Relative Consistency and Robust Stability Measures for Sequential Co-simulation

Slaven Glumac 1 Zdenko Kovačić 2

1 AVL-AST d.o.o., Croatia, slaven.glumac@avl.com
2 University of Zagreb Faculty of Electrical Engineering and Computing, Croatia, zdenko.kovacic@fer.hr

Abstract

The paper introduces a matrix co-simulation model of linear time invariant differential equations using general linear methods. This model is used to develop a calculation of relative consistency measure based on worst case defect calculation. It is shown how a robust stability measurement based on spectral radius can be used to measure robustness to slave parameter changes. Both consistency and stability measurements are calculated based on the linear model, and can be calculated prior to the co-simulation run. Finally, multi-objective optimization has been proposed to utilize introduced measurements for tuning the co-simulation master. 

Keywords: co-simulation, master, robust stability, relative consistency, multi-objective optimization

1 Introduction

Co-simulation is a multi-method coupling of a coupled system, also known as simulator coupling (Kübler and Schiehlen, 2000). A co-simulation run can be quite difficult to set up in practice since models of the system are usually black boxes. When no previous information is available about co-simulation slaves, the procedure for the choice of a co-simulation master usually boils down to repeated trial and error. This is usually due to the fact that the system is more than the sum of its parts. The reason to co-simulate multiple simulators is to get the notion of the coupled system behavior. However, without any information about the behavior it is difficult to set up a co-simulation master.

In this paper co-simulation slaves are not completely black boxes. It is presumed that Jacobian matrices of slaves are available. (Åkesson et al., 2012). FMI 2.0 standard (FMI 2.0) defines an optional interface to Jacobian matrices of a slave. A linearized model is used to make the prediction of quality for a coupled system co-simulation. A goal of this paper is to introduce quality measures independent of internal states of the slaves.

Local error control is analyzed and shown to be a feasible method for bounding the global co-simulation error (Arnold et al., 2014). However, methods for local error estimation are usually expensive, e.g. Richardson extrapolation requires three times more co-simulation steps executed. There are predictor/corrector methods (Busch, 2012; Schweizer and Lu, 2015) which allow for run-time local error estimation as a side-effect. Defect control (Enright, 2000) presents an alternative to local error control. The defect control has been used to control the error of differential algebraic equations which makes it an ideal candidate for the use in co-simulation environments. This paper expand the idea of defect control by introducing a worst case defect calculation in order to enable a relative consistency estimate prior to the co-simulation run.

Stability is an important measure of system quality which does not depend on the initial states. Zero-stability (Kübler and Schiehlen, 2000) is an important requirement for a co-simulation. However, a co-simulation cannot be more or less zero-stable, it can only be zero-stable or not. This leads to a search for a relative stability measure. In co-simulation there has been experimental work on determining stability regions for different co-simulation solvers (Busch, 2012; Schweizer et al., 2015). The authors have compared co-simulation masters based on the size of a plotted stability region. This paper tries to propose the use of stability radius estimate (Hinrichsen and Pritchard, 2005) in order to formalize this approach. Intuitively, robust stability is a particularly important property of a co-simulation. In practice, rapid prototyping is one of the main reasons for the use of co-simulation. During rapid prototyping, parameters or some parts of a single slave are expected to change. Robustness to such changes would mean that a user of a co-simulation does not have to adapt the master.

With quality measures defined the optimization becomes a feasible method for the choice of a co-simulation master. The optimization in co-simulation has been introduced as a means to improve the parameters of the simulated system in order to get a better signal response (Gedda et al., 2012). This paper introduces a problem of improvement of a co-simulation master as a multi-objective optimization problem (Kalyannoy, 2001).

The next section introduces a co-simulation model used in this paper and underlying assumptions about co-simulation slaves. A presented matrix model of a co-simulation step enables the calculation of quality measures introduced in the following sections. The third section shows the example of modeling a two-mass oscillator. This example is used for the verification of quality mea-
2 Co-simulation Model

2.1 Co-simulation Slave

A co-simulation slave according to the FMI 2.0 standard is a tuple:

\[ G_i = (V_i, U_i, Y_i, D_i, v_i, set_i, get_i, doStep_i) \]  

(1)

where \( V_i \) is a set of internal states of \( G_i \), \( U_i \) is a set of input variables, \( Y_i \) is a set of output variables, \( D_i \subset U_i \times Y_i \) is a set of output-input dependencies, \( v_i \) is an initial state of the FMU, \( set_i \) is a function which sets the value to an input variable, \( get_i \) is a function which returns the value of an output variable, and 

\[ doStep_i : (V_i, \mathbb{R}^+) \rightarrow V_i \]  

(2)

is a function which implements a simulation step, i.e. it integrates the internal state\(^2\). Let \( v \) be the internal state, and \( t \) the local time of the slave. The function:

\[ v' = doStep_i(v, h) \]  

(3)

will change the internal state to \( v' \), and advance the local time of the slave to \( t + h \).

