Demo: Stabilization Technique in INTO-CPS

Cláudio Gomes[2,4](✉), Casper Thule[1], Kenneth Lausdahl[5], Peter Gorm Larsen[1], and Hans Vangheluwe[2,3,4]

[1] DIGIT, Department of Engineering, Aarhus University, Aarhus, Denmark
{casper.thule,pgl}@eng.au.dk
[2] University of Antwerp, Antwerp, Belgium
{claudio.gomes,hans.vangheluwe}@uantwerp.be
[3] McGill University, Montreal, Canada
[4] Flanders Make, Lommel, Belgium
[5] Mjølner Informatics A/S, Aarhus, Denmark
Kenneth@lausdahl.com

Abstract. Despite the large number of applications and growing interest in the challenges that co-simulation poses, the field is fragmented into multiple application domains, with limited sharing of knowledge.

This demo promotes a deeper understanding of a well known stabilization feature in co-simulation, which is used in the INTO-CPS tool chain.

We develop the techniques that explain the empirical results of instability of the double mass-spring-damper system, and how to the stabilization feature improves the results. Moreover, we show how the restrictions of the Functional Mock-up Interface Standard impacts stability.

Keywords: Stability · Simulation · Co-simulation

1 Introduction

INTO-CPS provides an entire tool chain [8] that enables combining different tools and formalisms using co-simulation [6]. This demo provides the theoretical rationale for the stabilization feature of the Co-simulation Orchestration Engine from INTO-CPS called Maestro [12]. The feature will be illustrated with a small case study that is documented online [10].

This demo assumes that the reader is familiar with the main concepts in co-simulation (see, e.g., [7]).

This work was executed under the framework of the COST Action IC1404 – Multi-Paradigm Modelling for Cyber-Physical Systems (MPM4CPS), and partially supported by: Flanders Make vzw, the strategic research centre for the manufacturing industry; and PhD fellowship grants from the Agency for Innovation by Science and Technology in Flanders (IWT, dossier 151067).

In the next section, we describe the principles of stability analysis for linear Ordinary Differential Equations (ODEs), and linear discrete time systems. Then, in Sect. 3, we apply these principles to analyse the numerical stability of the commonly used Jacobi algorithm within the FMI context, and the stabilization method used in INTO-CPS. While the master algorithms are applicable outside the Functional Mockup Interface (FMI) context, the FMI version 2.0 has constraints that makes the stability analysis not applicable to other contexts.

2 Stability of Linear Systems

This section is based on [7].

Notation. We denote vectors with bold face, and we use capital letters for matrices and vector valued functions. Given a vector $\boldsymbol{x}$, we denote its transpose as $\boldsymbol{x}^T$. Furthermore, we denote the i-th element of vector $\boldsymbol{x}$ by x_i, so that $\boldsymbol{x} = \begin{bmatrix} x_1 \; x_2 \cdots x_n \end{bmatrix}^T$. Similarly, $F_i(\boldsymbol{x})$ denotes the i-th element of the vector returned by $F(\boldsymbol{x})$.

A linear ODE has the following form:

$$\dot{\boldsymbol{x}} = A\boldsymbol{x}, \tag{1}$$

where $\boldsymbol{x}(t)$ is a vector function, and A is a constant matrix. When an initial condition in the form $\boldsymbol{x}(0) = \boldsymbol{x_0}$ is specified, we denote Eq. (1) as an Initial Value Problem (IVP).

Example 1. The mass-spring-damper system, illustrated in Fig. 1a, is modelled by the following second order ordinary differential equation:

$$\ddot{x} = \frac{1}{m}(-cx - d\dot{x} + f_e(t)),$$

where x denotes the position of the mass, $c > 0$ is the stiffness coefficient of the spring, $d > 0$ is the damping constant of the damper, t is time, and $f_e(t)$ denotes an external force exerted on the mass.

