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 simulation 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 setup 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[1]. 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 (Kalyanmoy, 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-

[1]for linear systems

sures and demonstration of multi-objective optimization. The next section introduces a relative consistency and a robust stability measure. Both measures are minimized with the help of multi-objective optimization described in the fifth section. The section with conclusions and description of future work concludes the paper.

2 Co-simulation Model

2.1 Co-simulation Slave

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

$$G_i = (V_i, U_i, Y_i, D_i, v_i, set_i, get_i, doStep_i) \tag{1}$$

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

$$doStep_i : (V_i, \mathbb{R}^+) \to V_i \tag{2}$$

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:

$$v' = doStep_i(v, h) \tag{3}$$

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:

$$\dot{\mathbf{x}}_i(t) = \mathbf{A}_i\mathbf{x}_i(t) + \mathbf{B}_i\mathbf{u}_i(t) \tag{4a}$$
$$\mathbf{y}_i(t) = \mathbf{C}_i\mathbf{x}_i(t) + \mathbf{D}_i\mathbf{u}_i(t) \tag{4b}$$

where $\mathbf{x}_i : \mathbb{R}^+ \to \mathbb{R}^{|X_i|}$ is the state signal[3], $\mathbf{u}_i : \mathbb{R}^+ \to \mathbb{R}^{|U_i|}$ is the input signal, and $\mathbf{y}_i : \mathbb{R}^+ \to \mathbb{R}^{|Y_i|}$ is the output signal. A nonlinear differential equation:

$$\begin{aligned} \dot{\mathbf{x}}_i(t) &= \mathbf{f}_i(\mathbf{x}_i(t), \mathbf{u}_i(t)) \\ \mathbf{y}_i(t) &= \mathbf{g}_i(\mathbf{x}_i(t), \mathbf{u}_i(t)) \end{aligned} \tag{5}$$

can be transformed to (4) by linearization:

$$\mathbf{A}_i = \frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_i}\ ,\ \mathbf{B}_i = \frac{\partial \mathbf{f}_i}{\partial \mathbf{u}_i}\ ,\ \mathbf{C}_i = \frac{\partial \mathbf{g}_i}{\partial \mathbf{x}_i}\ ,\ \mathbf{D}_i = \frac{\partial \mathbf{g}_i}{\partial \mathbf{u}_i} \tag{6}$$

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

$$\begin{aligned} \mathbf{x}_i[k] &= \mathbf{x}_i(t_k) = \mathbf{x}_i(t_{k-1}+h) \\ &= \mathbf{x}_i(t_{k-1}) + \int_{t_{k-1}}^{t_k} \mathbf{A}_i\mathbf{x}_i(\tau) + \mathbf{B}_i\mathbf{u}_i(\tau)\mathrm{d}\tau \\ &= \mathbf{x}_i(t_{k-1}) + \int_{t_{k-1}}^{t_k} \mathbf{A}_i\mathbf{x}_i(\tau) + \mathbf{B}_i\mathbf{u}_i[k]\mathrm{d}\tau \end{aligned} \tag{7}$$

[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.

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:

$$\tilde{\mathbf{u}}(\tau) = \mathbf{u}[k]\ , \quad t_{k-1} < \tau \leqslant t_k \tag{8}$$

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

$$\begin{aligned} \mathbf{v}_i' &= \mathbf{G}_i^{\mathbf{vv}}\mathbf{v}_i + \mathbf{G}_i^{\mathbf{vu}}\mathbf{u}_i \\ \mathbf{x}_i' &= \mathbf{G}_i^{\mathbf{xv}}\mathbf{v}_i' \end{aligned} \tag{9}$$

where $\mathbf{v}_i : \mathbb{R}^{|V_i|}$.

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):

$$\mathbf{y}_i' = \mathbf{C}_i\mathbf{x}_i + \mathbf{D}_i\mathbf{u}_i \tag{10}$$

This formulation is consistent with the mathematical model of co-simulation in the FMI standard (FMI 2.0, Section 4.1.2), although the definition of the k^{th} signal sample is a bit different, both for input signals (8) and the output signal:

$$\tilde{\mathbf{y}}(t_k) = \mathbf{y}[k] \tag{11}$$

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:

$$N = (G, \mathbf{L}) \tag{12}$$

where $G = \{G_1, G_2, \dots G_n\}$ is a set of n co-simulation slaves, $U = U_1 \cup U_2 \cup \dots \cup U_n$ is a set of input variables of all co-simulation slaves, and $Y = Y_1 \cup Y_2 \cup \dots \cup Y_n$ is a set of output variables of all co-simulation slaves, $\mathbf{L} \in \mathbb{R}^{|U| \times |Y|}$ is a matrix representing output-input connections.

Let $\mathbf{u}(t)$, $\mathbf{y}(t)$, $\mathbf{x}(t)$ denote the column stacked values of all co-simulation slaves, respectively:

$$\mathbf{u}(t) = \begin{bmatrix} \mathbf{u}_1(t) \\ \mathbf{u}_2(t) \\ \vdots \\ \mathbf{u}_n(t) \end{bmatrix}, \quad \mathbf{y}(t) = \begin{bmatrix} \mathbf{y}_1(t) \\ \mathbf{y}_2(t) \\ \vdots \\ \mathbf{y}_n(t) \end{bmatrix}, \quad \mathbf{x}(t) = \begin{bmatrix} \mathbf{x}_1(t) \\ \mathbf{x}_2(t) \\ \vdots \\ \mathbf{x}_n(t) \end{bmatrix} \tag{13}$$