In this paper it is assumed that the \( i \)th co-simulation slave solves the following system of equations:

\[ x_i(t) = A_i x_i(t) + B_i u_i(t) \]  

(4a)

\[ y_i(t) = C_i x_i(t) + D_i u_i(t) \]  

(4b)

where \( x_i : \mathbb{R}^+ \rightarrow \mathbb{R}^{|X_i|} \) is the state signal\(^3\), \( u_i : \mathbb{R}^+ \rightarrow \mathbb{R}^{|U_i|} \) is the input signal, and \( y_i : \mathbb{R}^+ \rightarrow \mathbb{R}^{|Y_i|} \) is the output signal. A nonlinear differential equation:

\[ \dot{x}_i(t) = f_i(x_i(t), u_i(t)) \]  

\[ y_i(t) = g_i(x_i(t), u_i(t)) \]  

(5)

can be transformed to (4) by linearization:

\[ A_i = \frac{\partial f_i}{\partial x_i}, \quad B_i = \frac{\partial f_i}{\partial u_i}, \quad C_i = \frac{\partial g_i}{\partial x_i}, \quad D_i = \frac{\partial g_i}{\partial u_i} \]  

(6)

During co-simulation each slave performs the integration of its states (4a) in a \( doStep_i \) function (2):

\[ x_i[k] = x_i(t_k) = x_i(t_{k-1} + h) \]

\[ = x_i(t_{k-1}) + \int_{t_{k-1}}^{t_k} A_i x_i(\tau) + B_i u_i(\tau) d\tau \]  

\[ = x_i(t_{k-1}) + \int_{t_{k-1}}^{t_k} A_i x_i(\tau) + B_i u_i[k] d\tau \]  

(7)

where \( h \) is the communication step size. This paper limits the analysis to zero-order hold as seen in the above equation, i.e. the value of input signal is assumed to be constant throughout a slave integration:

\[ \hat{u}(\tau) = u[k], \quad t_{k-1} < \tau \leq t_k \]  

(8)

In this paper it is assumed that a linear model (4a) is integrated by a linear method:

\[ \dot{v}_i' = G^i v_i + G^i u_i \]  

\[ \dot{x}_i' = G^i v_i \]  

(9)

where \( v_i : \mathbb{R}^{|V_i|} \).

Integration of a linear system (4a) with any general linear methods can be formulated in the above manner. The derivation of matrices (9) can be done analogous to the derivation of the absolute stability matrix (Jackiewicz, 2009).

The function \( get_i \) should return the output values consistent to (4b):

\[ y_i[k] = y_i(t_k) \]  

(11)

In this paper the \( k \)th signal sample refers to a last signal value sampled in the iteration of a co-simulation master. This is described in more detail in next sections.

2.2 Co-simulation Network

A network of co-simulation slaves is a tuple:

\[ N = (G, L) \]  

(12)

where \( G = \{ G_1, G_2, \ldots, G_n \} \) is a set of \( n \) co-simulation slaves, \( U = U_1 \cup U_2 \cup \cdots \cup U_n \) is a set of input variables of all co-simulation slaves, and \( Y = Y_1 \cup Y_2 \cup \cdots \cup Y_n \) is a set of output variables of all co-simulation slaves, \( L \in \mathbb{R}^{[P] \times [P]} \) is a matrix representing output-input connections.

Let \( u_i(t), y_i(t), x_i(t) \) denote the column stacked values of all co-simulation slaves, respectively:

\[ u_i(t) = \begin{bmatrix} u_{1i}(t) \\ u_{2i}(t) \\ \vdots \\ u_{ni}(t) \end{bmatrix}, \quad y_i(t) = \begin{bmatrix} y_{1i}(t) \\ y_{2i}(t) \\ \vdots \\ y_{ni}(t) \end{bmatrix}, \quad x_i(t) = \begin{bmatrix} x_{1i}(t) \\ x_{2i}(t) \\ \vdots \\ x_{ni}(t) \end{bmatrix} \]  

(13)

Output-input connections are denoted as matrix \( L \):

\[ u(t) = Ly(t) \]  

(14)

\[ \tau \]  

2 The slave accepts any step size \( h \), i.e. does not do a step rejection in case of an event.

3 \( x_i \) is a set of internal state variables of the continuous system.
The extrapolation element can have access to all the techniques (Benedikt et al., 2013; Stettinger et al., 2014). This allows for the modeling of more advanced extrapolation techniques during the integration of co-simulation.

