ m_1 \cdot \dot{v_1} &= -c_1 \cdot x_1 - d_1 \cdot v_1 + F_e \\ x_1(0) &= p_1 \\ v_1(0) &= s_1 \end{aligned} \tag{9}$$

where $\dot{x}$ denotes the time derivative of x; c_1 is the spring stiffness constant and d_1 the damping coefficient; m_1 is the mass; p_1 and s_1 the initial position and velocity; and F_e denotes an external input force acting on the mass over time. The solutions $x_1(t)$ and $v_1(t)$ that satisfy eq. (9) constitute the behavior trace of the dynamical system. fig. 3b shows an example of such trace.

eq. (9) can be generalized to the state space form:

$$\dot{x} = f(x, u) \quad ; \quad y = g(x, u) \quad ; \quad x(0) = x_0 \tag{10}$$

where x is the state vector, u the input and y the output vectors, and x_0 is the initial state.

A solution $[x(t), y(t)]^T$ that obeys eq. (10) is the behavior trace of the system. If f is linear and time-invariant, an analytical form for $x(t)$ can be obtained [17]. An analytical solution obtained by the application of mathematical identities is an example of a behavior trace obtained via the translational approach, described in section 2.2. Alternatively, the behavior trace can be computed.

If $f(x, u)$ is sufficiently differentiable, x can be approximated with a truncated Taylor series [63, 227]:

$$x(t + h) = x(t) + f(x(t), u(t)) \cdot h + \mathcal{O}\left(h^2\right) \tag{11}$$

where

$$\mathcal{O}\left(h^{n+1}\right) = \max_i \left(\lim_{h \to 0} \frac{x^{n+1}\left(\zeta(t^*)\right)}{(n+1)!} h^{n+1} \right) = \text{const} \ \cdot h^{n+1}$$

denotes the order of the truncated residual term; $t^* \in [t, t + h]$; and $h \geq 0$ is the micro-step size. eq. (11) is the basis of a family of numerical solvers that iteratively compute an approximated behavior trace $\tilde{x}$. For example, the Forward Euler method is given by:

$$\begin{aligned} \tilde{x}(t + h) &:= \tilde{x}(t) + f(\tilde{x}(t), u(t)) \cdot h \\ \tilde{x}(0) &:= x(0) \end{aligned} \tag{12}$$

A CT simulation unit is assumed to have a behavior that is similar to one of a numerical solver computing a set of differential equations. We reinforce that this does not restrict CT simulation units to being mockups of CT systems, even though it is easier to introduced them as such. In the FMI Standard, a simulation unit is analogous to a Functional Mock-up Unit (FMU) for co-simulation. For example, a simulation unit S_1 of the mass-spring-damper, using the Forward Euler solver, can be written by embedding the solver (eq. (12)) into eq. (9):

$$\begin{aligned} \tilde{x}_1(t + h_1) &:= \tilde{x}_1(t) + v_1(t) \cdot h_1 \\ \tilde{v}_1(t + h_1) &:= \tilde{v}_1(t) + \frac{1}{m_1} \cdot \left(-c_1 \cdot \tilde{x}_1(t) - d_1 \cdot \tilde{v}_1(t) + F_e(t)\right) \cdot h_1 \\ \tilde{x}_1(0) &:= p_1 \\ \tilde{v}_1(0) &:= s_1 \end{aligned} \tag{13}$$

where h_1 is the micro-step size, $F_e(t)$ is the input, and $[x(t + h), v(t + h)]^T$ is the output.

4.2 CT Co-simulation Orchestration

Consider now a second system, depicted in the right hand side of fig. 5. It is governed by the differential equations:

$$
\begin{aligned}
\dot{x_2} &= v_2 \\
m_2 \cdot \dot{v_2} &= -c_2 \cdot x_2 - F_c \\
F_c &= c_c \cdot (x_2 - x_c) + d_c \cdot (v_2 - \dot{x_c}) \\
x_2(0) &= p_2 \\
v_2(0) &= s_2
\end{aligned} \tag{14}
$$

where c_c and d_c denote the stiffness and damping coefficients of the spring and damper, respectively; x_c denotes the displacement of the left end of the spring-damper. Combining with the Forward Euler solver, yields the following simulation unit:

$$
\begin{aligned}
\tilde{x}_2(t + h_2) &:= \tilde{x}_2(t) + \tilde{v}_2(t) \cdot h_2 \\
\tilde{v}_2(t + h_2) &:= \tilde{v}_2(t) + \frac{1}{m_2} \cdot (-c_2 \cdot \tilde{x}_2(t) - F_c(t)) \cdot h_2 \\
F_c(t) &= c_c \cdot (\tilde{x}_2(t) - x_c(t)) + d_c \cdot \left(\tilde{v}_2(t) - \dot{x_c(t)}\right) \\
\tilde{x}_2(0) &:= p_2 \\
\tilde{v}_2(0) &:= s_2
\end{aligned} \tag{15}
$$

where h_2 is the micro-step size, x_c and $\dot{x_c}$ are inputs, and F_c the output. Suppose S_1 (eq. (13)) and S_2 are coupled, setting $x_c = x_1$, $\dot{x_c} = v_1$ and $F_e = F_c$, so that the resulting co-simulation scenario represents the multi-body system depicted in fig. 5.

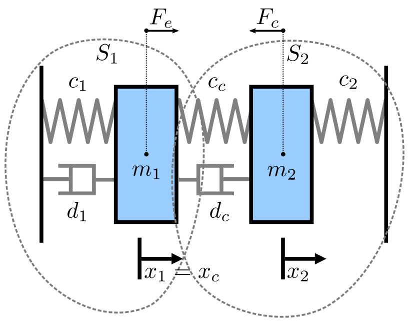


Figure 5: A multi-body system comprised of two mass-spring-damper subsystems.

In the co-modeling approach, the models in Equations 9 and 14 would be combined to get the following coupled model:

$$\begin{aligned}
\dot{x_1} &= v_1 \\
m_1 \cdot \dot{v_1} &= -c_1 \cdot x_1 - d_1 \cdot v_1 + F_c \\
\dot{x_2} &= v_2 \\
m_2 \cdot \dot{v_2} &= -c_2 \cdot x_2 - F_c \\
F_c &= c_c \cdot (x_2 - x_1) + d_c \cdot (v_2 - v_1) \\
x_1(0) &= p_1 \\
v_1(0) &= s_1 \\
x_2(0) &= p_2 \\
v_2(0) &= s_2
\end{aligned} \tag{16}$$

which can be written in the state space form (eq. (10)) as:

$$\begin{aligned}
\begin{bmatrix} \dot{x_1} \\ \dot{v_1} \\ \dot{x_2} \\ \dot{v_2} \end{bmatrix} &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{c_1+c_c}{m_1} & -\frac{d_1+d_c}{m_1} & \frac{c_c}{m_1} & \frac{d_c}{m_1} \\ 0 & 0 & 0 & 1 \\ \frac{c_c}{m_2} & \frac{d_c}{m_2} & -\frac{c_2+c_c}{m_2} & -\frac{d_c}{m_2} \end{bmatrix} \begin{bmatrix} x_1 \\ v_1 \\ x_2 \\ v_2 \end{bmatrix} \\
\begin{bmatrix} x_1(0) \\ v_1(0) \\ x_2(0) \\ v_2(0) \end{bmatrix} &= \begin{bmatrix} p_1 \\ s_1 \\ p_2 \\ s_2 \end{bmatrix}
\end{aligned} \tag{17}$$

The behavior trace of eq. (17) can be obtained either analytically, or with Forward Euler solver (eq. (12)).

In CT based co-simulation, to overcome the fact the each simulation unit's micro-step sizes are independent, a communication step size H (also known as macro-step size or communication grid size) has to be defined. H marks the times at which the simulation units exchange values of inputs/outputs.

Suppose a simulation unit S_i is at time $n \cdot H$, for some natural n, and it being asked by an orchestrator to execute until time $(n+1) \cdot H$. If S_i only gets its inputs valued at $n \cdot H$, then extrapolation must be used to associate a value with those inputs in any of the internal micro-steps of the simulation unit. In other words, when time $n \cdot H + m \cdot h_i$, for $m \leq \frac{H}{h_i}$ and micro-step size h_i, an extrapolation function $\phi_{u_i}(m \cdot h_i, u_i(n \cdot H), u_i((n-1) \cdot H), \ldots)$, built from input values known at previous communication time points, is used to approximate the value of $u_i(n \cdot H + m \cdot h_i)$. Notice that $m = \frac{H}{h_i}$ is allowed, even though, theoretically, the value of $u_i((n+1) \cdot H)$ can be obtained from the environment. The reason for this becomes clear in section 4.3.2. Analogously, interpolation techniques have to be used when the orchestrator makes the input value available at time $(n+1) \cdot H$ but the simulation unit is still at time $n \cdot H$.

For example, the input F_e of the simulation unit described in eq. (13) can be defined as:

$$F_e(n \cdot H + m \cdot h_1) := \phi_{F_e}(m \cdot h_1, F_e(n \cdot H), F_e((n-1) \cdot H), \ldots), \text{ for } m \leq \frac{H}{h_1} \tag{18}$$

Similarly, the inputs x_c and $\dot{x_c}$ of the simulation unit described in eq. (15) can be defined as

$$\begin{aligned} x_c(n \cdot H + m \cdot h_2) &:= \phi_{x_c}(m \cdot h_2, x_c(n \cdot H), x_c((n-1) \cdot H), \ldots) \\ \dot{x_c}(n \cdot H + m \cdot h_2) &:= \phi_{\dot{x_c}}(m \cdot h_2, \dot{x_c}(n \cdot H), \dot{x_c}((n-1) \cdot H), \ldots) \\ \text{for } m &\leq \frac{H}{h_2} \end{aligned} \tag{19}$$

In the simplest case, the extrapolations can be constant. In the coupled mass-spring-dampers example:

$$\begin{aligned} \phi_{F_e}(t, F_e(n \cdot H))) &= F_e(n \cdot H)) \\ \phi_{x_c}(t, x_c(n \cdot H)) &= x_c(n \cdot H) \\ \phi_{\dot{x_c}}(t, \dot{x_c}(n \cdot H)) &= \dot{x_c}(n \cdot H) \end{aligned} \tag{20}$$

In the state of the art, input extrapolation approaches can be classified by increasing degree of complexity: Constant; Linear; Polynomial; Extrapolated-Interpolation [49, 50, 75]; Context-aware [134]; and Estimated Dead-Reckoning Model [43, 221]; These can, and often are, combined in practical use cases. See Andersson [8], Arnold [10], Busch [50], Schweizer et al. [219] for an overview of linear and higher order extrapolation techniques and how these affect the accuracy of the co-simulation trace.

The orchestrator for this co-simulation scenario, at a time $t = n \cdot H$, gets the outputs of both simulation units and computes their inputs. Then, each simulation unit is instructed to compute its behavior trace until the next communication step size, at $t = (n+1) \cdot H$, making use of the extrapolating functions to get the inputs at each of the micro steps (Equations 18 and 19).

We are ready to formally define the behavior of a CT simulation unit S_i:

$$\begin{aligned} & S_i = \langle X_i, U_i, Y_i, \delta_i, \lambda_i, x_i(0), \phi_{U_i} \rangle \\ & \delta_i : \mathbb{R} \times X_i \times U_i \to X_i \\ & \lambda_i : \mathbb{R} \times X_i \times U_i \to Y_i \text{ or } \mathbb{R} \times X_i \to Y_i \\ & x_i(0) \in X_i \\ & \phi_{U_i} : \mathbb{R} \times U_i \times \ldots \times U_i \to U_i \end{aligned} \tag{21}$$

where:

- X_i is the state set, typically $\mathbb{R}^n$;
- U_i is the input set, typically $\mathbb{R}^m$;
- Y_i is the output set, typically $\mathbb{R}^p$;
- $\delta_i(t, x_i(t), u_i(t)) = x_i(t+H)$ or $\delta_i(t, x_i(t), u_i(t+H)) = x_i(t+H)$ is the function that instructs the simulation unit to compute a behavior trace from t to $t+H$, making use of the input extrapolation (or interpolation) function ϕ_{U_i};
- $\lambda_i(t, x_i(t), u_i(t)) = y_i(t)$ or $\lambda_i(t, x_i(t)) = y_i(t)$ is the output function; and
- $x_i(0)$ is the initial state.

For instance, the simulation unit in eq. (13) can be described as follows:

$$
\begin{aligned}
& S_1 = \left\langle \mathbb{R}^2, \mathbb{R}, \mathbb{R}^2, \delta_1, \lambda_1, \begin{bmatrix} p_1 \\ s_1 \end{bmatrix}, \phi_{F_e} \right\rangle \\
& \delta_1(t, \begin{bmatrix} \tilde{x}_1(t) \\ \tilde{v}_1(t) \end{bmatrix}, F_e(t)) = \begin{bmatrix} \tilde{x}_1(t+H) \\ \tilde{v}_1(t+H) \end{bmatrix} \\
& \lambda_1(t, \begin{bmatrix} \tilde{x}_1(t) \\ \tilde{v}_1(t) \end{bmatrix}) = \begin{bmatrix} \tilde{x}_1(t) \\ \tilde{v}_1(t) \end{bmatrix}
\end{aligned}
\tag{22}
$$

where $[\tilde{x}_1(t+H), \tilde{v}_1(t+H)]^T$ is obtained by the iterative application of the simulation unit in eq. (13) over a finite number of micro-steps, making use of the extrapolation of F_e (defined in eq. (18)):

$$
\begin{bmatrix} \tilde{x}_1(t+H) \\ \tilde{v}_1(t+H) \end{bmatrix} = \begin{bmatrix} \tilde{x}_1(t) \\ \tilde{v}_1(t) \end{bmatrix} + \begin{bmatrix} \dot{x_1}(t) \\ \dot{v_1}(t, \phi_{F_e}(t, F_e(t), \ldots)) \end{bmatrix} \cdot h + \begin{bmatrix} \dot{x_1}(t+h) \\ \dot{v_1}(t+h, \phi_{F_e}(t+h, F_e(t), \ldots)) \end{bmatrix} \cdot h + \ldots
$$

A continuous time co-simulation scenario with reference cs includes at least the following information[12]:

$$
\begin{aligned}
& \langle U_{cs}, Y_{cs}, D, \{S_i : i \in D\}, L, \phi_{U_{cs}} \rangle \\
& L : (\Pi_{i \in D} Y_i) \times Y_{cs} \times (\Pi_{i \in D} U_i) \times U_{cs} \to \mathbb{R}^m
\end{aligned}
\tag{23}
$$

where D is an ordered set of simulation unit references, each S_i is defined as in eq. (21), $m \in \mathbb{N}$, U_{cs} is the space of inputs external to the scenario, Y_{cs} is the space of outputs of the scenario, $\phi_{U_{cs}}$ a set of input approximation functions, and L induces the simulation unit coupling constraints (e.g., if $D = \{1, \ldots, n\}$, then the coupling is $L(y_1, \ldots, y_n, y_{cs}, u_1, \ldots, u_n, u_{cs}) = \bar{0}$).

As an example, the co-simulation scenario representing the system of fig. 5 is:

$$
\begin{aligned}
& cs = \langle \emptyset, \emptyset, \{1, 2\}, \{S_1, S_2\}, L, \emptyset \rangle \\
& L = \begin{bmatrix} x_c - v_1 \\ \dot{x_c} - x_1 \\ F_e - F_c \end{bmatrix}
\end{aligned}
\tag{24}
$$

where:

- S_1 is the simulation unit for the constituent system on the left (eq. (22)), and S_2 is the simulation unit for the remaining constituent system;
- x_c, $\dot{x_c}$ are the inputs of S_2, and F_e is the input of S_1; and
- x_1, v_1 are outputs of S_1 and F_c is the output of S_2.

Algorithm 3 summarizes in a generic way the tasks of the orchestrator for computing the co-simulation of a scenario cs with no external inputs. It represents the Jacobi communication approach: simulation units exchange values at time t and independently compute the trace until the next communication time $t + H$. The way the system in eq. (25) is solved depends on how the simulation units are coupled, that is, the definition of L. In the most trivial case, the system reduces to an assignment of an output $y_j(t)$ to each input $u_i(t)$, and so the orchestrator just gets the output of each simulation unit and copies it onto the input of some other simulation unit, in an appropriate order. Concrete examples of Algorithm 3 are described in [25, 52, 79, 93, 96, 105, 140, 253].

[12]Please note that this formalization is related to the formalization proposed by Broman et al. [46], with the main differences: i) it is not designed to formalize a subset of the FMI Standard, ii) it accommodates algebraic coupling conditions, and iii) it does not explicitly define port variables.

An alternative to the Jacobi communication approach is the Gauss-Seidel (a.k.a. sequential or zig-zag) approach, where an order of the simulation units' δ function is forced to ensure that, at time t, they get inputs from a simulation unit that is already at time $t+H$. Gauss-Seidel approach allows for interpolations of inputs, which is more accurate, but hinders the parallelization potential. Examples are described in [10, 11, 25, 59, 236].

Algorithm 3: Generic Jacobi based orchestrator for autonomous CT co-simulation scenarios.

```
Data: An autonomous scenario $cs = \langle \emptyset, Y_{cs}, D = \{1, \ldots, n\}, \{S_i\}, L, \emptyset \rangle$ and a
      communication step size $H$.
Result: A co-simulation trace.
$t := 0$ ;
$x_i := x_i(0)$ for $i = 1, \ldots, n$ ;
while true do
    Solve the following system for the unknowns:
        $$\begin{cases} y_1 = \lambda_1(t, x_1, u_1) \\ \ldots \\ y_n = \lambda_n(t, x_n, u_n) \\ L(y_1, \ldots, y_n, y_{cs}, u_1, \ldots, u_n) = \bar{0} \end{cases} \quad (25)$$
    $x_i := \delta_i(t, x_i, u_i)$, for $i = 1, \ldots, n$ ; // Instruct each simulation unit to advance
        to the next communication step
    $t := t + H$ ;                                          // Advance time
end
```

Similarly to DE based co-simulation, a CT co-simulation scenario, together with an orchestrator, should behave as a (co-)simulation unit of the form of eq. (21), and thus be coupled with other simulation units, forming hierarchical co-simulation scenarios: the state of the co-simulation unit is the set product of the states of the internal units; the inputs are given by U_{cs} and the outputs by Y_{cs}; the transition and output functions are implemented by the orchestrator; the communication step size H used by the orchestrator is analogous to a simulation unit's micro-step sizes, and the input extrapolation function is ϕ_{U_i}.

Algorithm 3 makes it clear that the simulation units can be coupled with very limited information about their internal details. In concrete:

- The output λ_i and state transition δ_i functions need to be executable but their internal details can remain hidden;
- the inputs u_i need to be accessible;
- the state variables can be hidden. These are represented merely to illustrate that the internal state of the simulation unit changes when executing δ_i.

However, the blind coupling can lead to compositionality problems, as will be discussed in the sections below. The common trait in addressing these is to require more from the individual simulation units: either more capabilities, or more information about the internal (hidden) dynamical system.

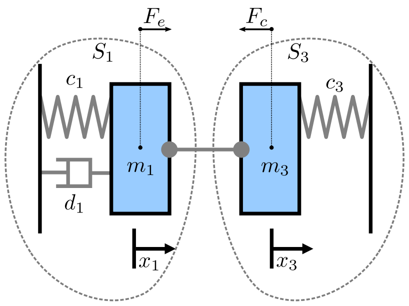


Figure 6: A multi-body system coupled by a mass-less link, based on the example provided in Schweizer and Lu [214].

4.3 Challenges

4.3.1 Modular Composition – Algebraic Constraints

In the co-simulation scenario described in eq. (24), the coupling condition L translates into a set of assignments from outputs to inputs. This is because the inputs of the simulation unit of the system in the left hand side of fig. 5 and the outputs of the simulation unit of the system represented in the right hand side of the same picture can be connected directly, and vice versa. In practice, the simulation units' models are not created with a specific coupling pattern in mind and L can be more complex. As an example, consider the system coupled by a massless rigid link, depicted in fig. 6. The first subsystem is the same as the one in the left hand side of fig. 5 and its simulation unit is in eq. (13). The second constituent system is governed by the following differential equations:

$$\begin{aligned} \dot{x_3} &= v_3 \\ m_3 \cdot \dot{v_2} &= -c_3 \cdot x_3 + F_c \\ x_3(0) &= p_3 \\ v_3(0) &= s_3 \end{aligned} \tag{26}$$

And the following simulation unit:

$$\begin{aligned} \tilde{x}_3(t+h_3) &= \tilde{x}_3(t) + v_3(t) \cdot h_3 \\ \tilde{v}_3(t+h_3) &= \tilde{v}_3(t) + \frac{1}{m_3} \cdot (-c_3 \cdot x_3(t) + F_c(t)) \cdot h_3 \\ \tilde{x}_3(0) &= p_3 \\ \tilde{v}_3(0) &= s_3 \end{aligned} \tag{27}$$

The input to the simulation unit S_3 is the coupling force F_c, and the output is the state of the mass $[\tilde{x}_3, \tilde{v}_3]^T$. The input to the simulation unit S_1 is the external force F_e and the outputs are the state of the mass $[\tilde{x}_1, \tilde{v}_1]^T$. Recall eq. (13). There is clearly a mismatch. The outputs $[\tilde{x}_1, \tilde{v}_1]^T$ of the first simulation unit cannot be coupled directly to the input F_c of the second simulation unit, and vice versa. However, the massless link restricts the states and inputs of the two units to be the same. Whatever the input forces may be, they are equal and opposite in sign. Hence, any orchestration algorithm has to find inputs that ensure the coupling constraints are satisfied:

$$L = \begin{bmatrix} \tilde{x}_1(n \cdot H) - \tilde{x}_3(n \cdot H) \\ \tilde{v}_1(n \cdot H) - \tilde{v}_3(n \cdot H) \\ F_e(n \cdot H) + F_c(n \cdot H) \end{bmatrix} = \bar{0} \tag{28}$$

This problem has been addressed in Arnold [10], Arnold and Günther [11], Gu and Asada [105, 106, 107], Schweizer and Lu [214, 215, 216], Schweizer et al. [218, 219], Sicklinger et al. [220]. The approach taken in Gu and Asada [105] is worth mentioning because it defines a Boundary Condition Coordinator (BCC) which behaves as an extra simulation unit, whose inputs are the outputs of the original two simulation units, and whose outputs are F_e and F_c. They show that the initial co-simulation scenario with the non-trivial constraint can be translated into a co-simulation, with a trivial constraint, by adding an extra simulation unit. This is illustrated in fig. 7.

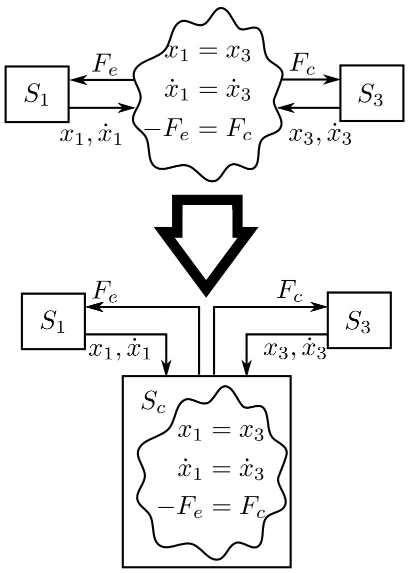


Figure 7: Transforming a co-simulation scenario with a non-trivial constraint into a simpler scenario by adding an extra simulation unit that induces a trivial constraint. This promotes separation of concerns.

Transforming the co-simulation scenario to make it simpler marks an important step in separating the concerns of the orchestrator [102]. In fact, the newly created simulation unit can be run with a smaller internal micro-step size, required to meet stability and accuracy criteria, as shown in Gu and Asada [105].

In many of the solutions proposed (e.g., [10, 11, 214, 219, 220]), information about the rate of change (or sensitivity) of outputs and states of each simulation unit, with respect to changes in its inputs is required to solve the non-trivial coupling condition. This information can be either provided directly as a Jacobian matrix of the system and output functions, or estimated by finite differences, provided that the simulation units can be rolled back to previous states. A frequent characteristic of co-simulation: the availability of certain capabilities from simulation units can mitigate the lack of other capabilities.

To show why the sensitivity information is useful, one of the tasks of the BCC is to ensure that $\tilde{x}_1 - \tilde{x}_3$ is as close to zero as possible, by finding appropriate inputs F_e and F_c. This is possible since $\tilde{x}_1$ and $\tilde{x}_3$ are functions of the inputs F_e and F_c, and $-F_e = F_c$. So the constraint can be written as

$$g(F_e) = \tilde{x}_1(F_e) - \tilde{x}_3(-F_e) = 0 \tag{29}$$

From one communication step to the next, g can be expanded with the Taylor series:

$$g(F_e((n+1)\cdot H)) = g(F_e(n\cdot H) + \Delta F_e) \approx g(F_e(n\cdot H)) + \frac{\partial g(F_e(n\cdot H))}{\partial F_e}\cdot \Delta F_e \tag{30}$$

From a known input $F_e(n\cdot H)$, Equations 29 and 30 can be combined to obtain the input $F_e((n+1)\cdot H)$ at the next communication step:

$$\begin{aligned} g(F_e(n\cdot H) + \Delta F_e) \approx g(F_e(n\cdot H)) + \frac{\partial g(F_e(n\cdot H))}{\partial F_e}\cdot \Delta F_e = 0 \leftrightarrow \\ g(F_e(n\cdot H)) = -\frac{\partial g(F_e(n\cdot H))}{\partial F_e}\cdot \Delta F_e \leftrightarrow \\ \Delta F_e = -\left[\frac{\partial g(F_e(n\cdot H))}{\partial F_e}\right]^{-1}\cdot g(F_e(n\cdot H)) \leftrightarrow \\ F_e((n+1)\cdot H) = F_e(n\cdot H) - \left[\frac{\partial g(F_e(n\cdot H))}{\partial F_e}\right]^{-1}\cdot g(F_e(n\cdot H)) \end{aligned} \tag{31}$$

with

$$\frac{\partial g(F_e(n\cdot H))}{\partial F_e} = \frac{\partial \tilde{x}_1(F_e(n\cdot H))}{\partial F_e} + \frac{\partial \tilde{x}_3(-F_e(n\cdot H))}{\partial F_c} \tag{32}$$

A simple orchestration algorithm will then perform the following steps, at each co-simulation step:

1. Let $\tilde{x}_1(nH), \tilde{x}_3(nH)$ be the current position outputs of the two simulation units S_1 and S_3;
2. Perform a co-simulation step with a known F_e, obtaining $\tilde{x}_1^p(nH), \tilde{x}_3^p(nH)$ as new outputs.
3. Rollback simulation units to state $\tilde{x}_1(nH), \tilde{x}_3(nH)$;
4. Perform a co-simulation step with $F_e + \Delta F_e$, obtaining $\tilde{x}_1^d(nH), \tilde{x}_3^d(nH)$;
5. Approximate $\frac{\partial g(F_e(n\cdot H))}{\partial F_e}$ by finite differences and eq. (32);
6. Obtain a corrected F_e^c by eq. (31);
7. Rollback simulation units to state $\tilde{x}_1(nH), \tilde{x}_3(nH)$;
8. Perform the final co-simulation step with F_e^c;
9. Commit states and advance time;

As can be seen in fig. 8, this coupling cannot be carried out without errors: the constraint $g(F_e((n+1)\cdot H))$ cannot be accurately forced to zero at first try. Furthermore, finding initial conditions and initial inputs that satisfy Equations 9, 26, and 28 is very important and usually requires a fixed point iteration. The above algorithm could be changed to perform an arbitrary

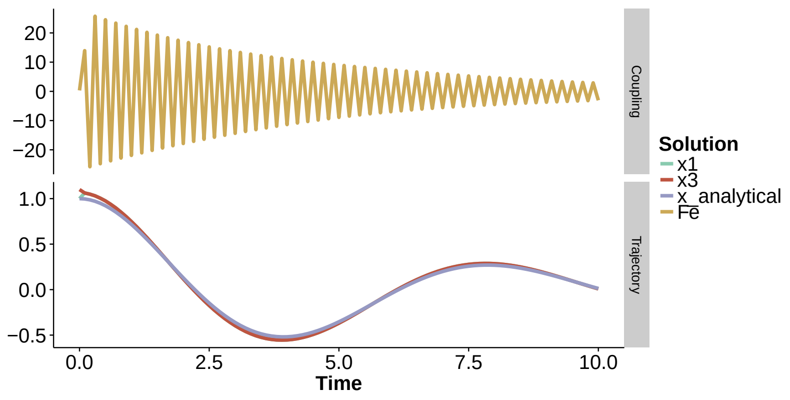


Figure 8: Co-simulation of algebraically coupled masses. Parameters are: $m_2 = 2, m_1 = c_1 = c_3 = d_1 = c_c = 1, H = 0.1, x_1(0) = 1.0, x_3(0) = 1.1, v_1(0) = v_3(0) = 0$. Notice the small disturbance at the initial conditions.

number of iterations, repeating steps 1–7 until $g(F_e((n+1) \cdot H))$ is close enough to zero. This would increase the accuracy but also increase the amount of computation.

These examples show that rollback capabilities are important. If a simulation unit is a black box, then the rollback capability has to be provided by the simulation unit itself and there is little that the orchestrator can do to make up for the lack of the feature. See Broman et al. [46] for an orchestrator that takes into account the existence of the rollback feature. If, on the other hand, the simulation unit provides access to its state, and allows the state to be set, as in Blockwitz et al. [35], then the orchestrator can implement the rollback by keeping track of the state of the simulation unit. Rollback also plays a key role when dealing with algebraic loops in the co-simulation scenario.

Finally, to explain why this sub-section refers to modular composition of simulation units, the example in fig. 6 makes explicit one of the problems in co-simulation: the "rigid" and protected nature of simulation units can make their coupled simulation very difficult. To contrast, in a white box approach where the equations of both constituent systems are available, the whole system is simplified, with the two masses being lumped together, and their coupling forces canceling each other out. The simplified system is a lumped mass-spring-damper, which is easily solvable. Such an approach is common in a-causal modeling languages, such as Modelica [1]. To be concrete, the coupled system is obtained by combining Equations 9, 26, and 28, and simplifying to:

$$\begin{aligned}
\dot{x_1} &= v_1 \\
(m_1 + m_3) \cdot \dot{v_1} &= -(c_1 + c_3) \cdot x_1 - d_1 \cdot v_1 \\
x_1(0) &= p_1 \\
v_1(0) &= s_1
\end{aligned} \tag{33}$$

fig. 9 compares the behavior trace produced by Algorithm 3 when applied to the co-simulation scenario described in eq. (24), with the analytical solution, obtained from the coupled model of eq. (17) (co-modelling). It is obvious that there is an error due to the extrapolation functions and the large communication step size $H = 0.1$.

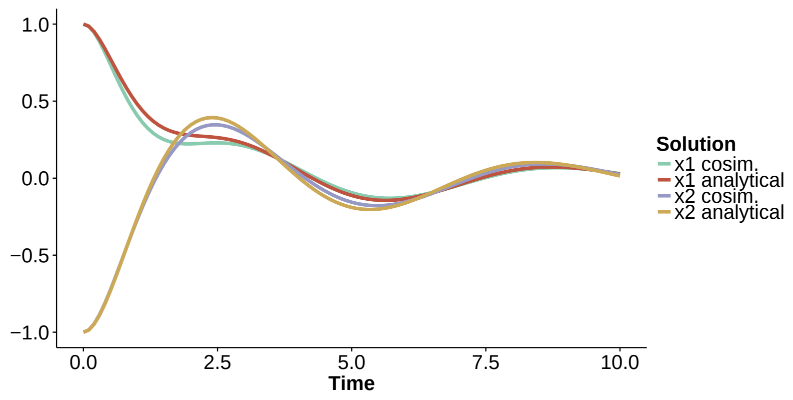


Figure 9: Comparison of co-simulation with co-modelling for the sample coupled system. Parameters are: $m_1 = m_2 = c_1 = c_2 = d_1 = c_c = 1, H = 0.1$.

This is more modular because (if the equations are made available) the same constituent system can be coupled to other systems in many different contexts, without further changes. As this sub-section shows, in co-simulation it is possible to get around the modularity aspect, but at a cost.

4.3.2 Algebraic loops

Algebraic loops occur whenever there is a variable that indirectly depends on itself. To see how algebraic loops arise in co-simulation scenarios, recall (see eq. (21)) that the state evolution and output of each simulation unit S_i can be written as:

$$\begin{aligned} x_i(t+H) &= \delta_i(t, x_i(t), u_i(t)) \\ y_i(t+H) &= \lambda_i(t, x_i(t+H), u_i(t+H)) \end{aligned} \tag{34}$$

To simplify things, assume that the simulation units are coupled by a set of assignments from outputs to inputs, i.e.,

$$u_i(t) := y_j(t) \tag{35}$$

where u_i is the input of simulation unit S_i and y_j the output of a simulation unit S_j, in the same co-simulation scenario.

With these definitions, it is easy to see that, depending on the coupling assignments of the co-simulation scenario, the output of a simulation unit may depend on itself, that is,

$$\begin{aligned} \mathbf{y_i(t+H)} &= \lambda_i(t, x_i(t+H), u_i(t+H)) \\ u_i(t+H) &= y_j(t+H) \\ y_j(t+H) &= \lambda_j(t, x_j(t+H), u_j(t+H)) \\ u_j(t+H) &= y_k(t+H) \\ &\cdots \\ u_z(t+H) &= \mathbf{y_i(t+H)} \end{aligned} \tag{36}$$

We distinguish two kinds of algebraic loops in co-simulation [141]: the ones spanning just input variables, and the ones that include state variables as well. The first kind can be avoided by using the input extrapolations as parameters to the output functions. The second kind arises when implicit numerical solvers are used, or when the input approximating functions are interpolations instead of extrapolations. In the previous example, the first kind can be removed by replacing $u_i(t+H)$ in eq. (34) by the corresponding extrapolation $\phi_{u_i}(H, u_i(n \cdot H), u_i((n-1) \cdot H), \ldots)$ which does not depend on $u_i((n+1) \cdot H)$, thus breaking the algebraic loop. These methods just ignore the algebraic loop though, and as is shown in Arnold et al. [14], Kübler and Schiehlen [141] (and empirically in Bastian et al. [25]), neglecting an algebraic loop can lead to a prohibitively high error in the co-simulation. A better way is to use a fixed point iteration technique. For algebraic loops involving state variables, the same co-simulation step has to be repeated until convergence. If the algebraic loop does not involve any state variable, then the iteration is just on the output functions.

