Test of Basic Co-Simulation Algorithms Using FMI

Kosmas Petridis[1] Christoph Clauß[2]

[1]Robert Bosch GmbH, Corporate Sector Research and Advance Engineering, Robert-Bosch-Campus 1, 71272 Renningen, Germany, kosmas.petridis@de.bosch.com

[2]Fraunhofer IIS EAS, Zeunerstrasse 38, 01069 Dresden, Germany, christoph.clauss@eas.iis.fraunhofer.de

Abstract

Since the FMI technology gains ground in industrial environment, the demand for robust co-simulation increases. In a master-slave concept the master algorithms define the quality of a co-simulation whereas the properties of the coupled FMUs for co-simulation restrict the variety of possible master algorithms. In this paper an existing experimental master tool with three basic master algorithms was improved to support FMI 2.0 as well as 1.0. For testing more than 20 Modelica examples were developed from which FMUs for co-simulation were generated by established simulation tools (e.g., Dymola, SimulationX). The examples demonstrate differences of the three master algorithms. Recommendations for tearing as well as improving the master algorithms are given.

Keywords: Co-Simulation; FMI; master algorithm;

1 Introduction

Nowadays simulation is of crucial importance in the development of mechatronic and cyberphysical systems. The main characteristic of such systems is that they consist of components of different physical domains like hydraulic, mechanic, electronic, and software. Through the strong coupling between the components the isolated investigation of single components is not sufficient. In fact the overall system has to be investigated. This means that we need to simulate the complete system. In general, the components are modelled and simulated in different established simulation tools. One commonly used method to simulate the complete system is co-simulation which can be classified into two types: the direct coupling between tools and the export and import of the simulation model into the other tool. To do this, there exist a lot of proprietary commercial and self-developed solutions but all of them are only applicable on a limited number of tool combinations. In addition these solutions need a high effort in maintenance because of the proprietary interface to the different simulation tools. A further disadvantage of these solutions is that the algorithms used for the coupling are strongly coupled with the interface. In addition usually only standard algorithms based on a constant macro step size are used. To avoid these limitations the Functional Mock-up Interface (FMI) was developed as an interface standard which allows the exchange and co-simulation of models. The standard allows the use of different coupling algorithms within the same interface. The coupling algorithms themselves are not part of the standard. Because of the increasing number of simulation tools, which support this standard, and the need from an industrial point of view (Bertsch et al, 2014) FMI represents a promising industry standard for model exchange.

2 Co-Simulation in Industrial Environment

One example where co-simulation is used to analyze the system is the simulation of injection valves (Petridis, 2013). The following physical domains are simulated with different simulation tools:

- Hydraulics and mechanics
- Electromagnetics and power electronics

Numerous additional examples for co-simulation like the simulation of high-pressure pumps, breaking systems, etc. exist.

Based on these applications we determined the following coupling cases:

- Simulator specific model with one imported FMU
- Simulator specific model with more than one imported FMU
- Software in the loop (SIL) platform with control algorithms and one or more FMU plant models

Thereby the type of coupling can be distinguished by:

- Coupling in one direction (see Figure 1) or with feedback (see Figure 2). The last one is also known as cycle.
- Analog coupling quantities (displacement, force, etc.) or discrete coupling quantities (sensor or actor signals)

The different simulation models can have the properties:

- Algebraic system without solver
- Differential or differential algebraic equation including solver (based on constant or variable solving step size) or without solver

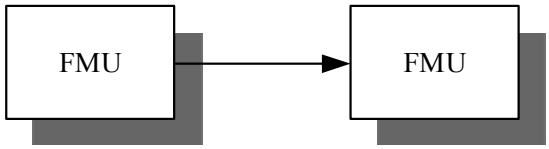

Figure 1. One directional coupling between two FMUs.

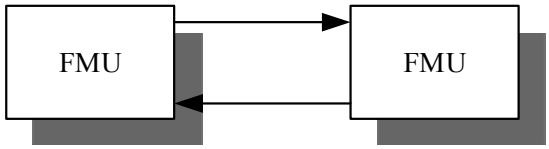

Figure 2. Coupling with feedback between two FMUs.

The described coupling configurations are an incomplete snapshot based on the current co-simulation applications. But it is a very useful orientation to determine the requirements on coupling algorithms.