Let system matrices of the co-simulation slaves (4) form a block diagonal matrix:

\[
\begin{align*}
A &= \text{blockDiag}(A_1, A_2, \ldots, A_n) \\
B &= \text{blockDiag}(B_1, B_2, \ldots, B_n) \\
C &= \text{blockDiag}(C_1, C_2, \ldots, C_n) \\
D &= \text{blockDiag}(D_1, D_2, \ldots, D_n)
\end{align*}
\]

The coupled system can be described by the following equations:

\[
\begin{align*}
\dot{x}(t) &= (A + BL(I - DL)^{-1}C)x(t) \quad (16a) \\
y(i) &= (I - DL)^{-1}Cx(t) \quad (16b)
\end{align*}
\]

An implicit assumption is that there are no algebraic loops in the system, i.e. \(\text{det}(I - DL) \neq 0\). This system is described to give a reference for the exact solution of the co-simulation.

### 2.3 Co-simulation Master

A co-simulation master is an algorithm that executes a co-simulation of the network (12). The master presented in this paper (Algorithm 1) executes a non-iterative sequential co-simulation. It is determined by the communication step size \(h\), extrapolation elements, and the calling sequence of co-simulation slaves.

A calling sequence of co-simulation slaves determines the execution order of co-simulation slaves. Let the function \(\sigma : I \rightarrow I\) define the calling sequence where \(I = \{1, 2, \ldots, n\}\) is the set of slave indices. The calling sequence is defined by the following expression:

\[
i = \sigma(i)
\]

which states that the co-simulation slave \(G_i\) is executed \(r\)th in the calling sequence.

The extrapolation is used to determine the values of continuous signal between4 and beyond known communication points. This paper assumes the zero-order hold approximation of inputs during integration of co-simulation slave (8). The extrapolation elements of the co-simulation slave \(G_i\) are determined by the following equations:

\[
\begin{align*}
w_i' &= F_i^{w}w_i + F_i^{wy}y \\
u_i' &= F_i^{w}w_i
\end{align*}
\]

The extrapolation element can have access to all the outputs of all the slaves in the co-simulation network5. This allows for modeling of more advanced extrapolation techniques (Benedikt et al., 2013; Stettinger et al., 2014).

The assignment of time to a discrete signal value is done at the end of a co-simulation iteration. Let \(\bar{u}(t)\) and \(\bar{y}(t)\) denote the signals reconstructed from discrete samples provided by the co-simulation run.

In this paper the following equality holds:

\[
\bar{u}(kh) = u[k], \quad \bar{y}(kh) = y[k] \quad (19)
\]

This definition is important for consistency measurements introduced in the next sections.

Let the stacked values of the co-simulation system vectors be denoted as:

\[
z = [x^T \ v^T \ y^T \ w^T \ u_i^T]^T \quad (20)
\]

This vector allows to model the co-simulation with a known model of linear time invariant co-simulation slaves (4), linear method of integration (9), and linear extrapolation methods (18). It allows to reformulate the equations of a co-simulation of black boxes (Algorithm 1) to a sequential multi-method integration of known models (Algorithm 2).

The integration of each co-simulation slave can be reformulated to:

\[
z' = G_i z \quad (21)
\]

where equation (9) should be satisfied:

\[
\begin{align*}
x_i' &= \left\{ \begin{array}{ll}
G_i^{v}v_i \nu_i + G_i^{v}G_i^{w}u_i, & j = i \\
x_j, & j \neq i
\end{array} \right.
\end{align*}
\]

\[
\begin{align*}
v_j' &= \left\{ \begin{array}{ll}
G_i^{v}v_i + G_i^{w}u_i, & j = i \\
v_j, & j \neq i
\end{array} \right.
\end{align*}
\]

\[
y_j' = y_j, \quad w_j' = w_j, \quad u_j' = u_j
\]

The output update of a co-simulation slave can be reformulated to:

\[
z' = H_i z \quad (23)
\]

where (10) should be satisfied:

\[
\begin{align*}
x_j' &= \left\{ \begin{array}{ll}
x_i + D_i u_i, & j = i \\
x_j, & j \neq i
\end{array} \right.
\end{align*}
\]

\[
y_j' = y_j, \quad w_j' = w_j, \quad u_j' = u_j
\]

The extrapolation elements can be reformulated to:

\[
z' = F_i z \quad (25)
\]

where (18) should be satisfied:

\[
\begin{align*}
x_j' &= x_j, \quad v_j' = v_j, \quad y_j' = y_j \\
w_j' &= \left\{ \begin{array}{ll}
F_i^{w}w_i + F_i^{wy}y, & j = i \\
w_j, & j \neq i
\end{array} \right.
\end{align*}
\]

\[
\begin{align*}
u_j' &= \left\{ \begin{array}{ll}
F_i^{w}F_i^{ww}w_i + F_i^{ww}F_i^{wy}y, & j = i \\
u_j, & j \neq i
\end{array} \right.
\end{align*}
\]