To see how algebraic loops involving state variables arise, suppose that, in the example above, ϕ_{u_i} is constructed from $u_i((n+1) \cdot H)$:

$$u_i(n \cdot H + m \cdot h_i) := \phi_{u_i}(m \cdot h_i, u_i((n+1) \cdot H), u_i(n \cdot H), u_i((n-1) \cdot H), \ldots) \tag{37}$$

If an order can be imposed in the evaluation of the simulation units that ensures $u_i((n+1) \cdot H)$ can be computed from some $\lambda_j(t, x_j((n+1) \cdot H), u_j((n+1) \cdot H))$ that does not indirectly depend on $u_i((n+1) \cdot H)$, then this approach —Gauss-Seidel— can improve the accuracy of the co-simulation, as shown in Arnold [10], Arnold and Günther [11], Arnold et al. [14], Busch [50], Kalmar-Nagy and Stanciulescu [128]. Obviously, the execution of simulation unit S_i has to start after simulation unit S_j has finished and its output $\lambda_j(t, x_j((n+1) \cdot H), u_j((n+1) \cdot H))$ can be evaluated. If the input $u_j((n+1)$ depends indirectly on $u_i((n+1) \cdot H)$, then an algebraic loop exists. The output function $\lambda_j(t, x_j((n+1) \cdot H), u_j((n+1) \cdot H))$ depends on the state of the simulation unit at $x_j((n+1) \cdot H)$, which in turn can only be obtained by executing the simulation unit from time $n \cdot H$ to $(n+1) \cdot H$, using the extrapolation of the input u_j, $\phi_{u_j}(m \cdot h_i, u_j((n+1) \cdot H, \ldots))$; any improvement in the input $u_j((n+1) \cdot H$, means that the whole co-simulation step has to be repeated, to get an improved $x_j((n+1) \cdot H)$ and by consequence, an improved output $\lambda_j(t, x_j((n+1) \cdot H), u_j((n+1) \cdot H))$.

A fixed point iteration technique that makes use of rollback to repeat the co-simulation step with corrected inputs is called dynamic iteration, waveform iteration, and strong or onion coupling [119, 231]. If the simulation units expose their outputs at every internal micro-step, then the waveform iteration can be improved [155]. Strong coupling approaches are typically the best in terms of accuracy, but worst in terms of performance. Approaches that do not perform any correction steps are the best in terms of performance, but worst in accuracy. A variant that attempts to obtain the middle-ground is the so-called semi-implicit method, where a fixed limited number of correction steps is performed. See Schweizer and Lu [214, 215, 216], Schweizer et al. [218, 219] for examples of this approach.

In the current FMI Standard for co-simulation, it is not possible to perform a fixed point iteration on the output variables only, in the *step mode*. A work-around this is to rollback the simulation units and repeat the co-simulation step, effectively treating the algebraic loop as involving the state variables too.

Until here, we have assumed full knowledge of the models being simulated in each simulation unit to explain how to identify, and deal with, algebraic loops. In practice, with general black-box simulation units, extra information is required to identify algebraic loops. According to Arnold et al. [13], Benveniste et al. [32], Broman et al. [46], a binary flag denoting whether an output depends directly on an input is sufficient. A structural analysis, for example, with Tarjan's strong component algorithm [225], can then be performed to identify the loops.

4.3.3 Consistent Initialization of Simulators

The definition of a simulation unit in eq. (21) assumes that an initial condition is part of the simulation unit. However, as seen in the example of section 4.3.1, the initial states of the simulation units can be coupled by algebraic constraints, through the output functions, which implies that the initial states of the simulation units cannot be set independently of the co-simulation in which they are used. For example, the constraint in eq. (28) has to be satisfied for the initial states:

$$\{\tilde{x}_1(0), \tilde{v}_1(0), \tilde{x}_3(0), \tilde{v}_3(0)\}.$$

In general, for a co-simulation scenario as defined in eq. (23), there is an extra coupling function L_0 that at the time $t = 0$, has to be satisfied. For example:

$$L_0(x_1(0), \ldots, x_n(0), y_1(0), \ldots, y_n(0), y_{cs}(0), u_1(0), \ldots, u_n(0), u_{cs}(0)) = \bar{0} \tag{38}$$

where:

- $x_i(0)$ denotes the initial state of simulation unit S_i; and
- $L_0 : X_1 \times \ldots \times X_n \times Y_1 \times \ldots \times Y_n \times U_1 \times \ldots \times U_n \to \mathbb{R}^m$ represents the initial constraint, not necessarily equal to L in eq. (23).

eq. (38) may have an infinite number of solutions – as in the case of the example provided in section 4.3.1 – or have algebraic loops. The initialization problem (or co-initialization) is identified in Blockwitz et al. [35] and addressed in Galtier et al. [96].

In the FMI Standard, there is a dedicated mode for the (possible fixed point iteration based) search of a consistent initial state in all simulation units.

4.3.4 Compositional Convergence – Error Control

The accuracy of a co-simulation trace is the degree to which it conforms to the real trace as described in section 2.2. Obtaining the real trace can be a challenge. Error —the difference between the co-simulation trace and the real trace— is then a measure of accuracy.

In the context of continuous co-simulation, the most accurate trace is the analytical solution to the coupled model that underlies the scenario. For example, the coupled model in eq. (17), corresponding to the multi-body system in fig. 5, is implicitly created from the co-simulation scenario described in eq. (24). Fortunately, the analytical solution can be obtained for this coupled model because it forms a linear time invariant system. In practice, the analytical solution for a coupled model cannot be found easily. Calculating the error precisely is impossible for most cases but getting an estimate in how the error grows is a well understood procedure in numerical analysis.

In simulation, the factors that influence the error are [63]: model, solver, micro-step size, and, naturally, the size of the time interval to be simulated. In co-simulation, the extrapolation functions introduce error in the inputs of the simulation units, which is translated into error in the state/outputs of these, causing a feedback on the error that can increase over time. Intuitively, the larger the co-simulation step size H, the larger is the error made by the extrapolation functions.

For example, when the Forward Euler solver (eq. (12)) is used to compute the approximated behavior trace of the dynamical system in eq. (10), in a single micro step, it is making an error in the order of

$$\left\| \underbrace{\left(x(t) + f(x(t)) \cdot h + \mathcal{O}\left(h^2\right)\right)}_{\text{by infinite Taylor series}} - \underbrace{\left(x(t) + f(x(t)) \cdot h\right)}_{\text{by Forward Euler}} \right\| = \mathcal{O}\left(h^2\right)$$

Obviously, the order in the error made at one step $\mathcal{O}\left(h^2\right)$, most commonly called the local error, depends on:

- f having no unbounded derivatives – to see why, observe that if the derivative of f is infinite, then the residual term cannot be bounded by a constant multiplied by h^2. Fortunately, since most continuous time dynamic systems model some real system, this assumption is satisfied.
- The solver used – other solvers, such as the midpoint method, are derived by truncating higher order terms of the Taylor series. For the midpoint method, the local truncation error is $\mathcal{O}\left(h^3\right)$;
- Naturally, the larger the micro step size h is, the larger the local error $\mathcal{O}\left(h^2\right)$ is.

The local error assumes that the solver only made one step, starting from an accurate point $x(t)$. To compute the approximate behavior trace, the only accurate point the solver starts from is the initial value $x(0)$. The rest of the trace is approximate and the error gets compounded over the multiple steps. For the Forward Euler method, if there is a limit to how f reacts to deviations on its parameter $\tilde{x}(t) = x(t) + e(t)$ from the true parameter $x(t)$, that is, if

$$\|f(x(t)) - f(x(t) + e(t))\| \leq \text{const } \cdot e(t)$$

and const $< \infty$, then the order of the total accumulation of error can be defined in terms of the micro-step size. This condition is called global Lipschitz continuity [80]. For the Forward Euler solver, the total (or global) error is $\mathcal{O}(h)$.

For a solver to be useful, it must be convergent, that is, the computed trace must coincide with the accurate trace when $h \to 0$ [252]. It means the error can be controlled by adjusting the micro step size h. The same concept of convergence applies to co-simulation but does, as the intuition suggests, decreasing the communication step size H lead to a more accurate co-simulation trace? This cannot be answered yet in general co-simulation because the behavior of the coupled model induced by the coupling of simulation units may not satisfy Lipschitz continuity.

In the context of CT co-simulation, according to Arnold and Günther [11], Arnold et al. [14], Busch and Schweizer [52], Hafner et al. [114], Kübler and Schiehlen [141], if the simulation units are convergent and the coupled model induced by the scenario coupling conditions can be written in the state space form of eq. (10), then the co-simulation unit induced by any of the Jacobi, Gauss-Seidel, or Strong coupling methods, is convergent, with any polynomial extrapolation technique for the inputs. Presence of algebraic loops, complex coupling constraints (such as the one shown in section 4.3.1), are factors that may make it impossible to write the coupled model in state space form. See Arnold [10] for more examples.

The local error vector, in a co-simulation, is defined as the deviation from the true trace after one co-simulation step H, starting from an accurate point.

$$\begin{array}{c} x_1(t+H) - \tilde{x}_1(t+H) \\ \cdots \\ x_n(t+H) - \tilde{x}_n(t+H) \\ y_1(t+H) - \tilde{y}_1(t+H) \\ \cdots \\ y_n(t+H) - \tilde{y}_n(t+H) \end{array} \tag{39}$$

where $\tilde{x}_i(t+H) = \delta_i(t, x_i(t), \phi_{u_i}(t))$, $\tilde{y}_i(t+H) = \lambda_i(t, \tilde{x}_i(t+H), \phi_{u_i}(t+H))$, and $x_i(t+H)$ and $y_i(t+H)$ are the true state vectors and outputs, respectively, for simulation unit S_i.

For a convergent co-simulation unit, some of the techniques used traditionally in simulation to *estimate* the error, have been applied in co-simulation:

Richardson extrapolation: This well-known technique is compatible with black-box simulation units as long as these provide rollback and state saving/restore capabilities [13, 15, 96]. The essential idea is to get an estimate of the local error by comparing $[\tilde{x}_i(t+H), \tilde{y}_i(t+H)]^T$ with a less accurate point $[\bar{x}_i(t+H), \bar{y}_i(t+H)]^T$. The less accurate point can be computed by the same orchestrator but using a larger communication step size. We have seen that larger communication step sizes affect the accuracy so if the two points are not too far apart, it means the communication step H does not need to be changed. It is importance to notice that the less accurate point $[\bar{x}_i(t+H), \bar{y}_i(t+H)]^T$ has to be computed from the accurate starting point $[\tilde{x}_i(t), \tilde{y}_i(t)]^T$.

Multi-Order Input Extrapolation: The outputs of two different order input approximation methods are compared [52, 54].

Milne's Device: Similar to the previous ones, but the extrapolation of the inputs is compared with its actual value, at the end of the co-simulation step. Iterative approaches such as the ones studied in Arnold [10], Arnold and Günther [11], Schweizer and Lu [214, 215, 216], Schweizer et al. [217, 218, 219] can readily benefit from this technique.

Parallel Embedded Method: This technique runs a traditional adaptive step size numerical method in parallel with the co-simulation [119]. The purpose is to piggy back in the auxiliary method, the decisions on the step size. The derivatives being integrated in each simulation unit have to be either provided, or estimated.

Conservation Laws: The local error is estimated based on the deviation from a known conservation law [207]. Extra domain knowledge about the coupling between simulation units is required. For example, if the couplings form power bonds [193], then energy should be conserved across a co-simulation step. In practice there is always an error due to the usual factors. The magnitude of the energy residual at a start and at end of a co-simulation step serves as an estimate of the local error. This technique has been implemented and studied in Sadjina et al. [207]. It has the advantage that it may not require rollback functionalities.

Embedded Solver Method: If the individual simulation units support adaptive step size, then the decisions made internally can be made public to help the orchestrator decide on the communication step size. To the best of our knowledge, there is no orchestrator proposed that performs this, but the FMI Standard allows simulation units to reject too large communication step sizes [35, 46].

After the error is deemed too large by one of the above methods, the correction can be applied pessimistically (rollback and repeating the same step) or optimistically (adapt the next step). To mitigate the overhead of a pessimistic approach, the corrections may be applied only to sensitive simulation units, as done in Verhoeven et al. [249].

Finally, the traditional simulation techniques can be applied to chose the right communication step size H: See Busch and Schweizer [52] for the PI-controller approach, and Gustafsson [112], Gustafsson et al. [113] for other techniques that can potentially be applied to co-simulation.

4.3.5 Compositional Stability

In the previous section we have presented conditions in which an orchestration engine can reduce the communication step size to an arbitrarily small value in order to meet arbitrary accuracy. Theoretically, this is useful as it tells the orchestrator that by reducing the local error, it also reduces the global error. In practice, the communication step size cannot be reduced to an arbitrarily small value without facing performance and roundoff error problems. Performance because, for smaller communication step sizes, it takes more steps to compute a behavior trace over a given interval of time. Round-off accuracy because in a digital computer, real numbers can only be represented approximately. Computations involving very small real numbers incur a non-negligible round-off error. So that means that in practice convergence does not imply that arbitrary accuracy can be achieved. A better question is to analyze what happens to the global error, as the co-simulation trace is computed with a non-null communication step size H.

Suppose that the analytical solution to the coupled model induced by the co-simulation scenario eventually goes to zero. This is the case for the coupled multi-body system of fig. 5, described in eq. (17), provided that at least one of the constants d_1 or d_2 is positive non-zero. Intuitively, this means that the system will lose energy over time, until it eventually comes to rest.

Let $x_1(t)$ denote the analytical solution of the position the mass m_1 in the system, and let $\tilde{x_1}(t)$ be the solution computed by a co-simulation unit. Then $e_{x_i}(t) = \|x_1(t) - \tilde{x_1}(t)\|$ denotes the global error at time t made by the co-simulation unit. If $\lim_{t\to\infty} x_1(t) = 0$, then $\lim_{t\to\infty} e_{x_i}(t) = \tilde{x_1}(t)$.

If the co-simulation unit is convergent, then for an arbitrarily small $H \to 0$, $\lim_{t\to\infty} e_{x_i}(t) \to 0$ will be arbitrarily small too. Since in practice we cannot take arbitrarily small H, we want to know whether there is some non-zero H such that $\lim_{t\to\infty} \tilde{x_1}(t) = 0$, thus driving $e_{x_i}(t)$ to zero as well. If that is the case, then it means that, assuming the system will eventually come to rest, the co-simulation unit will too. This property is called numerical stability.

Contrarily to convergence, numerical stability is a property that depends on the characteristics of the system being co-simulated. Numerical stability is always studied assuming that the system being co-simulated is stable. It makes no sense to show that the co-simulation trace will grow unbounded provided that the system does too. It is a comparison of two infinities. One of the ways numerical stability in co-simulation can be studied is by calculating the spectral radius of the error in the co-simulation unit, written as an autonomous linear discrete system [49, 51].

To give an example, recall that the coupled model induced by the co-simulation scenario described in eq. (24) can be written as:

$$
\begin{aligned}
\begin{bmatrix}\dot{x_1}\\ \dot{v_1}\end{bmatrix} &= \underbrace{\begin{bmatrix}0 & 1\\ -\frac{c_1}{m_1} & -\frac{d_1}{m_1}\end{bmatrix}}_{A_1}\begin{bmatrix}x_1\\ v_1\end{bmatrix} + \underbrace{\begin{bmatrix}0\\ \frac{1}{m_1}\end{bmatrix}}_{B_1} u_1 \\
y_1 &= \underbrace{\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}}_{C_1}\begin{bmatrix}x_1\\ v_1\end{bmatrix} \\
\begin{bmatrix}\dot{x_2}\\ \dot{v_2}\end{bmatrix} &= \underbrace{\begin{bmatrix}0 & 1\\ -\frac{c_2+c_c}{m_2} & -\frac{d_c}{m_2}\end{bmatrix}}_{A_2}\begin{bmatrix}x_2\\ v_2\end{bmatrix} + \underbrace{\begin{bmatrix}0 & 0\\ \frac{c_c}{m_2} & \frac{d_c}{m_2}\end{bmatrix}}_{B_2} u_2 \\
y_2 &= \underbrace{\begin{bmatrix}c_c & d_c\end{bmatrix}}_{C_2}\begin{bmatrix}x_2\\ v_2\end{bmatrix} + \underbrace{\begin{bmatrix}-c_c & -d_c\end{bmatrix}}_{D_2} u_2
\end{aligned}
\tag{40}
$$

with the coupling conditions $u_1 = y_2$ and $u_2 = y_1$.

In order to write the co-simulation model as an autonomous linear discrete system, we have to write what happens at a single co-simulation step $t \in [nH, (n+1)H]$ when executed by the orchestrator presented in Algorithm 3. Since the purpose is to analyze the stability of a co-simulation unit, and not the stability of each of the simulation units in the co-simulation, it is common to assume that the simulation units compute the analytical trace of the system. This enables the study of the stability properties of the co-simulation unit, starting from stable simulation units.

From time $t \in [nH, (n+1)H]$, simulation unit S_1 is computing the behavior trace of the following Initial Value Problem Ordinary Differential Equation (IVP-ODE):

$$
\begin{bmatrix}\dot{x_1}(t)\\ \dot{v_1}(t)\end{bmatrix} = A_1\begin{bmatrix}x_1(t)\\ v_1(t)\end{bmatrix} + B_1 u_1(nH) \tag{41}
$$

with initial conditions $\begin{bmatrix}x_1(nH) & v_1(nH)\end{bmatrix}^T$ given from the previous co-simulation step. The term $u_1(nH)$ denotes the fact that we are assuming a constant extrapolation of the input in the interval $t \in [nH, (n+1)H]$.

eq. (41) is linear and time invariant, so the value of $\begin{bmatrix}x_1((n+1)H)\\ v_1((n+1)H)\end{bmatrix}$ can be given analytically [49] as:

$$
\begin{bmatrix}x_1((n+1)H)\\ v_1((n+1)H)\end{bmatrix} = e^{A_1 H}\begin{bmatrix}x_1(nH)\\ v_1(nH)\end{bmatrix} + \left(\int_{nH}^{(n+1)H} e^{A_1((n+1)H-\tau)}d\tau\right) B_1 u_1(nH) \tag{42}
$$

or, replacing the integration variable with $s = \tau - nH$,

$$
\begin{bmatrix}x_1((n+1)H)\\ v_1((n+1)H)\end{bmatrix} = e^{A_1 H}\begin{bmatrix}x_1(nH)\\ v_1(nH)\end{bmatrix} + \underbrace{\left(\int_0^H e^{A_1(H-s)}ds\right)}_{K_1} B_1 u_1(nH) \tag{43}
$$

where $e^X = \sum_{k=0}^{\infty} \frac{1}{k!} X^k$ is the matrix exponential.

Rewriting eq. (43) as a discrete time system gives us the computation performed by simulation unit S_1 in a single co-simulation step, that is, the state transition function δ_1:

$$\begin{bmatrix} x_1^{(n+1)} \\ v_1^{(n+1)} \end{bmatrix} = e^{A_1 H} \begin{bmatrix} x_1^{(n)} \\ v_1^{(n)} \end{bmatrix} + K_1 B_1 u_1^{(n)} \tag{44}$$

where $z^{(n)} = z(nH)$.

At the end of the co-simulation step ($t = (n+1)H$) the output of the first simulation unit, that is, its output function λ_1, is given by plugging in eq. (44) to the output y_1 in eq. (40):

$$y_1^{(n+1)} = C_1 e^{A_1 H} \begin{bmatrix} x_1^{(n)} \\ v_1^{(n)} \end{bmatrix} + C_1 K_1 B_1 u_1^{(n)} \tag{45}$$

Repeating the same procedure for the second simulation unit, yields the state transition δ_2 and output functions λ_2:

$$\begin{aligned} \begin{bmatrix} x_2^{(n+1)} \\ v_2^{(n+1)} \end{bmatrix} &= e^{A_2 H} \begin{bmatrix} x_2^{(n)} \\ v_2^{(n)} \end{bmatrix} + K_2 B_2 u_2^{(n)} \\ y_2^{(n+1)} &= C_2 e^{A_2 H} \begin{bmatrix} x_2^{(n)} \\ v_2^{(n)} \end{bmatrix} + \left(C_2 K_2 B_2 + D_2 \right) u_2^{(n)} \end{aligned} \tag{46}$$

with $K_2 = \int_0^H e^{A_2 (H-u)} du$.

Since the coupling conditions are $u_1 = y_2$ and $u_2 = y_1$, we can combine Equations 46, 45, and 41 into a single discrete time system:

$$\begin{bmatrix} \begin{bmatrix} x_1^{(n+1)} \\ v_1^{(n+1)} \end{bmatrix} \\ y_1^{(n+1)} \\ \begin{bmatrix} x_2^{(n+1)} \\ v_2^{(n+1)} \end{bmatrix} \\ y_2^{(n+1)} \end{bmatrix} = \underbrace{\begin{bmatrix} e^{A_1 H} & \bar{0} & \bar{0} & K_1 B_1 \\ C_1 e^{A_1 H} & \bar{0} & \bar{0} & C_1 K_1 B_1 \\ \bar{0} & K_2 B_2 & e^{A_2 H} & \bar{0} \\ \bar{0} & C_2 K_2 B_2 + D_2 & C_2 e^{A_2 H} & \bar{0} \end{bmatrix}}_{A} \begin{bmatrix} \begin{bmatrix} x_1^{(n)} \\ v_1^{(n)} \end{bmatrix} \\ y_1^{(n)} \\ \begin{bmatrix} x_2^{(n)} \\ v_2^{(n)} \end{bmatrix} \\ y_2^{(n)} \end{bmatrix} \tag{47}$$

The above system is stable if the behavior traces remain bounded (e.g., by going to zero) as $n \to \infty$. This can be checked by observing whether the spectral radius $\rho(A) < 1$. For parameters $m_1 = m_2 = c_1 = c_2 = d_1 = c_c = d_c = 1, d_2 = 2$, a communication step size of $H = 0.001$, $\rho(A) = 0.9992$, which means that the co-simulation unit is stable. If the damping constant were $d_k = 6.0E6$, then the co-simulation unit would be unstable ($\rho(A) \approx 76.43$). A stable co-simulation is shown in fig. 10.

Different coupling methods, and different approximation functions yield different stability properties. See Busch [49, 50], Busch and Schweizer [51, 53] for the stability analysis of multiple coupling approaches and approximating functions. Stability of various co-simulation units has been also studied in Arnold [10], Gu and Asada [105], Kalmar-Nagy and Stanciulescu [128], Kübler and Schiehlen [142], Schweizer et al. [217]. The *rules of thumb* drawn from these papers can be summarized as:

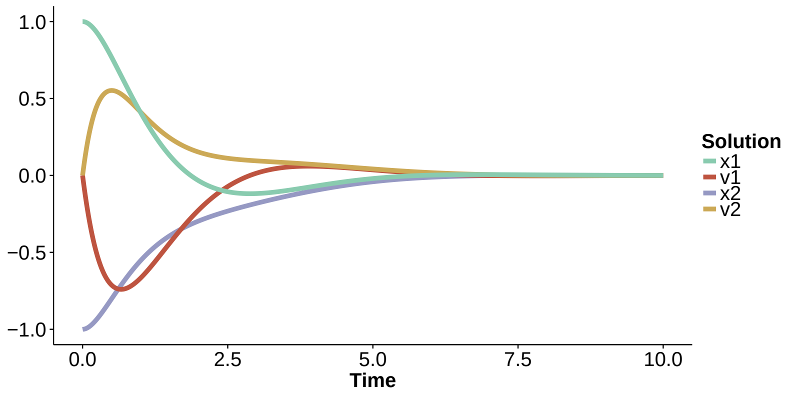


Figure 10: Behavior trace of co-simulator described in eq. (47). Parameters are: $m_1 = m_2 = c_1 = c_2 = d_1 = c_c = d_c = 1, d_2 = 2, H = 0.001$.

- Co-simulators that employ fixed point iteration techniques typically have better stability properties;
- Gauss-Seidel coupling approach has slightly better stability properties when the order in which the simulation units compute is appropriate. For example, the simulation unit with the highest mass should be computed first [10];

The main problem is that in co-simulation applied to industrial problems, the solvers and models may be coupled in a black box to protect IP, so there is little knowledge about the kind of solver and model being used and its stability properties. The best is then to always use iterative techniques that have been shown to have better stability properties. However, these techniques require rollback functionalities which can be difficult to support for certain simulation units. Even if those functionalities are available, the cost of computing a co-simulation trace can be prohibitively high when compared with non-iterative approaches. This creates a paradox where industrial co-simulation units should make use of iterative techniques but the performance toll may be too high.

4.3.6 Compositional Continuity

Let us for now assume that each CT simulation unit is a mock-up of a continuous system. A prerequisite is that the physical laws of continuity are obeyed. When using extrapolation in the inputs (e.g., constant extrapolation), these laws may not be obeyed, as discussed in Busch [50], Sadjina et al. [207]. Consider the point of view of a simulation unit S_i in co-simulation. Throughout a co-simulation step $t \in [nH, (n+1)H]$ the input $\phi_{u_i}(t, u_i(nH)) = u_i(nH)$ is kept constant. At the next co-simulation step $t \in [(n+1)H, (n+2)H]$, the input $\phi_{u_i}(t, u_i((n+1)H)) = u_i((n+1)H)$ may change radically if $u_i((n+1)H)$ is too far away from $u_i(nH)$.

The discontinuities in the inputs may wreak havoc in the performance of the simulation unit S_i, causing it to reduce inappropriately the micro step size, to reinitialize the solver [63], to discard useful information about the past (in multi-step solvers [8, 9]), and/or produce inaccurate values in its input extrapolation [191]. Furthermore, a discontinuity may be propagated to other simulation units, aggravating the problem.

Most numerical methods assume that the input is a discretized version of a continuous trace.

That means that, when a discontinuity occurs, simulation unit S_i cannot distinguish it from a very steep change in the continuous trace. The way traditional solvers deal with this behavior is to reduce the micro step size h_i until the change is not so steep. This works with a continuous signal with a steep change, but does not work with a discontinuity: even if the micro-step size h_i is reduced, the difference between $\lim_{t \to ((n+1)H)^-} \phi_{u_i}(t, u_i(nH)) = u_i(nH)$ and $\lim_{t \to ((n+1)H)^+} \phi_{u_i}(t, u_i((n+1)H)) = u_i((n+1)H)$ is still the same, as it depends on the communication step size H and not on the micro step size h_i. The solver will reduce the micro step size until a minimum is reached, at which point it gives up and finally advantages the micro step [63].

Most of the times this gives acceptable results but has a huge performance toll: when the solver is repeatedly retrying a small micro-step size, it does not advance the simulated time. This means that a huge computational effort goes to waste until the solver finally gives up [61].

We defer the discussion of the correct ways to deal with discontinuities to co-simulation scenario where discontinuities are welcome, section 5. In continuous co-simulation scenarios, discontinuities should not occur.

A solution to avoid discontinuities in the input approximations is to use the extrapolated interpolation methods instead of constant extrapolation [49, 50, 75]. These methods ensure at least that $\lim_{t \to ((n+1)H)^-} \phi_{u_i}(t, u_i(nH)) = \lim_{t \to ((n+1)H)^+} \phi_{u_i}(t, u_i((n+1)H))$.

To give an example, we derive one possible linear extrapolated interpolation method for ϕ_{u_i} over the interval $t \in [nH, (n+1)H]$. Since ϕ_{u_i} is linear, then $\phi_{u_i}(t, u_i(nH), u_i((n-1)H)) = b + a(t - nH)$, for some constants a, b. Let $\bar{u}_i(nH) = \phi_{u_i}(nH, u_i((n-1)H), u_i((n-2)H))$. To avoid discontinuities, we require that $\phi_{u_i}(nH, u_i(nH), u_i((n-1)H)) = \bar{u}_i(nH)$. And we want that $\phi_{u_i}((n+1)H, u_i(nH), u_i((n-1)H)) = u_i(nH)$.

So putting these constraints together gives

$$\begin{aligned} \phi_{u_i}(t, u_i(nH), u_i((n-1)H)) &= b + a(t - nH) \\ \bar{u}_i(nH) &= \phi_{u_i}(nH, u_i((n-1)H), u_i((n-2)H)) \\ \phi_{u_i}(nH, u_i(nH), u_i((n-1)H)) &= \bar{u}_i(nH) \\ \phi_{u_i}((n+1)H, u_i(nH), u_i((n-1)H)) &= u_i(nH) \end{aligned} \tag{48}$$

Solving this system for $\phi_{u_i}(t, u_i(nH), u_i((n-1)H))$ gives:

$$\phi_{u_i}(t, u_i(nH), u_i((n-1)H)) = u_i((n-1)H) + \frac{u_i(nH) - u_i((n-1)H)}{H}(t - nH) \tag{49}$$

4.3.7 Real-time Constraints

As introduced in section 2, the major challenge in real-time simulation is to ensure that a simulation unit is fast-enough to satisfy the timing constraint $t = \alpha\tau$. In real-time co-simulation, this challenge gets aggravated due to the presence of multiple simulation units, with different capabilities [222]. In order to enable real-time co-simulation, every simulation unit has to be fast enough. Furthermore, real-time co-simulation is often needed because one of the simulation unit is actually the original system, wrapped as a simulation unit. This means that measurements are performed to the state of the system, and this means noise in the signals. Therefore, the extrapolation functions used in the other simulation units have to be properly protected from the noise in the signal, using statistical techniques such as Kalman filtering [127]. Finally, the quality of the network is important, as the real-time simulation units needs to receive their inputs in a timely manner. To mitigate this, the orchestration algorithm has to compensate for any delays in the receiving of data, and provide inputs to the real-time simulation unit [221].

5 Hybrid Co-simulation Approach

Sections 3 and 4 described the essential characteristics and assumptions of simulation units for each kind of co-simulation approach. When compared to a CT unit, whose state evolves continuously in time and whose output may have to obey to physical laws of continuity, a DE unit state can assume multiple values at the same time (transiency) and its output can dramatically change over a short period of time. For an orchestrator, a CT unit has some flexibility (safe for algebraic loops and ugly coupling conditions) in computing the co-simulation. In contrast, a DE simulation unit has to get inputs and produce outputs at the precise time some event occurs. And due to the potentially drastic change in the outputs, there is no Lipschitz continuous condition that allows predicting how a delay in the output of the DE unit can affect the overall co-simulation trace.

For example, in the simulation unit of the mass-spring-damper system, eq. (22), with a constant extrapolation function, and running under the orchestrator in Algorithm 3, the change in the input can only affect the output after at least H units of time. For continuous time solvers in general, as can be seen for the explicit solver in eq. (12), a delayed response to the inputs is normal.

These differences between CT and DE units are at the heart of many challenges in hybrid co-simulation scenarios, mixing the two.

5.1 Hybrid Co-simulation Scenarios

We do not give a formal definition of a hybrid co-simulation scenarios because that is related to finding an appropriate standard for hybrid co-simulation, which is a non trivial challenge (see section 5.2.9) [47].

Instead, we define it broadly as mixing the characteristics and assumptions of both kinds of simulation units. These scenarios, together with an adequate orchestrator, can be used as mock-ups of hybrid systems [6, 57, 60, 162]. A thermostat regulating the temperature in a room is a classical example [161]. The continuous constituent system represents the temperature dynamics of the room, accounting for a source of heat (radiator). The discrete event part is a controller that turns on/off the radiator depending on the temperature.

The continuous time simulation unit S_1 simulates the following dynamics:

$$\begin{aligned} \dot{x} &= -\alpha\left(x - 30q\right) \\ x(0) &= x_0 \end{aligned} \tag{50}$$

where x is the output temperature in the room, $\alpha > 0$ denotes how fast the room can be heated (or cooled) down, and $q \in \{0, 1\}$ is the control input that turns on/off the radiator. The discrete event simulation unit S_2 simulates the statemachine shown in fig. 11, where one can think of the input event *tooHot* as happening when $x(t) \geq 21$ and *tooCold* when $x(t) \leq 19$. The output events *off* and *on* will assign the appropriate value to the input q of S_1. Therefore, the temperature $x(t)$ is kept within a comfort region.

Clearly, the two units cannot just be coupled together via input to output assignments. Any orchestrator for this co-simulation scenario has to reconcile the different assumptions about the inputs and output of each simulation unit.

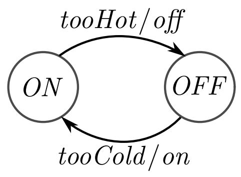


Figure 11: Statemachine model of the controller constituent system.

- The CT simulation unit expects a continuous input, whereas the output of the DE simulation unit is an event signal.
- The output of the CT simulation unit is a continuous signal, whereas the DE simulation units expects an event signal as input.

The coupling of continuous time and discrete event black box simulation units has been studied in the state of the art. In essence, two approaches are known, both based on creating a wrapper component around a simulation unit to adapt its behavior:
Hybrid DE – wrap every CT unit as a DE simulation unit, and use a DE based orchestration;
Hybrid CT – wrap every DE unit to become a CT unit and use a CT based orchestrator.

According to the formalization that we have proposed for CT and DE simulation units, the *Hybrid DE* approach, applied to the thermostat example may involve: wrapping S_1 as a DE simulation unit, S_1', with a time advance that matches the size of the co-simulation step; and keeping track of the output of S_1 in order to produce an output event whenever it crosses the thresholds. Conversely, any output event from S_2 has to be converted into a continuous signal for the input $q(t)$ of S_1.

Other examples of *Hybrid DE* are described in Awais et al. [19], Bolduc and Vangheluwe [37, 38], Camus et al. [55, 56], Fey et al. [84], Kofman and Junco [137], Kounev et al. [139], Kuhr et al. [146], Neema et al. [182], Nutaro [189], Quesnel et al. [202], Vangheluwe [244], Widl et al. [254], Yılmaz et al. [257], Zeigler [260].

The *Hybrid CT*, in our example, can be followed by wrapping the DE unit S_2 as a CT unit that takes as input the temperature continuous signal, and internally reacts to an event caused by the crossing of the threshold. Conversely, the output event of S_2 can be converted into a continuous signal $q(t)$.

Examples of the *Hybrid CT* include Denil et al. [70], Feldman et al. [83], Garro and Falcone [97], Lawrence et al. [150], Quaglia et al. [201], Tavella et al. [226], Tripakis [232].

Regardless of the approach taken, the properties of the constituent systems have to be retained: the fact that an otherwise discontinuous signal becomes continuous as a result of a linear or higher order extrapolation may not respect the properties of the coupled system. Knowledge of the domain and the simulation units is paramount.

A third alternative, compared to only using Hybrid CT or Hybrid DE, is to have different mechanisms of orchestrating the simulation units depending on the semantic domain. For instance, in the actor modeling language Ptolemy II [200], an actor has many similarities to a simulation unit. Instead of using either Hybrid CT or Hybrid DE, a so called *Director* block is used for a particular set of connected actors. In this context, the notion of superdense time is fundamental, as also discussed in [47].

