Demo: Stabilization Technique in INTO-CPS

Cláudio Gomes24, Casper Thule1, Kenneth Lausdahl1, Peter Gorm Larsen1, and Hans Vangheluwe23

1 Aarhus University, Denmark  
{casper.thule,pgl}@eng.au.dk  
2 University of Antwerp, Belgium  
{claudio.gomes,hans.vangheluwe}@uantwerp.be  
3 McGill University, Canada  
4 Flanders Make, Belgium

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 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]).

In the next section, we describe the principles of stability analysis for linear Ordinary Differential Equations (ODEs), and linear discrete time systems. Then, in Section 3, we apply these principles to analyze 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.

* 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).
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 $x$, we denote its transpose as $x^T$. Furthermore, we denote the $i$-th element of vector $x$ by $x_i$, so that $x = [x_1 \ x_2 \ \cdots \ x_n]^T$. Similarly, $F(x)$ denotes the $i$-th element of the vector returned by $F(x)$.

A linear ODE has the following form:

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

(1)

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

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

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

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 Equation (1) by introducing a new variable for velocity, $v = \dot{x}$, and letting the vector $x = [x \ v]^T$. Given an initial position $x_0$ and velocity $v_0$, we obtain the following:

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

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 Equation (1) is asymptotically stable when all its solutions tend to zero as time passes, regardless of the initial value specified. Formally,

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

(2)

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

$$\forall \lambda \in \text{Eig}(A), \ \text{Re}\{\lambda\} < 0.$$

(3)

This condition can be computed easily in most programming languages.
Fig. 1: Position (and its approximations) over time of the mass-spring-damper system. Parameters are: $h = 0.1, m = c = 1, d = 10^{-4}, f_e(t) = 0, x_0 = [1 \ 0]^T$.

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

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

where $I$ is the identity 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

$$x(t + h) = \tilde{A}x(t), \quad \text{(5)}$$

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 Equation (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 Figure 2. More details about this example are given in [6, Section 4.1]. For more examples of stability analysis in co-simulation, refer to [5,4,3,2].

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,

$$\begin{align*}
\dot{x}_j &= A_j x_j + B_j u_j \\
y_j &= C_j x_j + D_j u_j
\end{align*}$$

where $x_j$ is the state vector, $y_j$ is the output vector, $A_j, B_j, C_j, D_j$ are matrices, the initial state $x_j(t_i)$ 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 $u_j$ has to be extrapolated in the interval $[t_i, t_{i+1})$. In the simplest co-simulation strategy, this extrapolation is often implemented as a zero-order hold: $\tilde{u}_j(t) = u_j(t_i)$, for $t \in [t_i, t_{i+1})$. Then, Equation (6) can be re-written to represent the unforced system being integrated by each simulator:

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

(7)

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

$$\begin{bmatrix}
\tilde{x}_j(t_{i+1}) \\
\tilde{u}_j(t_{i+1})
\end{bmatrix} = z_{kj} \begin{bmatrix}
x_j(t_i) \\
\tilde{u}_j(t_i)
\end{bmatrix}$$

(8)

where, e.g., $\tilde{A}_j = I + h_j \begin{bmatrix}
A_j & B_j \\
0 & 0
\end{bmatrix}$ 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:

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

(9)

with

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

### 3.2 FMI Jacobi Algorithm

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

$$u_1 = y_2$$

and

$$u_2 = y_1.$$
The ideal Jacobi coupling would be described by:

\[
\begin{align*}
\mathbf{u}_1(t) &= \mathbf{y}_2(t) = C_2 \tilde{\mathbf{x}}_2(t) \\
\mathbf{u}_2(t) &= \mathbf{y}_1(t) = C_1 \tilde{\mathbf{x}}_1(t) + D_1 \mathbf{u}_1(t)
\end{align*}
\]

(11)

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

\[
\begin{align*}
\mathbf{u}_1(t_i) &= C_2 \tilde{\mathbf{x}}_2(t_i) \\
\mathbf{u}_2(t_i) &= C_1 \tilde{\mathbf{x}}_1(t_i) + D_1 \tilde{\mathbf{u}}_1(t_i).
\end{align*}
\]

(12)

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

\[
\begin{align*}
\tilde{\mathbf{x}}_1(t_{i+1}) &= M_{1,x_1} \tilde{\mathbf{x}}_1(t_i) + M_{1,u_1} C_2 \tilde{\mathbf{x}}_2(t_i) \\
\tilde{\mathbf{u}}_1(t_{i+1}) &= M_{2,x_1} \tilde{\mathbf{x}}_1(t_i) + M_{2,u_1} C_2 \tilde{\mathbf{x}}_2(t_i) \\
\tilde{\mathbf{x}}_2(t_{i+1}) &= M_{1,x_2} C_1 \tilde{\mathbf{x}}_1(t_i) + M_{1,u_2} D_1 \tilde{\mathbf{u}}_1(t) + M_{1,x_2} \tilde{\mathbf{x}}_2(t_i) \\
\tilde{\mathbf{u}}_2(t_{i+1}) &= M_{2,x_2} C_1 \tilde{\mathbf{x}}_1(t_i) + M_{2,u_2} D_1 \tilde{\mathbf{u}}_1(t) + M_{2,x_2} \tilde{\mathbf{u}}_2(t_i)
\end{align*}
\]