Such kind of reformulation allows for calculation of Algorithm 2 with the use of matrix operations.
Algorithm 1 Sequential co-simulation

Require: \(N, \sigma, h, t_0, t_{\text{end}}\)  
\[ k := 0 \quad \triangleright \text{Initialization phase} \]
\[ z := R_v v_0 \]
while \( \|\Delta z\| > \varepsilon \) do
  for \( r := 1 \) to \( n \) do
    \( i := \sigma(r) \)
    \( w_i := F^{(w)} w_i + F^{(y)} y \)  \( \triangleright \text{Extrapolation} \)
    \( u_i := F^{(w)} w_i \)
    for \( u_i \in U_i \) do
      \( v_i := \text{set}(v_i, u_i, u_i) \)
    end
    if \( y_i \in Y_i \) then
      \( y_{[y]} := \text{get}_j(s_j, y_j) \)  \( \triangleright \text{Read outputs} \)
    end
  end
  \[ z[0] := z \]
while \( t_0 + kh < t_{\text{end}} \) do
  for \( r := 1 \) to \( n \) do
    \( i := \sigma(r) \)
    \( w_i := F^{(w)} w_i \)
    for \( u_i \in U_i \) do
      \( v_i := \text{set}(v_i, u_i, u_i) \)
    end
    \( s_i := \text{doStep}_i(s_i, h) \)  \( \triangleright \text{Update internal state} \)
    if \( y_i \in Y_i \) then
      \( y_{[y]} := \text{get}_j(s_j, y_j) \)  \( \triangleright \text{Read outputs} \)
    end
  end
  \[ k := k + 1 \]
  \[ z[k] := z \]  \( \triangleright \text{Signal assignment} \)
end

of the computation phase of the algorithm can be reformulated as:
\[ z[k] = \Phi z[k-1] \]  \( (27) \)
The computation matrix \( \Phi \) can be calculated by tracing the steps of calculation phase of the algorithm:
\[ \Phi = \prod_{r=1}^{n} \Phi_{\sigma(r)} = \Phi_{\sigma(n)} \Phi_{\sigma(n-1)} \cdots \Phi_{\sigma(1)} \]  \( (28) \)
where \( \Phi_i \) represents the update of the overall co-simulation slave done by the co-simulation slave \( G_i \):
\[ \Phi_i = H_i G_i F_i \]  \( (29) \)

In this paper the initialization is done with the help of a fixed-point iteration. Initialization matrix \( \Phi_0 \) is used to model an iteration of the initialization algorithm:
\[ \Phi_0 = \Phi_{G_i=1, i \in I} \]  \( (30) \)
The initialization matrix is equal to computation matrix (27) with excluded solver updates\(^6\) (function \( \text{doStep}_i \)), i.e. integration matrices \( G_i \) equal to identity matrix. An iteration of the initialization phase of the algorithm can be reformulated as:
\[ z' = \Phi_0 z \]  \( (31) \)
\(^6\) A closer look at Algorithm 1 shows that the initialization and the computation phase are almost identical. The function call of \( \text{doStep}_i \), i.e. the integration is missing in the initialization phase.

Algorithm 2 Sequential multi-method integration

Require: \(N, \sigma, h, t_0, t_{\text{end}}\)  
\[ k := 0 \quad \triangleright \text{Initialization phase} \]
\[ z := R_v v_0 \]
while \( \|\Delta z\| > \varepsilon \) do
  for \( r := 1 \) to \( n \) do
    \( i := \sigma(r) \)
    \( z := F_i z \)  \( \triangleright \text{Extrapolation} \)
    \( z := H_i z \)  \( \triangleright \text{Read outputs} \)
  end
  \[ z[0] := z \]
while \( t_0 + kh < t_{\text{end}} \) do
  for \( r := 1 \) to \( n \) do
    \( i := \sigma(r) \)
    \( z := F_i z \)  \( \triangleright \text{Extrapolation} \)
    \( z := G_i z \)  \( \triangleright \text{Update internal state} \)
    \( z := H_i z \)  \( \triangleright \text{Read outputs} \)
    \( k := k + 1 \)
  end
  \[ z[k] := z \]  \( \triangleright \text{Signal assignment} \)
end

The initialization phase is started with a value assignment:
\[ z = R_v v_0 \]  \( (32) \)
where:
\[ v = v_0 \]
\[ x_i = G_i^{(w)} v_i \]  \( (33) \)
need to be assigned to respective positions in the co-simulation value vector \( z \). The initialization phase of the algorithm can be modeled with:
\[ z[0] = \Phi^w_0 R_v v_0 \]  \( (34) \)
where:
\[ \Phi^w_0 = \lim_{m \to \infty} \Phi^m_0 \]  \( (35) \)
The conditions for the initialization phase to converge to the above limit are stated in the next sections.