In the subsection below, different issues that arise in hybrid co-simulation will be described. These should be read in the light of hierarchical hybrid co-simulation scenarios, where compositionality is important.

5.2 Challenges

5.2.1 Semantic Adaptation

While a generic wrapper based on the underlying model of computation of the simulation unit can be used, as done in [68, 200], the realization of any of the approaches *Hybrid DE* or *Hybrid CT* depends on the concrete co-simulation scenario and the features of the simulation units [42, 178], as shown with the thermostat example. There is simply no best choice of wrappers for all scenarios. Even at the technical level, the manner in which the events or signals are sent to (or obtained from) the unit may need to be adapted [232]. To be concrete, the simulation unit S_2 can assume that all events are communicated by encoding them in a single string signal, as opposed to having a different signal signal to denote different events. To account for this variability, the most common adaptations can be captured in a configuration language, as was done in Denil et al. [70], Meyers et al. [169], or in a specialization of a model of computation, as done in Kuhr et al. [146], Muller and Widl [176], Pedersen et al. [195]. These approaches require that a person with the domain knowledge describes how the simulation units can be adapted.

Our choice of wrapper for the *Hybrid DE* approach is meant to highlight another problem with the adaptations of simulation units: the wrapper incorporates information that will ultimately have to be encoded in the software controller. As such, we argue that the need for sophisticated semantic adaptations should be smaller in later stages of the development of the components so that, for more refined models of the thermostat, the decision about when to turn off the radiator is not made by a wrapper of S_1.

5.2.2 Predictive Step Sizes

In the *Hybrid DE* approach, the time advance has to be defined (recall eq. (1)). Setting it to whatever co-simulation step size H the orchestrator decides will work, but the adapted simulation unit may produce many absent output events. Better adaptations have been proposed. In the thermostat example, S_1' can propose a time advance that coincides with the moment that $x(t)$ will leave the comfort region, thereby always being simulated at the relevant times.

Naturally, these approaches rely in information that may expose the IP of simulation units. Others try to adaptively guess the right time advance by monitoring other conditions of interest, set over the own dynamics of the adapted simulation unit, the most common approach being the quantization of the output space [38, 136, 137, 190, 261].

The capability to predict the time advance is also useful to enhance the performance/accuracy of CT based co-simulation, as shown in Broman et al. [46].

5.2.3 Event Location

Locating the exact time at which a continuous signal crosses a threshold is a well known problem [39, 41, 263] and intimately related to guessing the right time advance for predicting the step size [56, 96]. To address this, solutions typically require derivative information of the signal that causes the event, and/or the capability to perform rollbacks. In the thermostat example, a co-simulation that shows the output q of the controller changing from 0 to 1 at time t_e while the temperature of the room x actually crossed the confort zone at $t_e - k$, for $k > 0$, may not be accurate if k is too large. Note that k is a consequence of the decisions made in the orchestrator.

5.2.4 Discontinuity Identification

Until here, we have based our discussion in the knowledge of what kind of simulation units comprise a co-simulation. In a general hierarchical co-simulation, a simulation unit's output may be an event signal coming from a wrapper of a CT unit, or vice-versa. In any case, at runtime, a signal is often represented as a set of time-stamped points. Observing this sequence of points alone does not make it possible to discern a steep change in a continuous signal, from a true discontinuity, that occurs in an event signal [47, 154, 172, 263]. Extra information is currently used: *a*) a formalization of time which include the notion of absent signal, as proposed in Broman et al. [47], Lee and Zheng [154], Tavella et al. [226]; or *b*) an extra signal can be used to discern when a discontinuity occurs, as done in the FMI for Model Exchange [35], even facilitating the location of the exact time of the discontinuity; or *c*) symbolic information (e.g., Dirac impulses [72]) that characterize a discontinuity can be included, as done in Nilsson [186]

5.2.5 Discontinuity Handling

Once a discontinuity is located, how it is handled depends on the nature of the simulation units and their capabilities. If the simulation unit is a mock-up of a continuous system then, traditionally, discontinuities in the inputs should be handled by reinitializing the simulation unit [63]. This step can incur a too high performance cost, especially with multi-step numerical methods, and Andersson [8], Andersson et al. [9] proposes an improvement for these solvers. Furthermore, a discontinuity can cause other discontinuities, producing a cascade of re-initializations. During this process, which may not finish, care must be taken to ensure that physically meaningful properties such as energy distribution, are respected [174].

5.2.6 Algebraic Loops, Legitimacy, and Zeno Behavior

Algebraic loops are non-causal dependencies between simulation units that can be detected using feedthrough information, as explained in section 4.3.2. In CT based co-simulation, the solution to algebraic loops can be attained by a fixed point iteration technique, as covered in section 4.3.2. There is the possibility that the solution to an algebraic loop will fail to converge. The result is that, if left unchecked, the orchestrator would move an infinite number of input and output values between simulation units, at the same point in time.

In DE based co-simulation a related property is legitimacy [262], which is the undesirable version of the *transiency* property, explained in section 3. A illegitimate co-simulation scenario will cause the co-simulation orchestrator to move an infinite number of events with the same timestamp between units, never advancing time. Distance matrices, used to optimize parallel optimistic approaches, as explained in Fujimoto [94] and used in Ghosh et al. [101], can be leveraged to detect statically the presence of *some* classes of illegitimacy.

A similar behavior, but more difficult to detect is Zeno behavior. It occurs when there is an increasingly small interval of time between two consecutive events, up to the point that the the sum of all these intervals is finite [238]. However, while illegitimate behaviors are not desired in pure DE co-simulation, at least in the theoretical sense, Zenoness is a desired feature in some hybrid co-simulation scenarios. We say in the theoretical sense because, for the purposes of co-simulation, scenarios with Zenoness still have to be recognized and appropriate measures, such as regularization [125], have to be taken.

5.2.7 Stability

If a hybrid co-simulation represents a hybrid or switched system [126, 238], then it is possible that a particular sequence of events cases the the system to become unstable, even if all the individual continuous modes of operation are stable. New analyses are required to identify whether the CT units can yield unstable trajectories as a result of the events of wrapped DE simulation units, while keeping the IP hidden.

5.2.8 Theory of DE Approximated States

In a pure DE based co-simulation, if round-off errors are neglected, the computed trajectories are essentially exact. To the best of our knowledge, only Zeigler et al. [262] addresses theoretically how the error in a discrete event system can be propagated. In CT based co-simulation however, error is an accepted and well studied and techniques exist to control it.

In Hybrid co-simulation, there is a need for analysis techniques that provide bounds on the error propagation in the DE simulation units, when these are coupled to sources of error.

In addition, based on these analyzes, it should be possible for a DE simulation unit to recognize that its error has exceeded a given tolerance, and measures should be taken to reduce that error. Having these techniques in place allows a hybrid co-simulation orchestrator to take appropriate measures (e.g., adapt the communication step size, etc...) the keep the error bounded in every simulation unit.

5.2.9 Standards for Hybrid Co-simulation

While for CT co-simulation there is the Functional Mock-up Interface (FMI) standard [35], and for DE co-simulation there is the High Level Architecture (HLA) [2] standard, as of the time of writing, both standards have limitations for hybrid co-simulation. Bogomolov et al. [36], Garro and Falcone [97], Tavella et al. [226] use/propose extensions to the FMI standard and Awais et al. [18] proposes techniques to perform CT simulation conforming to HLA. Recognizing that hybrid co-simulation is far from well studied, Broman et al. [47] proposes a set of idealized test cases that any hybrid co-simulation unit, and underlying standard, should pass. In particular, it is important to have correct handling and representation of time, to achieve a sound approach for simultaneity.

Finally, even with a standardized interface, simulation units are not all equal: a fact that makes coding an orchestration algorithm a real challenge. A possible approach to deal with this heterogeneity, proposed in Gomes [102], is to assume that all units implement the same set of features, code the orchestration algorithm for those features, and delegate to wrappers the responsibility of leveraging extra features (or mitigating the lack of). In the section below, these features are classified.

6 Classification

Having described the multiple facets of co-simulation, this section summarizes our classification and methodology.

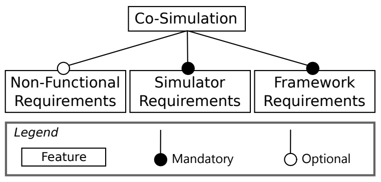


Figure 12: Top-level.

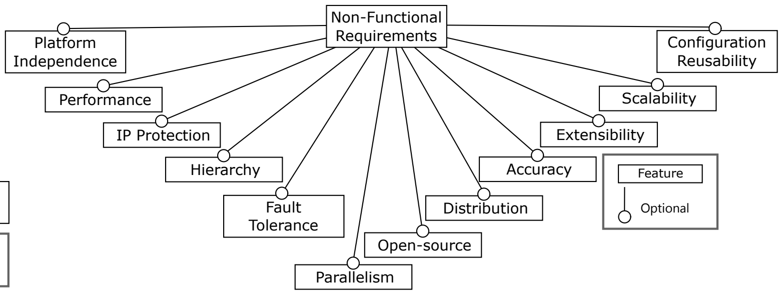


Figure 13: Non-Functional Requirements.

6.1 Methodology

To find an initial set of papers related to co-simulation, we used Google Scholar with the keywords "co-simulation", "cosimulation", "coupled simulation", and collected the first 10 pages of papers. Every paper was then filtered by the abstract, read in detail, and its references collected. To guide our reading to the most influential papers, we gave higher priority to most cited (from the papers that we have collected).

We read approximately 30 papers to create the initial version of the taxonomy. Then, as we read new papers, we constantly revised the taxonomy and classified them.

After a while, new references did not cause revisions to the taxonomy, which prompted us to classify the collected papers in a more systematic fashion: all the papers that we collected from 2011 (inclusive) up to, and including, 2016 were classified. Two main reasons justify the last 5 years interval: limited time; and most of the papers refer to, and are based on, prior work. As a consequence, the classification would be very similar for many of the related references prior to 2011.

From the papers classified, those that report case studies where noted to create fig. 1.

6.2 Taxonomy

The taxonomy is represented as a feature model [129] structured in three main categories, shown in fig. 12:

Non-Functional Requirements (NFRs): Groups concerns (e.g., performance, accuracy, and IP Protection) that the reference addresses.

Simulation unit Requirements (SRs): Features required/assumed from the simulation units by the orchestrator described in the paper. Examples: Information exposed, causality, local/remote availability, or rollback support.

Framework Requirements (FRs): Features provided by the orchestrator. Examples: dynamic structure, adaptive communication step size, or strong coupling support.

Each main group is detailed in Figures 13, 14, and 15. Abstract features denote concepts that can be easily detailed down but we chose not to, for the sake of brevity. Mandatory features are required for the activity of co-simulation while optional are not.

6.3 State of the Art

To give an example on how the taxonomy is used, consider the work in Van Acker et al. [236], where an FMI based multi-rate orchestration algorithm is generated from a description of the co-simulation scenario. In the paper, the description language introduced can be reused in a tool-agnostic manner. The orchestration code generator analyzes the co-simulation scenario, and: a) identifies algebraic loops using I/O feedthrough information; b) separates the fast moving simulation units from the slow moving ones, using the preferred step size information, and provides interpolation to the fast ones (multi-rate); and c) finds the largest communication step size that divides all step sizes suggested by simulation units and uses it throughout the whole co-simulation. We argue that the generated orchestrator is fast because all the decisions are made at code generation stage. The algebraic loops are solved via successive substitution of inputs, storing and restoring the state of the simulation units.

Based on these facts, Van Acker et al. [236] can be classified as follows:

Non-Functional requirements: Performance, IP protection, and Configuration reusability;

Simulation unit requirements: Causal simulation units, Locally available simulation units, Preferred step size information about the solver, Feedthrough I/O causality information, State values, No time constraints, and No rollback support;

Framework requirements: Continuous time domain, Multi-rate simulation, Fixed communication step size, Input/Output coupling, Fully implicit strong coupling support, Postmortem visualization of results, Postmortem visualization of results, FMI as the underlying standard, Gauss-seidel communication approach, and Support for three or more simulation units.

With similar reasoning, the FMI standard for co-simulation, version 2.0, can be classified according to the assumptions it makes about the participating simulation units. These are highlighted in fig. 14.

The remaining state of the art is classified in Figures 16 – 19. The raw data is available online[13].

6.4 Discussion

Observing fig. 16, Accuracy is the most observed NFR, with 31 reports, followed by IP protection and Performance. The least observed NFRs are Fault tolerance, Hierarchy and Extensibility.

Fault tolerance is especially important for long running co-simulations. One of the industrial partners of the INTO-CPS project has running co-simulations that takes a minimum of two weeks to complete.

We argue that Extensibility (the ability to easily accomodate new features) should be given more importance: if an heterogeneous set of simulation units participate in the same co-simulation scenario, the combination of capabilities provided (see fig. 14) can be huge. Thus, the orchestrator can either assume a common homogeneous set of capabilities, which is the most common approach, or can leverage the capabilities provided by each one. The later approach can lead to an extremely complex orchestration algorithm. In any case, extensibility is key to address new semantic adaptations (recall section 5.2.1).

As fig. 18 suggests, we could not find approaches that make use of the nominal values of state and output variables, even though these are capabilities supported in the FMI Standard (see fig. 14), and are useful to detect co-simulations that are not valid. A-causal approaches are important for modularity, as explained in section 4.3.1, but these are scarce too.

[13]`http://msdl.cs.mcgill.ca/people/claudio/pub/Gomes2016bClassificationRawData/raw_data.zip`

As for the framework requirements, in fig. 19, the least observed features are dynamic structure co-simulation, interactive visualization, multi-rate, algebraic coupling, and partial/full strong coupling support. This can be explained by the fact that these features depend upon the capabilities of the simulation units, which may not be mature.

Figures 16 – 19 do not tell the full story because they isolate each feature. Feature interaction is a common phenomenon, and among many possible interactions, we highlight the accuracy concern, domain of the co-simulation, number of simulation units supported, and IP protection. As can be seen from fig. 21, there is only one approach [146] that is both CT and DE based, up to any number of simulation units. Note that this does not mean that the work addresses all challenges that were identified in section 5. Accommodating the different CT and DE domains means that the approach assumes that the simulation units can behave both as a CT and as a DE unit.

The concern with IP protection is evident in fig. 16 but the number of DE and CT based approaches that provide some support for it is small, as shown in fig. 20. Similarly, as fig. 22 suggests, accuracy does not show up a lot in the DE and CT approaches, for more than two simulation units. Accuracy is particularly important in interactions between DE and CT simulation units.

In general, from the observed classification, there is a lack of research into approaches that are both DE and CT based, and that leverage the extra the features from the simulation units.

7 Concluding Remarks

As this work shows, there are many interesting challenges to be explored in co-simulation, which will play a key role in enabling the virtual development of complex heterogeneous systems in the decades to come. The early success can be attributed to a large number of reported applications. However, from the application references covered (see fig. 1), the large majority represent *ad-hoc* couplings between two simulators of two different domains (e.g., a network simulator with a power grid one, or a HVAC simulator with a building envelop one). This, however, excludes the (potentially many) unreported applications of co-simulation. As systems become complex, the demand for co-simulation scenarios that are large, hierarchical, heterogeneous, accurate, IP protected, and so on, will increase.

This survey presents an attempt at covering the main challenges in co-simulation. To tackle such a broad topic, we have covered the two main domains —continuous time and discrete event based co-simulation— separately and then surveyed the challenges that arise when the two domains are combined. A taxonomy is proposed and a classification of the works related to co-simulation in the last five years is carried out using that taxonomy.

From the challenges, we highlight semantic adaptation, modular coupling, stability and accuracy, and finding a standard for hybrid co-simulation as being particular important.

For early system analysis, the adaptations required to combine simulators from different formalisms, even conforming to the same standard, are very difficult to generalize to any co-simulation scenario. A possible work around this is to allow the system integrator to describe these, as proposed in [70].

One of the main conclusions of the classification is that there is lack of research into modular, stable and accurate coupling of simulators in dynamic structure scenarios. This is where a-causal approaches for co-simulation can play a key role. The use of bi-directional effort/flow ports can be a solution inspired by Bond-graphs [193], and there is some work in [207] already in this direction.

Finally, this document is an attempt to summarize, bridge, and enhance the future research in co-simulation, wherever it may lead us to.

Acknowledgment

The authors wish to thank Yentl Van Tendeloo, for the in depth review of DE based co-simulation, Kenneth Guldbrandt Lausdahl for providing valuable input and discussions throughout the making of this survey, and TWT GmbH for the valuable input on everyday challenges faced by a co-simulation master. This research was partially supported by Flanders Make vzw, the strategic research centre for the manufacturing industry, and a PhD fellowship grant from the Agency for Innovation by Science and Technology in Flanders (IWT). In addition, the work presented here is partially supported by the INTO-CPS project funded by the European Commission's Horizon 2020 programme under grant agreement number 664047. This project is financially supported by the Swedish Foundation for Strategic Research.

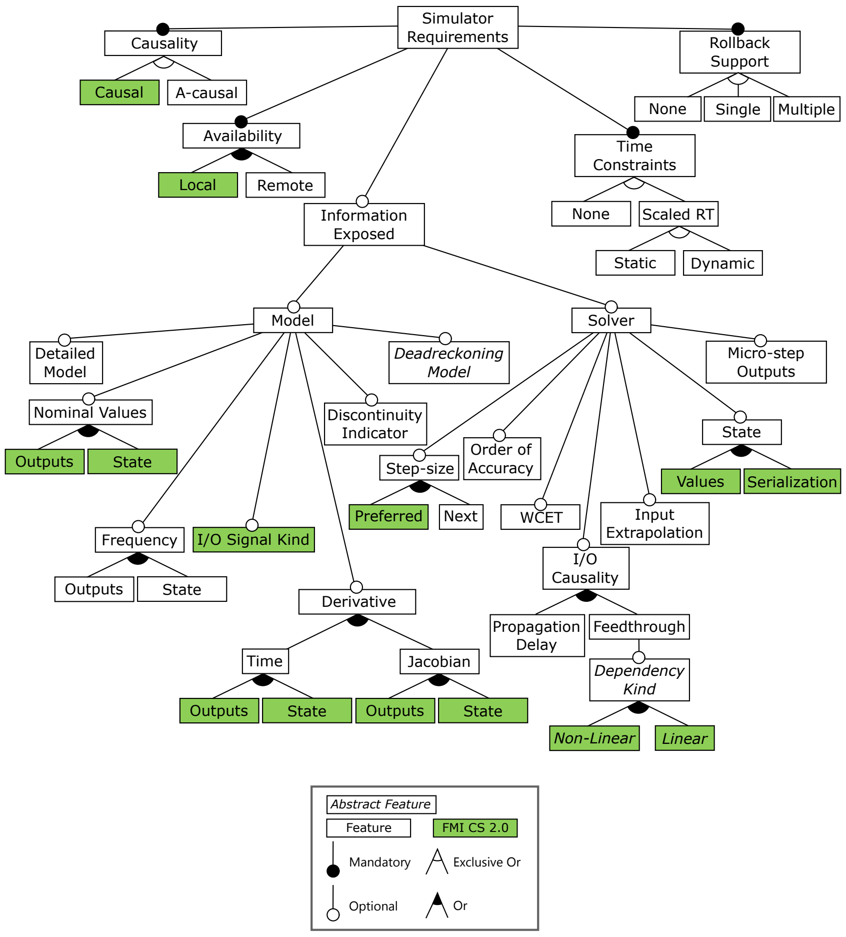


Figure 14: Simulation Unit Requirements and features provided in the FMI Standard for co-simulation, version 2.0.

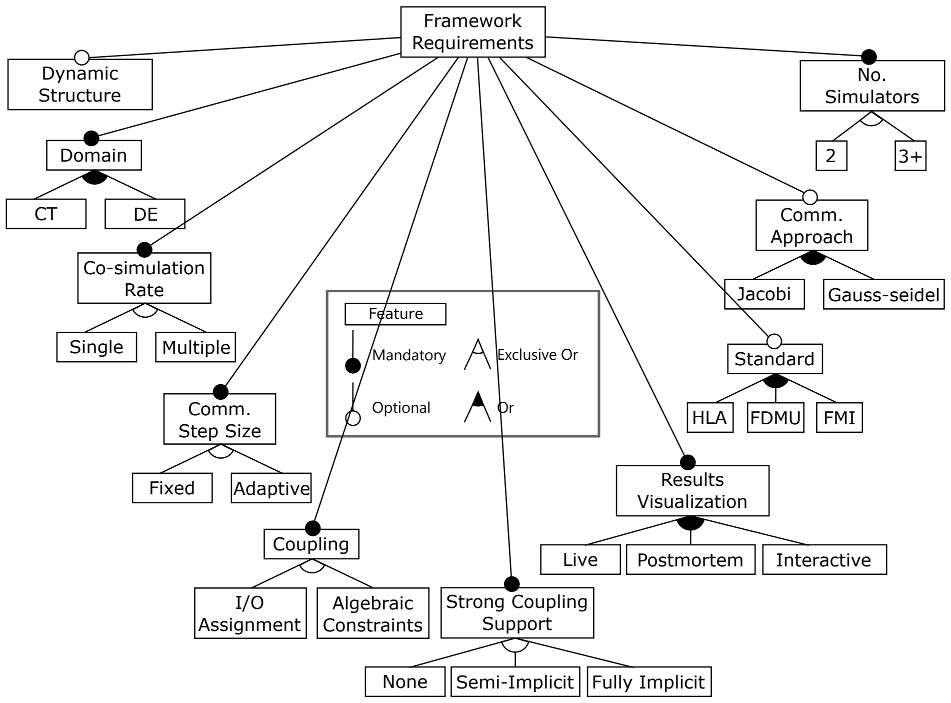


Figure 15: Framework Requirements.

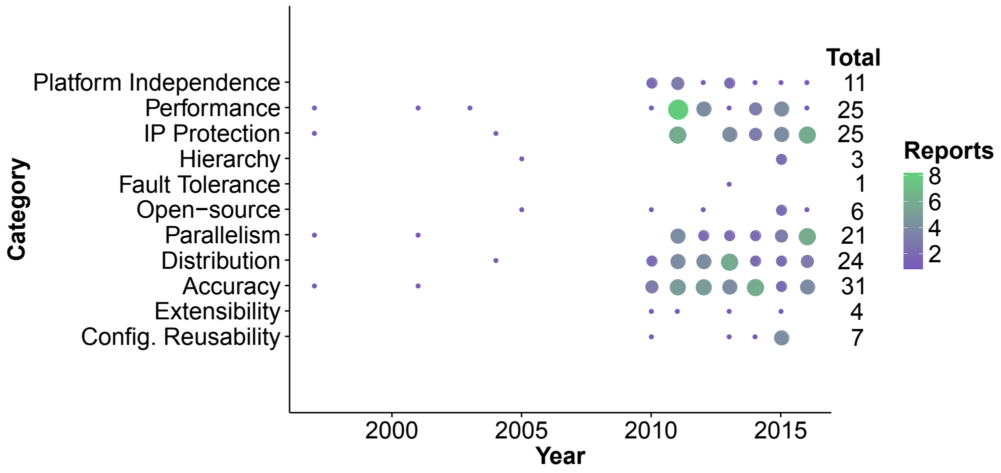


Figure 16: Classification with respect to non-functional requirements.

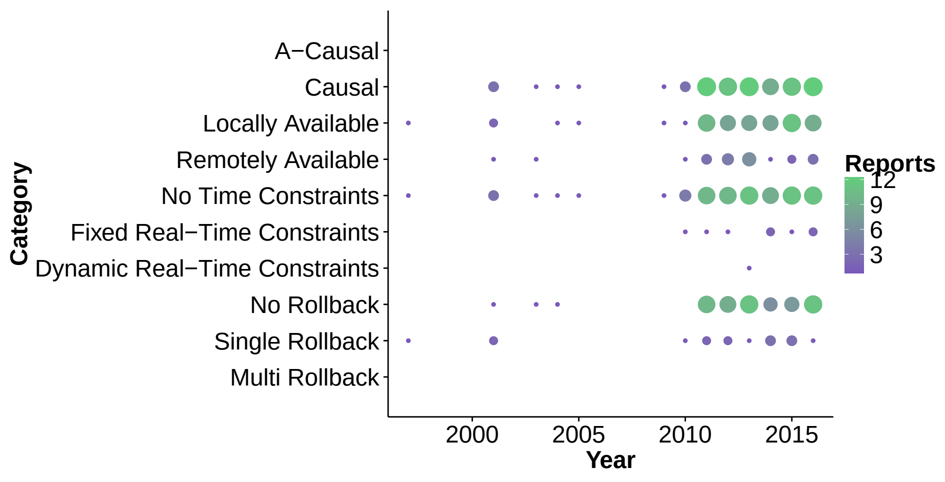


Figure 17: Classification with respect to simulation unit requirements: execution capabilities.

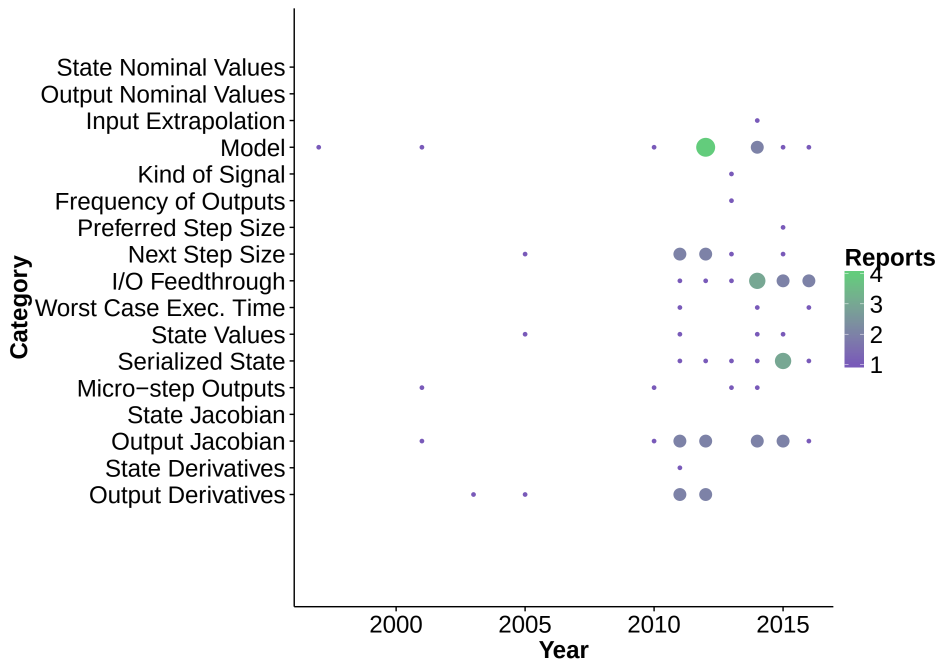


Figure 18: Classification with respect to simulation unit requirements: information exposed.

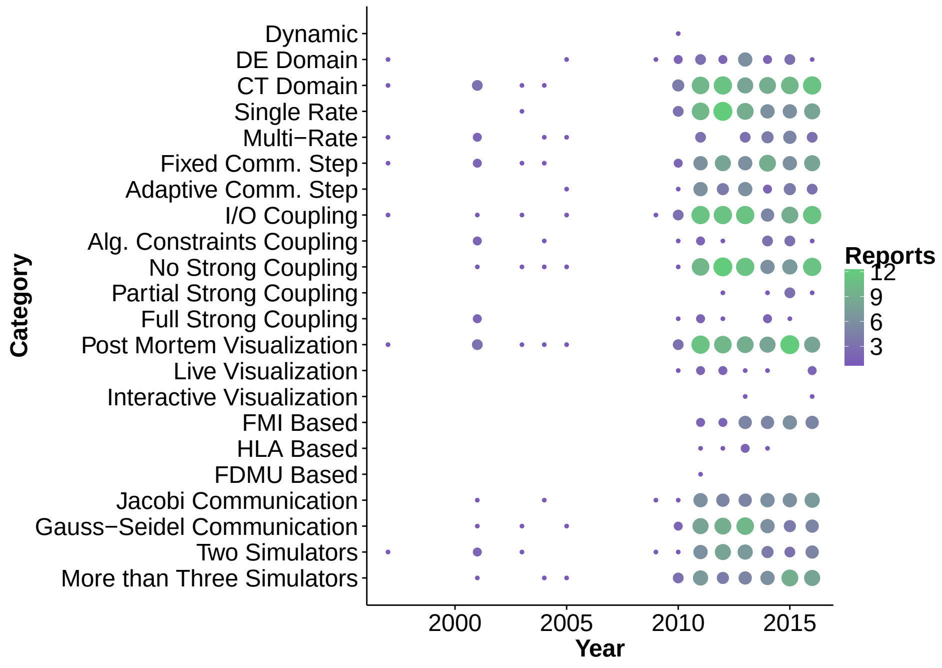


Figure 19: Classification with respect to framework requirements.

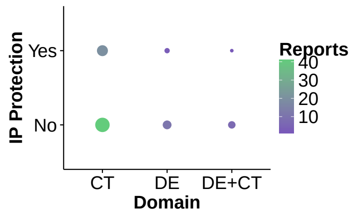


Figure 20: Formalisms vs IP Protection.

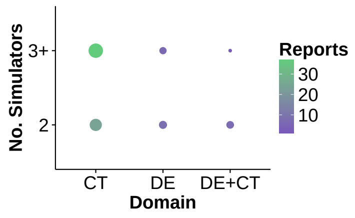


Figure 21: Formalisms vs Simulation units.

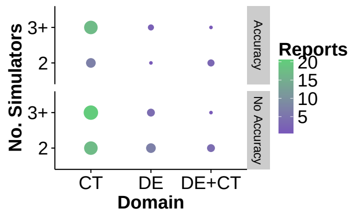


Figure 22: Accuracy vs Formalisms vs Simulation units.

References

[1] Modelica - A Unified Object-Oriented Language for Physical Systems Modeling, 2007. URL `https://www.modelica.org/documents/ModelicaSpec30.pdf`.

[2] IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Interface Specification, 2010. URL `https://standards.ieee.org/findstds/standard/1516-2010.html`.

[3] An-jelo Gian C. Abad, Lady Mari Faeda G. Guerrero, Jasper Kendall M. Ignacio, Dianne C Magtibay, Mark Angelo C Purio, and Evelyn Q Raguindin. A simulation of a power surge monitoring and suppression system using LabVIEW and multisim co-simulation tool. In *2015 International Conference on Humanoid, Nanotechnology, Information Technology,Communication and Control, Environment and Management (HNICEM)*, pages 1–3. IEEE, dec 2015. ISBN 978-1-5090-0360-0. doi: 10.1109/HNICEM.2015.7393204.

[4] Andreas Abel, Torsten Blochwitz, Alexander Eichberger, Peter Hamann, and Udo Rein. Functional mock-up interface in mechatronic gearshift simulation for commercial vehicles. In *9th International Modelica Conference. Munich*, 2012.

[5] Ahmad T. Al-Hammouri. A comprehensive co-simulation platform for cyber-physical systems. *Computer Communications*, 36(1):8–19, dec 2012. doi: 10.1016/j.comcom.2012.01.003.

[6] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P. H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, feb 1995. ISSN 03043975. doi: 10.1016/0304-3975(94)00202-T.

[7] Andrés A. Alvarez Cabrera, Krijn Woestenenk, and Tetsuo Tomiyama. An architecture model to support cooperative design for mechatronic products: A control design case. *Mechatronics*, 21(3):534–547, apr 2011. ISSN 09574158. doi: 10.1016/j.mechatronics.2011.01.009.

[8] Christian Andersson. *Methods and Tools for Co-Simulation of Dynamic Systems with the Functional Mock-up Interface.* PhD thesis, Lund University, 2016.

[9] Christian Andersson, Claus Führer, and Johan Åkesson. Efficient Predictor for Co-Simulation with Multistep Sub-System Solvers. *Technical Report in Mathematical Sciences*, 2016(1), 2016. ISSN 1403-9338. URL `http://lup.lub.lu.se/record/dbaf9c49-b118-4ff9-af2e-e1e3102e5c22`.

[10] Martin Arnold. Stability of Sequential Modular Time Integration Methods for Coupled Multibody System Models. *Journal of Computational and Nonlinear Dynamics*, 5(3):031003, may 2010. ISSN 15551423. doi: 10.1115/1.4001389.

[11] Martin Arnold and Michael Günther. Preconditioned Dynamic Iteration for Coupled Differential-Algebraic Systems. *BIT Numerical Mathematics*, 41(1):1–25, 2001. doi: 10.1023/A:1021909032551.

[12] Martin Arnold, Antonio Carrarini, Andreas Heckmann, and Gerhard Hippmann. Simulation techniques for multidisciplinary problems in vehicle system dynamics. In *Vehicle System Dynamics Supplement 40*, volume 40, pages 17–36, Vienna, Austria, 2003.

[13] Martin Arnold, Christoph Clauss, and Tom Schierz. Error Analysis and Error Estimates for Co-Simulation in FMI for Model Exchange and Co-Simulation V2.0. *Archive of Mechanical Engineering*, LX(1):75, jan 2013. ISSN 0004-0738. doi: 10.2478/meceng-2013-0005.

[14] Martin Arnold, Christoph Clauß, and Tom Schierz. Error Analysis and Error Estimates for Co-simulation in FMI for Model Exchange and Co-Simulation v2.0. In Sebastian Schöps, Andreas Bartel, Michael Günther, W E Jan ter Maten, and C Peter Müller, editors, *Progress in Differential-Algebraic Equations*, pages 107–125. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. ISBN 978-3-662-44926-4. doi: 10.1007/978-3-662-44926-4_6.

[15] Martin Arnold, Stefan Hante, and Markus A Köbis. Error analysis for co-simulation with force-displacement coupling. *PAMM*, 14(1):43–44, dec 2014. ISSN 16177061. doi: 10.1002/pamm.201410014.

[16] Memduha Aslan, Umut Durak, and Koray Taylan. MOKA: An Object-Oriented Framework for FMI Co-Simulation. 2015.

[17] Karl Johan Aström and Richard M Murray. *Feedback systems: an introduction for scientists and engineers.* Princeton university press, 2010. ISBN 1400828732.

[18] Muhammad Usman Awais, Wolfgang Mueller, Atiyah Elsheikh, Peter Palensky, and Edmund Widl. Using the HLA for Distributed Continuous Simulations. In *2013 8th EUROSIM Congress on Modelling and Simulation*, pages 544–549, Washington, DC, USA, sep 2013. IEEE. ISBN 978-0-7695-5073-2. doi: 10.1109/EUROSIM.2013.96.