(13)

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

\[
\begin{bmatrix}
\mathbf{x}_1(t_{i+1}) \\
\mathbf{u}_1(t_{i+1}) \\
\mathbf{x}_2(t_{i+1}) \\
\mathbf{u}_2(t_{i+1})
\end{bmatrix}
= 
\begin{bmatrix}
M_{1,x_1} & 0 & M_{1,u_1} C_2 & 0 \\
M_{2,x_1} & 0 & M_{2,u_1} C_2 & 0 \\
M_{1,x_2} C_1 & M_{1,u_2} D_1 & M_{1,x_2} & 0 \\
M_{2,x_2} C_1 & M_{2,u_2} D_1 & M_{2,x_2} & 0
\end{bmatrix}
\begin{bmatrix}
\mathbf{x}_1(t_i) \\
\mathbf{u}_1(t_i) \\
\mathbf{x}_2(t_i) \\
\mathbf{u}_2(t_i)
\end{bmatrix}
\]

(14)

3.3 INTO-CPS Method

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

\[
\begin{align*}
\mathbf{u}_1(t_{i+1}) &= C_2 \tilde{\mathbf{x}}_2(t_{i+1}) \\
\mathbf{u}_2(t_{i+1}) &= C_1 \tilde{\mathbf{x}}_1(t_{i+1}) + D_1 \mathbf{u}_1(t_{i+1})
\end{align*}
\]

(15)

The above equation can be expanded and simplified to:

\[
\begin{align*}
\tilde{\mathbf{x}}_1(t_{i+1}) &= M_{1,x_1} \tilde{\mathbf{x}}_1(t_i) + M_{1,u_1} C_2 \tilde{\mathbf{x}}_2(t_{i+1}) \\
\mathbf{u}_1(t_{i+1}) &= M_{2,x_1} \tilde{\mathbf{x}}_1(t_i) + M_{2,u_1} C_2 \tilde{\mathbf{x}}_2(t_{i+1}) \\
\tilde{\mathbf{x}}_2(t_{i+1}) &= M_{1,x_2} \tilde{\mathbf{x}}_2(t_i) + M_{1,u_2} C_1 \tilde{\mathbf{x}}_1(t_{i+1}) + M_{1,u_2} D_1 \mathbf{u}_1(t_{i+1}) \\
\mathbf{u}_2(t_{i+1}) &= M_{2,x_2} \tilde{\mathbf{x}}_2(t_i) + M_{2,u_2} C_1 \tilde{\mathbf{x}}_1(t_{i+1}) + M_{2,u_2} D_1 \mathbf{u}_1(t_{i+1})
\end{align*}
\]

(16)

which can be put in matrix form:

\[
\begin{bmatrix}
\mathbf{x}_1(t_{i+1}) \\
\mathbf{u}_1(t_{i+1}) \\
\mathbf{x}_2(t_{i+1}) \\
\mathbf{u}_2(t_{i+1})
\end{bmatrix}
= 
\begin{bmatrix}
M_{1,x_1} & 0 & 0 & 0 \\
M_{2,x_1} & 0 & 0 & 0 \\
0 & M_{1,x_2} & 0 & 0 \\
0 & 0 & M_{2,x_2} & 0
\end{bmatrix}
\begin{bmatrix}
\mathbf{x}_1(t_i) \\
\mathbf{u}_1(t_i) \\
\mathbf{x}_2(t_i) \\
\mathbf{u}_2(t_i)
\end{bmatrix}
+ 
\begin{bmatrix}
0 & 0 & M_{1,u_1} C_2 & 0 \\
0 & 0 & M_{2,u_1} C_2 & 0 \\
M_{1,u_2} C_1 & M_{1,u_2} D_1 & 0 & 0 \\
M_{2,u_2} C_1 & M_{2,u_2} D_1 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
\mathbf{x}_1(t_{i+1}) \\
\mathbf{u}_1(t_{i+1}) \\
\mathbf{x}_2(t_{i+1}) \\
\mathbf{u}_2(t_{i+1})
\end{bmatrix}
\]

(17)
Renaming the above equation to $\tilde{x}_{i+1} = \tilde{M}_i \tilde{x}_i + \tilde{M}_{i+1} \tilde{x}_{i+1}$, we get an equation in the form of Equation (5):

$$\tilde{x}_{i+1} = (I - \tilde{M}_{i+1})^{-1} \tilde{M}_i \tilde{x}_i$$  \hspace{1cm} (18)

In most cases in practice, $\rho((I - \tilde{M}_{i+1})^{-1} \tilde{M}_i)$ is smaller than the spectral radius of the matrix in Equation (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