3 Basic Co-Simulation Algorithms

Assume, *m* simulators $S_i,\ i = 1(1)m$, are to be coupled. Sometimes, such a simulator S_i is called "slave". Typically, it is "packaged" in a FMU. These simulators are assumed to exchange altogether *n* coupling variables $x_j(t) \in R^1,\ \ j = 1(1)n,\ \ t \in [0,T]$, which are time dependent. Some of the coupling variables are input variables for a simulator S_i, other coupling variables are output variables. It is assumed that no input variable is output variable of the same simulator. $[0,T]$ is the time interval to be simulated. If $x = (x_1,\ x_2,\ \ldots,\ x_n)^T \in R^n$ is the vector of all coupling variables then the call of each single simulator can be described by $x = Q_i S_i(P_i x)$.

P_i is a $(p_i \times n)$-matrix with one "1" at each row and all other entries identical zero. It denominates which coupling variable is the input of the simulator S_i. S_i has p_i input variables. Q_i is a $(n \times q_i)$-matrix with one "1" at each column and all other entries identical zero. It denominates which coupling variable is the output of the simulator S_i. S_i has q_i output variables. Since each coupling variable is output of exactly one simulator, $\sum_{i=1}^{m} q_i = n$ holds. Furthermore, the $(n \times n)$-matrix $Q = (Q_1,\ Q_2,\ldots, Q_m)$ has exactly one "1" in each column and in each line. Since no input variable is assumed to be an output variable of the same simulator $P_i Q_i = \theta$ holds with θ being a $(p_i \times q_i)$-zero-matrix. The matrices P_i and Q_i describe the connection graph, that means how the input variables and output variables of each simulator are connected.

To illustrate the notation of coupling, the following example is given in Figure 3. We have $m = 2$ simulators S_1 and S_2 with $n = 4$ coupling variables $x = (x_1,\ x_2,\ x_3, x_4)^T$. S_1 has $p_1 = 1$ input variables and $q_1 = 3$ output variables. Thus the dimension of P_1 is (1×4) and the dimension of Q_1 is (4×3). The matrices are $P_1 = (0\ 0\ 0\ 1)$ and $Q_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$. S_2 has $p_2 = 3$ input variables and $q_2 = 1$ output variables. Thus the dimension of P_2 is (3×4) and the dimension of Q_2 is (4×1). The matrices are $P_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$ and $Q_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$. The input variables of S_1 are described with $P_1 x = x_4$ and of S_2 with $P_2 x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$. The output of S_1 is given by the (3×1)-vector $S_1(P_1 x)$ and of S_2 by the (1×1) dimensional $S_2(P_2 x)$. The multiplication of these terms with the corresponding matrix Q

$$Q_1 S_1(P_1 x) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} S_1\big((x_4)\big)$$

$$Q_2 S_2(P_2 x) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} S_2\left(\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}\right)$$

corresponds to a mapping of the outputs of each simulator to the coupling vector x.

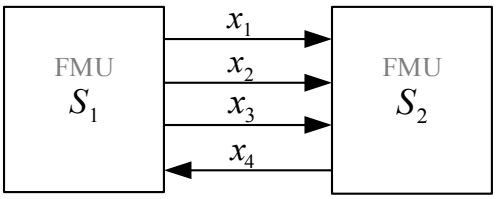

Figure 3. Example to describe the notation

Using this notation the task of the coupled simulation can be described as the following task: Find x^* which solves:

$$x^* = \sum_{i=1}^{m} Q_i S_i(P_i x^*) \qquad (1)$$

In general, this equation is a nonlinear equation in the space of time dependent functions. All solution methods which are available to solve nonlinear equations should be checked to solve this equation.

First fixed point iteration methods are possible which take equation (1) "as it is". There are several approaches. With k being the iteration index, and $x^0(t) = \big(x_1^0(t), \ldots, x_n^0(t)\big), t \in [0,T]$ the initialization of coupling variables, the Gauss-Jacobi method can be characterized by

$$x^{k+1} = \sum_{i=1}^{m} Q_i S_i(P_i x^k) \qquad (2)$$

DOI 10.3384/ecp15118865

In this case the simulators S_i can operate in parallel, since all simulators access to the same vector of coupling variables x^k.