[19] Muhammad Usman Awais, Peter Palensky, Atiyah Elsheikh, Edmund Widl, and Stifter Matthias. The high level architecture RTI as a master to the functional mock-up interface components. In *2013 International Conference on Computing, Networking and Communications (ICNC)*, pages 315–320, San Diego, USA, jan 2013. IEEE. ISBN 978-1-4673-5288-8. doi: 10.1109/ICCNC.2013.6504102.

[20] Muhammad Usman Awais, Peter Palensky, Wolfgang Mueller, Edmund Widl, and Atiyah Elsheikh. Distributed hybrid simulation using the HLA and the Functional Mock-up Interface. In *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, pages 7564–7569, Vienna, Austria, nov 2013. IEEE. ISBN 978-1-4799-0224-8. doi: 10.1109/IECON.2013.6700393.

[21] Fernando J. Barros. Modeling formalisms for dynamic structure systems. *ACM Transactions on Modeling and Computer Simulation*, 7(4):501–515, oct 1997. ISSN 10493301. doi: 10.1145/268403.268423.

[22] Fernando J Barros. Dynamic structure multiparadigm modeling and simulation. *ACM Transactions on Modeling and Computer Simulation*, 13(3):259–275, jul 2003. ISSN 10493301. doi: 10.1145/937332.937335.

[23] Fernando J. Barros. Semantics of dynamic structure event-based systems. In *Proceedings of the second international conference on Distributed event-based systems - DEBS '08*, DEBS '08, page 245, New York, USA, 2008. ACM Press. ISBN 9781605580906. doi: 10.1145/1385989.1386020.

[24] Paul I. Barton and C. C. Pantelides. Modeling of combined discrete/continuous processes. *AIChE Journal*, 40(6):966–979, jun 1994. ISSN 0001-1541. doi: 10.1002/aic.690400608.

[25] Jens Bastian, Christoph Clauß, Susann Wolf, and Peter Schneider. Master for Co-Simulation Using FMI. In *8th International Modelica Conference*, pages 115–120, Dresden, Germany, jun 2011. Fraunhofer Institute for Integrated Circuits IIS. doi: 10.3384/ecp11063115.

[26] Lionel Belmon, Yujung Geng, and Huaqiang He. Virtual Integration for hybrid powertrain development, using FMI and Modelica models. *10th International Modelica Conference*, 2014.

[27] Giovanni Beltrame, Donatella Sciuto, and Cristina Silvano. Multi-Accuracy Power and Performance Transaction-Level Modeling. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(10):1830–1842, oct 2007. ISSN 0278-0070. doi: 10.1109/TCAD.2007.895790.

[28] Abir Ben Khaled, Mongi Ben Gaid, Nicolas Pernet, and Daniel Simon. Fast multi-core co-simulation of Cyber-Physical Systems: Application to internal combustion engines. *Simulation Modelling Practice and Theory*, 47:79–91, sep 2014. ISSN 1569190X. doi: 10.1016/j.simpat.2014.05.002.

[29] M. Benedikt, D. Watzenig, J. Zehetner, and A. Hofer. Macro-step-size selection and monitoring of the coupoling error for weak coupled subsystems in the frequency-domain. *V International Conference on Computational Methods for Coupled Problems in Science and Engineering*, pages 1–12, 2013.

[30] Martin Benedikt and Anton Hofer. Guidelines for the Application of a Coupling Method for Non-iterative Co-simulation. In *2013 8th EUROSIM Congress on Modelling and Simulation*, pages 244–249. IEEE, sep 2013. ISBN 978-0-7695-5073-2. doi: 10.1109/EUROSIM.2013.52.

[31] Martin Benedikt and Franz Rudolf Holzinger. Automated configuration for non-iterative co-simulation. In *17th International Conference on Thermal, Mechanical and Multi-Physics Simulation and Experiments in Microelectronics and Microsystems (EuroSimE)*, pages 1–7, Montpellier, apr 2016. IEEE. ISBN 978-1-5090-2106-2. doi: 10.1109/EuroSimE.2016.7463355.

[32] Albert Benveniste, Benoît Caillaud, and Paul Le Guernic. Compositionality in Dataflow Synchronous Languages: Specification and Distributed Code Generation. *Information and Computation*, 163(1):125–171, nov 2000. ISSN 08905401. doi: 10.1006/inco.2000.9999.

[33] D. Bian, M. Kuzlu, M. Pipattanasomporn, S. Rahman, and Y. Wu. Real-time co-simulation platform using OPAL-RT and OPNET for analyzing smart grid performance. In *2015 IEEE Power & Energy Society General Meeting*, pages 1–5. IEEE, jul 2015. ISBN 978-1-4673-8040-9. doi: 10.1109/PESGM.2015.7286238.

[34] Torsten Blochwitz, Martin Otter, Martin Arnold, C. Bausch, Christoph Clauss, Hilding Elmqvist, Andreas Junghanns, Jakob Mauss, M. Monteiro, T. Neidhold, Dietmar Neumerkel, Hans Olsson, J.-V. Peetz, and S. Wolf. The Functional Mockup Interface for Tool independent Exchange of Simulation Models. In *8th International Modelica Conference*, pages 105–114, Dresden, Germany, jun 2011. Linköping University Electronic Press; Linköpings universitet. doi: 10.3384/ecp11063105.

[35] Torsten Blockwitz, Martin Otter, Johan Akesson, Martin Arnold, Christoph Clauss, Hilding Elmqvist, Markus Friedrich, Andreas Junghanns, Jakob Mauss, Dietmar Neumerkel, Hans Olsson, and Antoine Viel. Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models. In *9th International MODELICA Conference*, pages 173–184, Munich, Germany, nov 2012. Linköping University Electronic Press; Linköpings universitet. doi: 10.3384/ecp12076173.

[36] Sergiy Bogomolov, Marius Greitschus, Peter G. Jensen, Kim G. Larsen, Marius Mikucionis, Thomas Strump, and Stavros Tripakis. Co-Simulation of Hybrid Systems with SpaceEx and Uppaal. In *11th International Modelica Conference (MODELICA)*, pages 159–169, Paris, France, sep 2015. Linköping University Electronic Press. doi: 10.3384/ecp15118159.

[37] Jean-Sébastien Bolduc and Hans Vangheluwe. Expressing ODE models as DEVS: Quantization approaches. In *Proceedings of the AIS'2002 Conference (AI, Simulation and Planning in High Autonomy Systems), April 2002, Lisboa, Portugal/F. Barros and N. Giambiasi (eds.)*, pages 163–169, 2002.

[38] Jean-Sébastien Bolduc and Hans Vangheluwe. Mapping odes to devs: Adaptive quantization. In *Summer Computer Simulation Conference*, pages 401–407. Society for Computer Simulation International; 1998, 2003. ISBN 0094-7474.

[39] Massimo Bombino and Patrizia Scandurra. A model-driven co-simulation environment for heterogeneous systems. *International Journal on Software Tools for Technology Transfer*, 15 (4):363–374, aug 2013. ISSN 1433-2779. doi: 10.1007/s10009-012-0230-5.

[40] Spencer Borland. *Transforming statechart models to DEVS*. PhD thesis, 2003.

[41] F. Bouchhima, M. Briere, G Nicolescu, M Abid, and E. Aboulhamid. A SystemC/Simulink Co-Simulation Framework for Continuous/Discrete-Events Simulation. In *2006 IEEE International Behavioral Modeling and Simulation Workshop*, pages 1–6. IEEE, sep 2006. ISBN 0-7803-9742-8. doi: 10.1109/BMAS.2006.283461.

[42] Frédéric Boulanger, Cécile Hardebolle, Christophe Jacquet, and Dominique Marcadet. Semantic Adaptation for Models of Computation. In *Application of Concurrency to System Design (ACSD), 2011 11th International Conference on*, pages 153–162, 2011. ISBN 1550-4808 VO -. doi: 10.1109/ACSD.2011.17.

[43] Jonathan Brembeck, Andreas Pfeiffer, Michael Fleps-Dezasse, Martin Otter, Karl Wernersson, and Hilding Elmqvist. Nonlinear State Estimation with an Extended FMI 2.0 Co-Simulation Interface. In *Proceedings of the 10th International Modelica Conference. Lund, Sweden*, pages 53–62, 2014.

[44] T. Brezina, Z. Hadas, and J. Vetiska. Using of Co-simulation ADAMS-SIMULINK for development of mechatronic systems. In *14th International Conference Mechatronika*, pages 59–64. IEEE, jun 2011. ISBN 978-80-8075-477-8. doi: 10.1109/MECHATRON.2011.5961080.

[45] David Broman, Edward A. Lee, Stavros Tripakis, and Martin Törngren. Viewpoints, Formalisms, Languages, and Tools for Cyber-Physical Systems. In *Proceedings of the 6th International Workshop on Multi-Paradigm Modeling*, pages 49–54. ACM, 2012.

[46] David Broman, Christopher Brooks, Lev Greenberg, Edward A. Lee, Michael Masin, Stavros Tripakis, and Michael Wetter. Determinate composition of FMUs for co-simulation. In *Eleventh ACM International Conference on Embedded Software*, Montreal, Quebec, Canada, 2013. IEEE Press Piscataway, NJ, USA. ISBN 978-1-4799-1443-2.

[47] David Broman, Lev Greenberg, Edward A Lee, Michael Masin, Stavros Tripakis, and Michael Wetter. Requirements for Hybrid Cosimulation Standards. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, HSCC '15, pages 179–188, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3433-4. doi: 10.1145/2728606.2728629.

[48] Richard L. Burden and John Douglas Faires. *Numerical Analysis*. Cengage Learning, 9 edition, 2010. ISBN 0538733519.

[49] Martin Busch. *On the efficient coupling of simulation codes*. kassel university press GmbH, 2012. ISBN 3862192962.

[50] Martin Busch. Continuous approximation techniques for co-simulation methods: Analysis of numerical stability and local error. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, 96(9):1061–1081, sep 2016. ISSN 00442267. doi: 10.1002/zamm.201500196.

[51] Martin Busch and Bernhard Schweizer. Numerical stability and accuracy of different co-simulation techniques: analytical investigations based on a 2-DOF test model. In *1st Joint International Conference on Multibody System Dynamics*, pages 25–27, 2010.

[52] Martin Busch and Bernhard Schweizer. An explicit approach for controlling the macro-step size of co-simulation methods. *Proceedings of The 7th European Nonlinear Dynamics, ENOC*, pages 24–29, 2011.

[53] Martin Busch and Bernhard Schweizer. Stability of Co-Simulation Methods Using Hermite and Lagrange Approximation Techniques. In *ECCOMAS Thematic Conference on Multibody Dynamics*, pages 1–10, Brussels, Belgium, jul 2011.

[54] Martin Busch and Bernhard Schweizer. Coupled simulation of multibody and finite element systems: an efficient and robust semi-implicit coupling approach. *Archive of Applied Mechanics*, 82(6):723–741, jun 2012. ISSN 0939-1533. doi: 10.1007/s00419-011-0586-0.

[55] Benjamin Camus, Christine Bourjot, and Vincent Chevrier. Combining DEVS with multi-agent concepts to design and simulate multi-models of complex systems (WIP). In *Proceedings of the Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*, pages 85–90. Society for Computer Simulation International, 2015. ISBN 978-1-5108-0105-9.

[56] Benjamin Camus, Virginie Galtier, Mathieu Caujolle, Vincent Chevrier, Julien Vaubourg, Laurent Ciarletta, and Christine Bourjot. Hybrid Co-simulation of FMUs using DEV&DESS in MECSYCO. In *Proceedings of the Symposium on Theory of Modeling & Simulation - DEVS Integrative M&S Symposium (TMS/DEVS 16)*, Pasadena, CA, United States, jan 2016. Université de Lorraine, CNRS, Inria, LORIA, UMR 7503 ; CentraleSupélec UMI GT-CNRS 2958 Université Paris-Saclay ; EDF - R&D MIRE/R44.

[57] Luca P. Carloni, Roberto Passerone, Alessandro Pinto, and Alberto L. Angiovanni-Vincentelli. Languages and Tools for Hybrid Systems Design. *Foundations and Trends® in Electronic Design Automation*, 1(1/2):1–193, 2006. ISSN 1551-3939. doi: 10.1561/1000000001.

[58] Christopher D. Carothers, Kalyan S. Perumalla, and Richard M. Fujimoto. Efficient optimistic parallel simulations using reverse computation. *ACM Transactions on Modeling and Computer Simulation*, 9(3):224–253, jul 1999. ISSN 10493301. doi: 10.1145/347823.347828.

[59] Volker Carstens, Ralf Kemme, and Stefan Schmitt. Coupled simulation of flow-structure interaction in turbomachinery. *Aerospace Science and Technology*, 7(4):298–306, jun 2003. ISSN 12709638. doi: 10.1016/S1270-9638(03)00016-6.

[60] François Edouard Cellier. Combined Continuous/Discrete System Simulation Languages: Usefulness, Experiences and Future Development. *SIGSIM Simul. Dig.*, 9(1):18–21, 1977. ISSN 0163-6103. doi: 10.1145/1102505.1102514.

[61] François Edouard Cellier. *Combined Continuous Discrete Simulation by use of Digital Computers: Techniques and Tools*. PhD thesis, 1979.

[62] François Edouard Cellier. *Continuous system modeling*. Springer Science & Business Media, 1991.

[63] François Edouard Cellier and Ernesto Kofman. *Continuous System Simulation*. Springer Science & Business Media, 2006. ISBN 9780387261027.

[64] K.M. Chandy and J Misra. Distributed Simulation: A Case Study in Design and Verification of Distributed Programs. *IEEE Transactions on Software Engineering*, SE-5(5):440–452, sep 1979. ISSN 0098-5589. doi: 10.1109/TSE.1979.230182.

[65] W-T. Chang, A. Kalavade, and E. A. Lee. Effective Heterogenous Design and Co-Simulation. In Giovanni De Micheli and Mariagiovanna Sami, editors, *Hardware/Software Co-Design*, pages 187–212. Springer Netherlands, Dordrecht, 1996. ISBN 978-94-009-0187-2. doi: 10.1007/978-94-009-0187-2_8.

[66] Yung-Wei Chen, Chein-Shan Liu, and Jiang-Ren Chang. A chaos detectable and time step-size adaptive numerical scheme for nonlinear dynamical systems. *Journal of Sound and Vibration*, 299(4-5):977–989, feb 2007. ISSN 0022460X. doi: 10.1016/j.jsv.2006.08.028.

[67] Alex Chung Hen Chow and Bernard P. Zeigler. Parallel DEVS: A Parallel, Hierarchical, Modular, Modeling Formalism. In *Proceedings of the 26th Conference on Winter Simulation*, WSC '94, pages 716–722, San Diego, CA, USA, 1994. Society for Computer Simulation International. ISBN 0-7803-2109-X.

[68] Fabio Cremona, Marten Lohstroh, Stavros Tripakis, Christopher Brooks, and Edward A Lee. FIDE: an FMI integrated development environment. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing - SAC '16*, SAC '16, pages 1759–1766, New York, New York, USA, 2016. ACM Press. ISBN 9781450337397. doi: 10.1145/2851613.2851677.

[69] Makarand Datar, Ilinca Stanciulescu, and Dan Negrut. A co-simulation environment for high-fidelity virtual prototyping of vehicle systems. *International Journal of Vehicle Systems Modelling and Testing*, 7(1):54, jan 2012. ISSN 1745-6436. doi: 10.1504/IJVSMT.2012.045308.

[70] Joachim Denil, Bart Meyers, Paul De Meulenaere, and Hans Vangheluwe. Explicit Semantic Adaptation of Hybrid Formalisms for FMI Co-Simulation. In Society for Computer Simulation International, editor, *Proceedings of the Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*, pages 99–106, Alexandria, Virginia, 2015.

[71] Stefan Dietz, Gerhard HIPPMANN, and Gunter SCHUPP. Interaction of vehicles and flexible tracks by co-simulation of multibody vehicle systems and finite element track models. *Vehicle system dynamics*, 37:372–384, 2002. ISSN 0042-3114.

[72] Paul Adrien Maurice Dirac. *The principles of quantum mechanics.* Number 27. Oxford university press, 1981. ISBN 0198520115.

[73] W. Stuart Dols, Steven J. Emmerich, and Brian J. Polidoro. Coupling the multizone airflow and contaminant transport software CONTAM with EnergyPlus using co-simulation. *Building Simulation*, 9(4):469–479, aug 2016. ISSN 1996-3599. doi: 10.1007/s12273-016-0279-2.

[74] Edo Drenth, Mikael Törmänen, Krister Johansson, Bengt-Arne Andersson, Daniel Andersson, Ivar Torstensson, and Johan Åkesson. Consistent Simulation Environment with FMI based Tool Chain. In *Proceedings of 10th International Modelica Conference, Lund, Sweden*, 2014.

[75] Sven Dronka and Jochen Rauh. Co-simulation-interface for user-force-elements. In *Proceedings of SIMPACK user meeting*, 2006.

[76] John C. Eidson, Edward A. Lee, Slobodan Matic, Sanjit A. Seshia, and Jia Zou. Distributed Real-Time Software for Cyber-Physical Systems. *Proceedings of the IEEE*, 100(1):45–59, jan 2012. ISSN 0018-9219. doi: 10.1109/JPROC.2011.2161237.

[77] H. El Tahawy, D. Rodriguez, S. Garcia-Sabiro, and J.-J. Mayol. VHD/sub e/LDO: A new mixed mode simulation. In *Proceedings of EURO-DAC 93 and EURO-VHDL 93- European Design Automation Conference*, pages 546–551. IEEE Comput. Soc. Press, 1993. ISBN 0-8186-4350-1. doi: 10.1109/EURDAC.1993.410690.

[78] Atiyah Elsheikh, Muhammed Usman Awais, Edmund Widl, and Peter Palensky. Modelica-enabled rapid prototyping of cyber-physical energy systems via the functional mockup interface. In *2013 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, pages 1–6. IEEE, may 2013. ISBN 978-1-4799-1307-7. doi: 10.1109/MSCPES.2013.6623315.

[79] Olaf Enge-Rosenblatt, Christoph Clauß, André Schneider, Peter Schneider, and Olaf Enge. Functional Digital Mock-up and the Functional Mock-up Interface–Two Complementary Approaches for a Comprehensive Investigation of Heterogeneous Systems. In *Proc. of the 8th Int. Modelica Conference*, 2011.

[80] Kenneth Eriksson, Donald Estep, and Claes Johnson. *Applied Mathematics: Body and Soul.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-642-05659-8. doi: 10.1007/978-3-662-05796-4.

[81] Emeka Eyisi, Jia Bai, Derek Riley, Jiannian Weng, Wei Yan, Yuan Xue, Xenofon Koutsoukos, and Janos Sztipanovits. NCSWT: An integrated modeling and simulation tool for networked control systems. *Simulation Modelling Practice and Theory*, 27:90–111, sep 2012. ISSN 1569190X. doi: 10.1016/j.simpat.2012.05.004.

[82] Cyril Faure, Mongi Ben Gaid, Nicolas Pernet, Morgan Fremovici, Gregory Font, and Gilles Corde. Methods for real-time simulation of Cyber-Physical Systems: application to automotive domain. In *2011 7th International Wireless Communications and Mobile Computing Conference*, pages 1105–1110. IEEE, jul 2011. ISBN 978-1-4244-9539-9. doi: 10.1109/IWCMC.2011.5982695.

[83] Yishai A. Feldman, Lev Greenberg, and Eldad Palachi. Simulating Rhapsody SysML Blocks in Hybrid Models with FMI. In *Proceedings of the 10th International Modelica Conference*, pages 43–52. Linköping University Electronic Press, mar 2014. doi: 10.3384/ecp1409643.

[84] P. Fey, H.W. Carter, and P.A. Wilsey. Parallel synchronization of continuous time discrete event simulators. In *Proceedings of the 1997 International Conference on Parallel Processing (Cat. No.97TB100162)*, pages 227–231. IEEE Comput. Soc, 1997. ISBN 0-8186-8108-X. doi: 10.1109/ICPP.1997.622649.

[85] J. S. Fitzgerald and P. G. Larsen. Balancing Insight and Effort: the Industrial Uptake of Formal Methods. In Cliff B. Jones, Zhiming Liu, and Jim Woodcock, editors, *Formal Methods and Hybrid Real-Time Systems, Essays in Honour of Dines Bjørner and Chaochen Zhou on the Occasion of Their 70th Birthdays*, pages 237–254, Volume 4700, September 2007. Springer, Lecture Notes in Computer Science. ISBN 978-3-540-75220-2.

[86] John Fitzgerald, Peter Gorm Larsen, Ken Pierce, Marcel Verhoef, and Sune Wolff. Collaborative Modelling and Co-simulation in the Development of Dependable Embedded Systems. pages 12–26. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-16265-7. doi: 10.1007/978-3-642-16265-7_2.

[87] John Fitzgerald, Peter Gorm Larsen, and Marcel Verhoef. *Collaborative Design for Embedded Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. ISBN 978-3-642-54117-9. doi: 10.1007/978-3-642-54118-6.

[88] John Fitzgerald, Carl Gamble, Richard Payne, Peter Gorm Larsen, Stylianos Basagiannis, and Alie El-Din Mady. Collaborative Model-based Systems Engineering for Cyber-Physical Systems – a Case Study in Building Automation. In *INCOSE 2016*, Edinburgh, Scotland, jul 2016.

[89] John S. Fitzgerald, Peter Gorm Larsen, Ken G. Pierce, and Marcel Henri Gerard Verhoef. A formal approach to collaborative modelling and co-simulation for embedded systems. *Mathematical Structures in Computer Science*, 23(04):726–750, aug 2013. ISSN 0960-1295. doi: 10.1017/S0960129512000242.

[90] Alain Fourmigue, Bruno Girodias, Gabriela Nicolescu, and E.-M. Aboulhamid. Co-simulation based platform for wireless protocols design explorations. In *2009 Design, Automation & Test in Europe Conference & Exhibition*, pages 874–877. IEEE, apr 2009. ISBN 978-1-4244-3781-8. doi: 10.1109/DATE.2009.5090785.

[91] P. Frey, R. Radhakrishnan, H. W. Carter, and P. A. Wilsey. Optimistic synchronization of mixed-mode simulators. In *1998 First Merged International on Parallel Processing Symposium and Symposium on Parallel and Distributed Processing*, pages 694–699, 1998. ISBN 1063-7133 VO -. doi: 10.1109/IPPS.1998.670002.

[92] Jonathan Friedman and Jason Ghidella. Using model-based design for automotive systems engineering-requirements analysis of the power window example. Technical report, 2006.

[93] Markus Friedrich. *Parallel Co-Simulation for Mechatronic Systems.* PhD thesis, 2011.

[94] Richard M. Fujimoto. *Parallel and distributed simulation systems*, volume 300. Wiley New York, New York, USA, 1 edition, 2000. ISBN 0-7803-7307-3. doi: 10.1109/WSC.2001.977259.

[95] Jason C. Fuller, Selim Ciraci, Jeffrey A. Daily, Andrew R. Fisher, and M. Hauer. Communication simulations for power system applications. In *2013 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, pages 1–6. IEEE, may 2013. ISBN 978-1-4799-1307-7. doi: 10.1109/MSCPES.2013.6623314.

[96] Virginie Galtier, Gilles Plessis, and Les Renardi. FMI-Based Distributed Multi-Simulation with DACCOSIM. In *Spring Simulation Multi-Conference*, pages 804–811. Society for Computer Simulation International, 2015.

[97] Alfredo Garro and Alberto Falcone. On the integration of HLA and FMI for supporting interoperability and reusability in distributed simulation. In *Spring Simulation Multi-Conference*, pages 774–781. Society for Computer Simulation International, 2015.

[98] C. W. Gear and D. R. Wells. Multirate linear multistep methods. *BIT*, 24(4):484–502, dec 1984. ISSN 0006-3835. doi: 10.1007/BF01934907.

[99] L. Gheorghe, F. Bouchhima, G. Nicolescu, and H. Boucheneb. Formal Definitions of Simulation Interfaces in a Continuous/Discrete Co-Simulation Tool. In *Rapid System Prototyping, 2006. Seventeenth IEEE International Workshop on*, pages 186–192, 2006. ISBN 1074-6005 VO -. doi: 10.1109/RSP.2006.18.

[100] L. Gheorghe, F. Bouchhima, G. Nicolescu, and H. Boucheneb. A Formalization of Global Simulation Models for Continuous/Discrete Systems. In *Proceedings of the 2007 Summer Computer Simulation Conference*, SCSC '07, pages 559–566, San Diego, CA, USA, 2007. Society for Computer Simulation International. ISBN 1-56555-316-0.

[101] A. Ghosh, M. Bershteyn, R. Casley, C. Chien, A. Jain, M. Lipsie, D. Tarrodaychik, and O. Yamamoto. A hardware-software co-simulator for embedded system design and debugging. In *Design Automation Conference*, pages 155–164, 1995. doi: 10.1109/ASPDAC.1995.486217.

[102] Cláudio Gomes. Foundations for Continuous Time Hierarchical Co-simulation. In *ACM Student Research Competition (ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems)*, page to appear, Saint Malo, Brittany, France, 2016.

[103] Francisco González, Miguel Ángel Naya, Alberto Luaces, and Manuel González. On the effect of multirate co-simulation techniques in the efficiency and accuracy of multibody system dynamics. *Multibody System Dynamics*, 25(4):461–483, apr 2011. ISSN 1384-5640. doi: 10.1007/s11044-010-9234-7.

[104] Matthias Gries. Methods for evaluating and covering the design space during early design development. *Integration, the VLSI Journal*, 38(2):131–183, dec 2004. ISSN 01679260. doi: 10.1016/j.vlsi.2004.06.001.

[105] Bei Gu and H H Asada. Co-simulation of algebraically coupled dynamic subsystems. In *American Control Conference, 2001. Proceedings of the 2001*, volume 3, pages 2273–2278 vol.3, 2001. ISBN 0743-1619 VO - 3. doi: 10.1109/ACC.2001.946089.

[106] Bei Gu and H. Harry Asada. *Co-simulation of algebraically coupled dynamic subsystems.* PhD thesis, 2001.

[107] Bei Gu and H. Harry Asada. Co-Simulation of Algebraically Coupled Dynamic Subsystems Without Disclosure of Proprietary Subsystem Models. *Journal of Dynamic Systems, Measurement, and Control*, 126(1):1, apr 2004. ISSN 00220434. doi: 10.1115/1.1648307.

[108] Felix Günther, Georg Mallebrein, and Heinz Ulbrich. A Modular Technique for Automotive System Simulation. In *Proc. 9th International Modelica Conference*, Munich, Germany, 2012.

[109] M. Günther, A. Kværnø, and P. Rentrop. Multirate Partitioned Runge-Kutta Methods. *Bit*, 41(3):504–514, 2001. ISSN 00063835. doi: 10.1023/A:1021967112503.

[110] R. K. Gupta, C. N. Coelho Jr., and G. De Micheli. Synthesis and Simulation of Digital Systems Containing Interacting Hardware and Software Components. In *Proceedings of the 29th ACM/IEEE Design Automation Conference*, DAC '92, pages 225–230, Los Alamitos, CA, USA, 1992. IEEE Computer Society Press. ISBN 0-89791-516-X.

[111] Dinesh Rangana Gurusinghe, Saranga Menike, A I Konara, Athula D Rajapakse, Pradeepa Yahampath, U D Annakkage, Brian A Archer, and Tony Weekes. Co-simulation of Power System and Synchrophasor Communication Network on a Single Simulation Platform. *Technology and Economics of Smart Grids and Sustainable Energy*, 1(1):6, dec 2016. ISSN 2199-4706. doi: 10.1007/s40866-015-0003-9. URL `http://dx.doi.org/10.1007/s40866-015-0003-9http://link.springer.com/10.1007/s40866-015-0003-9`.

[112] Kjell Gustafsson. *Control of error and convergence in ODE solvers.* PhD thesis, 1992.

[113] Kjell Gustafsson, Michael Lundh, and Gustaf Söderlind. API stepsize control for the numerical solution of ordinary differential equations. *BIT*, 28(2):270–287, jun 1988. ISSN 0006-3835. doi: 10.1007/BF01934091.

[114] Irene Hafner, Bernhard Heinzl, and Matthias Roessler. An Investigation on Loose Coupling Co-Simulation with the BCVTB. *SNE Simulation Notes Europe*, 23(1), 2013. ISSN 2305-9974. doi: 10.11128/sne.23.tn.10173.

[115] Walid Hassairi, Moncef Bousselmi, Mohamed Abid, and Carlos Valderrama. Matlab/SystemC for the New Co-Simulation Environment by JPEG Algorithm. *MATLAB–A Fundamental Tool for Scientific Computing and Engineering Applications*, 2:120–138, 2012.

[116] D. R. Hickey, P. A. Wilsey, R. J. Hoekstra, E. R. Keiter, S. A. Hutchinson, and T. V. Russo. Mixed-signal simulation with the Simbus backplane. In *39th Annual Simulation Symposium, 2006*, Huntsville, AL, 2006. doi: 10.1109/ANSS.2006.25.

[117] Ken Hines and Gaetano Borriello. Selective focus as a means of improving geographically distributed embedded system co-simulation. In *8th IEEE International Workshop on Rapid System Prototyping, 1997*, pages 58–62, 1997. doi: 10.1109/IWRSP.1997.618825.

[118] Ken Hines and Gaetano Borriello. Dynamic Communication Models in Embedded System Co-simulation. In *Proceedings of the 34th Annual Design Automation Conference*, DAC '97, pages 395–400, New York, NY, USA, 1997. ACM. ISBN 0-89791-920-3. doi: 10.1145/266021.266178.

[119] Matthias Hoepfer. *Towards a Comprehensive Framework for Co- Simulation of Dynamic Models With an Emphasis on Time Stepping.* PhD thesis, 2011.

[120] Hua Lin, Santhoshkumar Sambamoorthy, Sandeep Shukla, James Thorp, and Lamine Mili. Power system and communication network co-simulation for smart grid applications. In *ISGT 2011*, pages 1–6. IEEE, jan 2011. ISBN 978-1-61284-218-9. doi: 10.1109/ISGT.2011.5759166.

[121] Z. Jackiewicz and M. Kwapisz. Convergence of Waveform Relaxation Methods for Differential-Algebraic Systems. *SIAM Journal on Numerical Analysis*, 33(6):2303–2317, dec 1996. ISSN 0036-1429. doi: 10.1137/S0036142992233098.

[122] Kenneth R Jackson. A survey of parallel numerical methods for initial value problems for ordinary differential equations. *IEEE Transactions on Magnetics*, 27(5):3792–3797, 1991. ISSN 0018-9464.

[123] David Jefferson and Henry A. Sowizral. Fast concurrent simulation using the time warp mechanism. Technical report, Rand Corporation, 1982.

[124] David R. Jefferson. Virtual time. *ACM Transactions on Programming Languages and Systems*, 7(3):404–425, jul 1985. ISSN 01640925. doi: 10.1145/3916.3988.

[125] Karl Henrik Johansson, Magnus Egerstedt, John Lygeros, and Shankar Sastry. On the regularization of Zeno hybrid automata. *Systems & Control Letters*, 38(3):141–150, oct 1999. ISSN 01676911. doi: 10.1016/S0167-6911(99)00059-6.

[126] Raphaël Jungers. *The joint spectral radius: theory and applications*, volume 385. Springer Science & Business Media, 2009. ISBN 3540959793.

[127] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960. ISSN 0021-9223.

[128] Tamas Kalmar-Nagy and Ilinca Stanciulescu. Can complex systems really be simulated? *Applied Mathematics and Computation*, 227:199–211, jan 2014. ISSN 00963003. doi: 10.1016/j.amc.2013.11.037.

[129] K. C. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson. Feature-Oriented Domain Analysis. Feasibility study,. Technical report, Carnegie Mellon University, 1990.

[130] M. Karner, M. Krammer, S. Krug, E. Armengaud, C. Steger, and R. Weiss. Heterogeneous co-simulation platform for the efficient analysis of FlexRay-based automotive distributed embedded systems. In *8th IEEE International Workshop on Factory Communication Systems (WFCS)*, pages 231–240, 2010. doi: 10.1109/WFCS.2010.5548627.

[131] Michael Karner, Eric Armengaud, Christian Steger, and Reinhold Weiss. Holistic Simulation of FlexRay Networks by Using Run-time Model Switching. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '10, pages 544–549, 3001 Leuven, Belgium, Belgium, 2010. European Design and Automation Association. ISBN 978-3-9810801-6-2.

[132] Michael Karner, Martin Krammer, Markus Schratter, Peter Wimmer, Daniel Watzenig, and ChristianMichael Gruber. A Comprehensive Approach for Modeling, Simulation and Virtual Validation of Integrated Safety Systems. In Jan Fischer-Wolfarth and Gereon Meyer, editors, *Advanced Microsystems for Automotive Applications 2013 SE - 10*, Lecture Notes in Mobility, pages 101–110. Springer International Publishing, 2013. ISBN 978-3-319-00475-4. doi: 10.1007/978-3-319-00476-1_10.

[133] Abir Ben Khaled, Mongi Ben Gaid, Daniel Simon, and Gregory Font. Multicore simulation of powertrains using weakly synchronized model partitioning. In *IFAC Proceedings Volumes*, volume 45, pages 448–455, Rueil-Malmaison, France, oct 2012. doi: 10.3182/20121023-3-FR-4025.00018.

[134] Abir Ben Khaled, Laurent Duval, Mohamed El Mongi Ben Gaïd, and Daniel Simon. Context-based polynomial extrapolation and slackened synchronization for fast multi-core simulation using FMI. In *10th International Modelica Conference 2014*, pages 225–234. Link{ö}ping University Electronic Press, 2014.

[135] James Ellis Kleckner. *Advanced mixed-mode simulation techniques.* PhD thesis, 1984.

[136] Ernesto Kofman. A Second-Order Approximation for DEVS Simulation of Continuous Systems. *SIMULATION*, 78(2):76–89, feb 2002. ISSN 0037-5497. doi: 10.1177/003754970207800206.

[137] Ernesto Kofman and Sergio Junco. Quantized-state systems: a DEVS Approach for continuous system simulation. *Transactions of The Society for Modeling and Simulation International*, 18(3):123–132, 2001. ISSN 0740-6797.

[138] Alexander Kossiakoff, William N. Sweet, Samuel J. Seymour, and Steven M. Biemer. *Structure of Complex Systems*, pages 41–67. John Wiley & Sons, Inc., 2011. ISBN 9781118001028. doi: 10.1002/9781118001028.ch3. URL `http://dx.doi.org/10.1002/9781118001028.ch3`.

[139] Velin Kounev, David Tipper, Martin Levesque, Brandon M. Grainger, Thomas Mcdermott, and Gregory F. Reed. A microgrid co-simulation framework. In *2015 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, pages 1–6. IEEE, apr 2015. ISBN 978-1-4799-7357-6. doi: 10.1109/MSCPES.2015.7115398.

