Stable Adaptive Co-simulation: A Switched Systems Approach

Cláudio Gomes[13], Benoît Legat[2] Raphaël M. Jungers[2], and Hans Vangheluwe[134]

[1] University of Antwerp, Department of Mathematics and Computer Science, Belgium,
[claudio.gomes, hans.vangheluwe]@uantwerp.be
[2] Université catholique de Louvain, ICTEAM, Belgium,
[benoit.legat, raphael.jungers]@uclouvain.be
[3] Flanders Make
[4] McGill University, School of Computer Science

Abstract. Co-simulation promotes the idea that domain specific simulation tools should cooperate in order to simulate the inter-domain interactions that are often observed in complex systems. To get trustworthy results, it is important that this technique preserves the stability properties of the original system.
In this paper, we show how to preserve stability for adaptive co-simulation schemes, which offer fine grained control over the performance/accuracy of the co-simulation. To this end, we apply the joint spectral radius theory to certify that an adaptive co-simulation scheme is stable, and, if that is not possible, we use recent results in this field to create a trace of decisions that lead to instability. With this trace, it is possible to adjust the adaptive co-simulation in order to make it stable.
Our approach is limited by the fact that computing the joint spectral radius is NP-Hard and undecidable in general. In practice, we successfully applied our results to the co-simulation of a double mass-spring-damper system.

Keywords: adaptive co-simulation, stability analysis, joint spectral radius, switched systems

1 Introduction

Co-simulation is the simulation of a complex system via cooperating simulators, mimicking the interactions between subsystems [30,19]. It promotes an efficient integration of the development process by leveraging existing, often specialized, modeling and simulation tools [46,9], and can be applied at any stage [47]. Moreover, the parallelization and decoupling of the computation allows for faster simulations [16,34,8]. Here, simulator means any process that exhibits behavior over time, so this definition encompasses physical prototypes, software components, and human operators [3,15].

Throughout this paper we assume that the simulators are independent of each other[5]. As a consequence, an orchestrator is required to ensure that the simulators exchange data during a co-simulation.

Co-simulation promotes the idea that each simulator decides how to best compute the behavior of the subsystem allocated to it, leaving to the orchestrator the decision of when (with respect to the simulated time) should the simulators exchange data, and in what order [19]. However, as prior work has shown (e.g., [12,42,19,17,28,6,30,5]), the decision on how to best compute the behavior of each subsystem depends on the specific arrangement of all subsystems—such arrangement being called the co-simulation scenario—, and on the decisions of the orchestrator. In sum: no decision concerning how to compute the co-simulation should be taken independently of the co-simulation scenario, which means that simulators should avoid "hard-coded" decisions.

It is currently a matter of research to find out which decisions are scenario dependent, and in this work, we assume that each simulator provides a mechanism to control some of these. Two factors are known to affect these decisions: (1) the co-simulation scenario; and (2) the requirements for the co-simulation.

Regarding the exact moment when these decisions need to be made, in the general case of systems that undergo structural changes (and therefore change the co-simulation scenario), the only possible time to make such decisions is when these changes occur, as demonstrated in [37]. The requirements for the co-simulation can change during the computation itself as well. The purpose of this is to inspect certain transient behavior of interest (e.g., see [23,40,7,27]). We will therefore focus on adaptive co-simulation, where the orchestrator and simulators change the way they compute the co-simulation during the co-simulation itself, as a factors (1) and (2) change.

In the scope of adaptive co-simulation, it is hard to predict which decisions are to be taken without actually computing the co-simulation. It is then natural to wonder whether it is possible to ensure trustworthy co-simulation results, in the face of such uncertainty.

In this paper, we show how to prove that a co-simulation of a stable system is numerically stable, provided that the set of all possible decisions (i.e., reactions to changes in factors (1) and (2)) is known. In particular, we propose to use the joint spectral radius theory [24] to certify the numerical stability of the co-simulation. Furthermore, when a co-simulation cannot be certified as numerically stable, we apply the results in [32,33] to provide a numerically stable co-simulation with a reduced set of possible decisions.

The challenges associated with our approach lie in scaling with respect to the size of the underlying system, number of simulators, and number of decisions; and how to protect the Intellectual Property in each simulator.

In the next section, we introduce an example that motivates our research problem, and will serve as a running example. Section 3 presents some preliminary

[5] Two well known standards for co-simulation—the Functional Mockup Interface Standard for co-simulation [10], and the High Level Architecture [1]—share this assumption.

concepts related to stability in co-simulation and the techniques that we based our contribution on. Then, section 4 details our contribution, and section 5 the related work. Finally, section 6 concludes.

2 Motivational Example

We motivate our work using a simple and well known system, that has been used to study the numerical stability of multiple orchestration algorithms (see, e.g., [12,14,13,29,42,26,4]).

A coupled mass-spring-damper system is shown in fig. 1. We consider two simulators—S_1, S_2—and the allocation depicted in the figure: simulator S_1 computes the behavior of the left-hand-side (LHS) mass, accepting the input coupling force F_c, and producing the position and velocity of the mass as outputs; and S_2 accepts the position and velocity computed by S_1, and produces the coupling force F_c. They are coupled as shown in fig. 2.

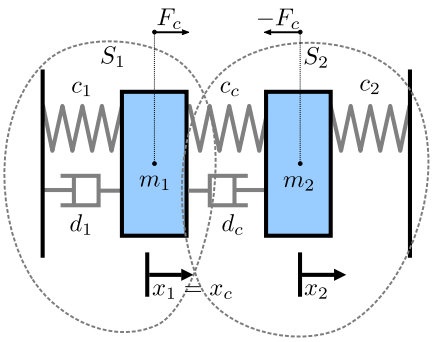

Fig. 1. Example double mass-spring-damper system.