Output-input connections are denoted as matrix $\mathbf{L}$:

$$\mathbf{u}(t) = \mathbf{L}\mathbf{y}(t) \tag{14}$$

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

$$\begin{aligned}\mathbf{A} &= blockDiag(\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_n)\\ \mathbf{B} &= blockDiag(\mathbf{B}_1, \mathbf{B}_2, \dots, \mathbf{B}_n)\\ \mathbf{C} &= blockDiag(\mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_n)\\ \mathbf{D} &= blockDiag(\mathbf{D}_1, \mathbf{D}_2, \dots, \mathbf{D}_n)\end{aligned} \tag{15}$$

The coupled system can be described by the following equations:

$$\dot{\mathbf{x}}(t) = \left(\mathbf{A} + \mathbf{BL}\left(\mathbf{I} - \mathbf{DL}\right)^{-1}\mathbf{C}\right)\mathbf{x}(t) \tag{16a}$$

$$\mathbf{y}(t) = \left(\mathbf{I} - \mathbf{DL}\right)^{-1}\mathbf{Cx}(t) \tag{16b}$$

An implicit assumption is that there are no algebraic loops in the system, i.e. $det\left(\mathbf{I} - \mathbf{DL}\right) \neq 0$. 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 $\sigma : I \to I$ define the calling sequence where $I = \{1, 2, \dots, n\}$ is the set of slave indices. The calling sequence is defined by the following expression:

$$i = \sigma(r) \tag{17}$$

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 between[4] 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:

$$\begin{aligned}\mathbf{w}_i' &= \mathbf{F}_i^{\mathbf{ww}}\mathbf{w}_i + \mathbf{F}_i^{\mathbf{wy}}\mathbf{y}\\ \mathbf{u}_i' &= \mathbf{F}_i^{\mathbf{uw}}\mathbf{w}_i'\end{aligned} \tag{18}$$

The extrapolation element can have access to all the outputs of all the slaves in the co-simulation network[5]. 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 $\tilde{\mathbf{u}}(t)$ and $\tilde{\mathbf{y}}(t)$ denote the signals reconstructed from discrete samples provided by the co-simulation run.

In this paper the following equality holds:

$$\tilde{\mathbf{u}}(kh) = \mathbf{u}[k], \quad \tilde{\mathbf{y}}(kh) = \mathbf{y}[k] \tag{19}$$

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:

$$\mathbf{z} = \begin{bmatrix}\mathbf{x}^T & \mathbf{v}^T & \mathbf{y}^T & \mathbf{w}^T & \mathbf{u}^T\end{bmatrix}^T \tag{20}$$

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:

$$\mathbf{z}' = \mathbf{G}_i\mathbf{z} \tag{21}$$

where equation (9) should be satisfied:

$$\begin{aligned}\mathbf{x}_i' &= \begin{cases}\mathbf{G}_i^{\mathbf{xv}}\mathbf{G}_i^{\mathbf{vv}}\mathbf{v}_i + \mathbf{G}_i^{\mathbf{xv}}\mathbf{G}_i^{\mathbf{vu}}\mathbf{u}_i, & j = i\\ \mathbf{x}_j, & j \neq i\end{cases}\\ \mathbf{v}_j' &= \begin{cases}\mathbf{G}_i^{\mathbf{vv}}\mathbf{v}_i + \mathbf{G}_i^{\mathbf{vu}}\mathbf{u}_i, & j = i\\ \mathbf{v}_j, & j \neq i\end{cases}\\ \mathbf{y}_j' &= \mathbf{y}_j, \quad \mathbf{w}_j' = \mathbf{w}_j, \quad \mathbf{u}_j' = \mathbf{u}_j\end{aligned} \tag{22}$$

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

$$\mathbf{z}' = \mathbf{H}_i\mathbf{z} \tag{23}$$

where (10) should be satisfied:

$$\begin{aligned}\mathbf{y}_j' &= \begin{cases}\mathbf{C}_i\mathbf{x}_i + \mathbf{D}_i\mathbf{u}_i & , j = i\\ \mathbf{y}_j & , j \neq i\end{cases}\\ \mathbf{x}_j' &= \mathbf{x}_j, \quad \mathbf{y}_j' = \mathbf{y}_j, \quad \mathbf{w}_j' = \mathbf{w}_j, \quad \mathbf{u}_j' = \mathbf{u}_j\end{aligned} \tag{24}$$

The extrapolation elements can be reformulated to:

$$\mathbf{z}' = \mathbf{F}_i\mathbf{z} \tag{25}$$

where (18) should be satisfied:

$$\begin{aligned}&\mathbf{x}_j' = \mathbf{x}_j, \quad \mathbf{v}_j' = \mathbf{v}_j, \quad , \quad \mathbf{y}_j' = \mathbf{y}_j\\ &\mathbf{w}_j' = \begin{cases}\mathbf{F}_i^{\mathbf{ww}}\mathbf{w}_i + \mathbf{F}_i^{\mathbf{wy}}\mathbf{y}, & j = i\\ \mathbf{w}_j, & j \neq i\end{cases}\\ &\mathbf{u}_j' = \begin{cases}\mathbf{F}_i^{\mathbf{uw}}\mathbf{F}_i^{\mathbf{ww}}\mathbf{w}_i + \mathbf{F}_i^{\mathbf{uw}}\mathbf{F}_i^{\mathbf{wy}}\mathbf{y}, & j = i\\ \mathbf{u}_j, & j \neq i\end{cases}\end{aligned} \tag{26}$$

Such kind of reformulation allows for calculation of Algorithm 2 with the use of matrix operations. An iteration

[4] The term extrapolation is used loosely, it refers both to interpolation and extrapolation.