Another variant is the Gauss-Seidel method (GSM) which needs an a priori defined calling sequence $r = (r_1, \dots, r_m)$ of the simulators. Each simulator uses the results of the already called simulators. One iteration step is finished if all simulators have been called. The Gauss-Seidel method can be summarized by

$$\xi^1 := x^k$$

$$\xi^{i+1} := Q_{r_i} S_{r_i}(P_{r_i}\xi^i) + \xi^i - Q_{r_i} Q_{r_i}^T \xi^i \qquad i = 1(1)m \tag{3}$$

$$x^{k+1} := \xi^{m+1}$$

The sequence r is defined by analyzing the matrices P_i and Q_i which is the same as to analyze the connection graph. The sequence shall be chosen such that as many as possible input coupling variables are "updated" before the simulation of each slave simulator.

A special case of the Gauss-Seidel method takes one iteration ($x^0 \to x^1$) only. That means that no "true" iteration takes place. This method is sufficient if a sequence *r* of simulator calls can be found at which each simulator takes input values only which are outputs of before called simulators.

Equation (1) can be reordered into

$$0 = x^* - \sum_{i=1}^{m} Q_i S_i(P_i x^*) =: x^* - Sx^* \tag{4}$$

To find a "root" of (4) Newton like methods can be applied, e.g. the classical Newton-Raphson method (NRM):

$$x^{k+1} = \left(\frac{\partial S}{\partial x} - I\right)^{-1}\left(\frac{\partial S}{\partial x}x^k - Sx^k\right) \tag{5}$$

In addition, for all groups of methods (2), (3), (5) many modifications are known (Schwetlick, 1979), e.g. the introduction of damping methods which limit large changes of the solution between two iterations.

So far all of these methods are applied to the complete functions $(t),\ t \in [0,T]$. This results in waveform relaxation methods and waveform Newton methods respectively which operate with functions within a function space on the time interval $[0,T]$.

Since FMI is not designed to exchange function space variables but values at certain time points the time interval is segmented into subintervals (communication intervals) $[0,T] = \cup\,[t_c, t_{c+1}]$. Each time t_c is a communication point at which the simulation of all slave simulators S_i is stopped for the exchange of values between the master and the slave simulators. $h_c = t_{c+1} - t_c$ is the communication step size (macro step size) between communication points which can be variable or constant. The above described task of simulator coupling (1) is solved for each communication interval $[t_c, t_{c+1}]$. The simulation of a slave simulator S_i within a communication interval is performed by the FMI doStep function. Both the methods (2) and (5) need repeated simulations of communication intervals, the "FMUState" must be stored (GetFMUState) and used again (SetFMUState).

The presented approaches yield a high variety of methods which have to be chosen depending on the properties of the simulation task and the restrictions of the FMUs to be coupled. A master should offer many coupling algorithms to be able to choose the best suitable one for a special coupling task. General criteria for the quality of master algorithms are the performance (it touches questions like this: are slave simulations repeated within communication intervals?), the correctness of the results (it touches questions of stability with respect to the communication step size) as well as the robustness (is an algorithm suitable for a large class of simulation tasks?). The choice of an ideally adapted master algorithm is an active field of research.

4 Tool for Testing Co-Simulation Algorithms

In (Bastian et al, 2011) a prototypical master tool for co-simulation is presented which was developed in the MODELISAR project by Fraunhofer IIS EAS Dresden. The aim was to investigate basic co-simulation algorithms while the Functional Mockup Interface was being created. The prototypical master tool coupled slaves written in C via a provisional interface which possessed the main functionality of FMI 1.0.

Recently, this "EAS master tool" was improved by supporting both the FMI 1.0 and FMI 2.0 for co-simulation. Furthermore, a graphical user interface for convenient handling was added. Currently, there are efforts to unify the interface of controlling master algorithms via an XML-File. These efforts are also supported in a preliminary way. This improved EAS master opens the opportunity to thoroughly test its master algorithms, since examples can be modeled easily using Modelica, and exported as FMU for co-simulation with commercial simulation tools (e.g., Dymola, SimulationX). These FMUs can be coupled via the EAS master tool.