3 Test System
3.1 Two-mass Oscillator

The example system is a two mass oscillator (Figure 1) The system consists of two co-simulation slaves, \( G_1 \) and \( G_2 \). Input variables of slaves \( G_1 \) and \( G_2 \) are:
\[ U_1 = \{ G_1.x, G_1.y \}, \quad U_2 = \{ G_2.F \} \]  \( (36) \)
respectively. Output variables of slaves \( G_1 \) and \( G_2 \) are:
\[ Y_1 = \{ G_1.F \}, \quad Y_2 = \{ G_2.x, G_2.y \} \]  \( (37) \)
respectively. The connection matrix is equal to:
\[ L = \begin{bmatrix} g_{1,F} & g_{1,x} & g_{1,y} \\ g_{2,x} & g_{2,F} & g_{2,y} \end{bmatrix} \]  \( (38) \)
where $i$ is an extrapolation element which tries to control the coupling. Slave $G_1$ provides force as an output, while slave $G_2$ provides the displacement and velocity of mass $m_2$.

Slave $G_1$ solves the following equations:

$$
\begin{align*}
A_1 &= \begin{bmatrix}
\frac{c_1}{m_1} & \frac{1}{m_1} \\
\frac{d_1}{m_1} & \frac{1}{m_1}
\end{bmatrix}, \\
B_1 &= \begin{bmatrix}
0 & 0 \\
0 & \frac{c_k}{m_k}
\end{bmatrix}, \\
C_1 &= \begin{bmatrix}
c_k & d_k
\end{bmatrix}, \\
D_1 &= \begin{bmatrix}
-c_k & -d_k
\end{bmatrix}
\end{align*}
$$

(39)

Slave $G_2$ solves the following equations:

$$
\begin{align*}
A_2 &= \begin{bmatrix}
\frac{c_2}{m_2} & \frac{1}{m_2} \\
\frac{d_2}{m_2} & \frac{1}{m_2}
\end{bmatrix}, \\
B_2 &= \begin{bmatrix}
0 \\
\frac{1}{m_2}
\end{bmatrix}, \\
C_2 &= \begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}, \\
D_2 &= \begin{bmatrix}
0 \\
0
\end{bmatrix}
\end{align*}
$$

(40)

Unless stated otherwise, the following system parameters are used in experiments throughout this paper:

$$
\begin{align*}
m_1 &= 10, & c_1 &= 1, & d_1 &= 1, & c_k &= 1, & d_k &= 2 \\
m_2 &= 10, & c_2 &= 1, & d_2 &= 2
\end{align*}
$$

(41)

The co-simulation slaves use CVODE (Hindmarsh et al., 2005) solvers with a tight tolerance bound in order to provide a solution as close as possible to the exact one. For the purpose of analysis the integration of a co-simulation slave is assumed to be analytic (Hartman, 2002):

$$
\begin{align*}
v_i[k] &= e^{\lambda h}v_i[k-1] + (e^{\lambda h} - I)A_i^{-1}B_iu_i[k] \\
x_i[k] &= v_i[k]
\end{align*}
$$

(42)

The integration matrices are defined as follows:

$$
\begin{align*}
G_i^{\alpha} &= e^{\lambda h}, & G_i^w &= (e^{\lambda h} - I)A_i^{-1}, & G_i^w &= I
\end{align*}
$$

(43)

where $i \in \{1, 2\}$.

### 3.2 Extrapolation

A zero-order hold (ZOH) element takes the last value and assigns it to the input:

$$
\begin{align*}
F_{i,ZOH}^{ww} &= 0, & F_{i,ZOH}^{wy} &= 1, & F_{i,ZOH}^{ww} &= I
\end{align*}
$$

(44)

where $i \in \{1, 2\}$ and forms $F_{i,ZOH}$ according to (26).

Nearly energy preserving coupling element (NEPCE) is an extrapolation element which tries to control the coupling error (Benedikt et al., 2013). Its implementation consists of an implementation of an integral controller. The implementation in this paper is modeled as:

$$
\begin{align*}
F_{i,NEPCE}^{ww} &= \alpha_i \\
F_{i,NEPCE}^{wy} &= I - \alpha_i \\
F_{i,NEPCE}^{ww} &= I
\end{align*}
$$

(45)

where $i \in \{1, 2\}$ and forms $F_{i,NEPCE}$ according to (26). It is interesting to note that a ZOH extrapolation element belongs to a subset of NEPCE extrapolation elements since:

$$
F_{i,ZOH} = F_{i,NEPCE}|_{\alpha_i=0}
$$

(46)

In later sections it is shown that tuning of NEPCE parameters can make a co-simulation more robust to parameter changes of the simulated system.