[5] Extrapolation matrices should be chosen with care to be consistent with the connection matrix $\mathbf{L}$ from (14).

Algorithm 1 Sequential co-simulation

```
Require: $N$, $\sigma$, $h$, $t_0$, $t_{end}$
  $k := 0$                                              ▷ Initialization phase
  $\mathbf{z} := \mathbf{R_v v}_0$
  while $\|\Delta\mathbf{z}\| > \varepsilon$ do
      for $r := 1$ to $n$ do
          $i = \sigma(r)$
          $\mathbf{w}_i := \mathbf{F}_i^{\mathbf{ww}}\mathbf{w}_i + \mathbf{F}_i^{\mathbf{wy}}\mathbf{y}$          ▷ Extrapolation
          $\mathbf{u}_i := \mathbf{F}_i^{\mathbf{ww}}\mathbf{w}_i$
          for $u_i \in U_i$ do
              $v_i := set_i(v_i, u_i, \mathbf{u}_{i[u_i]})$
          for $y_i \in Y_i$ do                           ▷ Read outputs
              $\mathbf{y}_{[y_i]} := get_j(s_j, y_j)$
  $\mathbf{z}_{[0]} = \mathbf{z}$
  while $t_0 + kh < t_{end}$ do                          ▷ Computation phase
      for $r := 1$ to $n$ do
          $i = \sigma(r)$
          $\mathbf{w}_i := \mathbf{F}_i^{\mathbf{ww}}\mathbf{w}_i + \mathbf{F}_i^{\mathbf{wy}}\mathbf{y}$          ▷ Extrapolation
          $\mathbf{u}_i := \mathbf{F}_i^{\mathbf{ww}}\mathbf{w}$
          for $u_i \in U_i$ do
              $v_i := set_i(v_i, u_i, \mathbf{u}_{i[u_i]})$
          $s_i = doStep_i(s_i, h)$                        ▷ Update internal state
          for $y_i \in Y_i$ do                           ▷ Read outputs
              $\mathbf{y}_{[y_i]} := get_j(s_j, y_j)$
      $k := k + 1$
      $\mathbf{z}_{[k]} := \mathbf{z}$                                    ▷ Signal assignment
```

of the computation phase of the algorithm can be reformulated as:

$$\mathbf{z}_{[k]} = \mathbf{\Phi}\mathbf{z}_{[k-1]} \tag{27}$$

The computation matrix $\mathbf{\Phi}$ can be calculated by tracing the steps of calculation phase of the algorithm:

$$\mathbf{\Phi} = \prod_{r=1}^{n} \mathbf{\Phi}_{\sigma(r)} = \mathbf{\Phi}_{\sigma(n)}\mathbf{\Phi}_{\sigma(n-1)} \cdots \mathbf{\Phi}_{\sigma(1)} \tag{28}$$

where $\mathbf{\Phi}_i$ represents the update of the overall co-simulation slave done by the co-simulation slave G_i:

$$\mathbf{\Phi}_i = \mathbf{H}_i\mathbf{G}_i\mathbf{F}_i \tag{29}$$

In this paper the initialization is done with the help of a fixed-point iteration. Initialization matrix $\mathbf{\Phi}_0$ is used to model an iteration of the initialization algorithm:

$$\mathbf{\Phi}_0 = \mathbf{\Phi}|_{\mathbf{G}_i = \mathbf{I},\ i \in I} \tag{30}$$

The initialization matrix is equal to computation matrix (27) with excluded solver updates[6] (function $doStep_i$), i.e. integration matrices $\mathbf{G}_i = \mathbf{I}$ equal to identity matrix. An iteration of the initialization phase of the algorithm can be reformulated as:

$$\mathbf{z}' = \mathbf{\Phi}_0\mathbf{z} \tag{31}$$

[6] A closer look at Algorithm 1 shows that the initialization and the computation phase are almost identical. The function call of $doStep_i$, i.e. the integration is missing in the initialization phase.

Algorithm 2 Sequential multi-method integration

```
Require: $N$, $\sigma$, $h$, $t_0$, $t_{end}$
  $k := 0$                                              ▷ Initialization phase
  $\mathbf{z} := \mathbf{R_v v}_0$
  while $\|\Delta\mathbf{z}\| > \varepsilon$ do
      for $r := 1$ to $n$ do
          $i = \sigma(r)$
          $\mathbf{z} := \mathbf{F}_i\mathbf{z}$                                    ▷ Extrapolation
          $\mathbf{z} := \mathbf{H}_i\mathbf{z}$                                    ▷ Read outputs
  $\mathbf{z}_{[0]} = \mathbf{z}$
  while $t_0 + kh < t_{end}$ do                          ▷ Computation phase
      for $r := 1$ to $n$ do
          $i = \sigma(r)$
          $\mathbf{z} := \mathbf{F}_i\mathbf{z}$                                    ▷ Extrapolation
          $\mathbf{z} := \mathbf{G}_i\mathbf{z}$                                    ▷ Update internal state
          $\mathbf{z} := \mathbf{H}_i\mathbf{z}$                                    ▷ Read outputs
      $k := k + 1$
      $\mathbf{z}_{[k]} := \mathbf{z}$                                    ▷ Signal assignment
```