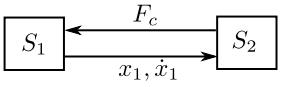

Fig. 2. Example arrangement of simulators.

The dynamics of the LHS mass are given by the following first order ODE:

$$\begin{aligned} &\dot{x}_1(t) = v_1(t); && m_1 \cdot \dot{v}_1(t) = -c_1 \cdot x_1(t) - d_1 \cdot v_1(t) + F_c(t); \\ &x_1(0) = p_1; && v_1(0) = s_1. \end{aligned} \tag{1}$$

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 $F_c(t)$ denotes the input force acting on the mass over time.

The right-hand-side mass is governed by the following first order ODE:

$$\begin{aligned} &\dot{x}_2(t) = v_2(t); && m_2 \cdot \dot{v}_2(t) = -c_2 \cdot x_2(t) - F_c(t); \\ &x_2(0) = p_2; && v_2(0) = s_2; \\ &F_c(t) = c_c \cdot (x_2(t) - x_1(t)) + d_c \cdot (v_2(t) - v_1(t)); \end{aligned} \tag{2}$$

where: c_c and d_c denote the stiffness and damping coefficients of the central spring and damper, respectively; c_2 denotes the stiffness constant for the right spring; p_2 and s_2 the initial position and velocity.

We assume that the co-simulation of this example is computed as shown in algorithm 1 (other orchestration algorithms exist—see [19, Section 4.2] for an overview). The function DoStep(H,S) instructs simulator S to simulate the behavior of its allocated subsystem in the time interval $t \to t + H$. This computation is done using a numerical method and, since the input is not available in the open interval $(t, t + H)$, an extrapolation scheme is used.

Algorithm 1: Jacobi orchestration algorithm for the simulators shown in fig. 2.

Data: The stop time t_f and a communication step size $H > 0$.

$t := 0$;
while $t \leq t_f$ **do**
  $\begin{bmatrix} x_1 & v_1 \end{bmatrix}^T :=$ GetOutput(S_1);
  SetInput(S_2,$\begin{bmatrix} x_1 & v_1 \end{bmatrix}^T$);
  $F_c :=$ GetOutput(S_2);
  SetInput(S_1,F_c);
  DoStep(H,S_1);
  DoStep(H,S_2);
  $t := t + H$;
end

Figure 3 shows multiple co-simulations of the system in fig. 1, using different configurations for the simulators:

`x1` denotes the correct trajectory of $x_1(t)$, for reference, obtained by coupling eqs. (1) and (2) and finding the analytical solution;

`x1_cs_1` denotes the trajectory obtained with a co-simulation where both simulators employ the forward Euler method, using a constant extrapolation of the inputs, and performing 10 internal integration steps per co-simulation step;

`x1_cs_2` is similar to `x1_cs_1`, except each simulator performs only one integration step per co-simulation step;

`x1_cs_3` is obtained with a co-simulation that adaptively combines the configuration used in `x1_cs_1` and `x1_cs_2`, i.e., it varies the number of internal integration steps per simulator.

Comparing the plotted trajectories, we see that there is something wrong with trajectory `x1_cs_2`. Due to the positive damping constants, the original system must always come to a rest, irrespective of the initial values. The co-simulation that produces `x1_cs_2` does not seem to obey this law.

To compare the performance of each co-simulation, we compute the number of model evaluations. For the co-simulations producing the trajectories `x1_cs_1`

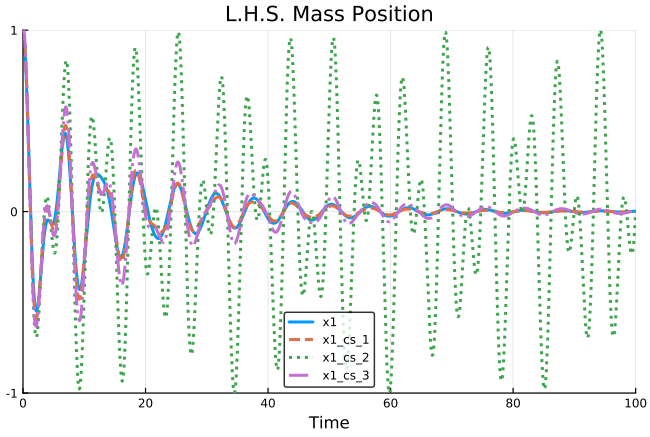

Fig. 3. LHS mass position co-simulations of the system in fig. 1. Parameters: $m_1 = c_1 = m_2 = c_2 = c_c = 1.0$; and $d_1 = d_2 = d_c = 0.1$. The co-simulation step used is $H = 0.1$.

and `x1_cs_2`, this is given as:

$$\frac{t_f}{H} \times \left(\mathsf{Steps}_{S_1} + \mathsf{Steps}_{S_2}\right),$$

where t_f is the maximum simulation time, and Steps_S denotes the number of internal integration steps performed by simulator S, per invocation of DoStep(H,S). The algorithm that computes trajectory `x1_cs_3` is designed to spend 70% of the time using the parameters used to compute `x1_cs_1` and the remaining time using the parameters used to compute `x1_cs_2`. It gives the following evaluations:

$$0.7 \times \mathsf{Evals}_{\mathtt{cs}_1} + 0.3 \times \mathsf{Evals}_{\mathtt{cs}_2}.$$

As can be seen in table 1, the adaptive co-simulation mimics the qualitative behavior of the system (i.e., eventually coming to a rest), with fewer model evaluations than `x1_cs_1`.

Table 1. Total number of model evaluations per co-simulation in fig. 3.

Trajectory	**Evaluations**
`x1_cs_1`	20000
`x1_cs_2`	2000
`x1_cs_3`	14600

This minimal example highlights one of the advantages of adaptive co-simulations: the ability to obtain better tradeoffs between mimicking the qualitative behavior of the original system and performance.