### 3.3 Sequential Co-simulation

<table>
<thead>
<tr>
<th>Table 1. Co-simulation Masters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>Seidel12</td>
</tr>
<tr>
<td>Seidel21</td>
</tr>
<tr>
<td>Control12</td>
</tr>
<tr>
<td>Control21</td>
</tr>
</tbody>
</table>

A calling sequence (17) for a sequential co-simulation of two slaves can be either:

$$
\sigma_{12}(r) = \begin{cases} 
1, & r = 1 \\
2, & r = 2
\end{cases}
$$

(47)
or:

\[ \sigma_{21}(r) = \begin{cases} 2, & r = 1 \\ 1, & r = 2 \end{cases} \]  

(48)

In the experiments presented in this paper, four co-simulation masters specified in Table 1 are used. Control master is named after the use of an error controller, namely \( F_{\text{unecc}} \). It is interesting to note that Seidel master is a subset of Control master:

\[ \text{Seidel} = \text{Control}|_\alpha=0 \]  

(49)

This fact stems directly from (46).

The signal response for each of them applied to (41) is presented in Figure 2. Through this paper the analysis of results has been done with the help of NumPy (Oliphant, 2007), Matplotlib (Hunter, 2007). The slaves and a reference configuration is available at repository BenchmarkFMUs (Glumac, 2017).

4 Relative Quality Measurements of Co-simulation

The goal of this section is to introduce measurements for the quality of co-simulation which do not require multiple co-simulation runs. The information from the previous sections should form the basis for such measurements.

4.1 Requirements for Initialization Phase

This paper uses fixed-point iteration (34) as an initialization algorithm for the co-simulation (Algorithm 1). In order for this algorithm to converge to an initial state, matrix \( \Phi_0 \) should be stable and have only eigenvalues with value 1 on the unit circle. Matrix \( \Phi_0 \) is stable if its spectral radius is:

\[ \rho(\Phi_0) < 1 \]  

(50)

and its eigenvalues on the unit circle are semi-simple (Elaydi, 1996). The stability alone is not enough for the initialization to terminate. The above stability constraint (50) still allows for bounded oscillations of the fixed-point iteration. In order to prevent them the following constraint should be satisfied:

\[ |\lambda| = 1 \Rightarrow \lambda = 1 \]  

(51)

i.e. the only eigenvalues on the unit circle should only have the value of 1. By using Jordan decomposition it can be seen that a sequence of matrices \( \Phi^k \) has eigenvalues moving on the unit circle whenever eigenvalues are different from \( e^{\theta_1} = 1 \), i.e. they have bounded oscillations.

The consistent initialization is defined in (Andersson, 2016), i.e. equations (14) and (16b) should be satisfied after the initialization phase:

\[ y[0] = Lu[0] \]  

(52a)

\[ y[0] = (I - DL)^{-1} Cx[0] \]  

(52b)

Conditions (50), (51) and (52) are introduced as a prerequisite for relative consistency measurement presented in the next subsection.

4.2 Relative Consistency

A local error control is a valid procedure to bound the global co-simulation error (Arnold et al., 2014). However, a local error depends on the initial state of the system and needs to be evaluated during co-simulation. Similar holds for the defect (Enright, 2000) which is used as the basis for the method proposed in this section. The goal of the method is to calculate the measure for consistency of the co-simulation with knowing only the structure of the system.

A connection defect is defined as:

\[ \delta[k] = \|y[k] - Lu[k]\| \]  

(53)

This definition of consistency is valid for a non-iterative co-simulation and can be measured during co-simulation with little or no extra cost. This measurement is useful as an indication whether or not to repeat a co-simulation run. However, an estimate for a single step of the co-simulation would be useful prior to the co-simulation run. The expression for the exact value of all the co-simulation values in the kth step can be calculated following (34):

\[ z[k] = \Phi^k \Phi_0^* R_u v_0 \]  

(54)

The defect in the same step can be calculated:

\[ \delta[k] = \|\Phi y - \Phi_0 y_0\| \]  

(55)

where \( S_y \) and \( S_u \) are selection matrices defined by the following equation:

\[ y = S_y z \]  

(56)

\[ u = S_u z \]  

(57)

The defect (55) still depends on the initial solver state \( v_0 \). In order to obtain a worst case estimate for defect in kth step, a matrix norm induced (Lancaster and Tismenetsky, 1985) by vector norm should be employed:

\[ \|M\| = \sup_{v_0 \neq 0} \frac{\|M v_0\|}{\|v_0\|} \]  

(58)

From the above definition it immediately follows that a defect (55) is bounded by:

\[ \|M v_0\| \leq \|M\| \|v_0\| \]  

(59)

This guarantees that minimizing the following relative consistency measure:

\[ \delta(\Phi) = \|\Phi y - \Phi_0 y_0\| \]  

(59)

will bound the defect with respect to internal solver state. The norm used for all of the experiments in this paper is 1-norm \( \|\cdot\|_1 \).
Relative Consistency and Robust Stability Measures for Sequential Co-simulation

Figure 3. The figure compares the relative consistency measurement (first subplot from the top) and global errors of position, velocity and force outputs (second, third, and forth subplot from the top, respectively).