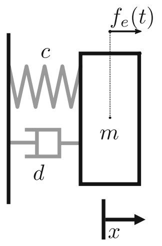
The above equation can be put into the form of Eq. (1) by introducing a new variable for velocity, $v = \dot{x}$, and letting the vector $\boldsymbol{x} = \begin{bmatrix} x \; v \end{bmatrix}^T$. Given an initial position x_0 and velocity v_0, we obtain the following:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = F(\begin{bmatrix} x \\ v \end{bmatrix}, f_e(t)) = \begin{bmatrix} v \\ (1/m)(-cx - dv + f_e(t)) \end{bmatrix}, \text{ with } \boldsymbol{x}(0) = \begin{bmatrix} x_0 \\ v_0 \end{bmatrix}.$$

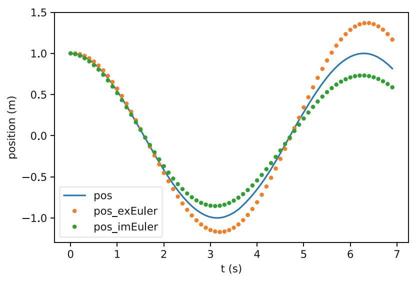
Figure 1b shows the solution of the position component of the mass-spring-damper IVP, introduced in Example 1, and will be explained below. The solution to the velocity component is omitted.

We say that the system in Eq. (1) is *asymptotically stable* when all its solutions tend to zero as time passes, regardless of the initial value specified. Formally,

$$\lim_{t \to \infty} \|\boldsymbol{x}(t)\| = 0, \text{ for all } x(t) \text{ satisfying Eq. (1).} \tag{2}$$



(a)



(b)

Fig. 1. Position (and its approximations) over time of the mass-spring-damper system. Parameters are: $h = 0.1, m = c = 1, d = 10^{-4}, f_e(t) = 0, \boldsymbol{x}_0 = \begin{bmatrix}1 \ 0\end{bmatrix}^T$.

An ODE in the form of Eq. (1) is asymptotically stable, i.e. it satisfies Eq. (2), if the real part of all eigenvalues of A is strictly negative. Formally,

$$\forall \lambda \in \text{Eig}(A), \ \mathbb{R}\text{e}\{\lambda\} < 0. \tag{3}$$

This condition can be computed easily in most programming languages.

To approximate the solution to the IVP in Example 1, one can use the forward Euler method:

$$\boldsymbol{x}(t+h) \approx \boldsymbol{x}(t) + A\boldsymbol{x}(t)h = (I + Ah)\boldsymbol{x}(t), \text{ with } \boldsymbol{x}(0) = \boldsymbol{x_0}, \tag{4}$$

where I is the identify matrix with the appropriate dimensions, and $h > 0$ is the given simulation step size.

In general, for a given matrix $\tilde{A}$, a system on the form

$$\boldsymbol{x}(t+h) = \tilde{A}\boldsymbol{x}(t), \tag{5}$$

is stable if $\rho(\tilde{A}) < 1$, where $\rho(\tilde{A})$ is the spectral radius [9] of $\tilde{A}$.

3 Stability Analysis of FMI Orchestration Algorithms

Our aim is to encode the co-simulation as a system in the form of Eq. (5). We perform this for a two-simulator system using two orchestration algorithms: the traditional Jacobi method, and the stabilization method used by INTO-CPS. A two simulator system introduced in [10] is illustrated in Fig. 2. More details about this example are given in [6, Sect. 4]. For more examples of stability analysis in co-simulation, refer to [2–5].

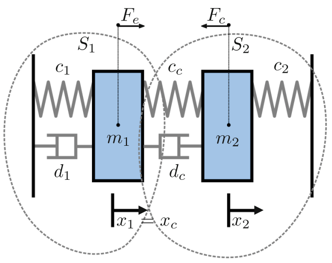


Fig. 2. Double mass-spring-damper with two subsystems: S_1 and S_2.

3.1 Co-simulation Unit Modelling

In the context of co-simulation, time is discretized into a countable set $T = \{t_0, t_1, t_2, \ldots\} \subset \mathbb{R}$, where $t_{i+1} = t_i + H_i$ is the time at step i and H_i is the communication step size at step i, with $i = 0, 1, \ldots$

Simulators exchange outputs only at times $t \in T$.

In the interval $t \in [t_i, t_{i+1}]$, each simulator S_j approximates the solution to a linear ODE,

$$\begin{aligned} \dot{\boldsymbol{x}}_j &= A_j \boldsymbol{x}_j + B_j \boldsymbol{u}_j \\ \boldsymbol{y}_j &= C_j \boldsymbol{x}_j + D_j \boldsymbol{u}_j \end{aligned} \tag{6}$$

where $\boldsymbol{x}_j$ is the state vector, $\boldsymbol{y}_j$ is the output vector, A_j, B_j, C_j, D_j are matrices, the initial state $\boldsymbol{x}_j(t_i)$ is computed in the most recent co-simulation step, and $j = 1, 2$.