Consider now the adaptive co-simulation `x1_cs_4` shown in fig. 4, which is similar to the policy used to compute `x1_cs_3`, except that more time is spent in the mode where the simulators only take one integration step. Despite being adaptive, it does not seem to come to a rest, which brings to our research problem: how can we tell a stable adaptive co-simulation, from an unstable one? And how can we ensure that the decisions taken during the co-simulation preserve the qualitative properties of the original system?

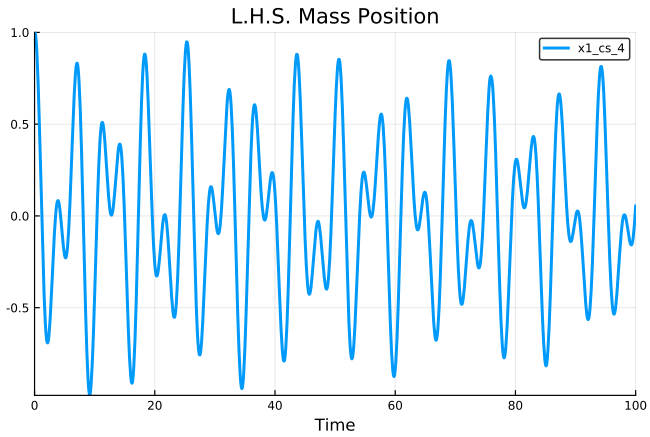

Fig. 4. Example wrong adaptive co-simulation.

The next section provides the necessary background to explore this problem in depth.

3 Background

3.1 Co-simulation

In this paper, we consider a simulator to be an executable unit that expects input signals at agreed-upon points in simulated time, and produces output signals at these times (see [19, Section 4] for a formal definition). The inputs (resp. outputs) correspond to the inputs (resp. outputs) of the subsystem that is allocated to that simulator.

A simulator often employs a numerical method to approximate the state of its subsystem over simulated time. Suppose that an input is provided at time t, and the simulator is instructed to compute until time $t+H$, with a given $H > 0$. Then, the simulator will perform a number of internal integration steps, while guessing what the input is throughout these steps. At time $t + H$, a new input point is provided, and the process is repeated.

A co-simulation scenario is a specific coupling of simulators, reflecting the coupling of the underlying subsystems. Here we assume that this coupling is a set of output-to-input assignments, giving rise to arrangements as exemplified in fig. 2.

An orchestrator is an algorithm that uses the co-simulation scenario to compute a co-simulation. It is responsible for asking simulators to produce outputs, setting their inputs, and controlling their computation over the simulated time, by deciding the co-simulation policy. A simple orchestrator is shown in algorithm 1.

A co-simulation policy is a sequence (over simulated time) of decisions that affect how the co-simulation is computed. In this paper, a policy encompasses:

Solver the numerical solver used by each simulator;

Internal Step Size the internal step size used by each simulator;

Input Approximation the input extrapolation scheme used by each simulator; and

Orchestration the order in which inputs are provided to each simulator (two well known examples are Jacobi and Gauss-Seidel orchestration [19]).

We consider these items because they are known to affect the stability of the co-simulation. For example, [12] studies the stability of the co-simulation using multiple input extrapolation schemes, and [42] studies the stability under different orchestration algorithms.

3.2 (Numerical) Stability

Consider the following first order ODE:

$$\dot{x} = Ax; \qquad x(0) \text{ is given;} \tag{3}$$

where $x(t)$ is a real-valued vector, and A is a square real matrix.

The solution $x(t)$ to the system in eq. (3) is stable if $x(t)$ tends to the origin, regardless of $x(0)$. In other words, $\lim_{t\to\infty} \|x(t)\| = 0$.

Suppose that the solution to eq. (3) is approximated by the following discrete time system:

$$\tilde{x}_{i+1} = \tilde{A}\tilde{x}_i; \qquad \tilde{x}_0 = x(0); \tag{4}$$

where $\tilde{x}_i$ is a real-valued vector, and $\tilde{A}$ is a square real matrix.

We say that the system in eq. (4) is stable if, for all $\tilde{x}_0$,

$$\lim_{i\to\infty} \|\tilde{x}_i\| = 0 \Leftrightarrow \lim_{i\to\infty} \|\tilde{A}^i\| = 0, \tag{5}$$

for any vector norm $\|\tilde{x}\|$, and any matrix norm $\|\tilde{A}\|$ satisfying the *submultiplicativity* property.

Assuming that the system in eq. (3) is stable, it is important that the approximating system in eq. (4) preserves this property, in which case, we denote it as being numerically stable. The importance of preserving this property lies in the fact that the computation of the approximation naturally introduces errors. If the system in eq. (4) is numerically stable, the error remains bounded.

The condition in eq. (5) can be studied by means of the spectral radius $\rho(\tilde{A})$ [44, Theorem 1.3.2]:

$$\rho(\tilde{A}) < 1 \Leftrightarrow \lim_{i\to\infty} \|\tilde{A}^i\| = 0,$$

where $\rho(\tilde{A})$ is given by Gelfand's formula or the maximum absolute eigenvalue:

$$\rho(\tilde{A}) = \lim_{i\to\infty} \|\tilde{A}^i\|^{1/i} = \max_j |\lambda_j|\,, \tag{6}$$

and λ_j is the j-th eigenvalue of $\tilde{A}$.

As detailed in [12,42,19] and references therein, the numerical stability of a co-simulation is analyzed by assuming that the underlying coupled system can be written as in eq. (3) and is stable, and computing the discrete time system in the form of eq. (4) that represents the co-simulation. Here, we illustrate how this is done for the example in fig. 1 (a more general description is given in the above references). This procedure can be generalized to any number of simulators, as long as the underlying coupled system can be written as in eq. (3) (for conditions that ensure this, see [5, Section 2]).