Figure 3 presents the analysis whether relative consistency provides a good indication of the global error. It can be seen that global error of each particular signal in the test examples follows the relative consistency measure. The experiment conducted gives confidence in using (59) as a minimization objective in order to improve the co-simulation master.

4.3 Robust Stability

The unstructured stability radius of a matrix (Hinrichsen and Son, 1989) is defined as the size of perturbations needed to bring the system to the verge of instability:

$$ r(\Phi) = \inf \{ \| \Delta \| : \rho(\Phi + \Delta) < 1 \} $$

(60)

where $\rho(\Phi)$ is the spectral radius of a matrix defined as:

$$ \rho(\Phi) = \max_{\Phi v = \lambda v} |\lambda| $$

(61)

A stability radius calculates an important value for co-simulation. If a co-simulation master is set-up with some known linear time-invariant slaves, it is expected that this master is robust with respect to the changes of slave parameters. With a bigger stability radius the master is expected to work more reliably if a co-simulation slave is switched. In practice, this allows for a better prototyping of a complex system without a need to tune the co-simulation master.

In this paper a simplification of calculation has been considered. For the unstructured stability radius (60), the following relation holds (Hinrichsen and Son, 1989):

$$ r(\Phi) \leq 1 - \rho(\Phi) $$

(62)

The equality holds only for normal matrices. This is the reason of the assumption that the spectral radius can be considered a good robust stability measure.

In order to check whether a spectral radius (61) is a good pointer of a stability radius, the robustness of system (41) to change of stiffness $c_k$ and damping $d_k$ coefficients is analyzed. By inspecting the results visible in Figure 4, it can be seen that Control masters have larger stability intervals compared to Seidel masters, both with respect to stiffness:

$$ \rho(\text{Seidel}_{12}) < 1, \quad c_k \in [10^{-5}, 0.89] $$

$$ \rho(\text{Seidel}_{21}) < 1, \quad c_k \in [10^{-5}, 0.89] $$

$$ \rho(\text{Control}_{12}) < 1, \quad c_k \in [10^{-5}, 3.59] $$

$$ \rho(\text{Control}_{21}) < 1, \quad c_k \in [10^{-5}, 3.59] $$

(63)

and damping:

$$ \rho(\text{Seidel}_{12}) < 1, \quad d_k \in [10^{-5}, 2.26] $$

$$ \rho(\text{Seidel}_{21}) < 1, \quad d_k \in [10^{-5}, 2.26] $$

$$ \rho(\text{Control}_{12}) < 1, \quad d_k \in [10^{-5}, 10^5] $$

$$ \rho(\text{Control}_{21}) < 1, \quad d_k \in [10^{-5}, 10^5] $$

(64)

This was done for simplicity of the exposition. In the work on stability radii there is a calculation of structured stability radius which will be a topic for future work.
studied problem (Figure 5). The figure presents the comparison of two optimization objectives on the search space defined by four co-simulation masters and the communication step-size \( h \). The plots allow a choice of a step-size and a master with the estimation of trade-off. This is a form of brute-force multi-objective optimization as it samples all the search space and evaluates the objectives in each point. After the evaluation, a person is very likely to put some kind of goal on the first objective and search for the best value in the second one. This approach, however, is limited to a few parameters in the search space. In the case of Figure 5, only the communication-step size and the master type are varied. Control masters had been assigned control parameters before the experiment. The choice of control parameters is described in the rest of this subsection.

Figure 6 shows the results of brute-force optimization of Control\(_{21}\) co-simulation masters (Table 1). Unlike the previous approach, this one plots the Pareto frontier in the parameters space. The search space is defined as:

\[
\mathcal{S} = \{\text{Control}_{21}|_{\alpha, h}: \alpha \in [0, 1.1], h \in [10^{-3}, 10^1]\}
\]  

The parameter \( \alpha \) is sampled uniformly in the search interval, while \( h \) is sampled uniformly in the log-scale of the search interval. The Pareto frontier is shown in the parameter space of the co-simulation master, i.e. both left-hand and right-hand heat-maps are spanned by the communication step-size \( h \) on the y-axis and the controller gain \( \alpha \) on the x-axis. The shade of gray presents the value of an objective in a heat-map. The orange, yellow, and cyan mark points are the positions of the Pareto frontier solutions in the search space. The yellow color marks the Seidel\(_{21}\) masters, while orange color marks the Control\(_{21}\) masters with \( \alpha \neq 0 \). From (46), it immediately follows that a Seidel\(_{21}\) master is a subset of possible Control\(_{21}\) masters, i.e.:

\[
\text{Seidel}_{21} = \text{Control}_{21}|_{\alpha=0}
\]  