The initialization phase is started with a value assignment:

$$\mathbf{z} = \mathbf{R_v v}_0 \tag{32}$$

where:

$$\begin{aligned} \mathbf{v} &= \mathbf{v}_0 \\ \mathbf{x}_i &= \mathbf{G}_i^{\mathbf{xv}}\mathbf{v}_i \end{aligned} \tag{33}$$

need to be assigned to respective positions in the co-simulation value vector $\mathbf{z}$. The initialization phase of the algorithm can be modeled with:

$$\mathbf{z}_{[0]} = \mathbf{\Phi}_0^{\infty}\mathbf{R_v v}_0 \tag{34}$$

where:

$$\mathbf{\Phi}_0^{\infty} = \lim_{m \to \infty} \mathbf{\Phi}_0^{m} \tag{35}$$

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:

$$U_1 = \{G_1.x, G_1.y\}, \quad U_2 = \{G_2.F\} \tag{36}$$

respectively. Output variables of slaves G_1 and G_2 are:

$$Y_1 = \{G_1.F\}, \quad Y_2 = \{G_2.x, G_2.y\} \tag{37}$$

respectively. The connection matrix is equal to:

$$\mathbf{L} = \begin{array}{c|ccc} & G_1.F & G_2.x & G_2.v \\ G_1.x & 0 & 1 & 0 \\ G_1.v & 0 & 0 & 1 \\ G_2.F & 1 & 0 & 0 \end{array} \tag{38}$$

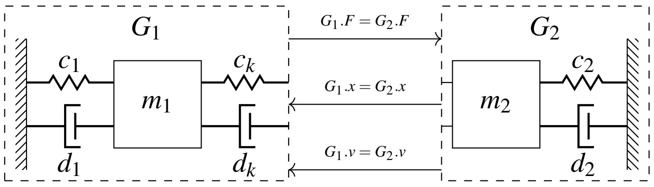

Figure 1. The example system used in the experiments throughout this paper is a two mass oscillator with force-displacement coupling. Slave G_1 provides force as an output, while slave G_2 provides the displacement and velocity of mass m_2.

Slave G_1 solves the following equations:

$$\mathbf{A}_1 = \begin{bmatrix} 0 & 1 \\ -\frac{c_1+c_k}{m_1} & -\frac{d_1+d_k}{m_1} \end{bmatrix} \quad \mathbf{B}_1 = \begin{bmatrix} 0 & 0 \\ \frac{c_k}{m_1} & \frac{d_k}{m_1} \end{bmatrix}$$
$$\mathbf{C}_1 = \begin{bmatrix} c_k & d_k \end{bmatrix} \qquad \mathbf{D}_1 = \begin{bmatrix} -c_k & -d_k \end{bmatrix} \tag{39}$$

Slave G_2 solves the following equations:

$$\mathbf{A}_2 = \begin{bmatrix} 0 & 1 \\ -\frac{c_2}{m_2} & -\frac{d_2}{m_1} \end{bmatrix} \quad \mathbf{B}_2 = \begin{bmatrix} 0 \\ \frac{1}{m_2} \end{bmatrix}$$
$$\mathbf{C}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mathbf{D}_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{40}$$

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

$$\begin{aligned} &m_1 = 10, \quad c_1 = 1, \quad d_1 = 1, \quad c_k = 1, \quad d_k = 2 \\ &m_2 = 10, \quad c_2 = 1, \quad d_2 = 2 \end{aligned} \tag{41}$$

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):

$$\begin{aligned} \mathbf{v}_i[k] &= e^{\mathbf{A}_i h}\mathbf{v}[k-1] + (e^{\mathbf{A}_i h} - \mathbf{I})\mathbf{A}_i^{-1}\mathbf{B}_i\mathbf{u}_i[k] \\ \mathbf{x}_i[k] &= \mathbf{v}_i[k] \end{aligned} \tag{42}$$

The integration matrices are defined as follows:

$$\mathbf{G}_i^{\mathbf{vv}} = e^{\mathbf{A}_i h}, \quad \mathbf{G}_i^{\mathbf{vu}} = (e^{\mathbf{A}_i h} - \mathbf{I})\mathbf{A}_i^{-1}, \quad \mathbf{G}_i^{\mathbf{xv}} = \mathbf{I} \tag{43}$$

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

3.2 Extrapolation

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

$$\mathbf{F}_{i,ZOH}^{\mathbf{ww}} = \mathbf{0}, \quad \mathbf{F}_{i,ZOH}^{\mathbf{wy}} = \mathbf{I}, \quad \mathbf{F}_{i,ZOH}^{\mathbf{uw}} = \mathbf{I} \tag{44}$$

where $i \in \{1,2\}$ and forms $\mathbf{F}_{i,ZOH}$ 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:

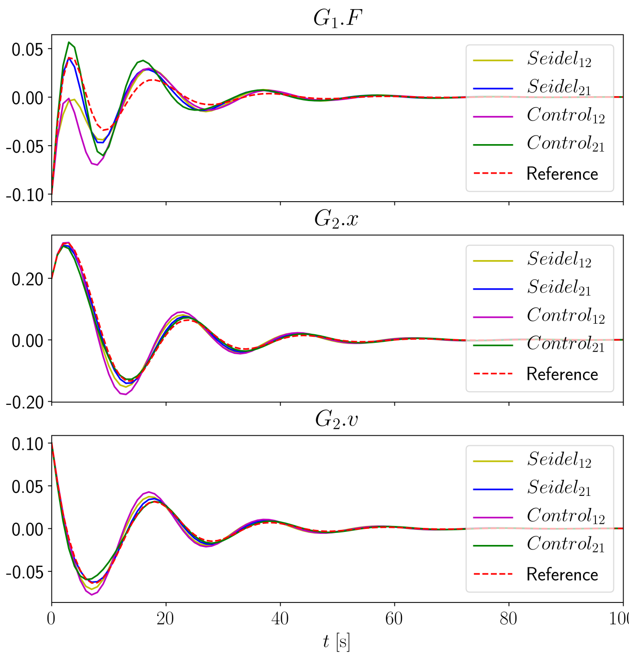

Figure 2. The plot shows output responses of four different co-simulation masters applied to the two-mass oscillator example system with co-simulation step size of $h = 1$.