Consider now the example of fig. 1, and suppose that the orchestrator (following algorithm 1) and simulators are at time t_i. In the interval $t \in [t_i, t_{i+1}]$, each simulator S_j, with $j = 1, 2$, is trying to approximate the solution to a linear ODE,

$$\begin{aligned} \dot{x}_j &= A_j \cdot x_j + B_j \cdot u_j \\ y_j &= C_j \cdot x_j + D_j \cdot u_j \end{aligned} \tag{7}$$

where A_j, B_j, C_j, D_j are matrices with appropriate dimensions, and the initial state $x_j(t_i)$ is the state computed in the most recent co-simulation step. We assume that either D_1 or D_2 is the null matrix, so that the coupled system can be written as eq. (3). In this example, $D_1 = \mathbf{0}$.

Without loss of generality (for more sophisticated input extrapolation techniques, see [12, Equation (9)]), we assume that each simulator uses a constant extrapolation to approximate the input in the interval $[t_i, t_{i+1})$. That is, $\tilde{u}_j(t) = u_j(t_i)$, for $t \in [t_i, t_{i+1})$. Then, eq. (7) can be re-written to represent the unforced system being integrated by each simulator:

$$\begin{bmatrix} \dot{x}_j \\ \dot{\tilde{u}}_j \end{bmatrix} = \begin{bmatrix} A_j & B_j \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} x_j \\ \tilde{u}_j \end{bmatrix} \tag{8}$$

We can represent the multiple internal integration steps of eq. (8), performed by the simulator S_j in the interval $t \in [t_i, t_{i+1}]$, as

$$\begin{bmatrix} x_j(t_{i+1}) \\ \tilde{u}_j(t_{i+1}) \end{bmatrix} = \tilde{A}_j^{k_j} \cdot \begin{bmatrix} x_j(t_i) \\ \tilde{u}_j \end{bmatrix} \tag{9}$$

where $\tilde{A}_j$ represents a single integration step of the numerical method (e.g., $\tilde{A}_j = \mathbf{I} + h_j \begin{bmatrix} A_j & B_j \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$ for the forward Euler method), $k_j = (t_{i+1} - t_i)/h_j$ is the number of internal steps, and $0 < h_j \leq H$ is the internal fixed step size that

divides H. Note that eq. (9) represents a discrete time system modeling the behavior of the simulator at a single co-simulation step, with no inputs. Now we just have to represent how the simulators exchange data at the end/beginning of a co-simulation step.

At the beginning of the co-simulation step i, we wish to enforce $u_1(t_i) = y_2(t_i)$ and $u_2(t_i) = y_1(t_i)$. This, together with eq. (7), gives,

$$\begin{aligned} u_1(t_i) &= C_2 \cdot x_2(t_i) + D_2 C_1 \cdot x_1(t_i). \\ u_2(t_i) &= C_1 \cdot x_1(t_i) \end{aligned} \tag{10}$$

Finally, eqs. (8) to (10) are combined to write the co-simulation step in the form of eq. (4) as

$$\begin{bmatrix} x_1(t_{i+1}) \\ x_2(t_{i+1}) \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \tilde{A}_1^{k_1} & \mathbf{0} \\ \mathbf{0} & \tilde{A}_2^{k_2} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & C_2 \\ \mathbf{0} & \mathbf{I} \\ C_1 & D_1 \cdot C2 \end{bmatrix}}_{\tilde{A}} \begin{bmatrix} x_1(t_i) \\ x_2(t_i) \end{bmatrix} \tag{11}$$

whose stability is easily checked with eq. (6).

We remark that eq. (11) represents an abstraction of how the co-simulation is computed, for analysis purposes. In practice, the co-simulation itself may include optimizations, parallelism, etc... which are neglected when building eq. (11).

3.3 Joint Spectral Radius

The definitions we present here are adapted from [24].

Consider the following switched discrete time system:

$$x_{i+1} = A_{\sigma(i)} x_i : \quad \sigma(i) \in \{0, \ldots, m-1\}, \ A_{\sigma(i)} \in \Sigma \tag{12}$$

where x_0 is given, $\{0, \ldots, m-1\}$ is the set of modes, $\sigma(i)$ is the mode active at step i, and $\Sigma = \{A_i\}_{i=0}^{m-1}$ is a sequence of m real square matrices.

We denote the sequence $\sigma(0), \sigma(1), \ldots$ as the *switching signal*, where $A_{\sigma(i)} \in \Sigma$ represents the matrix used to compute x_{i+1} from x_i in eq. (12). A switching signal $\sigma(0), \sigma(1), \ldots, \sigma(i)$ induces the matrix product $A_{\sigma(i-1)} \ldots A_{\sigma(1)} \cdot A_{\sigma(0)}$. Let

$$\Sigma^i = \left\{ A_{p_{i-1}} A_{p_{i-2}} \ldots A_{p_0} : A_{p_j} \in \Sigma, 0 \leq p_j < m, j = 0, \ldots, i-1 \right\}$$

be the set of all products induced by switching signals with length i. Note that, for any given switching signal $\sigma(0), \sigma(1), \ldots \sigma(i-1)$, $x_{i+1} = Ax_0$ for some $A \in \Sigma^i$.

The system in eq. (12) is stable if, for any x_0, and any switching signal, $\lim_{i \to \infty} \|x_i\| = 0$.