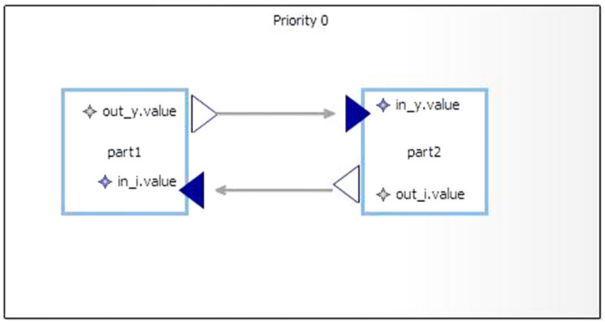

Figure 4. Graph window of the master GUI

The EAS master tool is controlled by a configuration text file with the following structure:

```
nval            2
nsim            2
tstart          0.0
tend            10.0
tstepmax        0.001
tstepstart      0.001
MasterMode      2
MasterDebug     2
OutputGnuplot   1
it_max_steps    100
it_tol_abs      1.0E-6
it_tol_rel      1.0E-4
simulator   0   fmus/part1.fmu
simulator   1   fmus/part2.fmu
graph #val #sim -1(out)/1(in) valueref
0   0   -1      r part1.out_y.value
0   1    1      r part2.in_y.value
1   1   -1      r part2.out_i.value
1   0    1      r part1.in_i.value
end
priority #sim priority
0   0
1   0
end
cycles  #prior    0(no)/1(yes)
0   1
end
```

The configuration file contains the number of coupling variables (nval), the number of FMUs (nsim), the simulation time interval ([tstart, tend]), the communication step size (tstepmax, tstepstart), the choosen master algorithm (mastermode), some numerical parameters, the paths to the FMUs, the connection graph, and information on directions (priority) and cycles in the graph. The graphical user interface generates the configuration file.

The "EAS master tool" comprises the following algorithmic approaches:

- Constant communication step size
  It is user-defined before simulation. Though variable communication step size basing on Richardson extrapolation was investigated successfully (Schierz et al, 2012; Schierz, 2013) it is not yet integrated into the tool.
- Sequence of calling the simulators
  The sequence $r = (r_1, \ldots, r_m)$ of calling the simulators is not yet automatically generated. It is to provide by the user.
- Gauss-Seidel method (3)
  It is applied to each communication interval. This method requires the FMUs to be able to repeat the simulation of communication intervals (repeated doStep calls).
- Gauss-Seidel method (3) with one iteration step (GS1)
  This simplified algorithm comes to a result in any case. But if there are cyclic dependencies the result may be no solution. Provided that the dependency sequence *r* is correct this method finds a solution if there are no cyclic dependencies.
- Newton-Raphson method (5)
  This method requires repetitions of simulating communication intervals (repeated doStep calls over the same communication interval). The Jacobian is calculated applying difference quotients which needs additional simulations. Jacobians delivered by the FMU are not yet evaluated.
- Directed graph with included cycles
  The dependencies between the simulators form a graph. The tool supports a unidirectional graph with included cycles. The iterating methods specified above can be restricted to the cycles, whereas GS1 is applied to the non-cyclic parts generally.

The "EAS master tool" solely interprets the information on the slave simulators given by FMI. Further information on the solution method used within the FMU is not exploited.

5 Application Test Examples

The „EAS master tool" was applied to a lot of small test examples which address more or less different difficulties or aspects of co-simulation. Each example was first modeled using Modelica tools (Dymola, SimulationX) without partitioning to generate the reference solution. Second, the example was split, and each part was exported as an FMU. Using the master tool the FMUs were coupled, and the example was simulated using the three above mentioned basic algorithms.

Table 1 gives an impression of the behavior of the three basic algorithms. The following subsections present four of the examples (row 7, 10, 5, and 23 in Table 1) in more detail. Finally, the last subsection collects some recommendations for succeeding co-simulations.

Table 1. Algorithm test results

	Addressed Purpose	*Cyc*	*FMU*	*GS1*	*GSM*	*NRM*
1	straightforward system with correct calling sequence, no difficulties	0	7	Y	Y	Y
2	linear system, diagonally dominant matrix, iterations needed	1	5	d	Y	Y
3	digital and analog variables, events	1	3	d	d	d