$$\begin{aligned} \mathbf{F}_{i,NEPCE}^{\mathbf{ww}} &= \boldsymbol{\alpha}_i \\ \mathbf{F}_{i,NEPCE}^{\mathbf{wy}} &= \mathbf{I} - \boldsymbol{\alpha}_i \\ \mathbf{F}_{1,NEPCE}^{\mathbf{uw}} &= \mathbf{I} \end{aligned} \tag{45}$$

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

$$\mathbf{F}_{i,ZOH} = \mathbf{F}_{i,NEPCE}|_{\boldsymbol{\alpha}_i = \mathbf{0}} \tag{46}$$

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 1. Co-simulation Masters

Master	σ	F_1	F_2
Seidel$_{12}$	σ_{12}	$\mathbf{F}_{1,ZOH}$	$\mathbf{F}_{2,ZOH}$
Seidel$_{21}$	σ_{21}	$\mathbf{F}_{1,ZOH}$	$\mathbf{F}_{2,ZOH}$
Control$_{12}$	σ_{12}	$\mathbf{F}_{1,NEPCE}$	$\mathbf{F}_{2,ZOH}$
Control$_{21}$	σ_{21}	$\mathbf{F}_{1,ZOH}$	$\mathbf{F}_{2,NEPCE}$

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

$$\sigma_{12}(r) = \begin{cases} 1, & r = 1 \\ 2, & r = 2 \end{cases} \tag{47}$$

or:

$$\sigma_{21}(r) = \begin{cases} 2, & r = 1 \\ 1, & r = 2 \end{cases} \tag{48}$$

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 $\mathbf{F}_{i,NEPCE}$. It is interesting to note that *Seidel* master is a subset of *Control* master:

$$Seidel = Control|_{\boldsymbol{\alpha}=\mathbf{0}} \tag{49}$$

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, 2015) and 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 $\mathbf{\Phi}_0$ should be stable and have only eigenvalues with value 1 on the unit circle. Matrix $\mathbf{\Phi}_0$ is stable if its spectral radius is:

$$\rho(\mathbf{\Phi}_0) \leqslant 1 \tag{50}$$

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:

$$|\lambda| = 1 \Rightarrow \lambda = 1 \tag{51}$$

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 $\mathbf{\Phi}^k$ has eigenvalues moving on the unit circle whenever eigenvalues are different from $e^{j0} = 1$, 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:

$$\mathbf{y}[0] = \mathbf{L}\mathbf{u}[0] \tag{52a}$$

$$\mathbf{y}[0] = (\mathbf{I} - \mathbf{D}\mathbf{L})^{-1}\,\mathbf{C}\mathbf{x}[0] \tag{52b}$$

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:

$$\delta[k] = \|\mathbf{y}[k] - \mathbf{L}\mathbf{u}[k]\| \tag{53}$$

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 k^{th} step can be calculated following (34):

$$\mathbf{z}[k] = \mathbf{\Phi}^k \mathbf{\Phi}_0^\infty \mathbf{R}_\mathbf{v} \mathbf{v}_0 \tag{54}$$

The defect in the same step can be calculated:

$$\delta[k] = \left\|(\mathbf{S}_\mathbf{y} - \mathbf{L}\mathbf{S}_\mathbf{u})\,\mathbf{\Phi}^k \mathbf{\Phi}_0^\infty \mathbf{v}_0\right\| \tag{55}$$

where $\mathbf{S}_\mathbf{y}$ and $\mathbf{S}_\mathbf{u}$ are selection matrices defined by the following equation:

$$\begin{aligned} \mathbf{y} &= \mathbf{S}_\mathbf{y}\mathbf{z} \\ \mathbf{u} &= \mathbf{S}_\mathbf{u}\mathbf{z} \end{aligned} \tag{56}$$

The defect (55) still depends on the initial solver state $\mathbf{v}_0$. In order to obtain a worst case estimate for defect in k^{th} step, a matrix norm induced (Lancaster and Tismenetsky, 1985) by vector norm should be employed:

$$\|\mathbf{M}\| = \sup_{\mathbf{v}_0 \neq \mathbf{0}} \frac{\|\mathbf{M}\mathbf{v}_0\|}{\|\mathbf{v}_0\|} \tag{57}$$

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

$$\|\mathbf{M}\mathbf{v}_0\| \leqslant \|\mathbf{M}\|\,\|\mathbf{v}_0\| \tag{58}$$

This guarantees that minimizing the following relative consistency measure:

$$\delta(\mathbf{\Phi}) = \left\|(\mathbf{S}_\mathbf{y} - \mathbf{L}\mathbf{S}_\mathbf{u})\,\mathbf{\Phi}^k \mathbf{\Phi}_0^\infty\right\| \tag{59}$$

will bound the defect with respect to internal solver state. The norm used for all of the experiments in this paper is 1-norm $\|\,.\,\|_1$.