The Joint Spectral Radius $\hat{\rho}(\Sigma)$ (JSR) is essentially a generalization of Gelfand's formula, in eq. (6), to arbitrary products of matrices in Σ [41]:

$$\begin{aligned} \hat{\rho}_i(\Sigma) &= \sup \left\{ \|A\|^{1/i} : A \in \Sigma^i \right\} \\ \hat{\rho}(\Sigma) &= \limsup_{i \to \infty} \hat{\rho}_i(\Sigma) \end{aligned} \tag{13}$$

Using the JSR, we can characterize the stability of the system in eq. (12) by noting that [24, Theorem 1], for any bounded set Σ,

$$\hat{\rho}(\Sigma) < 1 \Leftrightarrow \text{ for all } \sigma, \lim_{i \to \infty} \left\| A_{\sigma(i)} A_{\sigma(i-1)} \cdots A_{\sigma(0)} \right\| = 0. \tag{14}$$

To determine whether the system in eq. (12) is stable, note that the limit in eq. (13) exists, and any finite i satisfies:

$$\hat{\rho}(\Sigma) \leq \hat{\rho}_i(\Sigma) \text{ [24, Lemma 1.2].}$$

Therefore, if there exists i, such that $\hat{\rho}_i(\Sigma) < 1$, then the switched system is stable. Note however, that checking whether $\hat{\rho}(\Sigma) < 1$ is NP-Hard [11] and undecidable [24, Proposition 2.9] in general.

Other algorithms exist to estimate $\hat{\rho}(\Sigma)$, and we refer the reader to [20,22,25,39,36].

4 Stability Certification of Adaptive Co-simulations

In this section, we first describe how to use the concepts introduced in the previous section to determine the numerical stability of an adaptive co-simulation. Then, we propose a way to address the case when the adaptive co-simulation is not numerically stable.

4.1 Stability

Equation (11) represents a single co-simulation step, which, as can be seen from eqs. (7) to (10), represents a specific: system arrangement; coupling approach; simulator input approximation; internal solver method; internal simulator step size h_j; and communication step size H. If any of these items changes from one co-simulation step to the next, the co-simulation is adaptive, and is best described as a discrete time switched system, of the form of eq. (12), where Σ includes every possible variation of the matrix $\tilde{A}$ in eq. (11), constructed as explained in section 3.2.

To exemplify, in the co-simulation of the system in fig. 1, suppose that the decision space is as follows:

Arrangement is the one in fig. 2;

Coupling is the one in algorithm 1 but a Gauss-Seidel, Strong coupling, or others, could have been used [21];

Input Approximation is the constant extrapolation but higher order input approximations can be applied [12];

Solver can be forward Euler, or the midpoint method [48, Section II.1];
Solver Step Size can be $H/10$ or H;
Communication Step Size H is 0.1;
Then Σ contains 16 matrices, representing every possible combination of policies, per simulator, from one co-simulation step to the next.

Applying the result in eq. (14) ensures that any possible decision sequence taken by the co-simulation always produces a numerically stable co-simulation. This is a strong result in the sense that we do not need to know anything about how the decisions are made.

In the example proposed, $\hat{\rho}(\Sigma) \geq 1$. To see why this is the case, let $A_{\text{cs\_2}}$ denote that co-simulation step matrix that uses $H = 0.1$ and solver step size equal to H. Then, computing the spectral radius $\rho(A)$, one observes that $\rho(A) > 1$. This means that there is a switching signal (always use $A_{\text{cs\_2}}$ to compute the next co-simulation step) that causes the co-simulation to not be stable. In fact, the result is the trajectory `x1_cs_2`, plotted in fig. 3.

4.2 Stabilization

As the paragraph above shows, if there is a matrix $A \in \Sigma$ such that $\rho(A) \geq 1$, then we have that $\hat{\rho}(\Sigma) \geq 1$. This immediately suggests an optimization to be done before computing the JSR: exclude all unstable matrices. That is, we set

$$\Sigma_0 = \Sigma \setminus \{A\}, \forall A \in \Sigma : \rho(A) \geq 1.$$

After computing Σ_0, it can still be the case that $\hat{\rho}(\Sigma_0) \geq 1$, as the product of stable matrices is not necessarily stable (see, e.g., [24, Figure 1.2]). Furthermore, $\hat{\rho}(\Sigma_0) \geq 1$ does not imply that there exists a finite i and a $A \in \Sigma_0^i$ such that $\rho(A) \geq 1$ (see, e.g., [24, Section 2.4], with the case that $\rho(A) = 1$). This means that no algorithm can always ensure that a stable co-simulation is attained. Fortunately, in practice, the algorithm proposed in [32] works well.

The work in [32] approximates $\hat{\rho}(\Sigma_0)$, allowing us to check whether $\hat{\rho}(\Sigma_0) < 1$, and, more importantly, returns a sequence $p_0, \ldots, p_{i-1}$ such that $\rho(A_{p_{i-1}} \ldots A_{p_0}) \approx \hat{\rho}(M_0)$ to any desired level of accuracy. Computing $\Sigma_1 = \Sigma_0 \setminus A_{p_j}$ for one $j \in \{0, \ldots, i-1\}$ and iterating allows one to obtain a Σ_* such that $\hat{\rho}(\Sigma_*) < 1$.

In the adaptive co-simulation of the system introduced in fig. 3, we have that $\Sigma_* = \Sigma_0$ excludes the matrix $A_{\text{cs\_2}}$, and $\hat{\rho}(\Sigma_0) \leq 0.992905$.

4.3 Conservativeness

As the previous result shows, applying this procedure to the adaptive co-simulation introduced in the previous sub-section results in a stable adaptive co-simulation that will never use the matrix $A_{\text{cs\_2}}$. This is too restrictive. To see why, note that, as illustrated in plots of fig. 3, a careful use of the decisions embedded in matrix $A_{\text{cs\_2}}$ actually yields a co-simulation that outperforms the other non-adaptive co-simulations (see the stable trajectory `x1_cs_3` in table 1).

We propose a straightforward solution to this problem: apply the stabilization procedure to $Q = \Sigma^q$, which includes all products of length q of matrices in Σ for a given $q > 0$ (there is little use to including all products of length up to q because these are subsequences of the products of length m). The matrix products in the stabilized Q_* may include combinations of matrices that would otherwise have been removed.