[140] Martin Krammer, Johannes Fritz, and Michael Karner. Model-Based Configuration of Automotive Co-Simulation Scenarios. In *Spring Simulation Multi-Conference*, pages 246–253. Society for Computer Simulation International, 2015.

[141] R. Kübler and W. Schiehlen. Two Methods of Simulator Coupling. *Mathematical and Computer Modelling of Dynamical Systems*, 6(2):93–113, jun 2000. ISSN 1387-3954. doi: 10.1076/1387-3954(200006)6:2;1-M;FT093.

[142] R. Kübler and W. Schiehlen. Modular Simulation in Multibody System Dynamics. *Multibody System Dynamics*, 4(2-3):107–127, 2000. ISSN 1384-5640. doi: 10.1023/A:1009810318420.

[143] Michal Kudelski, Luca M. Gambardella, and Gianni A. Di Caro. RoboNetSim: An integrated framework for multi-robot and network simulation. *Robotics and Autonomous Systems*, 61(5):483–496, may 2013. ISSN 09218890. doi: 10.1016/j.robot.2013.01.003.

[144] Frederick Kuhl, Richard Weatherly, and Judith Dahmann. *Creating computer simulation systems: an introduction to the high level architecture*. Prentice Hall PTR, 1999. ISBN 0130225118.

[145] Thomas Kühne. What is a Model? In Jean Bezivin and Reiko Heckel, editors, *Language Engineering for Model-Driven Software Development*, volume 04101. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), 2005.

[146] T. Kuhr, T. Forster, T. Braun, and R. Gotzhein. FERAL - Framework for simulator coupling on requirements and architecture level. In *Eleventh IEEE/ACM International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, pages 11–22, 2013.

[147] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, jul 1978. ISSN 00010782. doi: 10.1145/359545.359563.

[148] Peter Gorm Larsen, Casper Thule, Kenneth Lausdahl, Victor Bardur, Carl Gamble, Etienne Brosse, Andrey Sadovykh, Alessandra Bagnato, and Luis Diogo Couto. Integrated Tool Chain for Model-Based Design of Cyber-Physical Systems. In Peter Gorm Larsen, Nico Plat, and Nick Battle, editors, *The 14th Overture Workshop: Towards Analytical Tool Chains*, pages 63–78, Cyprus, November 2016. Aarhus University, Department of Engineering. ECE-TR-28.

[149] Kenneth Lausdahl, Peter Gorm Larsen, Sune Wolf, Victor Bandur, Anders Terkelsen, Miran Hasanagić, Casper Thule Hansen, Ken Pierce, Oliver Kotte, Adrian Pop, Etienne Brosse, Jörg Brauer, and Oliver Möller. Design of the INTO-CPS Platform. Technical report, INTO-CPS Deliverable, D4.1d, December 2015.

[150] David P. Y. Lawrence, Cláudio Gomes, Joachim Denil, Hans Vangheluwe, and Didier Buchs. Coupling Petri nets with Deterministic Formalisms Using Co-simulation. In *Proceedings of the Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*, pages 6:1—-6:8, Pasadena, CA, USA, 2016.

[151] P. Le Marrec, C. A. Valderrama, F. Hessel, A. A. Jerraya, M. Attia, and O. Cayrol. Hardware, software and mechanical cosimulation for automotive applications. In *9th International Workshop on Rapid System Prototyping*, pages 202–206, 1998. doi: 10.1109/IWRSP.1998.676692.

[152] Bu-Sung Lee, Wentong Cai, Stephen J. Turner, and L. Chen. Adaptive dead reckoning algorithms for distributed interactive simulation. *International Journal of Simulation*, 1(1-2): 21–34, 2000.

[153] Edward A. Lee. Cyber Physical Systems: Design Challenges. In *11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC)*, pages 363–369, 2008. doi: 10.1109/ISORC.2008.25.

[154] Edward A. Lee and Haiyang Zheng. Operational semantics of hybrid systems. In Manfred Morari and Lothar Thiele, editors, *Hybrid Systems: Computation and Control*, volume 3414 of *Lecture Notes in Computer Science*, pages 25–53. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-25108-8. doi: 10.1007/978-3-540-31954-2_2.

[155] E. Lelarasmee, Albert E. Ruehli, and A. L. Sangiovanni-Vincentelli. The Waveform Relaxation Method for Time-Domain Analysis of Large Scale Integrated Circuits. In *IEEE Transactions*

on Computer-Aided Design of Integrated Circuits and Systems, volume 1, pages 131–145, 1982. ISBN 0278-00701. doi: 10.1109/TCAD.1982.1270004.

[156] Shengqin Li and Le He. Co-simulation Study of Vehicle ESP System Based on ADAMS and MATLAB. *Journal of Software*, 6(5), 2011.

[157] W. Li, A. Monti, M. Luo, and R. A. Dougal. VPNET: A co-simulation framework for analyzing communication channel effects on power systems. In *2011 IEEE Electric Ship Technologies Symposium*, pages 143–149. IEEE, apr 2011. ISBN 978-1-4244-9272-5. doi: 10.1109/ESTS.2011.5770857.

[158] Weilin Li, Min Luo, Lin Zhu, Antonello Monti, and Ferdinanda Ponci. A co-simulation method as an enabler for jointly analysis and design of MAS-based electrical power protection and communication. *SIMULATION*, mar 2013.

[159] Weilin Li, Xiaobin Zhang, and Huimin Li. Co-simulation platforms for co-design of networked control systems: An overview. *Control Engineering Practice*, 23:44–56, feb 2014. ISSN 09670661. doi: 10.1016/j.conengprac.2013.10.010.

[160] F. Liu, J. Cai, Y. Zhu, H. M. Tsai, and A. S. F. Wong. Calculation of Wing Flutter by a Coupled Fluid-Structure Method. *Journal of Aircraft*, 38(2):334–342, mar 2001. ISSN 0021-8669. doi: 10.2514/2.2766.

[161] John Lygeros. Lecture notes on hybrid systems, 2004.

[162] Oded Maler, Zohar Manna, and Amir Pnueli. From timed to hybrid systems. *Real-Time: Theory in Practice*, 600:447–484, 1992. doi: ˜10.1007/BFb0032003.

[163] M. Manbachi, A. Sadu, H. Farhangi, A. Monti, A. Palizban, F. Ponci, and S. Arzanpour. Impact of EV penetration on Volt–VAR Optimization of distribution networks using real-time co-simulation monitoring platform. *Applied Energy*, 169:28–39, may 2016. ISSN 03062619. doi: 10.1016/j.apenergy.2016.01.084.

[164] Toni Mancini, Federico Mari, Annalisa Massini, Igor Melatti, Fabio Merli, and Enrico Tronci. System Level Formal Verification via Model Checking Driven Simulation. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification SE - 21*, volume 8044 of *Lecture Notes in Computer Science*, pages 296–312. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-39798-1. doi: 10.1007/978-3-642-39799-8_21.

[165] Zohar Manna and Amir Pnueli. Verifying hybrid systems. In RobertL. Grossman, Anil Nerode, AndersP. Ravn, and Hans Rischel, editors, *Hybrid Systems SE - 2*, volume 736 of *Lecture Notes in Computer Science*, pages 4–35. Springer Berlin Heidelberg, 1993. ISBN 978-3-540-57318-0. doi: 10.1007/3-540-57318-6_22. URL `http://dx.doi.org/10.1007/3-540-57318-6{_}22`.

[166] J. A. Martinez, T. P. Kurzweg, S. P. Levitan, P. J. Marchand, and D. M. Chiarulli. Mixed-Technology System-Level Simulation. *Analog Integrated Circuits and Signal Processing*, 29 (1-2):127–149, 2001. ISSN 0925-1030. doi: 10.1023/A:1011294616831.

[167] William J. McCalla. *Fundamentals of computer-aided circuit simulation*, volume 37. Springer Science & Business Media, 1987. ISBN 1461320119.

[168] M Mews, J Svacina, and S Weissleder. From AUTOSAR Models to Co-simulation for MiL-Testing in the Automotive Domain. In *Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on*, pages 519–528, 2012. ISBN VO -. doi: 10.1109/ICST.2012.137.

[169] Bart Meyers, Joachim Denil, Frédéric Boulanger, Cécile Hardebolle, Christophe Jacquet, and Hans Vangheluwe. A DSL for Explicit Semantic Adaptation. In Edward Jones Tamás Mészáros Christophe Jacquet Daniel Balasubramanian, editor, *MPM 2013*, number 1112 in CEUR Workshop Proceedings, pages 47–56, Miami, United States, sep 2013.

[170] U. Miekkala and O. Nevanlinna. Convergence of Dynamic Iteration Methods for Initial Value Problems. *SIAM Journal on Scientific and Statistical Computing*, 8(4):459–482, jul 1987. ISSN 0196-5204. doi: 10.1137/0908046.

[171] Debasis Mitra. Asynchronous Relaxations for the Numerical Solution of Differential Equations by Parallel Processors. *SIAM Journal on Scientific and Statistical Computing*, 8(1):s43–s58, jan 1987. ISSN 0196-5204. doi: 10.1137/0908012.

[172] Pieter J. Mosterman. An Overview of Hybrid Simulation Phenomena and Their Support by Simulation Packages. In FritsW. Vaandrager and JanH. van Schuppen, editors, *Hybrid Systems: Computation and Control SE - 17*, volume 1569 of *Lecture Notes in Computer Science*, pages 165–177. Springer Berlin Heidelberg, 1999. ISBN 978-3-540-65734-7. doi: 10.1007/3-540-48983-5_17.

[173] Pieter J. Mosterman. HYBRSIM—a modelling and simulation environment for hybrid bond graphs. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 216(1), 2002. doi: 10.1243/0959651021541417.

[174] Pieter J. Mosterman and Gautam Biswas. A theory of discontinuities in physical system models. *Journal of the Franklin Institute*, 335(3):401–439, apr 1998. ISSN 00160032. doi: 10.1016/S0016-0032(96)00126-3.

[175] Pieter J. Mosterman, Justyna Zander, Gregoire Hamon, and Ben Denckla. A computational model of time for stiff hybrid systems applied to control synthesis. *Control Engineering Practice*, 20(1):2–13, 2012. ISSN 09670661. doi: 10.1016/j.conengprac.2011.04.013.

[176] W Muller and E Widl. Using FMI components in discrete event systems. In *Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES), 2015 Workshop on*, pages 1–6, 2015. ISBN VO -. doi: 10.1109/MSCPES.2015.7115397.

[177] Sadaf Mustafiz and Hans Vangheluwe. Explicit Modelling of Statechart Simulation Environments. In *Proceedings of the 2013 Summer Computer Simulation Conference*, SCSC '13, pages 21:1—-21:8, Vista, CA, 2013. Society for Modeling & Simulation International. ISBN 978-1-62748-276-9.

[178] Sadaf Mustafiz, Bruno Barroca, Cláudio Gomes, and Hans Vangheluwe. Towards Modular Language Design Using Language Fragments: The Hybrid Systems Case Study. In *13th International Conference on Information Technology - New Generations (ITNG)*, pages 785–797. 2016. doi: 10.1007/978-3-319-32467-8_68.

[179] Sadaf Mustafiz, Cláudio Gomes, Bruno Barroca, and Hans Vangheluwe. Modular Design of Hybrid Languages by Explicit Modeling of Semantic Adaptation. In *Proceedings of the Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*, DEVS '16, pages 29:1—-29:8, San Diego, CA, USA, 2016.

[180] Alexandre Muzy, Luc Touraille, Hans Vangheluwe, Olivier Michel, Mamadou Kaba Traoré, and David R. C. Hill. Activity Regions for the Specification of Discrete Event Systems. In *Proceedings of the 2010 Spring Simulation Multiconference*, SpringSim '10, pages 136:1—-136:7, San Diego, CA, USA, 2010. Society for Computer Simulation International. ISBN 978-1-4503-0069-8. doi: 10.1145/1878537.1878679.

[181] Andreas Naderlinger. Multiple Real-time Semantics on Top of Synchronous Block Diagrams. In *Proceedings of the Symposium on Theory of Modeling & Simulation - DEVS Integrative M&S Symposium*, DEVS 13, pages 6:1—-6:7, San Diego, CA, USA, 2013. Society for Computer Simulation International. ISBN 978-1-62748-032-1.

[182] Himanshu Neema, Jesse Gohl, Zsolt Lattmann, Janos Sztipanovits, Gabor Karsai, Sandeep Neema, Ted Bapty, John Batteh, Hubertus Tummescheit, and Chandrasekar Sureshkumar. Model-based integration platform for FMI co-simulation and heterogeneous simulations of cyber-physical systems. In *10th International Modelica Conference*, pages 10–12, 2014.

[183] Arthur Richard Newton and Alberto L. Sangiovanni-Vincentelli. Relaxation-Based Electrical Simulation. *SIAM Journal on Scientific and Statistical Computing*, 4(3):485–524, sep 1983. ISSN 0196-5204. doi: 10.1137/0904036.

[184] Y. Ni and J. F. Broenink. Hybrid systems modelling and simulation in DESTECS: a co-simulation approach. In M Klumpp, editor, *26th European Simulation and Modelling Conference, ESM 2012, Essen, Germany*, pages 32–36, Ghent, Belgium, 2012. EUROSIS-ETI.

[185] Claus Ballegaard Nielsen, Peter Gorm Larsen, John Fitzgerald, Jim Woodcock, and Jan Peleska. Systems of Systems Engineering: Basic Concepts, Model-Based Techniques, and Research Directions. *ACM Comput. Surv.*, 48(2):18:1—-18:41, sep 2015. ISSN 0360-0300. doi: 10.1145/2794381.

[186] Henrik Nilsson. Functional automatic differentiation with dirac impulses. *ACM SIGPLAN Notices*, 38(9):153–164, sep 2003. ISSN 03621340. doi: 10.1145/944746.944720.

[187] Kristoffer Norling, David Broman, Peter Fritzson, Alexander Siemers, and Dag Fritzson. Secure distributed co-simulation over wide area networks. In *Proceedings of the 48th Conference on Simulation and Modelling (SIMS 2007)*, pages 14–23. Citeseer, 2007.

[188] Thierry Nouidui, Michael Wetter, and Wangda Zuo. Functional mock-up unit for co-simulation import in EnergyPlus. *Journal of Building Performance Simulation*, 7(3):192–202, may 2014. ISSN 1940-1493. doi: 10.1080/19401493.2013.808265.

[189] James Nutaro. Designing power system simulators for the smart grid: Combining controls, communications, and electro-mechanical dynamics. In *2011 IEEE Power and Energy Society General Meeting*, pages 1–5. IEEE, jul 2011. ISBN 978-1-4577-1000-1. doi: 10.1109/PES.2011.6039456.

[190] James J. Nutaro. A method for bounding error in multi-rate and federated simulations. *Proceeding of the 2016 Winter Simulation Conference*, page to be accepted, 2016.

[191] Seaseung Oh and Suyong Chae. A Co-Simulation Framework for Power System Analysis. *Energies*, 9(3):131, 2016.

[192] Xiufeng Pang, Michael Wetter, Prajesh Bhattacharya, and Philip Haves. A framework for simulation-based real-time whole building performance assessment. *Building and Environment*, 54:100–108, aug 2012. ISSN 03601323. doi: 10.1016/j.buildenv.2012.02.003.

[193] Henry M. Paynter. *Analysis and design of engineering systems.* MIT press, 1961.

[194] Nicolai Pedersen, Jan Madsen, and Morten Vejlgaard-Laursen. Co-Simulation of Distributed Engine Control System and Network Model using FMI & SCNSL. *IFAC-PapersOnLine*, 48 (16):261–266, 2015. ISSN 24058963. doi: 10.1016/j.ifacol.2015.10.290.

[195] Nicolai Pedersen, Tom Bojsen, Jan Madsen, and Morten Vejlgaard-Laursen. FMI for Co-Simulation of Embedded Control Software. In *The First Japanese Modelica Conferences, May 23-24, Tokyo, Japan*, number 124, pages 70–77. Linköping University Electronic Press, may 2016. ISBN 1650-3740. doi: 10.3384/ecp1612470.

[196] Nikos C. Petrellis, Alexis N. Birbas, Michael K. Birbas, Evangelinos P. Mariatos, and George D. Papadopoulos. Simulating Hardware, Software and Electromechanical Parts Using Communicating Simulators. *Design Automation for Embedded Systems*, 3(2/3):187–198, 1998. ISSN 09295585. doi: 10.1023/A:1008846508549.

[197] Ralf Uwe Pfau. A priori step size adaptation for the simulation of non-smooth systems. *Communications in Numerical Methods in Engineering*, 23(2):85–96, jun 2006. ISSN 10698299. doi: 10.1002/cnm.884.

[198] K. Pierce, C. Gamble, Y. Ni, and J. F. Broenink. Collaborative Modelling and Co-simulation with DESTECS: A Pilot Study. In *IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pages 280–285, 2012. ISBN 1524-4547 VO -. doi: 10.1109/WETICE.2012.69.

[199] Uwe Pohlmann, Wilhelm Schäfer, Hendrik Reddehase, Jens Röckemann, and Robert Wagner. Generating functional mockup units from software specifications. In *Proceedings of the 9th International MODELICA Conference, September 3-5, 2012, Munich, Germany*, number 078, pages 765–774, 2012.

[200] Claudius Ptolemaeus. *System Design, Modeling, and Simulation: Using Ptolemy II.* Berkeley: Ptolemy.org, 2014. ISBN 1304421066.

[201] Davide Quaglia, Riccardo Muradore, Roberto Bragantini, and Paolo Fiorini. A SystemC/-Matlab co-simulation tool for networked control systems. *Simulation Modelling Practice and Theory*, 23:71–86, apr 2012. ISSN 1569-190X. doi: 10.1016/j.simpat.2012.01.003.

[202] Gauthier Quesnel, Raphaël Duboz, David Versmisse, and E Ramat. DEVS coupling of spatial and ordinary differential equations: VLE framework. *OICIMS*, 5:281–294, 2005.

[203] M. Radetzki and R. S. Khaligh. Accuracy-adaptive Simulation of Transaction Level Models. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '08, pages 788–791, New York, NY, USA, 2008. ACM. ISBN 978-3-9810801-3-1. doi: 10.1145/1403375.1403566.

[204] Derek Riley, Emeka Eyisi, Jia Bai, Xenofon Koutsoukos, Yuan Xue, and Janos Sztipanovits. Networked Control System Wind Tunnel (NCSWT): An Evaluation Tool for Networked Multi-agent Systems. In *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*, SIMUTools '11, pages 9–18, ICST, Brussels, Belgium, Belgium, 2011. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). ISBN 978-1-936968-00-8.

[205] Robin Roche, Sudarshan Natarajan, Ayan Bhattacharyya, and Siddharth Suryanarayanan. A Framework for Co-simulation of AI Tools with Power Systems Analysis Software. In *23rd International Workshop on Database and Expert Systems Applications*, pages 350–354. IEEE, sep 2012. ISBN 978-1-4673-2621-6. doi: 10.1109/DEXA.2012.9.

[206] J. A. Rowson. Hardware/Software Co-Simulation. In *31st Conference on Design Automation*, pages 439–440, 1994. ISBN 0738-100X VO -. doi: 10.1109/DAC.1994.204143.

[207] Severin Sadjina, Lars T. Kyllingstad, Eilif Pedersen, and Stian Skjong. Energy Conservation and Power Bonds in Co-Simulations: Non-Iterative Adaptive Step Size Control and Error Estimation. *arXiv preprint arXiv:1602.06434*, 2016.

[208] Salah Eddine Saidi, Nicolas Pernet, Yves Sorel, and Abir Ben Khaled. Acceleration of FMU Co-Simulation On Multi-core Architectures. In *The First Japanese Modelica Conferences, May 23-24, Tokyo, Japan*, number 124, pages 106–112. Linköping University Electronic Press, may 2016. ISBN 1650-3740. doi: 10.3384/ecp16124106.

[209] Resve A Saleh, Shyh-Jye Jou, and A Richard Newton. *Mixed-mode simulation and analog multilevel simulation*, volume 279. Springer Science & Business Media, 2013. ISBN 1475758545.

[210] Tom Schierz and Martin Arnold. Stabilized overlapping modular time integration of coupled differential-algebraic equations. *Applied Numerical Mathematics*, 62(10):1491–1502, oct 2012. ISSN 01689274. doi: 10.1016/j.apnum.2012.06.020.

[211] Tom Schierz, Martin Arnold, and Christoph Clauss. Co-simulation with communication step size control in an FMI compatible master algorithm. In *9th Int. Modelica Conf., Munich, Germany*, pages 205–214, nov 2012. doi: 10.3384/ecp12076205.

[212] Robert Schmoll and Bernhard Schweizer. Convergence Study of Explicit Co-Simulation Approaches with Respect to Subsystem Solver Settings. *PAMM*, 12(1):81–82, dec 2012. ISSN 1617-7061. doi: 10.1002/pamm.201210032.

[213] Stefan-Alexander Schneider, Johannes Frimberger, and Michael Folie. Significant Reduction of Validation Efforts for Dynamic Light Functions with FMI for Multi-Domain Integration and Test Platforms. In *10th International Modelica Conference*, 2014.

[214] B. Schweizer and D. Lu. Predictor/corrector co-simulation approaches for solver coupling with algebraic constraints. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift*

für Angewandte Mathematik und Mechanik, 95(9):911–938, sep 2015. ISSN 00442267. doi: 10.1002/zamm.201300191.

[215] Bernhard Schweizer and Daixing Lu. Semi-implicit co-simulation approach for solver coupling. *Archive of Applied Mechanics*, 84(12):1739–1769, dec 2014. ISSN 0939-1533. doi: 10.1007/s00419-014-0883-5.

[216] Bernhard Schweizer and Daixing Lu. Stabilized index-2 co-simulation approach for solver coupling with algebraic constraints. *Multibody System Dynamics*, 34(2):129–161, jun 2015. ISSN 1384-5640. doi: 10.1007/s11044-014-9422-y.

[217] Bernhard Schweizer, Pu Li, and Daixing Lu. Explicit and Implicit Cosimulation Methods: Stability and Convergence Analysis for Different Solver Coupling Approaches. *Journal of Computational and Nonlinear Dynamics*, 10(5):051007, sep 2015. ISSN 1555-1415. doi: 10.1115/1.4028503.

[218] Bernhard Schweizer, Pu Li, Daixing Lu, and Tobias Meyer. Stabilized implicit co-simulation methods: solver coupling based on constitutive laws. *Archive of Applied Mechanics*, 85(11): 1559–1594, nov 2015. ISSN 0939-1533. doi: 10.1007/s00419-015-0999-2.

[219] Bernhard Schweizer, Daixing Lu, and Pu Li. Co-simulation method for solver coupling with algebraic constraints incorporating relaxation techniques. *Multibody System Dynamics*, 36(1): 1–36, jan 2016. ISSN 1384-5640. doi: 10.1007/s11044-015-9464-9.

[220] S. Sicklinger, V. Belsky, B. Engelmann, H. Elmqvist, H. Olsson, R. Wüchner, and K. U. Bletzinger. Interface Jacobian-based Co-Simulation. *International Journal for Numerical Methods in Engineering*, 98(6):418–444, may 2014. ISSN 1097-0207. doi: 10.1002/nme.4637.

[221] G. Stettinger, M. Horn, M. Benedikt, and J. Zehetner. Model-based coupling approach for non-iterative real-time co-simulation. In *2014 European Control Conference (ECC)*, pages 2084–2089, 2014. doi: 10.1109/ECC.2014.6862242.

[222] Georg Stettinger, Josef Zehetner, Martin Benedikt, and Norbert Thek. Extending Co-Simulation to the Real-Time Domain. apr 2013. doi: 10.4271/2013-01-0421.

[223] H T Su, H. T. Su, K W Chan, K. W. Chan, L A Snider, and L. A. Snider. Parallel interaction protocol for electromagnetic and electromechanical hybrid simulation, 2005.

[224] Yongqi Sun, Stephanie Vogel, and Haiko Steuer. Combining Advantages of Specialized Simulation Tools and Modelica Models using Functional Mock-up Interface (FMI). In *Proceedings of the 8th International Modelica Conference*, 2011.

[225] Robert Tarjan. Depth-first search and linear graph algorithms. *12th Annual Symposium on Switching and Automata Theory (swat 1971)*, 1(2), jun 1971. ISSN 0272-4847. doi: 10.1109/SWAT.1971.10.

[226] Jean-Philippe Tavella, Mathieu Caujolle, Charles Tan, Gilles Plessis, Mathieu Schumann, Stéphane Vialle, Cherifa Dad, Arnaud Cuccuru, and Sébastien Revol. Toward an Hybrid Co-simulation with the FMI-CS Standard, apr 2016. URL `https://hal-centralesupelec.archives-ouvertes.fr/hal-01301183`.

[227] Brook Taylor. Methodus Incrementorum Directa et Inversa. *London*, 1715.

[228] T Tomiyama, V. D'Amelio, J Urbanic, and W ElMaraghy. Complexity of Multi-Disciplinary Design. *CIRP Annals - Manufacturing Technology*, 56(1):185–188, 2007. ISSN 00078506. doi: 10.1016/j.cirp.2007.05.044.

[229] Paweł Tomulik and Janusz Frączek. Simulation of multibody systems with the use of coupling techniques: a case study. *Multibody System Dynamics*, 25(2):145–165, feb 2011. ISSN 1384-5640. doi: 10.1007/s11044-010-9206-y.

[230] Mamadou K Traoré and Alexandre Muzy. Capturing the dual relationship between simulation models and their context. *Simulation Modelling Practice and Theory*, 14(2):126–142, feb 2006. ISSN 1569190X. doi: 10.1016/j.simpat.2005.03.002.

[231] Marija Trcka, Michael Wetter, and Jan Hensen. Comparison of co-simulation approaches for building and HVAC/R system simulation. In *Proceedings of the International IBPSA Conference, Beijing, China*, 2007.

[232] Stavros Tripakis. Bridging the semantic gap between heterogeneous modeling formalisms and FMI. In *2015 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*, pages 60–69. IEEE, jul 2015. ISBN 978-1-4673-7311-1. doi: 10.1109/SAMOS.2015.7363660. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7363660`.

[233] F. Tseng and G. Hulbert. Network-distributed multibody dynamics simulation—gluing algorithm. *Advances in Computational Multibody Dynamics*, pages 521–540, 1999.

[234] A. M. Uhrmacher. Dynamic Structures in Modeling and Simulation: A Reflective Approach. *ACM Trans. Model. Comput. Simul.*, 11(2):206–232, apr 2001. ISSN 1049-3301. doi: 10.1145/384169.384173.

[235] Adelinde M. Uhrmacher. Variable structure models: autonomy and control answers from two different modeling approaches. In *4th Annual Conference on AI, Simulation and Planning in High Autonomy Systems*, pages 133–139. IEEE Comput. Soc. Press, 1993. ISBN 0-8186-4020-0. doi: 10.1109/AIHAS.1993.410588.

[236] Bert Van Acker, Joachim Denil, Paul De Meulenaere, Hans Vangheluwe, Bert Vanacker, and Paul Demeulenaere. Generation of an Optimised Master Algorithm for FMI Co-simulation. In *Proceedings of the Symposium on Theory of Modeling & Simulation-DEVS Integrative*, pages 946–953. Society for Computer Simulation International, 2015.

[237] Herman Van der Auweraer, Jan Anthonis, Stijn De Bruyne, and Jan Leuridan. Virtual engineering at work: the challenges for designing mechatronic products. *Engineering with Computers*, 29(3):389–408, 2013. ISSN 0177-0667. doi: 10.1007/s00366-012-0286-6.

[238] Arjan J. Van Der Schaft and Johannes Maria Schumacher. *An introduction to hybrid dynamical systems*, volume 251. Springer London, 2000.

[239] Simon Van Mierlo. Explicitly Modelling Model Debugging Environments. In *ACM Student Research Competition at MODELS 2015*, pages 24–29. CEUR, 2015.

[240] Yentl Van Tendeloo and Hans Vangheluwe. Activity in PythonPDEVS. In *Proceedings of ACTIMS 2014*, 2014.

[241] Yentl Van Tendeloo and Hans Vangheluwe. PythonPDEVS: a distributed Parallel DEVS simulator. In *Proceedings of the 2015 Spring Simulation Multiconference*, SpringSim '15, pages 844–851. Society for Computer Simulation International, 2015.

[242] Yentl Van Tendeloo and Hans Vangheluwe. An Introduction to Classic DEVS. Technical report, 2017. URL `https://arxiv.org/pdf/1701.07697v1.pdf`.

[243] Luigi Vanfretti, Tetiana Bogodorova, and Maxime Baudette. Power system model identification exploiting the Modelica language and FMI technologies. In *2014 IEEE International Conference on Intelligent Energy and Power Systems (IEPS)*, pages 127–132. IEEE, jun 2014. ISBN 978-1-4799-2266-6. doi: 10.1109/IEPS.2014.6874164.

[244] Hans Vangheluwe. DEVS as a common denominator for multi-formalism hybrid systems modelling. In *CACSD. Conference Proceedings. IEEE International Symposium on Computer-Aided Control System Design (Cat. No.00TH8537)*, pages 129–134. IEEE, 2000. ISBN 0-7803-6566-6. doi: 10.1109/CACSD.2000.900199.

[245] Hans Vangheluwe. Foundations of Modelling and Simulation of Complex Systems. *EASST*, 10, 2008. doi: 10.14279/tuj.eceasst.10.162.148.

[246] Hans Vangheluwe, Juan De Lara, and Pieter J. Mosterman. An introduction to multi-paradigm modelling and simulation. In *Proceedings of AIS2002 (AI, Simulation & Planning)*, pages 9–20. SCS, 2002.

[247] Ken Vanherpen, Joachim Denil, Hans Vangheluwe, and Paul De Meulenaere. Model Transformations for Round-trip Engineering in Control Deployment Co-Design. In *Theory of Modeling and Simulation - DEVS*, TMS/DEVS '15, part of the Spring Simulation Multi-Conference, pages 820–827, Alexandria, Virginia, USA, apr 2015. Society for Computer Simulation International.

[248] Julien Vaubourg, Yannick Presse, Benjamin Camus, Christine Bourjot, Laurent Ciarletta, Vincent Chevrier, Jean-Philippe Tavella, and Hugo Morais. Multi-agent Multi-Model Simulation of Smart Grids in the MS4SG Project. pages 240–251. Springer International Publishing, Cham, 2015. ISBN 978-3-319-18944-4. doi: 10.1007/978-3-319-18944-4_20.

[249] A Verhoeven, B Tasic, T G J Beelen, E J W ter Maten, and R M M Mattheij. BDF compound-fast multirate transient analysis with adaptive stepsize control. *J. Numer. Anal. Ind. Appl. Math*, 3(3-4):275–297, 2008.

[250] Antoine Viel. Implementing stabilized co-simulation of strongly coupled systems using the Functional Mock-up Interface 2.0. *10th International Modelica Conference*, 2014.

[251] B. Wang and J. S. Baras. HybridSim: A Modeling and Co-simulation Toolchain for Cyber-physical Systems. In *Distributed Simulation and Real Time Applications (DS-RT), 2013 IEEE/ACM 17th International Symposium on*, pages 33–40, 2013. ISBN 1550-6525. doi: 10.1109/DS-RT.2013.12.

[252] G. Wanner and E. Hairer. *Solving ordinary differential equations I*, volume 1. Springer-Verlag, Berlin, 1991.

[253] Michael Wetter. Co-simulation of building energy and control systems with the Building Controls Virtual Test Bed. *Journal of Building Performance Simulation*, 4(3):185–203, nov 2010. ISSN 1940-1493. doi: 10.1080/19401493.2010.518631.

[254] E. Widl, W. Muller, A. Elsheikh, M. Hortenhuber, and P. Palensky. The FMI++ library: A high-level utility package for FMI for model exchange. In *Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES), 2013 Workshop on*, pages 1–6, 2013. doi: 10.1109/MSCPES.2013.6623316.

[255] Xiaorong Xie, Chuanyu Zhang, Huakun Liu, Chao Liu, Dongxiang Jiang, and Baorong Zhou. Continuous-Mass-Model-Based Mechanical and Electrical Co-Simulation of SSR and Its Application to a Practical Shaft Failure Event. *IEEE Transactions on Power Systems*, 31(6): 5172–5180, nov 2016. ISSN 0885-8950. doi: 10.1109/TPWRS.2016.2537001.

[256] Masahiro Yamaura, Nikos Arechiga, Shinichi Shiraishi, Scott Eisele, Joseph Hite2 Sandeep Neema2 Jason Scott, and Theodore Bapty. ADAS Virtual Prototyping using Modelica and Unity Co-simulation via OpenMETA. 2016.

[257] Faruk Yılmaz, Umut Durak, Koray Taylan, and Halit Oğuztüzün. Adapting Functional Mockup Units for HLA-compliant Distributed Simulation. In *10th International Modelica Conference*, 2014.

[258] Josef Zehetner, Di Wenpu Lu, and Daniel Watzenig. Design of modern vehicle electrical systems based on co-simulation and a model library. *ATZelektronik worldwide*, 8(4):20–24, 2013. doi: 10.1365/s38314-013-0182-x.

[259] Bernard P. Zeigler. *Theory of modelling and simulation.* New York, Wiley, 1976. ISBN 0471981524.

[260] Bernard P. Zeigler. Embedding dev&dess in devs. In *Proc. DEVS Integrative M&S Symp*, volume 7, 2006.

[261] Bernard P. Zeigler and J. S. Lee. Theory of quantized systems: formal basis for DEVS/HLA distributed simulation environment. volume 3369, pages 49–58, aug 1998. doi: 10.1117/12.319354.

[262] Bernard P. Zeigler, Herbert Praehofer, and Tag Gon Kim. *Theory of modeling and simulation: integrating discrete event and continuous complex dynamic systems.* Academic press, 2 edition, 2000.

[263] Fu Zhang, Murali Yeddanapudi, and Pieter Mosterman. Zero-crossing location and detection algorithms for hybrid system simulation. In *IFAC World Congress*, pages 7967–7972, 2008.

[264] Saipeng Zhang, Jun Liu, and Wei Su. *A Co-simulation Platform of Integrated Starter/Generator System Based on ANSYS*, pages 35–43. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016. ISBN 978-3-662-49367-0. doi: 10.1007/978-3-662-49367-0_5.

[265] X. Zhang and J. F. Broenink. A structuring mechanism for embedded control systems using co-modelling and co-simulation. In *Proceedings of the 2nd International Conference on Simulation and Modeling Methodologies, Technologies and Applications, SIMULTECH 2012, Rome, Italy*, pages 131–136, Portugal, 2012. SciTePress.