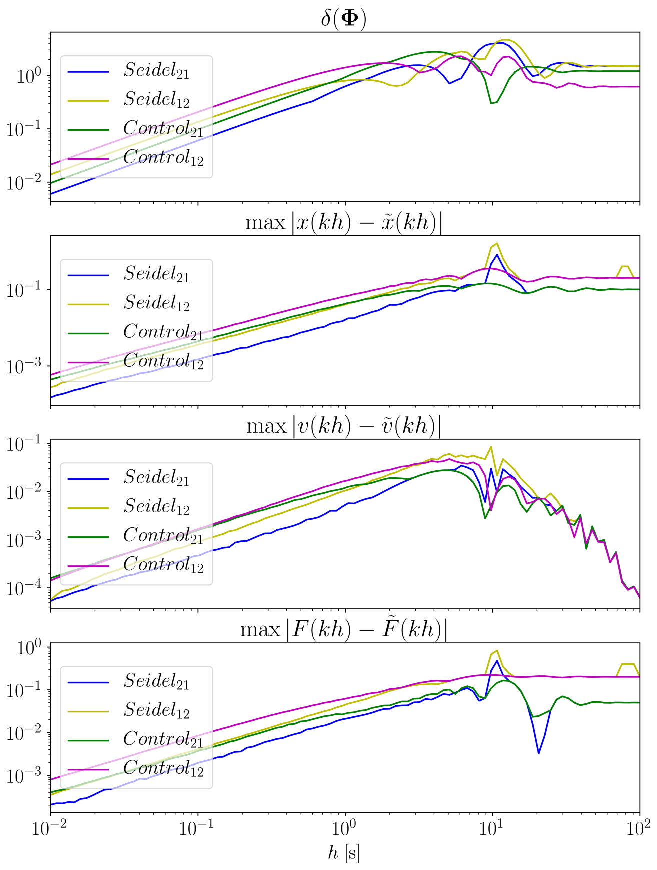

Figure 3. The figure compares the relative consistency measurement (first subplot from the top) and global errors of position, velocity and force outputs (second, third, and forth subplot from the top, respectively).

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:

$$r(\mathbf{\Phi}) = \inf\{\|\mathbf{\Delta}\| : \rho(\mathbf{\Phi}+\mathbf{\Delta}) < 1\} \tag{60}$$

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

$$\rho(\mathbf{\Phi}) = \max_{\mathbf{\Phi v}=\lambda \mathbf{v}} |\lambda| \tag{61}$$

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

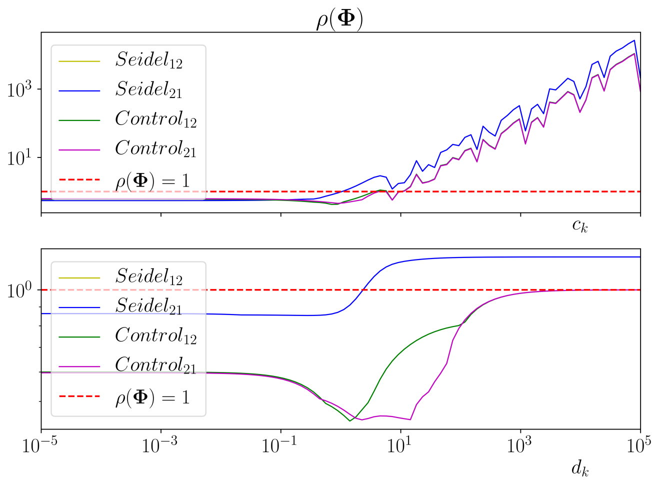

Figure 4. The figure shows the spectral radius of a two-mass oscillator co-simulation with respect to change of coupling stiffness c_k (upper plot) and coupling damping d_k (lower plot). The plots are used to indicate the size of stability region in terms of system parameters.

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):

$$r(\mathbf{\Phi}) \leqslant 1-\rho(\mathbf{\Phi}) \tag{62}$$

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[7].

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:

$$\begin{aligned}
\rho(Seidel_{12}) < 1, &\quad c_k \in [10^{-5}, 0.89]\\
\rho(Seidel_{21}) < 1, &\quad c_k \in [10^{-5}, 0.89]\\
\rho(Control_{12}) < 1, &\quad c_k \in [10^{-5}, 3.59]\\
\rho(Control_{21}) < 1, &\quad c_k \in [10^{-5}, 3.59]
\end{aligned} \tag{63}$$

and damping:

$$\begin{aligned}
\rho(Seidel_{12}) < 1, &\quad d_k \in [10^{-5}, 2.26]\\
\rho(Seidel_{21}) < 1, &\quad d_k \in [10^{-5}, 2.26]\\
\rho(Control_{12}) < 1, &\quad d_k \in [10^{-5}, 10^{5}]\\
\rho(Control_{21}) < 1, &\quad d_k \in [10^{-5}, 10^{5}]
\end{aligned} \tag{64}$$

[7] 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.

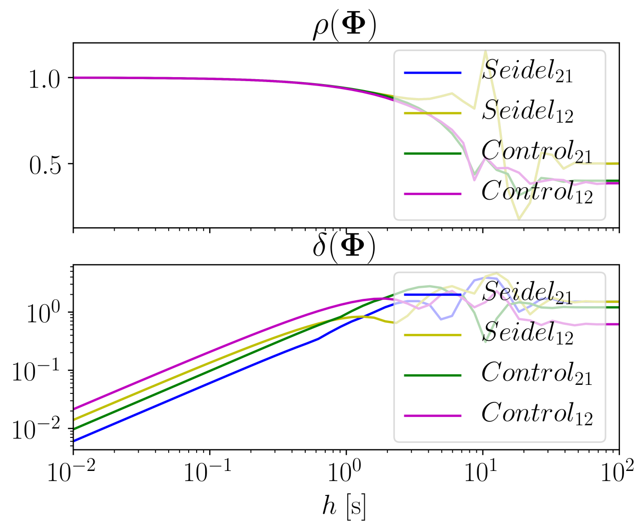

Figure 5. Comparison of the spectral radii (upper plot) and relative consistencies (lower plot) for different co-simulation masters with respect to the change of the communication step-size.