In the adaptive co-simulation example, we set $q = 2$ and we obtain Q_* that only excludes the matrix $A_{\text{cs\_2}}A_{\text{cs\_2}}$, which means that the policies embedded in $A_{\text{cs\_2}}$ can still be used, provided that they are alternated with any other policies. Applying the algorithm in [32] yields $\hat{\rho}(\Sigma_0^2) \leq 0.982986$.

4.4 Implementation

If the stabilization procedure terminates, we are left with Q_*: a set of sequences of matrix products of length q. Since we abstracted how each decision is taken at each co-simulation step, we still need to ensure that, at run-time, the decisions taken by the adaptive co-simulation remain within the allowed decisions (in the set Q_*).

To shed light on this problem, note that each sequence $p_0, \ldots, p_{q-1}$ that induces the matric product $A_{p_{q-1}} \ldots A_{p_0} \in \Sigma^q$, can be associated with one, and only one, natural number $d_{p_0 \ldots p_{q-1}} \in \mathbb{N}_0$ computed as a conversion from base-m digit to a decimal number:

$$d_{p_0 \ldots p_{q-1}} = \sum_{j=0}^{q-1} p_j \cdot m^j, \tag{15}$$

where m is the number of matrices in Σ.

We therefore propose to allocate a m^q-bit array, where the position $d_{p_0 \ldots p_{q-1}}$ of the array indicates whether the matrix product $A_{p_{q-1}} \ldots A_{p_0} \in Q_*$. Then, at time t_i with $i \geq q-1$, the previous q policies are used to reconstruct $d_{\sigma_{i-q-1} \ldots \sigma_i}$ and check whether the corresponding subsequence is safe to take. If the policy $\sigma(i)$ is not safe to take, then the immediate neighbors of $d_{\sigma_{i-q-1} \ldots \sigma_i}$ in the bit array can be inspected to find whether there are safe policies that can be selected. fig. 5 illustrates a scenario where $q = 3, m = 16$ and the orchestrator is about to decide to use the policies embedded in matrix A_{14}. A quick look-up to position 2158, obtained with eq. (15), shows that this is not allowed. The neighbouring positions show alternative matrices that can be used.

5 Related Work

There is a huge body of work in techniques to determine the stability of a discrete time switched system. For introductions and overviews on the topic, please see [35,45,24,38,2].

We highlight the work in [31], where an algorithm is proposed to search for a stable periodic switching signal. This work differs from our because we are

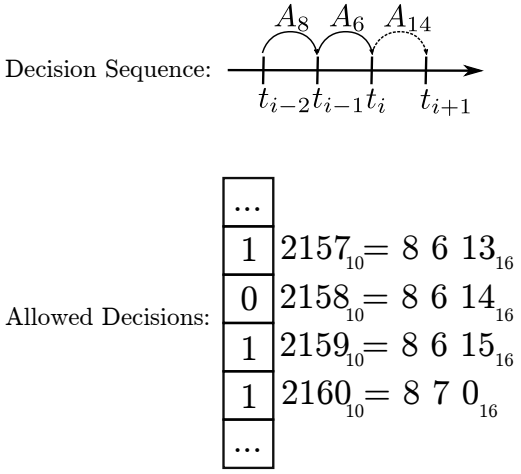

Fig. 5. Runtime structures of decision sequence monitor.

interested is removing all non-stable periodic switching signals, and retaining all the stable ones.

To the best of our knowledge, there is no work that applies the above results to the topic of adaptive co-simulation. For applications to the stability analysis of non-adaptive orchestration algorithms, we refer the reader to [43,42,12,18].

6 Conclusion

In this paper, we introduce the problem of ensuring numerical stability in the context of adaptive co-simulation. To address this problem, we describe how to model the adaptive co-simulation as a discrete timed switched system, incorporating all possible policies. Then we use recent results [32] to determine whether the adaptive co-simulation is numerically stable. If it is not, we propose an attempt to make it stable by reducing the set of allowed policies. Finally, we describe how to implement a simple monitor that ensures that the adaptive co-simulation only takes the accepted policies during execution.

The experiments made throughout the paper to exemplify our approach is available for download[6].

There are two major limitations in this work: (i) the stability of the adaptive co-simulation is not decidable in general, so the algorithm we propose here may not terminate; and (ii) for large numbers of policies, the computation of the joint spectral radius becomes impractical.

To address these, we will explore whether the construction of adaptive co-simulation yields any structure that can be leveraged (e.g., common reducibility) to accelerate the computation of the joint spectral radius.