[266] Zhenkai Zhang, Emeka Eyisi, Xenofon Koutsoukos, Joseph Porter, Gabor Karsai, and Janos Sztipanovits. A co-simulation framework for design of time-triggered automotive cyber physical systems. *Simulation Modelling Practice and Theory*, 43:16–33, 2014. ISSN 1569190X. doi: 10.1016/j.simpat.2014.01.001.

[267] Chenguang Zhao, Hongman Yan, Dong Liu, Hong Zhu, Yun Wang, and Yunhui Chen. Co-simulation research and application for Active Distribution Network based on Ptolemy II and Simulink. In *2014 China International Conference on Electricity Distribution (CICED)*, pages 1230–1235. IEEE, sep 2014. ISBN 978-1-4799-4126-1. doi: 10.1109/CICED.2014.6991903.

[268] Vojin Živojnovic and Heinrich Meyr. Compiled HW/SW Co-simulation. In *Proceedings of the 33rd Annual Design Automation Conference*, DAC '96, pages 690–695, New York, NY, USA, 1996. ACM. ISBN 0-89791-779-0. doi: 10.1145/240518.240649.

[269] M. Zwolinski, C. Garagate, Z. Mrcarica, T. J. Kazmierski, and A. D. Brown. Anatomy of a simulation backplane. *IEE Proceedings - Computers and Digital Techniques*, 142(6):377–385(8), nov 1995. ISSN 1350-2387.

Table 1: Historical Perspective of Co-simulation.

Time	Concept	Description
<80s	Single Formalism	The equations describing dynamic behavior are integrated together.
80s	Dynamic Iteration	Large circuits are decomposed into coupled constituent systems and dynamic iteration techniques are used [98, 155, 167, 170, 171, 183].
90s	Multi-Formalism	Software and Hardware are developed and simulated concurrently [65, 110, 206, 268] at multiple levels of abstraction [77, 117, 118]. Orchestration methods are explored in Carothers et al. [58], Fey et al. [84], Frey et al. [91], Tseng and Hulbert [233].
Late 90s and Early 2000s	Standard Interfaces	Recognized as key for co-simulation [116, 144, 196, 223, 269]
2010s	IP Protection, X-in-the-loop, and Scale	Important to enhance industrial applicability of co-simulation [7, 18–20, 74, 96, 237].

A Historical Perspective of Co-simulation

This section provides an historical perspective that relates the major concepts in co-simulation to the time at which they are recognized in the studied state of the art, summarized in table 1.

A.1 One Formalism and Dynamic Iteration

Traditionally, the equations describing the dynamical behavior of large circuits were integrated together. These systems are sparsely coupled, reflecting the connections of the corresponding circuits, and many techniques were developed that take advantage of this structure [167].

The crucial idea that improved the simulation speed in up to two orders of magnitude is to decompose the large system into a set of coupled constituent systems and integrate them independently.

The decomposition of the circuit implies the definition of inputs and outputs for each of the resulting constituent systems. The coupling is then the assignment of outputs to inputs.

For a subsystem S_i, we call the subsystems, whose outputs are assigned to any of the inputs of S_i, for neighbor subsystems.

The essence of the dynamic iteration approach is to integrate each subsystem independently, for a period of time $T_n \to T_{n+1}$, using the extrapolated outputs of the neighbor subsystems as inputs [155, 170, 171, 183].

Naturally, the fact that outputs are extrapolated introduces inaccuracy in the solution of the subsystem, so the integration can be repeated for the same period of time, with corrected outputs, until some form of convergence criteria is met [121]. The extrapolated outputs of a subsystem S_j can be corrected by collecting the outputs during the integration of S_j.

It is easy to see that this approach only requires communication between constituent systems at times T_n and T_{n+1} and that the integration of each subsystem can be done independently and in parallel [122], using any numerical method with any step size control policy. The signals exchanged are functions in the interval $[T_n, T_{n+1}]$.

The advantages of independent step size control policy become evident when one observes that many circuits have components that change at different rates. If the whole system were to be simulated, the simulation unit would have to use the smallest time step that ensures sufficient accuracy for the fastest changing component, which would be a huge waste of computational effort for the slow components. This is the similarity to multi-rate numerical methods [98].

To the best of our knowledge, dynamic iteration techniques and multi-rate numerical are the first to resemble co-simulation. The coordination software that implements these techniques expect any number of subsystems but assumes that the subsystems are all specific in the same formalism: differential equations.

A.2 Two Formalisms: Digital and Analog Co-simulation

Co-simulation, in its modern definition, was applied to enable the virtual development of coupled software and hardware systems [65, 110, 206, 268]. In this application domain, co-simulation decreases the need to build prototype board circuits to validate the composition of the software and the hardware part. It enables software and hardware to be developed and validated concurrently. To the best of our knowledge, this was one of the first uses of co-simulation in the modern sense. The co-simulation frameworks developed in this application domain typically assumed two simulation units and two formalisms.

The hardware/software systems quickly became more complex and a new idea was introduced: use multiple models at different levels of abstraction of each subsystem. Simulations could be made arbitrarily faster in some intervals by solving the more abstract models, and arbitrarily accurate in other intervals, by solving the more detailed ones [77, 135, 166, 209]. In the particular case of analog-digital co-simulation, each level of abstraction was solved by a different tool: a continuous time tool and a discrete event tool. The separation into continuous time and discrete event made the abstract synchronization problem and synchronization methods between simulation units in these two domains were developed [58, 84, 91, 233]. We could call these some of the first master algorithms.

A.3 Multi-abstraction/Multi-Formalism Co-simulation

The heterogeneity aspect of co-simulation comes into play at this time: multiple formalisms can be used to describe the same subsystem at multiple levels of abstraction: state machines can describe a rough approximation of the modes, while differential equations can describe the detailed dynamics of the electronic circuit. Depending on the purpose of the co-simulation, a subsystem and its neighbors can be solved in detail, whereas subsystems that are “farther away” can be simulated with higher levels of abstraction [117, 118]. For the domain of Hw/sw co-simulation, RTL and TLM classify the multiple abstraction levels of models [27, 203] and switching between these multiple levels of abstraction have been studied in [131].

As the number and heterogeneity of simulation tools to be coupled increases, the need to provide a common interface to couple any number of tools is recognized in Hickey et al. [116], Kuhl et al. [144], Petrellis et al. [196], Zwolinski et al. [269] and later in Blochwitz et al. [34].

In parallel with the previous advancements, co-simulation has also been in use for heterogeneous physical systems, such as automotive [105, 142, 151, 213], railway [12, 71] and HVAC [107, 231], to name just a few. The common motivation is the fact that co-simulation enables specialized simulation units to cooperatively simulated the system, with huge savings in time and cost, when compared to a monolithic modeling approach.

A.4 Black-box Co-simulation

Later, distributed and concurrent development processes, enabled by co-simulation, are studied and IP protection is identified as a desired characteristic [7, 237] to enable suppliers and integrators to exchange co-simulation units without having to disclose sensitive information and avoiding *vendor lock-in* contracts.

A.5 Real-time Co-simulation

Furthermore, co-simulation is used at every stage of the development process, from early system validation, to X-in-the-Loop co-simulation, bringing hard real-time constraints to the set of challenges [74].

A.6 Many simulation units: Large Scale Co-simulation

More recently, with the acknowledgment that there is a need to be able to simulate even larger systems of systems, scale and distribution become inherent challenges in co-simulation [18–20, 96].

B State of the Art in Co-simulation Frameworks

This section provides the detailed classification of each reference.

Ref 1. Pedersen2015: Co-Simulation of Distributed Engine Control System and Network Model using FMI & SCNSL [194].

Summary. This work describes a co-simulation master in the context of the maritime industry.
sr : causality : causal
sr : rel_time : analytic
sr : rollback : none
sr : availability : local
fr : results_visualization : post_mortem
fr : alg_loop : explicit
fr : sim_step_size : fixed
fr : sim_rate : single
fr : domain:ct
fr : coupling_model:io_assignments
fr : standard:fmi
fr : communication_model:jacobi

fr : num_sim:three_more

□

Ref 2. Lin2011: Power system and communication network co-simulation for smart grid applications [120].

Summary. This work describes a co-simulation between power system and network simulator.
sr : info : predict_step_sizes
sr : causality : causal
sr : rel_time : analytic
sr : rollback : none
sr : availability : local
fr : results_visualization : post_mortem
fr : alg_loop : explicit
fr : sim_rate : single
fr : coupling_model:io_assignments
fr : num_sim:two
fr : domain:de
fr : sim_step_size : variable
fr : communication_model:gauss_seidel

□

Ref 3. Hoepfer2011: Towards a Comprehensive Framework for Co- Simulation of Dynamic Models With an Emphasis on Time Stepping [119].

Summary. This work describes a co-simulation approach that finds an appropriate co-simulation step size.
nfr : performance
nfr : accuracy
nfr : ip_protection
sr : availability : local
sr : info : derivatives : out
sr : info : derivatives : state
sr : info : statevars
sr : causality : causal
sr : rollback : none
sr : rel_time : analytic
fr : num_sim:three_more
fr : domain:ct
fr : sim_rate : single
fr : sim_step_size : variable
fr : communication_model:jacobi
fr : communication_model:gauss_seidel
fr : alg_loop : explicit
fr : results_visualization : post_mortem

□

Ref 4. Faure2011: Methods for real-time simulation of Cyber-Physical Systems: application to automotive domain [82].

Summary. This work addresses co-simulation with real-time simulators.
nfr : performance
nfr : parallelism
sr : availability : local
sr : rel_time : fixed_real_scaled_time_simulation
sr : info : wcet
sr : rollback : none
fr : results_visualization : post_mortem
fr : num_sim:three_more
fr : coupling_model:io_assignments
fr : domain:ct
fr : communication_model:jacobi
fr : alg_loop : explicit
fr : sim_rate : single
fr : sim_step_size : fixed

□

Ref 5. Tomulik2011: Simulation of multibody systems with the use of coupling techniques: a case study [229].

Summary. This work discusses a co-simulation method for couplings with algebraic constraints. One of the results is that this kind of coupling should be done with many derivatives of the coupling variables.
nfr : accuracy
sr : rollback : single
sr : availability : local
sr : rel_time : analytic
sr : causality : causal
sr : info : derivatives : out
sr : info : jacobian : out
fr : results_visualization : post_mortem
fr : communication_model:jacobi
fr : alg_loop : implicit
fr : coupling_model: algebraic_constraints
fr : domain:ct
fr : num_sim:three_more
fr : sim_rate : single
fr : sim_step_size : fixed

□

Ref 6. Sun2011: Combining Advantages of Specialized Simulation Tools and Modelica Models using Functional Mock-up Interface (FMI) [224].

Summary. This work describes the application of co-simulation to the power production domain.
nfr : ip_protection
nfr : performance
sr : causality : causal
sr : rel_time : analytic
sr : rollback : none
sr : availability : local
fr : results_visualization : post_mortem
fr : sim_rate : single
fr : coupling_model:io_assignments
fr : standard:fmi
fr : domain:ct
fr : communication_model:gauss_seidel
fr : num_sim:two
fr : alg_loop : explicit
fr : sim_step_size : variable

□

Ref 7. Bastian2011a: Master for Co-Simulation Using FMI [25].

Summary. This work describes a co-simulation approach.
nfr : ip_protection
nfr : platform_independence
nfr : parallelism
sr : causality : causal
sr : rollback : none
sr : info : stateserial
sr : info : jacobian : out
sr : availability : local
fr : results_visualization : post_mortem
fr : coupling_model:io_assignments
fr : sim_step_size : fixed
fr : sim_rate : single
fr : alg_loop : implicit
fr : domain:ct
fr : num_sim:three_more
fr : standard:fmi
fr : communication_model:jacobi
fr : communication_model:gauss_seidel

□

Ref 8. Friedrich2011: Parallel Co-Simulation for Mechatronic Systems [93].

Summary. This work describes a co-simulation framework based on the Jacobi iteration scheme.
nfr : parallelism
nfr : performance
nfr : distribution

nfr : ip_protection
sr : availability : remote
sr : causality : causal
sr : rel_time : analytic
sr : rollback : none
fr : results_visualization : post_mortem
fr : alg_loop : explicit
fr : domain:ct
fr : coupling_model:io_assignments
fr : coupling_model: algebraic_constraints
fr : num_sim:three_more
fr : communication_model:jacobi
fr : sim_rate : single
fr : sim_step_size : fixed

□

Ref 9. Gonzalez2011: On the effect of multirate co-simulation techniques in the efficiency and accuracy of multibody system dynamics [103].

Summary. This work deals with multi-rate co-simulation. Essentially, one of the simulators (the fast one) drives the simulation, while the slow one provides extrapolated inputs, to avoid excessive computation.
nfr : accuracy
nfr : performance
sr : causality : causal
sr : rollback : none
sr : availability : local
sr : rel_time : analytic
fr : results_visualization : post_mortem
fr : alg_loop : explicit
fr : communication_model:gauss_seidel
fr : coupling_model:io_assignments
fr : num_sim:two
fr : domain:ct
fr : sim_rate : multi
fr : sim_step_size : fixed

□

Ref 10. Nutaro2011: Designing power system simulators for the smart grid: Combining controls, communications, and electro-mechanical dynamics [189].

Summary. This work describes a tool that is formed by the coupling of a DEVS simulator with some other modules that wrap CT as DEVS simulators.
nfr : distribution
nfr : accuracy
sr : causality : causal
sr : rollback : single

sr : rel_time : analytic
sr : availability : local
fr : alg_loop : explicit
fr : domain:de
fr : communication_model:gauss_seidel
fr : num_sim:two
fr : sim_rate : single
fr : sim_step_size : variable
fr : coupling_model:io_assignments
fr : results_visualization : post_mortem

□

Ref 11. Busch2012: Asynchronous method for the coupled simulation of mechatronic systems [54].

Summary. This work describes co-simulation approaches between two simulation tools. The main contribution is a semi-implicit method that applies a correction based on the jacobian of the subsystem's coupling variables.
nfr : accuracy
nfr : distribution
sr : causality : causal
sr : rollback : single
sr : availability : remote
sr : info : jacobian : out
sr : rel_time : analytic
fr : sim_rate : single
fr : sim_step_size : fixed
fr : results_visualization : post_mortem
fr : communication_model:gauss_seidel
fr : alg_loop : semi_implicit
fr : alg_loop : explicit
fr : alg_loop : implicit
fr : coupling_model:io_assignments
fr : domain:ct
fr : num_sim:two

□

Ref 12. Pohlmann2012: Generating functional mockup units from software specifications [199].

Summary. This work describes an application of co-simulation to robotics. □

Ref 13. Schmoll2012: Convergence Study of Explicit Co-Simulation Approaches with Respect to Subsystem Solver Settings [212].

Summary. This paper describes global error analysis for co-simulation, that takes into account sub-system solvers (instead of analytical solvers, as more commonly done).
nfr : accuracy
sr : rollback : none
sr : info : full_model

sr : causality : causal
sr : rel_time : analytic
sr : availability : local
fr : alg_loop : explicit
fr : domain:ct
fr : num_sim:two
fr : coupling_model:io_assignments
fr : communication_model:jacobi
fr : sim_rate : single
fr : sim_step_size : fixed
fr : results_visualization : post_mortem

□

Ref 14. Ni2012: Hybrid systems modelling and simulation in DESTECS: a co-simulation approach [184].

Summary. This work present a coupling of the tools Crescendo and 20-sim.
sr : causality : causal
sr : rel_time : analytic
sr : rollback : none
sr : availability : local
fr : communication_model:gauss_seidel
fr : results_visualization : post_mortem
fr : coupling_model:io_assignments
fr : alg_loop : explicit
fr : domain:ct
fr : sim_rate : single
fr : sim_step_size : fixed
fr : num_sim:two

□

Ref 15. Hassairi2012: Matlab/SystemC for the New Co-Simulation Environment by JPEG Algorithm [115].

Summary. This work introduces guidelines for the implementation of co-simulation between Matlab and SystemC. The case study is the JPEG Algorithm.
sr : info : full_model
sr : causality : causal
sr : rel_time : analytic
sr : availability : local
fr : domain:ct
fr : coupling_model:io_assignments
fr : sim_step_size : fixed
fr : sim_rate : single
fr : alg_loop : explicit
fr : communication_model:gauss_seidel
fr : results_visualization : live

fr : num_sim:two

□

Ref 16. Schierz2012: Stabilized overlapping modular time integration of coupled differential-algebraic equations [210].

Summary. This work discusses co-simulation techniques for simulators coupled via algebraic constraints.
nfr : accuracy
sr : availability : local
sr : rel_time : analytic
sr : causality : causal
sr : rollback : none
sr : info : full_model
sr : info : jacobian:out
fr : results_visualization : post_mortem
fr : sim_rate : single
fr : num_sim:three_more
fr : alg_loop : explicit
fr : sim_step_size : fixed
fr : domain:ct
fr : coupling_model: algebraic_constraints
fr : communication_model:gauss_seidel
fr : communication_model:jacobi

□

Ref 17. Gunther2012: A Modular Technique for Automotive System Simulation [108].

Summary. This work describes the MDPCosim framework.
nfr : performance
The decomposition of the system for co-simulation is done for performance reasons.
nfr : parallelism
nfr : accuracy
sr : availability : local
IPC communication is used.
sr : causality : causal
sr : info : derivatives : out
sr : rollback : none
sr : info : predict_step_sizes
fr : results_visualization : post_mortem
fr : alg_loop : explicit
fr : coupling_model:io_assignments
fr : domain:ct
fr : communication_model:jacobi
fr : sim_step_size : variable
The step size control approach is based on looking at the derivatives.
fr : sim_rate : single

fr : num_sim:three_more

□

Ref 18. Quaglia2012: A SystemC/Matlab co-simulation tool for networked control systems [201].

Summary. Work describing another tool coupling.
nfr : distribution
sr : causality : causal
sr : rel_time : analytic
sr : rollback : none
sr : availability : remote
fr : results_visualization : post_mortem
fr : alg_loop : explicit
fr : coupling_model:io_assignments
fr : domain:ct
fr : num_sim:two
fr : sim_rate : single
fr : sim_step_size : fixed
fr : communication_model:gauss_seidel

□

Ref 19. Al-Hammouri2012: A comprehensive co-simulation platform for cyber-physical systems [5].

Summary. The work describes the integration of two tools: Modelica, and NS-2.
sr : causality : causal
sr : rel_time : analytic
sr : availability : local
Communication is done over named pipes.
sr : rollback : none
fr : results_visualization : post_mortem
fr : num_sim:two
fr : coupling_model:io_assignments
fr : alg_loop : explicit
fr : domain:ct
fr : communication_model:gauss_seidel
fr : domain:de
fr : sim_rate : single
fr : sim_step_size : variable

□

Ref 20. Eyisi2012: NCSWT: An integrated modeling and simulation tool for networked control systems [81].

Summary. This work describes the coupling of two tools: Matlab and NS-2. The coupling is done through HLA standard. The preliminary version of the tool is described in [204].
nfr : platform_independence
nfr : performance

nfr : distribution
sr : info : predict_step_sizes
sr : causality : causal
sr : rel_time : analytic
sr : rollback : none
sr : availability : remote
fr : alg_loop : explicit
fr : num_sim:two
fr : coupling_model:io_assignments
fr : domain:de
fr : standard:hla
fr : communication_model:gauss_seidel
fr : sim_rate : single
fr : sim_step_size : variable
fr : results_visualization : live

□

Ref 21. Riley2011: Networked Control System Wind Tunnel (NCSWT): An Evaluation Tool for Networked Multi-agent Systems [204].

Summary. This work describes the coupling of two tools: Matlab and NS-2. The coupling is done through HLA standard.
nfr : platform_independence
nfr : performance
nfr : distribution
sr : info : predict_step_sizes
sr : causality : causal
sr : rel_time : analytic
sr : rollback : none
sr : availability : remote
fr : alg_loop : explicit
fr : num_sim:two
fr : coupling_model:io_assignments
fr : domain:de
fr : standard:hla
fr : communication_model:gauss_seidel
fr : sim_rate : single
fr : sim_step_size : variable
fr : results_visualization : live

□

Ref 22. Roche2012: A Framework for Co-simulation of AI Tools with Power Systems Analysis Software [205].

Summary. This work describes a co-simulation between two tools in the power grid domain with matlab running the co-simulation.
nfr : distribution

nfr : open_source
sr : causality : causal
sr : rel_time : analytic
sr : rollback : none
sr : availability : remote
fr : coupling_model:io_assignments
fr : domain:ct
fr : communication_model:gauss_seidel
fr : alg_loop : explicit
fr : results_visualization : post_mortem
fr : num_sim:two
fr : sim_rate : single
fr : sim_step_size : fixed

□

Ref 23. Fitzgerald2010: Collaborative Modelling and Co-simulation in the Development of Dependable Embedded Systems [86].

Summary. This work describes the coupling between two tools: Overture and 20-sim.
nfr : accuracy
nfr : distribution
nfr : platform_independence
sr : availability : remote
sr : causality : causal
sr : rel_time : analytic
fr : sim_step_size : variable
fr : sim_rate : single
fr : domain:de
fr : domain:ct
fr : num_sim:two
fr : coupling_model:io_assignments
fr : alg_loop : explicit
fr : results_visualization : live
fr : communication_model:gauss_seidel

□

Ref 24. Fitzgerald2013: A formal approach to collaborative modelling and co-simulation for embedded systems [89].

Summary. This work describes the coupling between two tools: Overture and 20-sim; already described in [86].
nfr : accuracy
nfr : distribution
nfr : platform_independence
sr : availability : remote
sr : causality : causal
sr : rel_time : analytic

fr : sim_step_size : variable
fr : sim_rate : single
fr : domain:de
fr : domain:ct
fr : num_sim:two
fr : coupling_model:io_assignments
fr : alg_loop : explicit
fr : results_visualization : live
fr : communication_model:gauss_seidel

□

Ref 25. Kudelski2013: RoboNetSim: An integrated framework for multi-robot and network simulation [143].

Summary. This work describes the integration of three simulators (ARGoS, NS-2 and NS-3) that can be used in co-simulation scenarios with two simulators.
nfr : distribution
nfr : platform_independence
sr : rollback : none
sr : causality : causal
sr : rel_time : analytic
sr : availability : remote
fr : results_visualization : post_mortem
fr : alg_loop : explicit
fr : communication_model:jacobi
fr : domain:de
fr : num_sim:two
fr : sim_rate : single
fr : sim_step_size : fixed
fr : coupling_model:io_assignments

□

Ref 26. Broman2013: Determinate Composition of FMUs for Co-simulation [46].

Summary. This work describes a master algorithm that ensures a determinate execution.
nfr : ip_protection
sr : availability : local
sr : rollback : none
sr : rel_time : analytic
sr : causality : causal
sr : info : causality : feedthrough
sr : info : predict_step_sizes
sr : info : stateserial
fr : coupling_model:io_assignments
fr : standard:fmi
fr : sim_step_size : variable
fr : domain:ct

fr : communication_model:jacobi
fr : num_sim:three_more
fr : sim_rate : single
fr : alg_loop : explicit
fr : results_visualization : post_mortem

□

Ref 27. Benedikt2013: Guidelines for the Application of a Coupling Method for Non-iterative Co-simulation [30].

Summary. This work describes a co-simulation approach where energy information about the signals is used, and those errors are compensated in a corrector step.
nfr : accuracy
nfr : performance
sr : rollback : none
sr : rel_time : analytic
sr : causality : causal
sr : info : record_outputs
sr : availability : local
fr : results_visualization : post_mortem
fr : alg_loop : explicit
fr : num_sim:two
fr : coupling_model:io_assignments
fr : communication_model:gauss_seidel
fr : domain:ct
fr : sim_rate : single
fr : sim_step_size : fixed

□

Ref 28. Benedikt2013b: Macro-step-size selection and monitoring of the coupoling error for weak coupled subsystems in the frequency-domain [29].

Summary. The work describes a method for finding appropriate communication step sizes in co-simulations between LTI systems. Essentially, it provides rules of thumb to chose a communication step size based on the maximum instantaneous frequency of components.
nfr : accuracy
sr : availability : local
sr : rollback : none
sr : rel_time : analytic
sr : causality : causal
sr : info : frequency_outputs
fr : results_visualization : post_mortem
fr : sim_rate : single
fr : coupling_model:io_assignments
fr : alg_loop : explicit
fr : communication_model:gauss_seidel
fr : domain:ct

fr : sim_step_size : fixed
fr : num_sim:two

□

Ref 29. Fuller2013: Communication simulations for power system applications [95].

Summary. This work describes a co-simulation between two co-simulation tools (ns-3 and GridLAB-D) for smart grid development.
nfr : scalability
nfr : faulttolerance
nfr : ip_protection
nfr : distribution
The tools keeps track of messages in transit.
sr : causality : causal
sr : rel_time : analytic
sr : availability : remote
sr : rollback : none
fr : coupling_model:io_assignments
fr : alg_loop : explicit
fr : domain:de
fr : num_sim:two
fr : sim_rate : single
fr : sim_step_size : variable
fr : results_visualization : post_mortem
fr : communication_model:gauss_seidel

□

Ref 30. Bombino2013: A model-driven co-simulation environment for heterogeneous systems [39].

Summary. This work describes the coupling between two simulation tools.
nfr : distribution
sr : rollback : none
sr : causality : causal
sr : rollback : single
sr : rel_time : dy_real_scaled_time_simulation
sr : availability : remote
fr : alg_loop : explicit
fr : coupling_model:io_assignments
fr : communication_model:gauss_seidel
fr : domain:ct
fr : num_sim:two
fr : results_visualization : interactive_live
fr : sim_rate : single
fr : sim_step_size : fixed

□

Ref 31. Wang2013: HybridSim: A Modeling and Co-simulation Toolchain for Cyber-physical Systems [251].

Summary. The approach described in this reference allows to arrange and process co-simulation units, Modelica models and TinyOS applications. SysML is used to configure the co-simulation master. The coordination of simulators is done through the FMI standard.

nfr : ip_protection
nfr : config_reusability
sr : causality : causal
sr : rel_time : analytic
sr : rollback : none
sr : availability : local
fr : coupling_model:io_assignments
fr : alg_loop : explicit
fr : communication_model:gauss_seidel
fr : num_sim:two
fr : domain:ct
fr : standard:fmi
fr : sim_rate : single
fr : sim_step_size : variable
fr : results_visualization : post_mortem

□

Ref 32. Hafner2013: An Investigation on Loose Coupling Co-Simulation with the BCVTB [114].

Summary. This work discusses the consistency and stability of the Jacobi and Gauss-Seidel co-simulation methods. Later, it presents a case study in HVAC systems.

nfr : accuracy
sr : causality : causal
sr : rel_time : analytic
sr : rollback : none
sr : availability : local
fr : coupling_model:io_assignments
fr : results_visualization : post_mortem
fr : communication_model:jacobi
fr : communication_model:gauss_seidel
fr : alg_loop : explicit
fr : domain:ct
fr : num_sim:three_more
fr : sim_rate : single
fr : sim_step_size : fixed

□

Ref 33. Zhao2014: Co-simulation research and application for Active Distribution Network based on Ptolemy II and Simulink [267].

Summary. This work describes the co-simulation between Ptolemy II and Simulink.

nfr : distribution
sr : causality : causal
sr : rel_time : analytic

sr : rollback : none
sr : availability : remote
fr : coupling_model:io_assignments
fr : domain:ct
fr : communication_model:gauss_seidel
fr : num_sim:two
fr : sim_rate: single
fr : sim_step_size : fixed
fr : alg_loop : explicit
fr : results_visualization : post_mortem

□

Ref 34. Li2011c: VPNET: A co-simulation framework for analyzing communication channel effects on power systems [157].

Summary. This work describes the coupling of two simulation tools (VTB and OPNET) to achieve co-simulation.
sr : availability : local
sr : rollback : none
sr : causality : causal
sr : rel_time : analytic
fr : results_visualization : post_mortem
fr : alg_loop : explicit
fr : coupling_model:io_assignments
fr : communication_model:gauss_seidel
fr : domain:ct
The coordination is a sample discrete time system.
fr : num_sim:two
fr : sim_rate: single
fr : sim_step_size : fixed

□

Ref 35. Awais2013b: Distributed hybrid simulation using the HLA and the Functional Mock-up Interface [20].

Summary. The main difference between this work and [18] is that this proposes a variable step size wrapper around CT components. The approach taken to do this is quantization.
nfr : distribution
nfr : parallelism
sr : rollback : none
sr : causality : causal
sr : rel_time : analytic
sr : availability : remote
sr : availability : local
fr : alg_loop : explicit
fr : communication_model:gauss_seidel
fr : communication_model:jacobi

fr : standard:fmi
fr : standard:hla
fr : num_sim:three_more
fr : sim_rate : multi
fr : sim_step_size : variable
fr : coupling_model:io_assignments
fr : domain:de
fr : results_visualization : post_mortem

□

Ref 36. Awais2013a: Using the HLA for Distributed Continuous Simulations [18].

Summary. This work addresses the need to adapt CT simulators as DE simulators, in order to be used in a hybrid co-simulation scenario that is fundamentally DE oriented.
nfr : distribution
nfr : parallelism
sr : rollback : none
sr : causality : causal
sr : rel_time : analytic
sr : availability : remote
sr : availability : local
fr : alg_loop : explicit
fr : communication_model:gauss_seidel
fr : communication_model:jacobi
fr : standard:fmi
fr : standard:hla
fr : num_sim:three_more
fr : sim_rate : multi
fr : sim_step_size : fixed
fr : coupling_model:io_assignments
fr : domain:de
fr : results_visualization : post_mortem

□

Ref 37. Kuhr2013: FERAL - Framework for simulator coupling on requirements and architecture level [146].

Summary. They describe a framework that borrows many concepts from Ptolemy, but that is fundamentally event based co-simulation. It allows for the specialization of basic directors for the semantic adaptation of simulation units.
nfr : ip_protection
nfr : extensibility
sr : info : signal
sr : causality : causal
sr : rollback : none
sr : availability : local
sr : rel_time : analytic

fr : sim_rate : multi
fr : communication_model:gauss_seidel
fr : standard:fmi
fr : domain:de
fr : domain:ct
fr : num_sim:three_more
fr : sim_step_size : variable

□

Ref 38. Viel2014: Implementing stabilized co-simulation of strongly coupled systems using the Functional Mock-up Interface 2.0. [250].

Summary. This work describes the implementation of the method described in [10] in the context of the FMI standard.
nfr : accuracy
nfr : ip_protection
sr : info : jacobian : out
sr : info : input_extrapolation
sr : info : record_outputs
sr : info : stateserial
sr : causality : causal
sr : rel_time : analytic
sr : availability : local
sr : rollback : none
fr : domain:ct
fr : num_sim:three_more
fr : sim_rate : single
fr : sim_step_size : fixed
fr : alg_loop : implicit
fr : results_visualization : post_mortem
fr : coupling_model: algebraic_constraints
fr : communication_model:gauss_seidel
fr : standard:fmi

□

Ref 39. Sicklinger2014: Interface Jacobian-based Co-Simulation [220].

Summary. Describes a co-simulation method that makes use of the Jacobian information for fixed point computations.
nfr : performance
nfr : accuracy
sr : availability : local
sr : rel_time : analytic
sr : rollback : single
sr : info : jacobian : out
sr : causality : causal
fr : results_visualization : post_mortem

fr : communication_model:gauss_seidel
fr : communication_model:jacobi
fr : coupling_model: algebraic_constraints
fr : domain:ct
fr : alg_loop : implicit
fr : num_sim:three_more
fr : sim_rate: single
fr : sim_step_size : fixed

□

Ref 40. Zhang2014: A co-simulation framework for design of time-triggered automotive cyber physical systems [266].

Summary. The work describes a co-simulation that integrates SystemC and CarSim.
sr : availability : local
sr : rollback : none
sr : causality : causal
sr : info : full_model
sr : rel_time : analytic
fr : coupling_model:io_assignments
fr : domain:de
fr : domain:ct
fr : alg_loop : explicit
fr : results_visualization : post_mortem
fr : sim_step_size : fixed
fr : sim_rate: single
fr : num_sim:two

□

Ref 41. Kounev2015: A microgrid co-simulation framework [139].

Summary. Describes the coupling of two simulators written in MATLAB and OMNeT++.
nfr : performance
sr : availability : local
sr : rel_time : analytic
sr : rollback : none
The DEV's orchestration is conservative.
sr : causality : causal
sr : info : predict_step_sizes
fr : coupling_model:io_assignments
fr : results_visualization : post_mortem
fr : communication_model:gauss_seidel
fr : domain:de
fr : num_sim:two
fr : sim_rate: single
fr : sim_step_size : variable
fr : alg_loop : explicit

□

Ref 42. Bogomolov2015: Co-Simulation of Hybrid Systems with SpaceEx and Uppaal [36].

Summary. The orchestration algorithm is the one described in [46]. The work exploits the standard by allowing zero step transitions.

sr : info : causality : feedthrough
sr : rollback : none
sr : info : stateserial
sr : causality : causal
sr : rel_time : analytic
sr : availability : local
fr : coupling_model:io_assignments
fr : standard:fmi
fr : num_sim:two
fr : communication_model:jacobi
fr : alg_loop : explicit
fr : sim_rate : single
fr : sim_step_size : variable
This is due to the rejection of steps, not due to accuracy.
fr : domain:ct
fr : domain:de
They abuse the FMI standard to be able to support state transitions.
fr : results_visualization : post_mortem

□

Ref 43. Bian2015: Real-time co-simulation platform using OPAL-RT and OPNET for analyzing smart grid performance [33].

Summary. Not many details are provided about the co-simulation orchestration. However, due to the fact that it is real-time, we can infer certain features.

nfr : performance
sr : rollback : none
sr : causality : causal
sr : rel_time : fixed_real_scaled_time_simulation
sr : availability : remote
Communication is done through UDP.
fr : sim_step_size : fixed
fr : sim_rate : single
Two simulators do not give more than this.
fr : domain:ct
fr : coupling_model:io_assignments
fr : num_sim:two
fr : alg_loop : explicit
fr : results_visualization : post_mortem

□

Ref 44. Dols2016: Coupling the multizone airflow and contaminant transport software CONTAM with EnergyPlus using co-simulation [73].

Summary. The work described the coupling of the CONTAM and EnergyPlus tools to achieve HVAC simulation. The coupling is done through FMI. The coupling is done through the compiled binaries. The case study highlights the problems with an explicit method for co-simulation, even if the Gauss-seidel. Instabilities occur.

nfr : ip_protection
sr : rollback : none
sr : rel_time : analytic
sr : causality : causal
sr : availability : remote
fr : results_visualization : post_mortem
fr : domain:ct
fr : num_sim:two
fr : standard:fmi
fr : alg_loop : explicit
fr : coupling_model:io_assignments
fr : communication_model:gauss_seidel
fr : sim_step_size : fixed
It uses a 5-minute synchronization step.
fr : sim_rate : single

□

Ref 45. BenKhaled2012: Multicore simulation of powertrains using weakly synchronized model partitioning [133].