5 Optimal Choice of a Co-simulation Master

There can be multiple objectives defined for a co-simulation master. Two were defined in the previous section, the spectral radius (61) and the relative consistency (59):

$$\begin{aligned} J_1(\mathbf{\Phi}) &= \rho(\mathbf{\Phi}) \\ J_2(\mathbf{\Phi}) &= \delta(\mathbf{\Phi}) \end{aligned} \tag{65}$$

In order to make an optimal choice of a co-simulation master with respect to the above objectives, multi-objective optimization (Kalyanmoy, 2001) should be employed. Multi-objective optimization is a technique for finding a non-dominated subset of solutions, i.e. the Pareto frontier $\mathscr{P}$:

$$\mathscr{P} = \{\mathbf{\Phi} \in \mathscr{S} : \neg(\exists \mathbf{\Phi}_{other}. \quad \mathbf{\Phi} \preceq \mathbf{\Phi}_{other})\} \tag{66}$$

The above definition states that solutions in the Pareto frontier $\mathscr{P}$ are not dominated by any other solution in the search space $\mathscr{S}$. A solution is dominated by another one $\mathbf{\Phi} \preceq \mathbf{\Phi}_{other}$ if it is worse or equal with respect to all objectives, and strictly worse with respect to at least one objective:

$$\begin{gathered} \mathbf{\Phi} \preceq \mathbf{\Phi}_{other} \\ \Updownarrow \\ J_1(\mathbf{\Phi}) < J_1(\mathbf{\Phi}_{other}) \wedge J_2(\mathbf{\Phi}) \leqslant J_2(\mathbf{\Phi}_{other}) \\ \vee \\ J_1(\mathbf{\Phi}) \leqslant J_1(\mathbf{\Phi}_{other}) \wedge J_2(\mathbf{\Phi}) < J_2(\mathbf{\Phi}_{other}) \end{gathered} \tag{67}$$

A multi-objective optimization is useful when objectives are conflicting [8], which seems to be the case in the 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.

[8] Otherwise, a minimization of one objective would minimize the other. In this case a single-objective minimization could be more efficient.

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:

$$\begin{aligned} \mathscr{S} = \{Control_{21}|_{\alpha,h}{:}\alpha \in [0, 1.1], \\ h \in [10^{-3}, 10^{1}]\} \end{aligned} \tag{68}$$

The parameter α 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 α 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.:

$$Seidel_{21} = Control_{21}|_{\alpha=0} \tag{69}$$

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 heat-map) 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

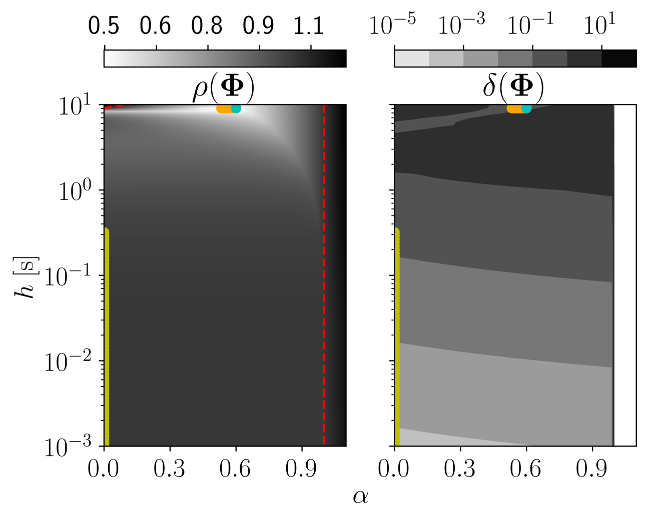

Figure 6. The Pareto frontier of $Control_{21}$ co-simulation is presented in the parameter space (α, h). The yellow points present $Seidel_{21}$ masters, while orange points present the rest of $Control_{21}$ masters. The cyan point represent $Control_{21}|_{\alpha=0.6,h=9.11}$ with objective values (70). The dashed red line represents a stability margin for the system, i.e. $\rho(\mathbf{\Phi}) = 1$.

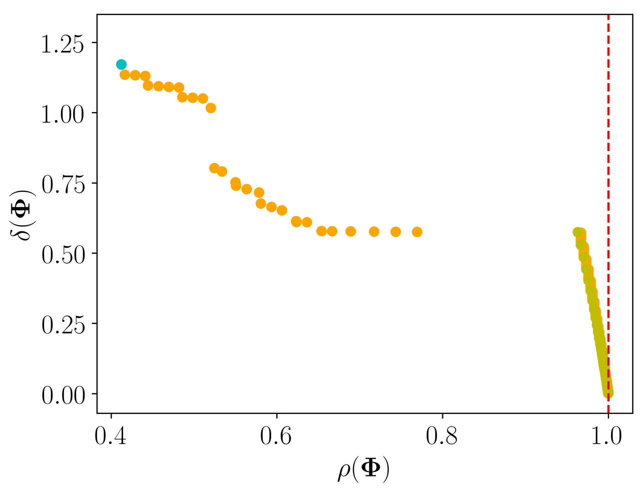

Figure 7. The Pareto frontier of $Control_{12}$ co-simulation is presented in the objective space (ρ, δ). The yellow points present $Seidel_{12}$ masters, while orange points present the rest of $Control_{12}$ masters. The cyan point represent $Control_{12}|_{\alpha_1=0.59,\alpha_2=0.36,h=8.9}$ with objective values (71). The dashed red line represents a stability margin for the system, i.e. $\rho(\mathbf{\Phi}) = 1$.