[6] http://msdl.cs.mcgill.ca/people/claudio/hybrid_cosim_analysis

References

1. 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
2. Ahmadi, A.A., Jungers, R., Parrilo, P.A., Roozbehani, M.: Analysis of the joint spectral radius via lyapunov functions on path-complete graphs. In: Proceedings of the 14th international conference on Hybrid systems: computation and control - HSCC '11, p. 13. ACM Press, Chicago, IL, USA (2011). DOI 10.1145/1967701.1967706. URL http://portal.acm.org/citation.cfm?doid=1967701.1967706
3. Alvarez Cabrera, A.A., Woestenenk, K., Tomiyama, T.: An architecture model to support cooperative design for mechatronic products: A control design case. Mechatronics **21**(3), 534–547 (2011). DOI 10.1016/j.mechatronics.2011.01.009
4. Arnold, M.: Stability of Sequential Modular Time Integration Methods for Coupled Multibody System Models. Journal of Computational and Nonlinear Dynamics **5**(3), 9 (2010). DOI 10.1115/1.4001389
5. Arnold, M., Clauß, C., Schierz, T.: Error Analysis and Error Estimates for Co-simulation in FMI for Model Exchange and Co-Simulation v2.0. In: S. Schöps, A. Bartel, M. Günther, W.E.J. ter Maten, C.P. Müller (eds.) Progress in Differential-Algebraic Equations, pp. 107–125. Springer Berlin Heidelberg, Berlin, Heidelberg (2014). DOI 10.1007/978-3-662-44926-4_6
6. Bastian, J., Clauß, C., Wolf, S., Schneider, P.: Master for Co-Simulation Using FMI. In: 8th International Modelica Conference, pp. 115–120. Dresden, Germany (2011). DOI 10.3384/ecp11063115
7. Beltrame, G., Sciuto, D., Silvano, C.: Multi-Accuracy Power and Performance Transaction-Level Modeling. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **26**(10), 1830–1842 (2007). DOI 10.1109/TCAD.2007.895790
8. Ben Khaled, A., Ben Gaid, M., Pernet, N., Simon, D.: Fast multi-core co-simulation of Cyber-Physical Systems: Application to internal combustion engines. Simulation Modelling Practice and Theory **47**, 79–91 (2014). DOI 10.1016/j.simpat.2014.05.002
9. Blochwitz, T., Otter, M., Arnold, M., Bausch, C., Clauss, C., Elmqvist, H., Junghanns, A., Mauss, J., Monteiro, M., Neidhold, T., Neumerkel, D., Olsson, H., Peetz, J.V., Wolf, S.: The Functional Mockup Interface for Tool independent Exchange of Simulation Models. In: 8th International Modelica Conference, pp. 105–114. Linköping University Electronic Press; Linköpings universitet, Dresden, Germany (2011). DOI 10.3384/ecp11063105
10. Blockwitz, T., Otter, M., Akesson, J., Arnold, M., Clauss, C., Elmqvist, H., Friedrich, M., Junghanns, A., Mauss, J., Neumerkel, D., Olsson, H., Viel, A.: Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models. In: 9th International Modelica Conference, pp. 173–184. Linköping University Electronic Press, Munich, Germany (2012). DOI 10.3384/ecp12076173
11. Blondel, V.D., Tsitsiklis, J.N.: The boundedness of all products of a pair of matrices is undecidable. Systems & Control Letters **41**(2), 135–140 (2000). DOI 10.1016/S0167-6911(00)00049-9. URL http://linkinghub.elsevier.com/retrieve/pii/S0167691100000499
12. Busch, M.: Continuous approximation techniques for co-simulation methods: Analysis of numerical stability and local error. ZAMM - Journal of Applied Mathematics and Mechanics **96**(9), 1061–1081 (2016). DOI 10.1002/zamm.201500196

13. Busch, M., Schweizer, B.: 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, pp. 25–27 (2010)
14. Busch, M., Schweizer, B.: Stability of Co-Simulation Methods Using Hermite and Lagrange Approximation Techniques. In: ECCOMAS Thematic Conference on Multibody Dynamics, pp. 1–10. Brussels, Belgium (2011)
15. Faure, C., Ben Gaid, M., Pernet, N., Fremovici, M., Font, G., Corde, G.: Methods for real-time simulation of Cyber-Physical Systems: application to automotive domain. In: 2011 7th International Wireless Communications and Mobile Computing Conference, pp. 1105–1110. IEEE (2011). DOI 10.1109/IWCMC.2011.5982695
16. Friedrich, M.: Parallel Co-Simulation for Mechatronic Systems. Ph.D. thesis (2011)
17. Gomes, C.: Foundations for Continuous Time Hierarchical Co-simulation. In: ACM Student Research Competition (ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems), p. to appear. Saint Malo, Brittany, France (2016)
18. Gomes, C., Karalis, P., Navarro-López, E.M., Vangheluwe, H.: Approximated Stability Analysis of Bi-Modal Hybrid Co-simulation Scenarios. In: 1st Workshop on Formal Co-Simulation of Cyber-Physical Systems, p. to appear. Trento, Italy (2017)
19. Gomes, C., Thule, C., Broman, D., Larsen, P.G., Vangheluwe, H.: Co-simulation: State of the art. Tech. rep. (2017). URL http://arxiv.org/abs/1702.00686
20. Gripenberg, G.: Computing the joint spectral radius. Linear Algebra and its Applications **234**, 43–60 (1996). DOI 10.1016/0024-3795(94)00082-4. URL http://linkinghub.elsevier.com/retrieve/pii/0024379594000824
21. Gu, B., Asada, H.H.: Co-simulation of algebraically coupled dynamic subsystems. In: American Control Conference, vol. 3, pp. 2273–2278. Arlington, VA, USA (2001). DOI 10.1109/ACC.2001.946089
22. Guglielmi, N., Zennaro, M.: An algorithm for finding extremal polytope norms of matrix families. Linear Algebra and its Applications **428**(10), 2265–2282 (2008). DOI 10.1016/j.laa.2007.07.009. URL http://linkinghub.elsevier.com/retrieve/pii/S0024379507003126
23. Hines, K., Borriello, G.: Selective focus as a means of improving geographically distributed embedded system co-simulation. In: 8th IEEE International Workshop on Rapid System Prototyping, 1997, pp. 58–62 (1997). DOI 10.1109/IWRSP.1997.618825
24. Jungers, R.: The joint spectral radius: theory and applications, vol. 385. Springer Science & Business Media (2009)
25. Jungers, R.M., Cicone, A., Guglielmi, N.: Lifted Polytope Methods for Computing the Joint Spectral Radius. SIAM Journal on Matrix Analysis and Applications **35**(2), 391–410 (2014). DOI 10.1137/130907811. URL http://epubs.siam.org/doi/10.1137/130907811
26. Kalmar-Nagy, T., Stanciulescu, I.: Can complex systems really be simulated? Applied Mathematics and Computation **227**, 199–211 (2014). DOI 10.1016/j.amc.2013.11.037
27. Karner, M., Armengaud, E., Steger, C., Weiss, R.: 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, pp. 544–549. European Design and Automation Association, 3001 Leuven, Belgium, Belgium (2010)