Summary. According to [28], this work explores variable step solvers.

nfr : parallelism
nfr : performance
sr : info : causality : feedthrough
sr : info : full_model
sr : rel_time : fixed_real_scaled_time_simulation
sr : rollback : none
sr : availability : local
fr : standard:fmi
fr : coupling_model:io_assignments
fr : num_sim:three_more
fr : domain:ct
fr : sim_rate : single
fr : sim_step_size : fixed
fr : alg_loop : explicit
fr : results_visualization : post_mortem
fr : communication_model:jacobi
fr : communication_model:gauss_seidel

□

Ref 46. BenKhaled2014: Fast multi-core co-simulation of Cyber-Physical Systems: Application to internal combustion engines [28].

Summary. This paper focus on the parallelization of co-simulation. The approach is to start with a single model and partition it into multiple models, which are then executed in separate FMUs in parallel. The partitioning is important for accuracy reasons (e.g., break the algebraic loops at less sensitive variables).

nfr : parallelism
nfr : accuracy
nfr : performance
nfr : scalability
sr : info : full_model
sr : info : wcet
sr : info : causality : feedthrough
sr : causality : causal
sr : rel_time : analytic
sr : rollback : none
sr : availability : local
fr : communication_model:gauss_seidel
fr : sim_step_size : fixed
fr : sim_rate : single
fr : standard:fmi
fr : domain:ct
fr : num_sim:three_more
fr : alg_loop : explicit
It breaks the loops by establishing an order and delaying one of the variables in the loop.

□

Ref 47. Saidi2016: Acceleration of FMU Co-Simulation On Multi-core Architectures [208].

Summary. The paper addresses the problem of performance in FMI co-simulation. The solution proposed is to go parallel. The parallelization approach is the same as the one presented in [133]. Since FMI does not enforce thread safety across multiple instances of the same FMU, the work presented ensures that these do not execute concurrently by using mutexes or changing the scheduling policy.

nfr : parallelism
nfr : performance
nfr : ip_protection
nfr : scalability
sr : info : wcet
sr : info : causality : feedthrough
sr : causality : causal
sr : rel_time : analytic
sr : rollback : none
sr : availability : local
fr : communication_model:jacobi
fr : standard:fmi
fr : domain:ct
fr : num_sim:three_more
fr : alg_loop : explicit

□

Ref 48. Yamaura2016: ADAS Virtual Prototyping using Modelica and Unity Co-simulation via OpenMETA [256].

Summary. The co-simulation framework includes 4 tools. The communication between the tools is realized using OpenMeta. The work uses Unity for the modelling and simulation of the environment, allowing for live interaction. Communication is over UDP but there is no report on extra caution due to network delays and failures.

nfr : parallelism
sr : info : full_model
sr : causality : causal
sr : rel_time : analytic
sr : rollback : none
sr : availability : remote
fr : num_sim:three_more
fr : domain:ct
fr : results_visualization : live
fr : results_visualization : interactive_live
fr : coupling_model:io_assignments
fr : sim_rate : single
fr : alg_loop : explicit

□

Ref 49. Camus2015: Combining DEVS with multi-agent concepts to design and simulate multi-models of complex systems (WIP) [55].

Summary. This work is the preliminary description of [56].

nfr : parallelism
nfr : distribution
nfr : ip_protection
nfr : accuracy
sr : rollback : none
sr : info : stateserial
sr : causality : causal
sr : rel_time : analytic
sr : availability : local
fr : domain:de
fr : alg_loop : explicit
fr : num_sim:three_more
fr : sim_rate : multi
fr : sim_step_size : variable
fr : standard:fmi
fr : coupling_model:io_assignments
fr : communication_model:gauss_seidel
fr : results_visualization : post_mortem

□

Ref 50. Camus2016: Hybrid Co-simulation of FMUs using DEV & DESS in MECSYCO [56].

Summary. It proposes to use a FMU wrapper around DEV and DESS models, meaning that the co-simulation proceeds using a DE approach. It handles black box FMUs and the algorithm used to drive the co-simulation is the conservative parallel DEVS simulator. It requires that the FMU is able to perform rollback (through the use of state set and get).

nfr : parallelism
nfr : distribution
nfr : ip_protection
nfr : accuracy
sr : rollback : none
sr : info : stateserial
sr : causality : causal
sr : rel_time : analytic
sr : availability : local
fr : domain:de
fr : alg_loop : explicit
fr : num_sim:three_more
fr : sim_rate : multi
fr : sim_step_size : variable
fr : standard:fmi
fr : coupling_model:io_assignments
fr : communication_model:gauss_seidel
fr : results_visualization : post_mortem

□

Ref 51. Pedersen2016: FMI for Co-Simulation of Embedded Control Software [195].

Summary. The paper describes the adaptation of an embedded system to comply with FMI and thus interface with other FMUs. To validate the implementation, they run a co-simulation.

nfr : distribution
nfr : parallelism
sr : rel_time : fixed_real_scaled_time_simulation
fr : domain:ct
fr : num_sim:two
fr : sim_rate : single
fr : sim_step_size : fixed
fr : communication_model:gauss_seidel
fr : standard:fmi
fr : coupling_model:io_assignments
fr : alg_loop : explicit
fr : results_visualization : live

□

Ref 52. Oh2016: A Co-Simulation Framework for Power System Analysis [191].

Summary. The paper proposes a co-simulation framework that takes into account network delays and compensates for that. It proposes to use cubic spline extrapolation to compensate for the delay but recognizes that if there are faults in the line (resulting in voltage drops), the derivatives used in the extrapolation assume gigantic proportions, thus wreaking havoc in the simulation. To address that, the framework employs an algorithm to detect discontinuities. The detection is simple: they check the derivative of the signal to see whether it exceeds a pre-determined empirically threshold. Basically, it looks for and Dirac delta. Figure 7 shows the effect of not handling a discontinuity.

nfr : distribution
nfr : accuracy
nfr : parallelism
fr : num_sim:two
fr : sim_rate : single
fr : sim_step_size : fixed
fr : communication_model:gauss_seidel
Due to the parallel interface protocol that they use.
fr : coupling_model:io_assignments
fr : domain:ct
fr : alg_loop : explicit
sr : availability : remote
sr : causality : causal
sr : rel_time : analytic
sr : rollback : none

□

Ref 53. Xie2016: Continuous-Mass-Model-Based Mechanical and Electrical Co-Simulation of SSR and Its Application to a Practical Shaft Failure Event [255].

Summary. Between two simulators. As it is explained in the paper, prior to co-simulation, the most common approach would be to run two simulations: one complete for one sub-system, and then another for the second sub-system, using the first as inputs. This is an open loop approach, whose results can be misleading due to ignoring the feedback loops. Each simulator advances in parallel and their communication is made with a barrier.

nfr : parallelism
fr : communication_model:jacobi
fr : num_sim:two
fr : domain:ct
fr : sim_step_size : fixed
fr : results_visualization : post_mortem
fr : sim_rate : single
sr : availability : local
sr : causality : causal
sr : rel_time : analytic
sr : rollback : none
fr : alg_loop : explicit
fr : coupling_model:io_assignments

□

Ref 54. Manbachi2016: Impact of EV penetration on Volt–VAR Optimization of distribution networks using real-time co-simulation monitoring platform [163].

Summary. It describes an application of co-simulation in the distribution of energy in smart grids, supported by a real-time co-simulation framework. The simulators involved are the RTDS, which simulates the distribution network model, and the VVO Engine, coded in MATLAB.

sr : rel_time : fixed_real_scaled_time_simulation
fr : num_sim:two
sr : causality : causal
sr : rollback : none
sr : availability : local
fr : coupling_model:io_assignments
fr : sim_rate : single
fr : sim_step_size : fixed
fr : alg_loop : explicit
fr : communication_model:jacobi

□

Ref 55. Schierz2012a: Co-simulation with communication step size control [211].

Summary. Describes a master algorithm. Does not allow for interpolation of inputs. Needs rollback. It touches upon accuracy, as it suggests an adaptive step size control mechanism. It does not address algebraic loops. It assumes that there is no feedthrough information.

nfr : performance
nfr : accuracy
sr : info : derivatives : out
sr : info : stateserial
sr : causality : causal
sr : rel_time : analytic
sr : rollback : single
sr : availability : local
fr : standard:fmi
fr : coupling_model:io_assignments
fr : num_sim:three_more
fr : domain:ct
fr : sim_rate : single
fr : sim_step_size : variable
fr : alg_loop : explicit
fr : results_visualization : post_mortem
fr : communication_model:jacobi

□

Ref 56. Fourmigue2009: Co-simulation based platform for wireless protocols design explorations [90].

Summary. Application of co-simulation to wireless network development. One of the simulators is the actual Linux operating system, and the other is represents a wireless network protocol simulator.
sr : causality : causal
sr : rel_time : analytic
sr : availability : local
fr : coupling_model:io_assignments
fr : num_sim:two
fr : domain:de
fr : communication_model:jacobi

□

Ref 57. Liu2001: Calculation of Wing Flutter by a Coupled Fluid-Structure Method [160].

Summary. A fully implicit method, dealing with parallelism.
nfr : parallelism
nfr : performance
sr : causality : causal
sr : rel_time : analytic
sr : rollback : single
sr : availability : remote
fr : coupling_model:io_assignments
fr : num_sim:two
fr : domain:ct
fr : alg_loop : implicit
fr : results_visualization : post_mortem

□

Ref 58. Carstens2003: Coupled simulation of flow-structure interaction in turbomachinery [59].

Summary. Relates to the application of a co-simulation algorithm to the simulation of the deformation in the blades of a transonic compressor rotor under airflow. One of the simulators calculates deformation of the blades, while the other calculates the flow dynamics around the blades.
The communication of orchestration algorithm in use is shifted by half a step.
nfr : performance
They highlight the need for it, because the computation of a rotor is just too expensive.
sr : info : derivatives : out
sr : causality : causal
sr : rel_time : analytic
sr : rollback : none
sr : availability : remote
It seems that they perform the computation in separate computers.
fr : coupling_model:io_assignments
fr : num_sim:two
fr : domain:ct
fr : sim_rate : single
fr : sim_step_size : fixed
fr : alg_loop : explicit

fr : results_visualization :post_mortem
fr :communication_model:gauss_seidel
Although it is a gauss seidel shifted in time. □

Ref 59. Stettinger2014: Model-based coupling approach for non-iterative real-time co-simulation [221].

Summary. Proposes to address the challenges in real-time co-simulation by using a model based coupling approach. The master has to keep track of two values for each packet of data: receiving time delay t_r – the time it takes for a packet to reach the master from the simulator –, and sending time delay t_s – the time it takes for a packet to leave the master and reach the simulator. When a sample is delayed, the master acts as a replacement for it. Basically, it is a dead reckoning model.
nfr :performance
nfr : parallelism
nfr :accuracy
sr : causality :causal
fr :domain:ct
fr :num_sim:two
sr : availability : local
fr :coupling_model:io_assignments
sr : rollback :none
sr : rel_time : fixed_real_scaled_time_simulation
fr :sim_rate: single
fr : sim_step_size : fixed
fr :alg_loop : explicit
fr : results_visualization :post_mortem
fr :communication_model:jacobi
fr :communication_model:gauss_seidel
□

Ref 60. Benedikt2016: Automated configuration for non-iterative co-simulation [31].

Summary. Describes how a co-simulation master can configure some parameters throughout the co-simulation execution. This is the idea behind adaptive master algorithms.
nfr : ip_protection
nfr :accuracy
sr : info : causality :feedthrough
sr : causality :causal
sr : availability : local
sr : rollback :none
sr : rel_time :analytic
fr :domain:ct
fr :num_sim:three_more
fr :coupling_model:io_assignments
fr :sim_rate: single
fr : sim_step_size : variable
fr :alg_loop : explicit
fr : results_visualization :post_mortem
□

Ref 61. Busch2011: An explicit approach for controlling the macro-step size of co-simulation methods [52].

Summary. Presents an approach to estimate the local truncation error caused by the extrapolations of inputs in a co-simulation. The sub-systems are assumed to make no error. It does not require rollback or re-initialization.

Categories:
nfr : accuracy
Because they study the global error and control the local error.
nfr : ip_protection
nfr : performance
They control the step size, which increases performance. And they study how to get an optimal step size.
sr : info : causality : feedthrough
sr : rollback : none
sr : causality : causal
fr : domain:ct
fr : num_sim:three_more
sr : rel_time : analytic
fr : sim_rate : multi
fr : sim_step_size : variable
fr : alg_loop : explicit
sr : availability : local
fr : results_visualization : post_mortem
fr : coupling_model:io_assignments
fr : communication_model:jacobi
In theory, they seem to support any communication model. In the paper they studied assuming the Jacobi.

□

Ref 62. Quesnel2005: DEVS coupling of spatial and ordinary differential equations: VLE framework [202].

Summary. Proposes a way to wrap a continuous time ODE simulator as a DEVS model. It requires that the state variables, and derivatives are available.

Categories:
nfr : hierarchy
nfr : open_source
sr : info : derivatives : out
sr : info : statevars
sr : info : predict_step_sizes
sr : causality : causal
fr : domain:de
fr : num_sim:three_more
sr : rel_time : analytic
fr : sim_rate : multi
Any discrete event framework is by definition multi-rate.

fr : sim_step_size : variable

Any discrete event framework is by definition in this category.

fr : alg_loop : explicit

sr : availability : local

fr : results_visualization : post_mortem

fr : coupling_model:io_assignments

fr : communication_model:gauss_seidel

A discrete event framework is in this category as there is no extrapolation of inputs. Also, Gauss seidel does not violate the causality of inputs and outputs, because it sorts according to these dependencies. Events are processed to retain their causality.

□

Ref 63. Arnold2014a: Error analysis for co-simulation with force-displacement coupling [15].

Summary. Describes an FMI based master called SNiMoWrapper.

Categories:

nfr : accuracy

Because they study the global error and control the local error.

nfr : ip_protection

sr : info : causality : feedthrough

sr : rollback : none

sr : causality : causal

sr : rel_time : analytic

sr : availability : local

fr : domain:ct

fr : num_sim:three_more

fr : sim_rate : multi

fr : sim_step_size : fixed

fr : alg_loop : explicit

fr : results_visualization : post_mortem

fr : coupling_model:io_assignments

fr : communication_model:jacobi

fr : standard:fmi

□

Ref 64. Arnold2014: Error Analysis and Error Estimates for Co-simulation in FMI for Model Exchange and Co-Simulation v2.0 [14].

Summary. Studies the error control method known as Richard's extrapolation.

Categories:

nfr : accuracy

Because they study the global error and control the local error.

nfr : ip_protection

sr : info : causality : feedthrough

sr : info : statevars

sr : rollback : single

sr : causality : causal

fr : domain:ct
fr : num_sim:three_more
sr : rel_time : analytic
fr : sim_rate : multi
fr : sim_step_size : variable
fr : alg_loop : explicit
sr : availability : local
fr : results_visualization : post_mortem
fr : coupling_model:io_assignments
fr : communication_model:jacobi
fr : communication_model:gauss_seidel
fr : standard:fmi

□

Ref 65. Arnold2001: Preconditioned Dynamic Iteration for Coupled Differential-Algebraic Systems [11].

Summary. Studies the convergence of the Gauss-Seidel dynamic iteration method and proposes a way to ensure it. The way to do it though, requires information from the model.

Categories:
nfr : accuracy
Because they study the global error.
sr : info : jacobian : out
sr : info : record_outputs
sr : info : full_model
sr : rollback : single
sr : causality : causal
sr : rel_time : analytic
sr : availability : local
fr : domain:ct
fr : num_sim:three_more
fr : sim_rate : multi
fr : sim_step_size : fixed
fr : alg_loop : implicit
fr : results_visualization : post_mortem
fr : coupling_model: algebraic_constraints
fr : communication_model:gauss_seidel

□

Ref 66. Schweizer2014: Semi-implicit co-simulation approach for solver coupling [215].

Summary. Proposes a predictor corrector master that evaluates the macro step twice and uses a perturbation on the inputs to get an estimate of the required partial derivatives. This approach is then generalized to multiple kinds of joints in the mechanical domain. A double pendulum, double mass-spring-damper and a slider crank mechanism are used as numerical examples.

Categories:
sr : rollback : single

sr : causality : causal
sr : rel_time : analytic
sr : availability : local
fr : domain:ct
fr : num_sim:two
fr : sim_rate: multi
fr : sim_step_size : fixed
fr : alg_loop : semi_implicit
fr : results_visualization :post_mortem
fr : coupling_model: algebraic_constraints
fr : communication_model:jacobi

□

Ref 67. Schweizer2015d: Stabilized implicit co-simulation methods: solver coupling based on constitutive laws [218].

Summary. It presents an implicit and semi-explicit methods for the co-simulation of scenarios coupled via applied forces. The difference between this paper and the previous ones by the same author seems to be in the fact that the coupling constraints are integrated and differentiated, to enrich the information being used to ensure that the original coupling constraints are met.
Categories:
sr : rollback : single
sr : causality : causal
sr : rel_time : analytic
sr : availability : local
fr : domain:ct
fr : num_sim:three_more
fr : sim_rate: multi
fr : sim_step_size : fixed
fr : alg_loop : semi_implicit
fr : results_visualization :post_mortem
fr : coupling_model: algebraic_constraints
fr : communication_model:jacobi

□

Ref 68. Sadjina2016: Energy Conservation and Power Bonds in Co-Simulations: Non-Iterative Adaptive Step Size Control and Error Estimation [207].

Summary. Proposes a master for co-simulation that requires the identification of power bonds between sub-systems. It assumes that the scenario is energy conserving and thus calculate the energy residual as an error to be minimized. The step size is adapted via a PI-Controller. When the step size is reduced, it is only on the next co-simulation step, so the method is explicit.
nfr : ip_protection
nfr : accuracy
Due to step size control.
sr : rollback : none
sr : causality : causal

sr : rel_time : analytic
sr : availability : local
fr : domain:ct
fr : num_sim:three_more
fr : sim_rate : multi
fr : sim_step_size : variable
fr : alg_loop : explicit
fr : results_visualization : post_mortem
fr : coupling_model:io_assignments
fr : communication_model:jacobi

□

Ref 69. Busch2016: Continuous approximation techniques for co-simulation methods: Analysis of numerical stability and local error [50].

Summary. Analyses the stability and local error of multiple co-simulation approaches with multiple extrapolation approaches for the inputs. It considers Gauss-Seidel and Jacobi. It also talks about a method called the extrapolated interpolation method, which ensure no discontinuities at the inputs of the subsystems.

sr : rollback : none
The method is explicit.
sr : causality : causal
fr : domain:ct
fr : num_sim:three_more
sr : rel_time : analytic
fr : sim_rate : single
fr : sim_step_size : fixed
fr : alg_loop : explicit
sr : availability : local
fr : results_visualization : post_mortem
fr : coupling_model:io_assignments
fr : communication_model:jacobi
fr : communication_model:gauss_seidel

□

Ref 70. Arnold2010: Stability of Sequential Modular Time Integration Methods for Coupled Multibody System Models [10].

Summary. Studies stability of a gauss Seidel co-simulation method proposed in previous work: [11]. Based on that analysis, it proposes an implicit stabilization technique that uses Gauss-Seidel iteration. The resulting method is implicit but the equations that are being solved are linear.

Categories:
nfr : accuracy
Because they study the global error.
sr : info : jacobian : out
sr : info : record_outputs
sr : info : full_model

sr : rollback : single
sr : causality : causal
sr : rel_time : analytic
sr : availability : local
fr : domain:ct
fr : num_sim:three_more
fr : sim_rate : single
fr : sim_step_size : fixed
fr : alg_loop : implicit
fr : results_visualization : post_mortem
fr : coupling_model: algebraic_constraints
fr : communication_model:gauss_seidel

□

Ref 71. Gu2001: Co-simulation of algebraically coupled dynamic subsystems [105].

Summary. Describes a technique to deal with algebraically coupled sub-systems using a control theoretic approach. The highlights of this method are: it supports scenarios of arbitrary index; the boundary condition coordinator is seen as a co-simulation unit (this is an elegant approach) and the method is explicit. The beauty of making the BCC as a co-simulation unit, is that it can, just like any other sub-system be run at a different rate and in the paper they show that by running it at a higher rate, the stability of the co-simulation increases.
sr : rollback : none
sr : causality : causal
fr : domain:ct
fr : num_sim:two
sr : rel_time : analytic
fr : sim_rate : multi
fr : sim_step_size : fixed
fr : alg_loop : explicit
sr : availability : local
fr : results_visualization : post_mortem
fr : coupling_model: algebraic_constraints
fr : communication_model:jacobi

□

Ref 72. Gu2004: Co-Simulation of Algebraically Coupled Dynamic Subsystems Without Disclosure of Proprietary Subsystem Models [107].

Summary. Describes a technique to solve causal conflicts using a Boundary Condition Coordinator (BCC). Causal conflicts arise naturally from the coupling of different sub-systems and they are a relevant challenge that needs to be overcome in order to perform correct co-simulation. While in [105], the BCC requires the knowledge of the state variables of the simulations, in [107], some modifications are made to ensure that this information is not required.
nfr : ip_protection
nfr : distribution
sr : rollback : none

sr : causality : causal
fr : domain:ct
fr : num_sim:three_more
sr : rel_time : analytic
fr : sim_rate : multi
fr : sim_step_size : fixed
fr : alg_loop : explicit
sr : availability : local
fr : results_visualization : post_mortem
fr : coupling_model: algebraic_constraints
fr : communication_model:jacobi

□

Ref 73. Schweizer2016: Co-simulation method for solver coupling with algebraic constraints incorporating relaxation techniques [219].

Summary. A master algorithm capable of dealing with algebraic constraints is described. It requires the derivatives of the coupled variables to be available. It executes each communication step twice, being a semi-implicit method. It uses a predict step and a corrector step. The final corrected coupling variables are obtained by polynomial extrapolation and relaxation (to avoid instabilities).

Categories:
sr : rollback : single
sr : causality : causal
fr : domain:ct
fr : num_sim:three_more
sr : rel_time : analytic
fr : sim_rate : multi
sr : info : jacobian : out
fr : sim_step_size : fixed
fr : alg_loop : semi_implicit
sr : availability : local
fr : results_visualization : post_mortem
fr : coupling_model: algebraic_constraints
fr : communication_model:jacobi

□

Ref 74. Schweizer2015: Predictor/corrector co-simulation approaches for solver coupling with algebraic constraints [214].

Summary. Proposes a predictor corrector master that evaluates the macro step twice and uses a perturbation on the inputs to get an estimate of the required partial derivatives.

Categories:
sr : rollback : single
sr : causality : causal
fr : domain:ct
fr : num_sim:three_more
sr : rel_time : analytic

fr : sim_rate : multi
fr : sim_step_size : fixed
fr : alg_loop : semi_implicit
sr : info : jacobian : out
sr : availability : local
fr : results_visualization : post_mortem
fr : coupling_model: algebraic_constraints
fr : communication_model:jacobi

□

Ref 75. Schweizer2015a: Stabilized index-2 co-simulation approach for solver coupling with algebraic constraints [216].

Summary. A master algorithm capable of dealing with algebraic constraints is described. It requires the derivatives of the coupled variables to be available. It executes each communication step twice, being a semi-implicit method. It uses a predict step and a corrector step. The predictor step allows the method to estimate the sensitivity of the state variables with respect to the applied forces/torques.

Categories:
sr : rollback : single
sr : causality : causal
fr : domain:ct
fr : num_sim:three_more
sr : rel_time : analytic
fr : sim_rate : multi
sr : info : jacobian : out
fr : sim_step_size : fixed
fr : alg_loop : semi_implicit
sr : availability : local
fr : results_visualization : post_mortem
fr : coupling_model: algebraic_constraints
fr : communication_model:jacobi

□

Ref 76. Andersson2016: Methods and Tools for Co-Simulation of Dynamic Systems with the Functional Mock-up Interface [8].

Summary. This is a Phd Thesis. A linear extrapolation based master is proposed that is convergent and does not require fixed point iterations. Then, a modification to multi-step methods is proposed to increase their performance when executing in a co-simulation environment. This modification avoids the need to restart when dealing with discontinuities.

Categories:
nfr : ip_protection
nfr : platform_independence
nfr : open_source
sr : rollback : none
sr : causality : causal

fr : domain:ct
fr : num_sim:three_more
sr : rel_time : analytic
fr : sim_step_size : fixed
sr : availability : local
fr : results_visualization : post_mortem
fr : coupling_model:io_assignments
fr : communication_model:jacobi
fr : standard:fmi

□

Ref 77. Krammer2015: Model-Based Configuration of Automotive Co-Simulation Scenarios [140].

Summary. The language is further developed in [140] with the addition of three novel diagrams to represent different aspects of the co-simulation configuration:

- Architectural – coupling of executable units;
- Tools – assignment of tools to models;
- Connections – connections (it was not clear what does this diagram do);

In addition, they define a couple of well formedness properties that can be checked more easily with the model-based approach. They give a brief summary of the tool ICOS.

Categories:
nfr : config_reusability
nfr : parallelism
nfr : hierarchy
nfr : extensibility
fr : domain:ct
fr : num_sim:three_more
sr : rel_time : analytic
sr : availability : local
fr : coupling_model:io_assignments
fr : results_visualization : post_mortem
sr : info : full_model

□

Ref 78. Galtier2015: FMI-Based Distributed Multi-Simulation with DACCOSIM [96].

Summary. DACCOSIM is able to perform Distributed simulations and multi-core simulations. The term “computation node” is used for a collection of FMU wrappers (which include an FMU) and a local master / global master. The FMU wrappers, and thereby not the masters, are responsible for passing outputs to connected inputs. This is to avoid bottlenecks. A component node contains a master and some FMUs, which are wrapped in so-called “FMU-wrappers”. The masters take responsibility of coordinated step sizes in case an FMU needs to roll back.

nfr : performance

Because of the possibility of splitting the simulation over a cluster / multi-core and the focus on performance in the article. Additionally because of their use of variable step size

nfr : config_reusability

It is possible to create multiple co-simulation configuration files in the simulation configuration GUI. These can be stored and therefore reused.

nfr : ip_protection

Some level of IP protection because of FMI..

nfr : parallelism

Because of the possibility of splitting the simulation over a cluster

nfr : distribution

Because a co-simulation can be executed on a cluster of computers

sr : availability : remote

sr : availability : local

nfr : hierarchy

DACCOSIM is weak hierarchical because it has the notion of local and global masters.

nfr : scalability

The framework is considered to be scalable because of the multi-core and distributed architecture.

nfr : platform_independence

There are two versions of the DACCOSIM library. A cross-platform version relying on JAVA and a Windows version using C++ and QTronic SDK.

nfr : accuracy

The article provides an example where the result of the co-simulation using DACCOSIM is compared to the simulation using Dymola and the results are very close to each other. Accuracy is ensured by each FMU examining its outputs and estimating how far they are from the exact value.

nfr : open_source

The framework is distributed under an open source license from January 2016.

sr : info : stateserial

The framework can perform a single rollback using the state variable serialization.

sr : causality : causal

Because the framework is based on FMI for co-simulation it is considered to be causal.

fr : domain:ct

The framework supports multiple formalisms because it is based on FMI for co-simulation.

fr : num_sim:three_more

The frameworks is capable of supporting many FMUs and thereby many simulation units. DACCOSIM offers its own algorithm depending on global/local masters.

fr : coupling_model:io_assignments

sr : rel_time : analytic

There is no mentioning of any other time models than this in the article.

fr : sim_rate : single

The simulation rate is the same for all FMUs.

fr : sim_step_size : variable

The framework uses Coordinated Variable Step

fr : alg_loop : explicit

The framework uses Euler's method and Richardson's method. Whether this is default, parameterizable or fully customizable is unknown based on this article.

fr : communication_model:jacobi

See Co-initialization bullet 2 in the article.

fr : standard:fmi

It is based on the FMI standard.
fr : results_visualization :post_mortem

□

Ref 79. Fey1997: Parallel synchronization of continuous time discrete event simulators [84].

Summary. Presents two synchronization approaches, detailed in three different synchronization protocols, to coordinate simulation scenarios that include one discrete event simulator and one continuous time simulator. The discrete event simulator can implement any parallel simulation approach that we know, such as Time-Warp. This means that, even internally, the DE simulator can be forced to rollback due to straggler messages. The focus is on parallel approaches.
Categories:
nfr : performance
nfr : ip_protection
nfr : parallelism
nfr : accuracy
fr : domain:de
fr : domain:ct
fr : num_sim:two
fr : sim_rate : multi
fr : sim_step_size : fixed
fr : results_visualization :post_mortem
sr : rel_time : analytic
sr : availability : local
sr : rollback : single
fr : coupling_model:io_assignments
sr : info : full_model

□

Ref 80. Acker2015: Generation of an Optimised Master Algorithm for FMI Co-simulation [236].

Summary. Essentially, this paper shows how a compiled approach increases the performance of the co-simulation. It also shows that, because there are so many decisions to be made when designing the master, a compiled approach allows for a more elegant, and specifically tailored master, to be generated.
nfr : performance
nfr : ip_protection
nfr : config_reusability
nfr : open_source
sr : rel_time : analytic
sr : availability : local
sr : info : causality : feedthrough
sr : info : statevars
sr : rollback : none
sr : causality : causal
sr : info : preferred_step_sizes
fr : domain:ct

fr : num_sim:three_more
fr : sim_rate : multi
fr : sim_step_size : fixed
fr : results_visualization : post_mortem
fr : standard:fmi
fr : communication_model:gauss_seidel
fr : alg_loop : implicit
fr : coupling_model:io_assignments

□

Ref 81. Enge-Rosenblatt2011: Functional Digital Mock-up and the Functional Mock-up Interface - Two Complementary Approaches for a Comprehensive Investigation of Heterogeneous Systems [79].

Summary. The paper describes and compares two approaches to performing co-simulation of heterogeneous systems, namely the Functional Digital Mock-up (FDMU) and the Functional Mock-up Interface (FMI). Besides describing these approaches it also introduces the "FDMU framework", a framework that implements the Functional Digital Mock-up approach. Furthermore, proposals are presented for combining FDMU and FMI approaches.

The FDMU approach is a tool-independent and web service-based framework build on the Web Service standards. It is capable of coupling different simulation tools and provide visualization based on CAD models.

FDMU consists of three main concepts: functional building blocks (FBB), wrappers, FDMU master, and FDMU Console. The functional building block can wrap geometric information (CAD Models), behavioral models, and a simulator tool. It is the responsibility of the wrappers to establish a connection between the different simulation tools and the FDMU Master Simulator. Finally, the FDMU master ensures correct communication between the simulators. The FDMU Console is the user's front-end.

nfr : performance
Communication overhead of a web service-based approach.
nfr : ip_protection
Because of the the web service-based approach IP protection should be possible.
nfr : parallelism
Because of the web service-based approach it is parallel by nature. It uses thread-safe queues and deadlock-free transmission of data.
nfr : distribution
Because of the web service-based approach it is easy distributable.
nfr : scalability
The distributed systems paradigm ensures scalability.
nfr : platform_independence
web service-based approach.
nfr : extensibility
A new wrapper can be implemented.
sr : causality : causal
Every input of an FBB must have an appropriate output belonging to another FBB.
fr : domain:ct
fr : coupling_model:io_assignments

fr : num_sim:three_more
fr : sim_rate : multi
sr : availability : remote
fr : results_visualization : live
The framework provides an interactive 3D visualization based on CAD.
fr : standard:fdmu

□

Ref 82. Karner2010a: Heterogeneous co-simulation platform for the efficient analysis of FlexRay-based automotive distributed embedded systems [130].

Summary. Motivation: FlexRay is a wired network for automotive high-speed control applications and no solutions exist that simulates all parts of the network.

What: a co-simulation platform called TEODACS FlexRayExprt.Sim. The simulation approach used in the platform covers mechanics and all parts of the network from physical layer to application layer, which is not done by other solutions. The framework CISC SyAD is used to perform the microelectronics co-simulation, CarMaker/AVL InMotion for the mechanics, and they are bridged by TEODACS FlexRayEprt.Sim. The platform uses a very interesting approach to faster co-simulations, namely the use of model switching, where a less detailed model replaces a more detailed model in parts of the simulation.

The paper provides an overview of existing approaches such as transaction based modeling, HDLs such as SystemC and Verilog, and cable harness and topology modeling along with why these contain shortcomings to this domain. Furthermore, the paper provides some details of the implementation of the models used in the co-simulation and showcases how the platform can analyse a system with specific examples.
nfr : accuracy
Because of model switching.
nfr : performance
Because of model switching.
fr : dynamic_structure
Because the structure of the co-simulation is changed via model switching.
fr : domain:ct
Because SyAD supprots multiple formalisms and CarMaker / AVL InMotion.
fr : domain:de
fr : coupling_model:io_assignments
fr : num_sim:three_more
Multiple FlexRay nodes can be added.
sr : rel_time : analytic
From model switching and similar it is clear that analytic simulation is used.
fr : results_visualization : post_mortem

□

Ref 83. Aslan2015: MOKA: An Object-Oriented Framework for FMI Co-Simulation [16].

Summary. The paper describes MOKA, which is a framework for performing co-simulations and creating FMUs using FMI 2.0 for co-simulation. The framework turns the creation of FMUs into an object-oriented process by using C++. An FMU is created by inheriting one of the classes and implementing virtual functions thereby avoiding writing boilerplate code. The implementation of FMUs is realised by the concepts of FMUBlock, which is to be inherited, FMUPort, and FMUStateVariables. FMUBlock is to be extended by a concrete FMU slave and implements common computation phase functions for slaves. It contains FMUPort for data exchange and FMUStateVariables for state tracking during the simulation. The FMUPort classes provides the data exchange interface of a slave. It abstracts the value references by automatically assigning a value reference to the variable. The BaseStateVariable class also functions as base that is to be extended. It provides virtual functions for state variable services. The StateVariable inherits from BaseStateVariable and represents state variables for the slave. The framework also provides a template for the FMU Master so the master code changes minimally for different scenarios.

The article exemplifies an application of the MOKA framework where two FMUs are used: The bouncing ball and integer counter example from the QTronic SDK, where the bouncing ball has been re-developed with MOKA.

In future work it is stated that development of a DSL in a current study, so that different scenarios can be executed without altering the master code.

nfr : ip_protection
nfr : config_reusability
sr : causality : causal
sr : rel_time : analytic
sr : rollback : none
sr : availability : local
fr : coupling_model:io_assignments
fr : num_sim:three_more
fr : standard:fmi
fr : domain:ct
fr : alg_loop : explicit
fr : sim_rate : single
fr : communication_model:gauss_seidel
fr : results_visualization : post_mortem

□

Ref 84. Wetter2010: Co-simulation of building energy and control systems with the Building Controls Virtual Test Bed [253].