points as cyan $(\alpha = 0.6, h = 9.11)$:

$$\begin{aligned}\rho(Control_{21}|_{\alpha=0.6,h=9.11}) &= 0.45\\ \delta(Control_{21}|_{\alpha=0.6,h=9.11}) &= 0.78\end{aligned} \tag{70}$$

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 α_1, α_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 $(\alpha_1 = 0.59, \alpha_2 = 0.36, h = 8.9)$:

$$\begin{aligned}\rho(Control_{12}|_{\alpha_1=0.59,\alpha_2=0.36,h=8.9}) &= 0.41\\ \delta(Control_{12}|_{\alpha_1=0.59,\alpha_2=0.36,h=8.9}) &= 1.1\end{aligned} \tag{71}$$

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 unfeasible 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

Johan Åkesson, Willi Braun, Petter Lindholm, and Bernhard Bachmann. Generation of sparse jacobians for the function mock-up interface 2.0. In *Proceedings of the 9th International MODELICA Conference; September 3-5; 2012; Munich; Germany*, pages 185–196. Linköping University Electronic Press, 2012.

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

Martin Arnold, Christoph Clauß, and Tom Schierz. Error analysis and error estimates for co-simulation in fmi for model exchange and co-simulation v2. 0. In *Progress in Differential-Algebraic Equations*, pages 107–125. Springer, 2014.

Martin Benedikt, Daniel Watzenig, Josef Zehetner, and Anton Hofer. *NEPCE - A nearly energy-preserving coupling element for weak-coupled problems and co-simulation*, pages 1–12. ., 2013. ISBN 978-84-941407-6-1.

Martin Busch. *Zur effizienten Kopplung von Simulationsprogrammen*. kassel university press GmbH, 2012.

Saber N. Elaydi. *An Introduction to Difference Equations*. Springer-Verlag, Berlin, Heidelberg, 1996. ISBN 0-387-94582-2.

WH Enright. Continuous numerical methods for odes with defect control. *Journal of Computational and Applied Mathematics*, 125(1-2):159–170, 2000.

FMI 2.0. Functional Mock-up Interface for Model Exchange and Co-Simulation, Version 2.0. `https://svn.modelica.org/fmi/branches/public/specifications/v2.0/FMI_for_ModelExchange_and_CoSimulation_v2.0.pdf`, 2014. Accessed: 2018-10-26.

Sofia Gedda, Christian Andersson, Johan Åkesson, and Stefan Diehl. Derivative-free parameter optimization of functional mock-up units. In *Proceedings of the 9th International MODELICA Conference; September 3-5; 2012; Munich; Germany*, pages 819–828. Linköping University Electronic Press, 2012.

Slaven Glumac. Benchmarkfmus. `https://github.com/sglumac/BenchmarkFMUs`, 2017. Accessed: 2018-11-04.

P. Hartman. *Ordinary Differential Equations*. Society for Industrial and Applied Mathematics, second edition, 2002. doi:10.1137/1.9780898719222. URL `https://epubs.siam.org/doi/abs/10.1137/1.9780898719222`.

Alan C Hindmarsh, Peter N Brown, Keith E Grant, Steven L Lee, Radu Serban, Dan E Shumaker, and Carol S Woodward. Sundials: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):363–396, 2005.

D. Hinrichsen and N. K. Son. The complex stability radius of discrete-time systems and symplectic pencils. In *Proceedings of the 28th IEEE Conference on Decision and Control,*, pages 2265–2270 vol.3, Dec 1989. doi:10.1109/CDC.1989.70573.

Diederich Hinrichsen and Anthony J Pritchard. *Mathematical systems theory I: modelling, state space analysis, stability and robustness*, volume 48. Springer Berlin, 2005.

John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90–95, 2007.

Zdzislaw Jackiewicz. *General linear methods for ordinary differential equations*. John Wiley & Sons, 2009.

Deb Kalyanmoy. *Multi objective optimization using evolutionary algorithms*. John Wiley and Sons, 2001.

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

Peter Lancaster and Miron Tismenetsky. *The theory of matrices: with applications*. Elsevier, 1985.

Travis E. Oliphant. *Guide to NumPy*. CreateSpace Independent Publishing Platform, USA, 2nd edition, 2015. ISBN 151730007X, 9781517300074.

Bernhard Schweizer and Daixing Lu. Predictor/corrector co-simulation approaches for solver coupling with algebraic constraints. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 95(9):911–938, 2015.

Bernhard Schweizer, Pu Li, and Daixing Lu. Explicit and implicit cosimulation methods: Stability and convergence analysis for different solver coupling approaches. *Journal of Computational and Nonlinear Dynamics*, 10(5):051007, 2015.

Georg Stettinger, Martin Horn, Martin Benedikt, and Josef Zehetner. *A model-based approach for prediction-based interconnection of dynamic systems*, pages 3286–3291. ., 2014. ISBN 978-1-4673-6088-3. doi:10.1109/CDC.2014.7039897.