Since the simulators only exchange outputs at times $t_i, t_{i+1} \in T$, the input $\boldsymbol{u}_j$ has to be extrapolated in the interval $[t_i, t_{i+1})$. In the simplest co-simulation strategy[1], this extrapolation is often implemented as a zero-order hold: $\tilde{\boldsymbol{u}}_j(t) = \boldsymbol{u}_j(t_i)$, for $t \in [t_i, t_{i+1})$. Then, Eq. (6) can be re-written to represent the unforced system being integrated by each simulator:

$$\begin{bmatrix} \dot{\boldsymbol{x}}_j \\ \dot{\tilde{\boldsymbol{u}}}_j \end{bmatrix} = \begin{bmatrix} A_j & B_j \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_j \\ \tilde{\boldsymbol{u}}_j \end{bmatrix} \tag{7}$$

We can represent the multiple internal integration steps of Eq. (7), performed by the simulator S_j in the interval $t \in [t_i, t_{i+1}]$, as

$$\begin{bmatrix} \tilde{\boldsymbol{x}}_j(t_{i+1}) \\ \tilde{\boldsymbol{u}}_j(t_{i+1}) \end{bmatrix} = \tilde{A}_j^{k_j} \begin{bmatrix} \tilde{\boldsymbol{x}}_j(t_i) \\ \tilde{\boldsymbol{u}}_j \end{bmatrix} \tag{8}$$

where, 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 $0 < h_j \leq H_i$ is the internal fixed step size that divides H_i.

Therefore, each co-simulation unit can be modelled as a discrete time system:

$$\begin{bmatrix} \tilde{\boldsymbol{x}}_j(t_i + H) \\ \tilde{\boldsymbol{u}}_j(t_i + H) \end{bmatrix} = \begin{bmatrix} M_{1,x_j} & M_{1,u_j} \\ M_{2,x_j} & M_{2,u_j} \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{x}}_j(t_i) \\ \boldsymbol{u}_j(t_i) \end{bmatrix} \tag{9}$$

with

$$\tilde{A}_j^{k_j} = \begin{bmatrix} M_{1,x_j} & M_{1,u_j} \\ M_{2,x_j} & M_{2,u_j} \end{bmatrix}.$$

[1] The derivation presented can be applied to more sophisticated input extrapolation techniques, see [1, Eq. (9)].

3.2 FMI Jacobi Algorithm

We assume without loss of generality that the two simulators are coupled in a feedback loop, that is,

$$\boldsymbol{u}_1 = \boldsymbol{y}_2 \text{ and } \boldsymbol{u}_2 = \boldsymbol{y}_1. \tag{10}$$

And, to avoid algebraic loops and keep the exposition short, we assume that either D_1 or D_2 (recall Eq. (6)) is the zero matrix. Let $D_2 = \mathbf{0}$.

The ideal Jacobi coupling would be described by:

$$\begin{aligned}\boldsymbol{u}_1(t) &= \boldsymbol{y}_2(t) = C_2\tilde{\boldsymbol{x}}_2(t)\\ \boldsymbol{u}_2(t) &= \boldsymbol{y}_1(t) = C_1\tilde{\boldsymbol{x}}_1(t) + D_1\boldsymbol{u}_1(t)\end{aligned} \tag{11}$$

However, due the FMI restrictions [11, Restriction 1], the actual coupling is:

$$\begin{aligned}\boldsymbol{u}_1(t_i) &= C_2\tilde{\boldsymbol{x}}_2(t_i)\\ \boldsymbol{u}_2(t_i) &= C_1\tilde{\boldsymbol{x}}_1(t_i) + D_1\tilde{\boldsymbol{u}}_1(t_i).\end{aligned} \tag{12}$$