Summary. Describes a co-simulation framework called Building Controls Virtual Test Bed (BCVTB) that can be used for real-time simulation and co-simulation. It is a modular extensible open-source platform to interface different simulation programs with each other. The intention is to give users the option to use the best tools suited to model various aspects of building energy and control systems, or use programs where they have expertise. The middleware to couple any number of simulation programs also provides libraries such that it can be extended. Furthermore, the paper describes how they gathered capabilities the framework should support. The framework is based on Ptolemy II, which is extended by some java packages. The simulator package adds functionality that allows an actor to perform system calls to start any executable on Windows, OSX or Linux. It simply starts a simulation program, sends input tokens to the simulation program, receives new values and sends them to its output port. Algorithms are also provided on how simulators are coupled. These are also exemplified with specific simulators. It is also described how to connect client programs. The article describes how the interfaces are created for simulink, matlab, modelica, and system cals. Furthermore, a specific example is presented.

nfr : config_reusability
nfr : distribution
nfr : platform_independence
nfr : extensibility
nfr : open_source
sr : causality : causal
sr : rel_time : analytic
sr : rel_time : fixed_real_scaled_time_simulation
fr : domain:ct
In the paper, their explanation is focused on the CT domain.
fr : num_sim:three_more
fr : sim_rate: single
fr : sim_step_size : fixed
fr : results_visualization : post_mortem
fr : coupling_model:io_assignments
fr : communication_model:jacobi

□

Ref 85. Neema2014: Model-based integration platform for FMI co-simulation and heterogeneous simulations of cyber-physical systems [182].

Summary. The article concerns integrating FMI as an HLA federate and extending the Command and Control Wind Tunnel (C2WT) metamodel to include FMI-specifics. This enables the C2WT tool to use FMUs as part of a simulation. The C2WT tool is describes as a multi-model integration platform that allows users to model and synthesize complex, heterogeneous, command and control simulations. The tool therefore has support for multiple simulation engines and an introduction to the tool is given in the paper. Furthermore, a case study on Vehicle Thermal Management using FMUs are presented and The focompared to a simulation in a different environment. The work is sponsored by the US DoD.

nfr : platform_independence
nfr : distribution
nfr : config_reusability
fr : domain:de
fr : num_sim:three_more
fr : sim_rate : multi
fr : sim_step_size : fixed
fr : sim_step_size : variable
sr : rel_time : fixed_real_scaled_time_simulation
sr : rel_time : analytic
fr : results_visualization : live
fr : standard:hla
fr : standard:fmi
fr : communication_model:jacobi

□

Ref 86. Larsen16c: Integrated Tool Chain for Model-Based Design of Cyber-Physical Systems [148].

Summary. This article presents an overview of the INTO-CPS project and thereby a Co-Simulation tool. The projects concerns production of a well-founded tool chain for model-based design of CPSs, and therefore consists of a semantic foundation and several baseline tools such as Modelio, Overture, 20-sim, OpenModelica and RT-Tester. Furthermore, an application called the INTO-CPS application is the entry point for configuring co-simulations and uses the co-simulation orchestration engine (COE) to perform the actual simulations. This COE is based on the FMI standard. The entire tool chain and the semantic foundation are presented in this paper. This is related to [148].

nfr : config_reusability
nfr : ip_protection
sr : info : nominal_values:output
sr : info : nominal_values:state
sr : info : derivatives : out
sr : info : derivatives : state
sr : info : jacobian : out
sr : info : jacobian : state
sr : info : preferred_step_sizes
sr : info : causality : feedthrough
sr : causality : causal
sr : availability : local
sr : info : statevars
sr : info : record_outputs
sr : rel_time : analytic
sr : info : stateserial
fr : standard:fmi
sr : info : signal
fr : num_sim:three_more
fr : domain:de
fr : domain:ct
fr : sim_rate : single
fr : results_visualization : post_mortem
fr : results_visualization : live

□

Ref 87. INTOCPSD41d: Design of the INTO-CPS Platform [149].

Summary. This is an EU deliverable related to the INTO-CPS project. It contains the technical documentation of the INTO-CPS platform at the end of 2015 (the first year of the project). Part of this project is the Co-simulation Orchestration Engine (COE). This is related to [148].

sr : info : predict_step_sizes
Supports the FMI suggested extension fmi2getMaxStepSize
nfr : performance
The COE supports variable step size, which can increase performance.
sr : rollback : single
Supports rollback to last successful state.

nfr : performance

The COE supports variable step size, which can increase performance.

nfr : accuracy

Contains various constraints such as zero-crossing, bounded difference and sampling rate. Furthermore, support for allowed min and max values. □

C Co-Simulation Scenario Categorization

This section describes each category and lists the references that belong to that category, classified in the previous section.

C.1 Non-Functional Requirements

C.1.1 Fault Tolerance

A co-simulation platform is fault tolerant if, for example, when one simulation unit fails, other can take its place. This is particularly important for long running simulations. To be fault tolerant, certain features need to available: periodically store the state of simulation units; record all inputs to each simulation unit. If a simulation unit fails and the state is periodically stored, then the simulation can be paused while the state is restored in a new instance of the simulation unit. The history of input values passed to each simulation unit can be used to bring the simulation unit to the current state.

References in this category:

Fuller2013

C.1.2 Configuration Reusability

This category refers to the fact that frameworks can provide a way to configure co-simulation scenarios that can be reused. This means that the configuration is considered external to the execution of the co-simulation. External in this context means that the configuration can reused without altering the binaries for the co-simulation application.

If a tool/frame does not provide a way to reuse configurations for co-simulation, then it is a time-consuming, error-prone and non-trivial process to set up co-simulations [140].

References in this category:

Wang2013

Krammer2015

Galtier2015

Acker2015

Aslan2015

Wetter2010

Neema2014

C.1.3 Performance

Performance is a relative measure: a co-simulation platform is performant when it is able to simulate a great deal in a short amount of time while needing little resources. This can be achieved by using variable step integration methods and signal extrapolation techniques. Parallelism also plays a role but mostly on the time aspect of performance.

References in this category:
Hoepfer2011
Faure2011
Sun2011
Friedrich2011
Gonzalez2011
Gunther2012
Eyisi2012
Riley2011
Benedikt2013
Sicklinger2014
Kounev2015
Bian2015
BenKhaled2012
BenKhaled2014
Saidi2016
Schierz2012a
Liu2001
Carstens2003
Stettinger2014
Busch2011
Galtier2015
Fey1997
Acker2015
Enge−Rosenblatt2011
Karner2010a

C.1.4 IP Protection

IP Protection deals with not requiring the models participating in the co-simulation to provide detailed structure, variables, etc... There are multiple levels of protection ranging from fully protected to not protected at all. A good IP protection enables component suppliers to provide the system integrators with detailed simulations of their components avoiding expensive lock-in contracts.

There are multiple techniques can be be employed to ensure some degree of protection. For instance, making the models (and corresponding simulation units) available as a web service is a possible solution. Another example is any framework that implements the FMI Standard [34, 35], which allows models and simulation units to be exported as a single functional unit, in binary format, that can be imported into a co-simulation.

References in this category:
Hoepfer2011
Sun2011
Bastian2011a
Friedrich2011
Broman2013

Fuller2013
Wang2013
Kuhr2013
Viel2014
Dols2016
Saidi2016
Camus2015
Camus2016
Benedikt2016
Busch2011
Arnold2014a
Arnold2014
Sadjina2016
Gu2004
Andersson2016
Galtier2015
Fey1997
Acker2015
Enge−Rosenblatt2011
Aslan2015

C.1.5 Parallelism

A co-simulation framework is parallel when it makes use of multiple processes/threads to perform the co-simulation. This is typically in the same computer or same local network.

Techniques such as signal extrapolation help improve the speed-up gained from parallelism. Furthermore waveform relaxation techniques and the Jacobi iterations promote parallelism [167].

References in this category:
Faure2011
Bastian2011a
Friedrich2011
Gunther2012
Awais2013b
Awais2013a
BenKhaled2012
BenKhaled2014
Saidi2016
Yamaura2016
Camus2015
Camus2016
Pedersen2016
Oh2016
Xie2016
Liu2001
Stettinger2014
Krammer2015

Galtier2015
Fey1997
Enge−Rosenblatt2011

C.1.6 Distribution

A co-simulation framework is parallel and distributed when it allows each simulation unit to be remote, across a wide area network.

This is very important since suppliers can, instead of transferring the simulation units in executable form across computers, can make them available over the web. This offers much more control over how the simulation units are used.

The same techniques used in parallelism can be used to promote distribution, but fault tolerance is also important.

References in this category:
Friedrich2011
Nutaro2011
Busch2012
Quaglia2012
Eyisi2012
Riley2011
Roche2012
Fitzgerald2010
Fitzgerald2013
Kudelski2013
Fuller2013
Bombino2013
Zhao2014
Awais2013b
Awais2013a
Camus2015
Camus2016
Pedersen2016
Oh2016
Gu2004
Galtier2015
Enge−Rosenblatt2011
Wetter2010
Neema2014

C.1.7 Hierarchy

A hierarchical co-simulation framework is able to abstract a co-simulation scenario as a black box simulation unit. This is very intuitive and promotes abstraction of complex systems.

References in this category:
Quesnel2005
Krammer2015
Galtier2015

C.1.8 Scalability

A co-simulation framework is scalable when it supports a large number of simulation units. It is intimately related to performance and paralelism.

References in this category:
Fuller2013
BenKhaled2014
Saidi2016
Galtier2015
Enge−Rosenblatt2011

C.1.9 Platform Independence

A co-simulation framework is platform independent when it works on multiple computing platforms. For this to be achieved, a platform independent language, such as Java, can be used to coordinate the simulation.

References in this category:
Bastian2011a
Eyisi2012
Riley2011
Fitzgerald2010
Fitzgerald2013
Kudelski2013
Andersson2016
Galtier2015
Enge−Rosenblatt2011
Wetter2010
Neema2014

C.1.10 Extensibility

A co-simulation framework is extensible when it can be easily extended to support new kinds of simulation units, with new kinds of capabilities. A higher level, domain specific, language can be used to specify the behaviour in a platform agnostic way. Code is then generated from this description. The hypothesis is that the high level description can be more easily extended to describe new behaviour and that the code generation process can be adapted accordingly.

References in this category:
Kuhr2013
Krammer2015
Enge−Rosenblatt2011
Wetter2010

C.1.11 Accuracy

A co-simulation is accurate when the error between the trace produced and the correct trace is minimal. This can be achieved by error control mechanisms.

References in this category:

Hoepfer2011
Tomulik2011
Gonzalez2011
Nutaro2011
Busch2012
Schmoll2012
Schierz2012
Gunther2012
Fitzgerald2010
Fitzgerald2013
Benedikt2013
Benedikt2013b
Hafner2013
Viel2014
Sicklinger2014
BenKhaled2014
Camus2015
Camus2016
Oh2016
Schierz2012a
Stettinger2014
Benedikt2016
Busch2011
Arnold2014a
Arnold2014
Arnold2001
Sadjina2016
Arnold2010
Galtier2015
Fey1997
Karner2010a

C.1.12 Open source

We consider open source the frameworks that make available the source code under certain licenses that are not paid for in any way.

References in this category:
Roche2012
Quesnel2005
Andersson2016
Galtier2015
Acker2015
Wetter2010

C.2 Simulator Requirements

This sub section covers the taxonomy that focuses on individual simulators' capabilities.

C.2.1 Information Exposed

Frequency of State The instantaneous frequency of the state of the sub-system can be used to adjust the communication step size.

Frequency of Outputs The instantaneous frequency of the output of the sub-system can be used to adjust the communication step size, as is done in [29].
References in this category:
Benedikt2013b

Detailed Model Simulators that make the equations of the dynamic system available fall into this category.
References in this category:
Schmoll2012
Hassairi2012
Schierz2012
Zhang2014
BenKhaled2012
BenKhaled2014
Yamaura2016
Arnold2001
Arnold2010
Krammer2015
Fey1997

Nominal Values of Outputs This information indicates the order of magnitude of output signals.

Nominal Values of State This information indicates the order of magnitude of state signals.

I/O Signal Kind The kind of output signal helps the master algorithm understand what assumptions are in a signal.
References in this category:
Kuhr2013

Time Derivative

Output References in this category:
Hoepfer2011
Tomulik2011
Gunther2012
Schierz2012a
Carstens2003
Quesnel2005

State References in this category:
Hoepfer2011

Jacobian

Output References in this category:
Tomulik2011
Bastian2011a
Busch2012
Schierz2012
Viel2014
Sicklinger2014
Arnold2001
Arnold2010
Schweizer2016
Schweizer2015
Schweizer2015a

State

Discontinuity Indicator A discontinuity indicator is a signal that indicates the presence of a discontinuity in the output of the simulation unit.

Deadreckoning model A deadreckoning model is a function that can be used by other simulation units to extrapolate the behavior of this simulation unit.

Preferred Step Size References in this category:
Acker2015

Next Step Size The next step size indicates the next communication time that is appropriate for the current simulator.
References in this category:
Lin2011
Gunther2012
Eyisi2012
Riley2011
Broman2013
Kounev2015
Quesnel2005

Order of Accuracy The order of accuracy can be used to determine the appropriate input extrapolation functions to be used in a co-simulation scenario.

I/O Causality *Feedthrough*

References in this category:
Broman2013
Bogomolov2015
BenKhaled2012
BenKhaled2014
Saidi2016
Benedikt2016
Busch2011
Arnold2014a
Arnold2014
Acker2015

Propagation Delay

The propagation delay indicates how many micro-steps have to be performed before a change in the input affects an output. In Simulink, this is the number of delay blocks in chain from an input to an output.

Input Extrapolation This information denotes the kind of input extrapolation being performed by the simulation unit.

References in this category:
Viel2014

State Variables References in this category:
Hoepfer2011
Quesnel2005
Arnold2014
Acker2015

Serialized State References in this category:
Bastian2011a
Broman2013
Viel2014
Bogomolov2015
Camus2015
Camus2016
Schierz2012a
Galtier2015

Micro-Step Outputs This information denotes the output of the simulation unit, evaluated at each of the micro-steps.

References in this category:
Benedikt2013
Viel2014
Arnold2001
Arnold2010

WCET This denotes the worst case excution time.
References in this category:
Faure2011
BenKhaled2014
Saidi2016

C.2.2 Causality

Causal References in this category:
Pedersen2015
Lin2011
Hoepfer2011
Tomulik2011
Sun2011
Bastian2011a
Friedrich2011
Gonzalez2011
Nutaro2011
Busch2012
Schmoll2012
Ni2012
Hassairi2012
Schierz2012
Gunther2012
Quaglia2012
Al−Hammouri2012
Eyisi2012
Riley2011
Roche2012
Fitzgerald2010
Fitzgerald2013
Kudelski2013
Broman2013
Benedikt2013
Benedikt2013b
Fuller2013
Bombino2013
Wang2013
Hafner2013
Zhao2014
Li2011c
Awais2013b
Awais2013a
Kuhr2013
Viel2014
Sicklinger2014

Zhang2014
Kounev2015
Bogomolov2015
Bian2015
Dols2016
BenKhaled2014
Saidi2016
Yamaura2016
Camus2015
Camus2016
Oh2016
Xie2016
Manbachi2016
Schierz2012a
Fourmigue2009
Liu2001
Carstens2003
Stettinger2014
Benedikt2016
Busch2011
Quesnel2005
Arnold2014a
Arnold2014
Arnold2001
Schweizer2014
Schweizer2015d
Sadjina2016
Busch2016
Arnold2010
Gu2001
Gu2004
Schweizer2016
Schweizer2015
Schweizer2015a
Andersson2016
Galtier2015
Acker2015
Enge−Rosenblatt2011
Aslan2015
Wetter2010

A-Causal

C.2.3 Time Constraints

Analytic Simulation References in this category:
Pedersen2015
Lin2011
Hoepfer2011
Tomulik2011
Sun2011
Friedrich2011
Gonzalez2011
Nutaro2011
Busch2012
Schmoll2012
Ni2012
Hassairi2012
Schierz2012
Quaglia2012
Al−Hammouri2012
Eyisi2012
Riley2011
Roche2012
Fitzgerald2010
Fitzgerald2013
Kudelski2013
Broman2013
Benedikt2013
Benedikt2013b
Fuller2013
Wang2013
Hafner2013
Zhao2014
Li2011c
Awais2013b
Awais2013a
Kuhr2013
Viel2014
Sicklinger2014
Zhang2014
Kounev2015
Bogomolov2015
Dols2016
BenKhaled2014
Saidi2016
Yamaura2016
Camus2015
Camus2016
Oh2016

Xie2016
Schierz2012a
Fourmigue2009
Liu2001
Carstens2003
Benedikt2016
Busch2011
Quesnel2005
Arnold2014a
Arnold2014
Arnold2001
Schweizer2014
Schweizer2015d
Sadjina2016
Busch2016
Arnold2010
Gu2001
Gu2004
Schweizer2016
Schweizer2015
Schweizer2015a
Andersson2016
Krammer2015
Galtier2015
Fey1997
Acker2015
Karner2010a
Aslan2015
Wetter2010
Neema2014

Scaled Real Time Simulation
Fixed

A simulator is fixed scaled real time when it simulated time progresses according to a fixed linear relationship with the real time.

References in this category:
Faure2011
Bian2015
BenKhaled2012
Pedersen2016
Manbachi2016
Stettinger2014
Wetter2010
Neema2014

Dynamic

A simulator is dynamic scaled real time when the relation between the simulated time and the real time can be changed throughout the simulation.

References in this category:

Bombino2013

C.2.4 Rollback Support

None References in this category:

Pedersen2015
Lin2011
Hoepfer2011
Faure2011
Sun2011
Bastian2011a
Friedrich2011
Gonzalez2011
Schmoll2012
Ni2012
Schierz2012
Gunther2012
Quaglia2012
Al−Hammouri2012
Eyisi2012
Riley2011
Roche2012
Kudelski2013
Broman2013
Benedikt2013
Benedikt2013b
Fuller2013
Bombino2013
Wang2013
Hafner2013
Zhao2014
Li2011c
Awais2013b
Awais2013a
Kuhr2013
Viel2014
Zhang2014
Kounev2015
Bogomolov2015
Bian2015
Dols2016
BenKhaled2012
BenKhaled2014

Saidi2016
Yamaura2016
Camus2015
Camus2016
Oh2016
Xie2016
Manbachi2016
Carstens2003
Stettinger2014
Benedikt2016
Busch2011
Arnold2014a
Sadjina2016
Busch2016
Gu2001
Gu2004
Andersson2016
Acker2015
Aslan2015

Single Single rollback means that the simulation unit is capable of saving a certain state in the past (simulated time) and revert to that state. Once reverted, the simulation unit cannot revert further in the past.

References in this category:
Tomulik2011
Nutaro2011
Busch2012
Bombino2013
Sicklinger2014
Schierz2012a
Liu2001
Arnold2014
Arnold2001
Schweizer2014
Schweizer2015d
Arnold2010
Schweizer2016
Schweizer2015
Schweizer2015a
Fey1997

Multiple Multiple rollback means that the simulation unit is capable of saving a certain state in the past (simulated time) and revert to that state. Once reverted, the simulation unit revert further into the past as many times as necessary.

C.2.5 Availability

Remote References in this category:
Friedrich2011
Busch2012
Quaglia2012
Eyisi2012
Riley2011
Roche2012
Fitzgerald2010
Fitzgerald2013
Kudelski2013
Fuller2013
Bombino2013
Zhao2014
Awais2013b
Awais2013a
Bian2015
Dols2016
Yamaura2016
Oh2016
Liu2001
Carstens2003
Galtier2015
Enge−Rosenblatt2011

Local References in this category:
Pedersen2015
Lin2011
Hoepfer2011
Faure2011
Tomulik2011
Sun2011
Bastian2011a
Gonzalez2011
Nutaro2011
Schmoll2012
Ni2012
Hassairi2012
Schierz2012
Gunther2012
Al−Hammouri2012
Broman2013
Benedikt2013
Benedikt2013b
Wang2013

Hafner2013
Li2011c
Awais2013b
Awais2013a
Kuhr2013
Viel2014
Sicklinger2014
Zhang2014
Kounev2015
Bogomolov2015
BenKhaled2012
BenKhaled2014
Saidi2016
Camus2015
Camus2016
Xie2016
Manbachi2016
Schierz2012a
Fourmigue2009
Stettinger2014
Benedikt2016
Busch2011
Quesnel2005
Arnold2014a
Arnold2014
Arnold2001
Schweizer2014
Schweizer2015d
Sadjina2016
Busch2016
Arnold2010
Gu2001
Gu2004
Schweizer2016
Schweizer2015
Schweizer2015a
Andersson2016
Krammer2015
Galtier2015
Fey1997
Acker2015
Aslan2015

C.3 Framework Requirements

C.3.1 Standard

High Level Architecture References in this category:
Eyisi2012
Riley2011
Awais2013b
Awais2013a
Neema2014

Functional Mock-up Interface References in this category:
Pedersen2015
Sun2011
Bastian2011a
Broman2013
Wang2013
Awais2013b
Awais2013a
Kuhr2013
Viel2014
Bogomolov2015
Dols2016
BenKhaled2012
BenKhaled2014
Saidi2016
Camus2015
Camus2016
Pedersen2016
Schierz2012a
Arnold2014a
Arnold2014
Andersson2016
Galtier2015
Acker2015
Aslan2015
Neema2014

Functional Digital Mock-up References in this category:
Enge−Rosenblatt2011

C.3.2 Coupling

Input/Output Assignments References in this category:
Pedersen2015
Lin2011
Faure2011

Sun2011
Bastian2011a
Friedrich2011
Gonzalez2011
Nutaro2011
Busch2012
Schmoll2012
Ni2012
Hassairi2012
Gunther2012
Quaglia2012
Al−Hammouri2012
Eyisi2012
Riley2011
Roche2012
Fitzgerald2010
Fitzgerald2013
Kudelski2013
Broman2013
Benedikt2013
Benedikt2013b
Fuller2013
Bombino2013
Wang2013
Hafner2013
Zhao2014
Li2011c
Awais2013b
Awais2013a
Zhang2014
Kounev2015
Bogomolov2015
Bian2015
Dols2016
BenKhaled2012
Yamaura2016
Camus2015
Camus2016
Pedersen2016
Oh2016
Xie2016
Manbachi2016
Schierz2012a
Fourmigue2009
Liu2001
Carstens2003

Stettinger2014
Benedikt2016
Busch2011
Quesnel2005
Arnold2014a
Arnold2014
Sadjina2016
Busch2016
Andersson2016
Krammer2015
Galtier2015
Fey1997
Acker2015
Enge−Rosenblatt2011
Karner2010a
Aslan2015
Wetter2010

Algebraic Constraints References in this category:
Tomulik2011
Friedrich2011
Schierz2012
Viel2014
Sicklinger2014
Arnold2001
Schweizer2014
Schweizer2015d
Arnold2010
Gu2001
Gu2004
Schweizer2016
Schweizer2015
Schweizer2015a

C.3.3 Number of Simulation Units

Two References in this category:
Lin2011
Sun2011
Gonzalez2011
Nutaro2011
Busch2012
Schmoll2012
Ni2012
Hassairi2012
Quaglia2012

Al−Hammouri2012
Eyisi2012
Riley2011
Roche2012
Fitzgerald2010
Fitzgerald2013
Kudelski2013
Benedikt2013
Benedikt2013b
Fuller2013
Bombino2013
Wang2013
Zhao2014
Li2011c
Zhang2014
Kounev2015
Bogomolov2015
Bian2015
Dols2016
Pedersen2016
Oh2016
Xie2016
Manbachi2016
Fourmigue2009
Liu2001
Carstens2003
Stettinger2014
Schweizer2014
Gu2001
Fey1997

Three or More References in this category:
Pedersen2015
Hoepfer2011
Faure2011
Tomulik2011
Bastian2011a
Friedrich2011
Schierz2012
Gunther2012
Broman2013
Hafner2013
Awais2013b
Awais2013a
Kuhr2013
Viel2014

Sicklinger2014
BenKhaled2012
BenKhaled2014
Saidi2016
Yamaura2016
Camus2015
Camus2016
Schierz2012a
Benedikt2016
Busch2011
Quesnel2005
Arnold2014a
Arnold2014
Arnold2001
Schweizer2015d
Sadjina2016
Busch2016
Arnold2010
Gu2004
Schweizer2016
Schweizer2015
Schweizer2015a
Andersson2016
Krammer2015
Galtier2015
Acker2015
Enge−Rosenblatt2011
Karner2010a
Aslan2015
Wetter2010
Neema2014

C.3.4 Domain

CT References in this category:
Pedersen2015
Hoepfer2011
Faure2011
Tomulik2011
Sun2011
Bastian2011a
Friedrich2011
Gonzalez2011
Busch2012
Schmoll2012
Ni2012

Hassairi2012
Schierz2012
Gunther2012
Quaglia2012
Al−Hammouri2012
Roche2012
Fitzgerald2010
Fitzgerald2013
Broman2013
Benedikt2013
Benedikt2013b
Bombino2013
Wang2013
Hafner2013
Zhao2014
Li2011c
Kuhr2013
Viel2014
Sicklinger2014
Zhang2014
Bogomolov2015
Bian2015
Dols2016
BenKhaled2012
BenKhaled2014
Saidi2016
Yamaura2016
Pedersen2016
Oh2016
Xie2016
Schierz2012a
Liu2001
Carstens2003
Stettinger2014
Benedikt2016
Busch2011
Arnold2014a
Arnold2014
Arnold2001
Schweizer2014
Schweizer2015d
Sadjina2016
Busch2016
Arnold2010
Gu2001
Gu2004

Schweizer2016
Schweizer2015
Schweizer2015a
Andersson2016
Krammer2015
Galtier2015
Fey1997
Acker2015
Enge−Rosenblatt2011
Karner2010a
Aslan2015
Wetter2010

DE References in this category:
Lin2011
Nutaro2011
Al−Hammouri2012
Eyisi2012
Riley2011
Fitzgerald2010
Fitzgerald2013
Kudelski2013
Fuller2013
Awais2013b
Awais2013a
Kuhr2013
Zhang2014
Kounev2015
Bogomolov2015
Camus2015
Camus2016
Fourmigue2009
Quesnel2005
Fey1997
Karner2010a
Neema2014

C.3.5 Dynamic structure

References in this category:
Karner2010a

C.3.6 Co-simulation Rate

Single References in this category:
Pedersen2015
Lin2011

Hoepfer2011
Faure2011
Tomulik2011
Sun2011
Bastian2011a
Friedrich2011
Nutaro2011
Busch2012
Schmoll2012
Ni2012
Hassairi2012
Schierz2012
Gunther2012
Quaglia2012
Al−Hammouri2012
Eyisi2012
Riley2011
Roche2012
Fitzgerald2010
Fitzgerald2013
Kudelski2013
Broman2013
Benedikt2013
Benedikt2013b
Fuller2013
Bombino2013
Wang2013
Hafner2013
Zhao2014
Li2011c
Viel2014
Sicklinger2014
Zhang2014
Kounev2015
Bogomolov2015
Bian2015
Dols2016
BenKhaled2012
BenKhaled2014
Yamaura2016
Pedersen2016
Oh2016
Xie2016
Manbachi2016
Schierz2012a
Carstens2003

Stettinger2014
Benedikt2016
Busch2016
Arnold2010
Galtier2015
Aslan2015
Wetter2010

Multi Multi-rate co-simulation denotes that the framework distinguishes between slow and fast sub-systems and dimensions the communication step size accordingly, providing for interpolation/extrapolation of the slow systems.
References in this category:
Gonzalez2011
Awais2013b
Awais2013a
Kuhr2013
Camus2015
Camus2016
Busch2011
Quesnel2005
Arnold2014a
Arnold2014
Arnold2001
Schweizer2014
Schweizer2015d
Sadjina2016
Gu2001
Gu2004
Schweizer2016
Schweizer2015
Schweizer2015a
Fey1997
Acker2015
Enge−Rosenblatt2011
Neema2014

C.3.7 Communication Step Size

Fixed References in this category:
Pedersen2015
Faure2011
Tomulik2011
Bastian2011a
Friedrich2011
Gonzalez2011
Busch2012

Schmoll2012
Ni2012
Hassairi2012
Schierz2012
Quaglia2012
Roche2012
Kudelski2013
Benedikt2013
Benedikt2013b
Bombino2013
Hafner2013
Zhao2014
Li2011c
Awais2013a
Viel2014
Sicklinger2014
Zhang2014
Bian2015
Dols2016
BenKhaled2012
BenKhaled2014
Pedersen2016
Oh2016
Xie2016
Manbachi2016
Carstens2003
Stettinger2014
Arnold2014a
Arnold2001
Schweizer2014
Schweizer2015d
Busch2016
Arnold2010
Gu2001
Gu2004
Schweizer2016
Schweizer2015
Schweizer2015a
Andersson2016
Fey1997
Acker2015
Wetter2010
Neema2014

Variable References in this category:

Lin2011
Hoepfer2011
Sun2011
Nutaro2011
Gunther2012
Al−Hammouri2012
Eyisi2012
Riley2011
Fitzgerald2010
Fitzgerald2013
Broman2013
Fuller2013
Wang2013
Awais2013b
Kuhr2013
Kounev2015
Bogomolov2015
Camus2015
Camus2016
Schierz2012a
Benedikt2016
Busch2011
Quesnel2005
Arnold2014
Sadjina2016
Galtier2015
Neema2014

C.3.8 Strong Coupling Support

None – Explicit Method References in this category:

Pedersen2015
Lin2011
Hoepfer2011
Faure2011
Sun2011
Friedrich2011
Gonzalez2011
Nutaro2011
Busch2012
Schmoll2012
Ni2012
Hassairi2012
Schierz2012
Gunther2012

Quaglia2012
Al−Hammouri2012
Eyisi2012
Riley2011
Roche2012
Fitzgerald2010
Fitzgerald2013
Kudelski2013
Broman2013
Benedikt2013
Benedikt2013b
Fuller2013
Bombino2013
Wang2013
Hafner2013
Zhao2014
Li2011c
Awais2013b
Awais2013a
Zhang2014
Kounev2015
Bogomolov2015
Bian2015
Dols2016
BenKhaled2012
BenKhaled2014
Saidi2016
Yamaura2016
Camus2015
Camus2016
Pedersen2016
Oh2016
Xie2016
Manbachi2016
Schierz2012a
Carstens2003
Stettinger2014
Benedikt2016
Busch2011
Quesnel2005
Arnold2014a
Arnold2014
Sadjina2016
Busch2016
Gu2001
Gu2004

Galtier2015
Aslan2015

Partial – Semi-Implicit Method References in this category:
Busch2012
Schweizer2014
Schweizer2015d
Schweizer2016
Schweizer2015
Schweizer2015a

Full – Implicit Method References in this category:
Tomulik2011
Bastian2011a
Busch2012
Viel2014
Sicklinger2014
Liu2001
Arnold2001
Arnold2010
Acker2015

C.3.9 Results Visualization

Postmortem The results are available after the simulation. References in this category:
Pedersen2015
Lin2011
Hoepfer2011
Faure2011
Tomulik2011
Sun2011
Bastian2011a
Friedrich2011
Gonzalez2011
Nutaro2011
Busch2012
Schmoll2012
Ni2012
Schierz2012
Gunther2012
Quaglia2012
Al−Hammouri2012
Roche2012
Kudelski2013
Broman2013
Benedikt2013

Benedikt2013b
Fuller2013
Wang2013
Hafner2013
Zhao2014
Li2011c
Awais2013b
Awais2013a
Viel2014
Sicklinger2014
Zhang2014
Kounev2015
Bogomolov2015
Bian2015
Dols2016
BenKhaled2012
Camus2015
Camus2016
Xie2016
Schierz2012a
Liu2001
Carstens2003
Stettinger2014
Benedikt2016
Busch2011
Quesnel2005
Arnold2014a
Arnold2014
Arnold2001
Schweizer2014
Schweizer2015d
Sadjina2016
Busch2016
Arnold2010
Gu2001
Gu2004
Schweizer2016
Schweizer2015
Schweizer2015a
Andersson2016
Krammer2015
Galtier2015
Fey1997
Acker2015
Karner2010a
Aslan2015

Wetter2010

Live References in this category:
Hassairi2012
Eyisi2012
Riley2011
Fitzgerald2010
Fitzgerald2013
Yamaura2016
Pedersen2016
Enge−Rosenblatt2011
Neema2014

Interactive References in this category:
Bombino2013
Yamaura2016

C.3.10 Communication Approach

Jacobi References in this category:
Pedersen2015
Hoepfer2011
Faure2011
Tomulik2011
Bastian2011a
Friedrich2011
Schmoll2012
Schierz2012
Gunther2012
Kudelski2013
Broman2013
Hafner2013
Awais2013b
Awais2013a
Sicklinger2014
Bogomolov2015
BenKhaled2012
Saidi2016
Xie2016
Manbachi2016
Schierz2012a
Fourmigue2009
Stettinger2014
Busch2011
Arnold2014a
Arnold2014

Schweizer2014
Schweizer2015d
Sadjina2016
Busch2016
Gu2001
Gu2004
Schweizer2016
Schweizer2015
Schweizer2015a
Andersson2016
Galtier2015
Wetter2010
Neema2014

Gauss-Seidel References in this category:
Lin2011
Hoepfer2011
Sun2011
Bastian2011a
Gonzalez2011
Nutaro2011
Busch2012
Ni2012
Hassairi2012
Schierz2012
Quaglia2012
Al−Hammouri2012
Eyisi2012
Riley2011
Roche2012
Fitzgerald2010
Fitzgerald2013
Benedikt2013
Benedikt2013b
Fuller2013
Bombino2013
Wang2013
Hafner2013
Zhao2014
Li2011c
Awais2013b
Awais2013a
Kuhr2013
Viel2014
Sicklinger2014
Kounev2015

Dols2016
BenKhaled2012
BenKhaled2014
Camus2015
Camus2016
Pedersen2016
Oh2016
Carstens2003
Stettinger2014
Quesnel2005
Arnold2014
Arnold2001
Busch2016
Arnold2010
Acker2015
Aslan2015

D List of Acronyms

CPS	Cyber-Physical System
CT	Continuous Time
DE	Discrete Event
DEVS	Discrete Event System Specification
DTSS	Discrete Time System Specification
FMI	Functional Mock-up Interface
FR	Framework Requirement
GVT	Global Virtual Time
IP	Intellectual Property
IVP	Initial Value Problem
NFR	Non-Functional Requirement
ODE	Ordinary Differential Equation
SR	Simulator Requirement