DOI 10.3384/ecp15118865

4	like 2, but not diagonally dominant	1	5	w	w	Y
5	like 2, time dependent matrix, which loses diagonal dominance, see section 5.3	1	5	w	w	Y
6	precision test, “too large” communication step size	0	3	d	d	d
7	precision test, iterations needed, see section 5.1	1	3	d	Y	Y
8	precision test, iterations needed	1	3	d	Y	Y
9	nonlinear equation, iteration needed,	1	3	w	Y	Y
10	changing signal flow due to varying resistors, but fixed directions for FMI, see section 5.2	1	3	w	e	Y
11	like 10, but changed fixed directions	1	3	w	e	Y
12	like 10, further changed fixed directions	1	3	d	Y	Y
13	like 10, further changed fixed directions	1	3	w	e	Y
14	extended circuit with changing signal flow, but fixed directions for FMI	1	6	d	Y	Y
15	like 10, jumping input variables, events should be hit	1	3	w	e	Y
16	Rossler DAE, large simulation interval, iterations needed	1	3	w	d	d
17	retarding DAE	0	2	Y	Y	Y
18	like 5, other time dependent matrix, which loses diagonal dominance	1	5	w	w	Y
19	linear equation, not contractive matrix	0	2	d	d	Y
20	like 7, extended, test of initialization	1	4	e	e	e
21	higher index problem due to wrong signal flow direction; correct, that no FMU exportable	1	2	e	e	e
22	like 21, suitable direction	1	2	Y	Y	Y
23	“nearly” higher index problem, like 21, see section 5.4	1	2	d	d	Y

Legend: Cyc – number of cycles, FMU – number of FMUs, GS1 – Gauss-Seidel method with one iteration, GSM – iterating Gauss-Seidel method, NRM – Newton-Raphson method, Y – correct, d – small differences compared with unpartitioned reference solution, w – completely wrong result, e – error, or simulation fails, or FMU not exportable. The difference between “d” and “w” is estimated subjectively to give a rough impression whether the calculated solution is “far away” or “near” the correct solution.

5.1 Precision Test Example

The equations according to Table 2 are segmented such that the equations of each row in the table are simulated with their own simulator within an FMU. The columns “In” and “Out” describe the coupling variables.

Table 2. Equations of the precision test example

In	*Equations*	*Out*
x_2	$x_1 = -x_2$	x_1
x_1	$\frac{\partial x_2}{\partial t} = x_1, \quad x_2(0) = 1$	x_2
x_2	$e^{-t} - x_2 = y$	y

This example (row 7 in Table 1) is designed such that $y(t)$ is zero. Therefore, the magnitude of y is an indicator of precision of the numerical solution. All three implemented methods calculate the correct result. Both GSM and NRM are similar precise (Figure 5), but GS1 is more inaccurate (Figure 6). The communication step size was 0.1.

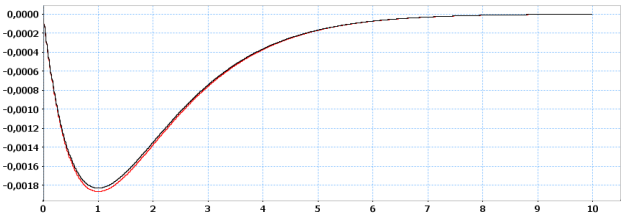

Figure 5. *y(t)* calculated by GSM and NRM

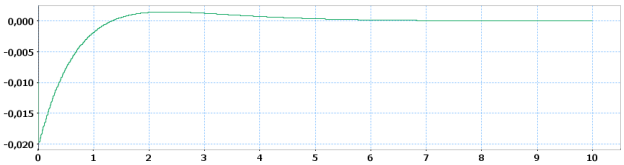

Figure 6. *y(t)* calculated by GS1

5.2 Varying Resistors

This example (Figure 7, row 10 of Table 1) from the electronic domain is designed such that the voltage vi alternately depends on $x1$ or $x2$. The reason for this behavior is the variation of the resistances of *var1* and *var2* (Figure 8).