Applying Eq. (12) to t_{i+1} and using Eq. (9), yields:

$$\begin{aligned}\tilde{\boldsymbol{x}}_1(t_{i+1}) &= M_{1,x_1}\tilde{\boldsymbol{x}}_1(t_i) + M_{1,u_1}C_2\tilde{\boldsymbol{x}}_2(t_i)\\ \tilde{\boldsymbol{u}}_1(t_{i+1}) &= M_{2,x_1}\tilde{\boldsymbol{x}}_1(t_i) + M_{2,u_1}C_2\tilde{\boldsymbol{x}}_2(t_i)\\ \tilde{\boldsymbol{x}}_2(t_{i+1}) &= M_{1,u_2}C_1\tilde{\boldsymbol{x}}_1(t_i) + M_{1,u_2}D_1\tilde{\boldsymbol{u}}_1(t) + M_{1,x_2}\tilde{\boldsymbol{x}}_2(t_i)\\ \tilde{\boldsymbol{u}}_2(t_{i+1}) &= M_{2,u_2}C_1\tilde{\boldsymbol{x}}_1(t_i) + M_{2,u_2}D_1\tilde{\boldsymbol{u}}_1(t) + M_{2,x_2}\tilde{\boldsymbol{x}}_2(t_i)\end{aligned} \tag{13}$$

which can be arranged to the form of Eq. (5):

$$\begin{bmatrix}\tilde{\boldsymbol{x}}_1(t_{i+1})\\ \tilde{\boldsymbol{u}}_1(t_{i+1})\\ \tilde{\boldsymbol{x}}_2(t_{i+1})\\ \tilde{\boldsymbol{u}}_2(t_{i+1})\end{bmatrix} = \begin{bmatrix} M_{1,x_1} & 0 & M_{1,u_1}C_2 & 0\\ M_{2,x_1} & 0 & M_{2,u_1}C_2 & 0\\ M_{1,u_2}C_1 & M_{1,u_2}D_1 & M_{1,x_2} & 0\\ M_{2,u_2}C_1 & M_{2,u_2}D_1 & M_{2,x_2} & 0\end{bmatrix} \begin{bmatrix}\tilde{\boldsymbol{x}}_1(t_i)\\ \tilde{\boldsymbol{u}}_1(t_i)\\ \tilde{\boldsymbol{x}}_2(t_i)\\ \tilde{\boldsymbol{u}}_2(t_i)\end{bmatrix} \tag{14}$$

3.3 INTO-CPS Method

The method used in INTO-CPS is a sucessive substitution fixed point iteration, described by:

$$\begin{aligned}\boldsymbol{u}_1(t_{i+1}) &= C_2\tilde{\boldsymbol{x}}_2(t_{i+1})\\ \boldsymbol{u}_2(t_{i+1}) &= C_1\tilde{\boldsymbol{x}}_1(t_{i+1}) + D_1\boldsymbol{u}_1(t_{i+1})\end{aligned} \tag{15}$$

The above equation can be expanded and simplified to:

$$\begin{aligned}\tilde{\boldsymbol{x}}_1(t_{i+1}) &= M_{1,x_1}\tilde{\boldsymbol{x}}_1(t_i) + M_{1,u_1}C_2\tilde{\boldsymbol{x}}_2(t_{i+1})\\ \boldsymbol{u}_1(t_{i+1}) &= M_{2,x_1}\tilde{\boldsymbol{x}}_1(t_i) + M_{2,u_1}C_2\tilde{\boldsymbol{x}}_2(t_{i+1})\\ \tilde{\boldsymbol{x}}_2(t_{i+1}) &= M_{1,x_2}\tilde{\boldsymbol{x}}_2(t_i) + M_{1,u_2}C_1\tilde{\boldsymbol{x}}_1(t_{i+1}) + M_{1,u_2}D_1\boldsymbol{u}_1(t_{i+1})\\ \boldsymbol{u}_2(t_{i+1}) &= M_{2,x_2}\tilde{\boldsymbol{x}}_2(t_i) + M_{2,u_2}C_1\tilde{\boldsymbol{x}}_1(t_{i+1}) + M_{2,u_2}D_1\boldsymbol{u}_1(t_{i+1})\end{aligned} \tag{16}$$

which can be put in matrix form:

$$\begin{bmatrix} \tilde{\boldsymbol{x}}_1(t_{i+1}) \\ \boldsymbol{u}_1(t_{i+1}) \\ \tilde{\boldsymbol{x}}_2(t_{i+1}) \\ \boldsymbol{u}_2(t_{i+1}) \end{bmatrix} = \begin{bmatrix} M_{1,x_1} & 0 & 0 & 0 \\ M_{2,x_1} & 0 & 0 & 0 \\ 0 & 0 & M_{1,x_2} & 0 \\ 0 & 0 & M_{2,x_2} & 0 \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{x}}_1(t_i) \\ \boldsymbol{u}_1(t_i) \\ \tilde{\boldsymbol{x}}_2(t_i) \\ \boldsymbol{u}_2(t_i) \end{bmatrix} + \begin{bmatrix} 0 & 0 & M_{1,u_1}C_2 & 0 \\ 0 & 0 & M_{2,u_1}C_2 & 0 \\ M_{1,u_2}C_1 & M_{1,u_2}D_1 & 0 & 0 \\ M_{2,u_2}C_1 & M_{2,u_2}D_1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{x}}_1(t_{i+1}) \\ \boldsymbol{u}_1(t_{i+1}) \\ \tilde{\boldsymbol{x}}_2(t_{i+1}) \\ \boldsymbol{u}_2(t_{i+1}) \end{bmatrix} \tag{17}$$

Renaming the above equation to $\bar{\boldsymbol{x}}_{i+1} = \bar{M}_i \bar{\boldsymbol{x}}_i + \bar{M}_{i+1} \bar{\boldsymbol{x}}_{i+1}$, we get an equation in the form of Eq. (5):

$$\bar{\boldsymbol{x}}_{i+1} = (I - \bar{M}_{i+1})^{-1} \bar{M}_i \bar{\boldsymbol{x}}_i \tag{18}$$

In most cases in practice, $\rho((I - \bar{M}_{i+1})^{-1} \bar{M}_i)$ is smaller than the spectral radius of the matrix in Eq. (14). The practical results of this analysis are shown in the case study described in [10].

This can be generalized. However, in practice, one must be aware of the internal details of each co-simulation unit, which is usually difficult. As such, this analysis can be used to determine the best orchestration algorithm, without providing guarantees.

References

1. Busch, M.: Continuous approximation techniques for co-simulation methods: analysis of numerical stability and local error. J. Appl. Math. Mech. **96**(9), 1061–1081 (2016)
2. Gomes, C., Jungers, R., Legat, B., Vangheluwe, H.: Minimally constrained stable switched systems and application to co-simulation. Technical report. arXiv:1809.02648 (2018), http://arxiv.org/abs/1809.02648
3. Gomes, C., Legat, B., Jungers, R., Vangheluwe, H.: Minimally constrained stable switched systems and application to co-simulation. In: IEEE Conference on Decision and Control, Miami Beach, FL, USA (2018). To be published
4. Gomes, C., Legat, B., Jungers, R.M., Vangheluwe, H.: Stable adaptive co-simulation: a switched systems approach. In: IUTAM Symposium on Co-Simulation and Solver Coupling, Darmstadt, Germany (2017). To appear
5. Gomes, C., Thule, C., Broman, D., Larsen, P.G., Vangheluwe, H.: Co-simulation: state of the art. Technical report, February 2017. http://arxiv.org/abs/1702.00686
6. Gomes, C., Thule, C., Broman, D., Larsen, P.G., Vangheluwe, H.: Co-simulation: a survey. ACM Comput. Surv. **51**(3) (2018). Article 49
7. Gomes, C., Thule, C., Larsen, P.G., Denil, J., Vangheluwe, H.: Co-simulation of continuous systems: a tutorial. arXiv:1809.08463 [cs, math], September 2018. http://arxiv.org/abs/1809.08463
8. Larsen, P.G., Fitzgerald, J., Woodcock, J., Gamble, C., Payne, R., Pierce, K.: Features of integrated model-based co-modelling and co-simulation technology. In: Cerone, A., Roveri, M. (eds.) SEFM 2017. LNCS, vol. 10729, pp. 377–390. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-74781-1_26

9. Strang, G.: Introduction to Linear Algebra, vol. vol, p. 3. Wellesley-Cambridge Press, Wellesley (1993)
10. Thule, C.: Mass-spring-damper Case Study (2018). https://github.com/INTO-CPS-Association/example-mass_spring_damper
11. Thule, C., Gomes, C., Deantoni, J., Larsen, P.G., Brauer, J., Vangheluwe, H.: Towards verification of hybrid co-simulation algorithms. In: 2nd Workshop on Formal Co-Simulation of Cyber-Physical Systems, Toulouse, France. Springer, Cham (2018). To be published
12. Thule, C., Lausdahl, K., Larsen, P.G., Meisl, G.: Maestro: The INTO-CPSCo-simulation orchestration engine (2018). Submitted to Simulation Modelling Practice and Theory