A closer look at the Pareto frontier reveals that Seidel\(_{21}\) masters (yellow points) dominate the rest of Control\(_{21}\) for smaller communication step-sizes. The exception is a set of orange points, Control\(_{21}\) masters at the top of the heat-maps. These points have smaller values of spectral radius (left heatmap) which corresponds to robustness to changes in system parameters (Figure 4). In addition, a communication step-size can be viewed as both a parameter and an objective. The bigger values of the communication step-size allow less burden for the communication network and allow more freedom for internal solvers to choose internal time-steps. In turn, this should probably decrease the CPU load. This is a rationale to prefer points with bigger values of communication step-size on the Pareto frontier. A smaller spectral radius and a bigger step-size are the reason to highlight one of the orange points with bigger values of communication step-size on the Pareto frontier. A smaller spectral radius and a bigger step-size allow less burden for the communication network and allow more freedom for internal solvers to choose internal time-steps. In turn, this should probably decrease the CPU load. This is a rationale to prefer points with bigger values of communication step-size on the Pareto frontier. A smaller spectral radius and a bigger step-size are the reason to highlight one of the orange points with bigger values of communication step-size on the Pareto frontier.
points as cyan \((\alpha = 0.6, h = 9.11)\):
\[
\begin{align*}
\rho(Control_{21}[\alpha=0.6,h=9.11]) &= 0.45 \\
\delta(Control_{21}[\alpha=0.6,h=9.11]) &= 0.78
\end{align*}
\] (70)

This master has been used in previous images to present responses and objective values of all Control_{21} masters.

The presentation in Figure 6 is feasible for search spaces with two real parameters. This is not the case for Control_{12} masters which have three parameters \(\alpha_1, \alpha_2\) and \(h\). The Pareto frontier can be presented in the objective space (Figure 7). Again, the Seidel masters (yellow) seem to dominate other Control masters (orange) on a subset of the Pareto frontier. Orange points close to the yellow are also close in the parameter space, but they have \(\alpha_1 > 0\). Again, the subset of Control masters (orange points on the left) has better stability properties for lower communication step-sizes. One of these points has been highlighted by cyan \((\alpha_1 = 0.59, \alpha_2 = 0.36, h = 8.9)\):
\[
\begin{align*}
\rho(Control_{12}[\alpha_1=0.59,\alpha_2=0.36,h=8.9]) &= 0.41 \\
\delta(Control_{12}[\alpha_1=0.59,\alpha_2=0.36,h=8.9]) &= 1.1
\end{align*}
\] (71)

This master has been used in the previous images to present responses and objective values of all Control_{21} masters.

The methods in this section are brute-force optimization techniques. They can easily become infeasible as the search space scales poorly with the number of co-simulation slaves. The calling sequence (17) of co-simulation slaves is a permutation function, and as such the number of its configurations is factorial of the number of co-simulation slaves. However, the goal of this section is to demonstrate the use of objectives in choice of a co-simulation master. The formulation of objectives (65) enables the use of more advanced multi-objective optimization algorithms (Kalyanmoy, 2001) which may tackle high dimensional search spaces.

6 Conclusion and Future Work

This paper introduces relative consistency measure (59) and robust stability measure (61) of a co-simulation. Both measures have been used as objectives in multi-objective optimization (66) in order to increase the efficacy of tuning a co-simulation master.

The relative consistency measure is a worst case defect measurement with respect to unknown internal states of slaves. The experiments conducted show the global error follows the similar trend to the proposed consistency measure. This gives a suggestion that the relative consistency is a good measure to indicate how well a co-simulation master suites the coupled system it co-simulates. Since this measure is not dependent on an initial state, it gives the measure for any possible run of a linear system. This measure may be applicable for a non-iterative communication-step size control which may be one of the topics for future work.

The analysis in this paper has been restricted to sequential masters. Jacobi master does parallel updates of multiple co-simulation slaves which cannot be modeled with a sequential matrix multiplication. Parallel execution should be modeled as a single matrix. This matrix would align equations for doStep functions of multiple co-simulation slaves. The construction of such matrix should be a part of future work.

Since practical systems are usually not linear, a robust
stability measure is introduced. It is argued that minimization of spectral radius (61) can be used to approximate unstructured stability radius (60). It is experimentally shown that a co-simulation with a lower spectral radius is more robust to changes of coupling stiffness and damping in the two-mass oscillator system. However, the spectral radius is only an approximation of the unstructured stability radius. Furthermore, it is not expected that the whole structure of the co-simulation is uncertain. The parameters of a co-simulation master are assumed to be known and certain, while the structured uncertainty should be concentrated on parameters of co-simulation slaves. For these reasons one of the main topics for future research will be to develop a calculation method of the structured stability radius (Hinrichsen and Pritchard, 2005) for the co-simulation.

References