28. Karner, M., Steger, C., Weiss, R., Armengaud, E.: Optimizing HW/SW Co-simulation based on run-time model switching. In: Specification & Design Languages, 2009. FDL 2009. Forum on, pp. 1–6 (2009)
29. Kübler, R., Schiehlen, W.: Modular Simulation in Multibody System Dynamics. Multibody System Dynamics **4**(2-3), 107–127 (2000). DOI 10.1023/A:1009810318420
30. Kübler, R., Schiehlen, W.: Two Methods of Simulator Coupling. Mathematical and Computer Modelling of Dynamical Systems **6**(2), 93–113 (2000). DOI 10.1076/1387-3954(200006)6:2;1-M;FT093
31. Kundu, A., Chatterjee, D.: Stabilizing discrete-time switched linear systems. In: Hybrid systems: computation and control, pp. 11–20. ACM Press, Berlin, Germany (2014). DOI 10.1145/2562059.2562114. URL http://dl.acm.org/citation.cfm?doid=2562059.2562114
32. Legat, B., Jungers, R.M., Parrilo, P.A.: Generating Unstable Trajectories for Switched Systems via Dual Sum-Of-Squares Techniques. In: Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control - HSCC '16, pp. 51–60. ACM Press, New York, New York, USA (2016). DOI 10.1145/2883817.2883821. URL http://dl.acm.org/citation.cfm?doid=2883817.2883821
33. Legat, B., Parrilo, P.A., Jungers, R.M.: Certifying unstability of Switched Systems using Sum of Squares Programming. Tech. rep. (2017). URL http://arxiv.org/abs/1710.01814
34. Lelarasmee, E., Ruehli, A.E., Sangiovanni-Vincentelli, A.L.: 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, vol. 1, pp. 131–145 (1982). DOI 10.1109/TCAD.1982.1270004
35. Lin, H., Antsaklis, P.J.: Stability and Stabilizability of Switched Linear Systems: A Survey of Recent Results. IEEE Transactions on Automatic Control **54**(2), 308–322 (2009). DOI 10.1109/TAC.2008.2012009. URL http://ieeexplore.ieee.org/document/4782010/
36. Maesumi, M.: An efficient lower bound for the generalized spectral radius of a set of matrices. Linear Algebra and its Applications **240**, 1–7 (1996). DOI 10.1016/0024-3795(94)00171-5. URL http://linkinghub.elsevier.com/retrieve/pii/0024379594001715
37. Mosterman, P.J.: An Overview of Hybrid Simulation Phenomena and Their Support by Simulation Packages. In: F.W. Vaandrager, J.H. van Schuppen (eds.) Hybrid Systems: Computation and Control SE - 17, *Lecture Notes in Computer Science*, vol. 1569, pp. 165–177. Springer Berlin Heidelberg, Berg en Dal, The Netherlands (1999). DOI 10.1007/3-540-48983-5_17
38. Parrilo, P.A., Jadbabaie, A.: Approximation of the Joint Spectral Radius of a Set of Matrices Using Sum of Squares. In: Hybrid Systems: Computation and Control, pp. 444–458. Springer, Berlin, Heidelberg, Pisa, Italy (2007). DOI 10.1007/978-3-540-71493-4_35. URL http://link.springer.com/10.1007/978-3-540-71493-4{_}35
39. Parrilo, P.A., Jadbabaie, A.: Approximation of the joint spectral radius using sum of squares. Linear Algebra and its Applications **428**(10), 2385–2402 (2008). DOI 10.1016/j.laa.2007.12.027. URL http://www.sciencedirect.com/science/article/pii/S0024379508000281http://linkinghub.elsevier.com/retrieve/pii/S0024379508000281
40. Radetzki, M., Khaligh, R.S.: Accuracy-adaptive Simulation of Transaction Level Models. In: Proceedings of the Conference on Design, Automation and Test in

Europe, DATE '08, pp. 788–791. ACM, New York, NY, USA (2008). DOI 10.1145/1403375.1403566
41. Rota, G.C., Strang, W.: A note on the joint spectral radius. In: Proceedings of the Netherlands Academy 22, pp. 379–381 (1960)
42. Schweizer, B., Li, P., Lu, D.: Explicit and Implicit Cosimulation Methods: Stability and Convergence Analysis for Different Solver Coupling Approaches. Journal of Computational and Nonlinear Dynamics **10**(5), 051,007 (2015). DOI 10.1115/1.4028503
43. Schweizer, B., Li, P., Lu, D., Meyer, T.: Stabilized implicit co-simulation methods: solver coupling based on constitutive laws. Archive of Applied Mechanics **85**(11), 1559–1594 (2015). DOI 10.1007/s00419-015-0999-2
44. Stuart, A., Humphries, A.R.: Dynamical systems and numerical analysis, vol. 2. Cambridge University Press (1998)
45. Sun, Z., Ge, S.: Analysis and synthesis of switched linear control systems. Automatica **41**(2), 181–195 (2005). DOI 10.1016/j.automatica.2004.09.015. URL http://linkinghub.elsevier.com/retrieve/pii/S0005109804002778
46. Tomiyama, T., D'Amelio, V., Urbanic, J., ElMaraghy, W.: Complexity of Multi-Disciplinary Design. CIRP Annals - Manufacturing Technology **56**(1), 185–188 (2007). DOI 10.1016/j.cirp.2007.05.044
47. Van der Auweraer, H., Anthonis, J., De Bruyne, S., Leuridan, J.: Virtual engineering at work: the challenges for designing mechatronic products. Engineering with Computers **29**(3), 389–408 (2013). DOI 10.1007/s00366-012-0286-6
48. Wanner, G., Hairer, E.: Solving ordinary differential equations I: Nonstiff Problems, vol. 1, springer s edn. Springer-Verlag (1991)