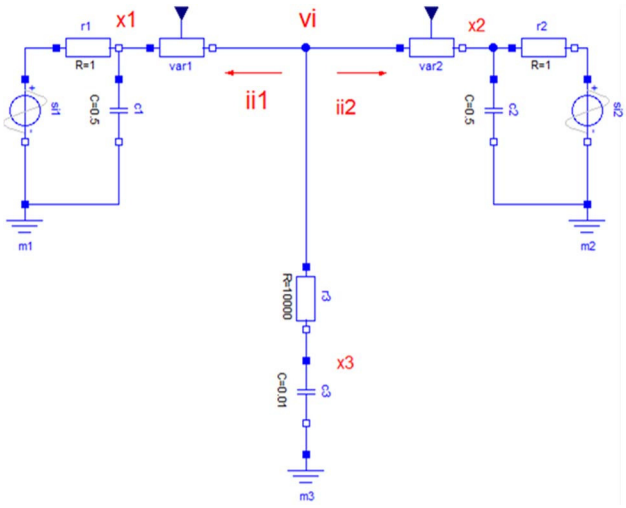

Figure 7. Circuit with varying resistors

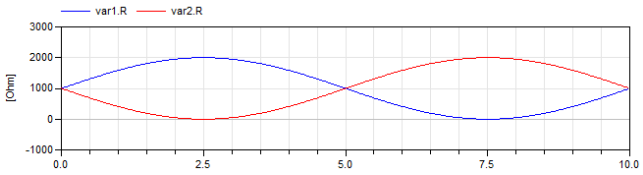

Figure 8. Varying resistances

Similar to Table 2 the equations of this example are presented in Table 3.

Table 3. Equations of the varying resistors example

In	*Equations*	*Out*
vi	$0.5(\partial x_1/\partial t) + x_1 - ii_1 - \sin(3\pi t) = 0$ $(vi - x_1)/(1000\sin(0.2\pi t) + 1001) = ii_1$	ii_1
vi	$0.5(\partial x_2/\partial t) + x_2 - ii_2 - \sin(2\pi t) = 0$ $(vi - x_2)/(-1000\sin(0.2\pi t) + 1001) = ii_2$	ii_2
ii_1, ii_2	$0.0001(x_3 - vi) + 0.01(\partial x_3/\partial t) = 0$ $ii_1 + ii_2 - 0.0001(x_3 - vi) = 0$	vi

Because of the varying resistances no unique exchange direction of coupling variables is possible. Therefore, both GSM and GS1 do not converge. Only NRM calculates a correct result (Figure 9).The trajectory is not quite smooth due to a large communication step size of 0.1.

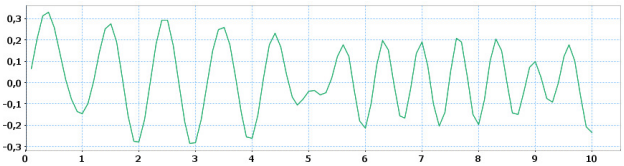

Figure 9. Coupling variable *vi(t)*

5.3 Linear System of Equations

In the following linear system of equations (row 5 of Table 1) the matrix varys depending on the time. Therefore, a similar behaviour like in Section 5.2 can be observed.

Table 4. Linear system of equations

In	*Equations*	*Out*
	$r_1 = 1, r_2 = t, r_3 = 1$	r_1, r_3, r_3
x_2, x_3, r_1	$3x_1 + (0.1 + t)x_2 + 0.2x_3 = r_1$	x_1
x_1, x_3, r_2	$0.1x_1 + 3x_2 + (0.1 + t)x_3 = r_2$	x_2
x_1, x_2, r_3	$(0.1 + t)x_1 + 0.2x_2 + 4x_3 = r_3$	x_3
x_1, x_2, x_3	$x_1 + x_2 + x_3 = y$	y

Five FMUs are coupled according to Table 4. Both GSM and GS1 do not converge, since due to the time dependence the fixed point iteration is not contractive for increasing t. The NRM calculates the correct result (Figure 10).

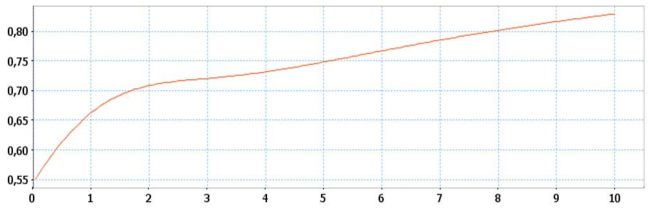

Figure 10. Result variable *y(t)*

5.4 Resistor-Capacitor-Circuit

This example (row 23 in Table 1) from the electric domain is a simple resistor-capacitor-circuit where the resistor is divided into two parts. The equations are allocated to two FMUs according to Table 4.

Table 4. Equations of the Resistor-Capacitor-Example

In	*Equations*	*Out*
i	$i(1 - R_{Part2}) = y - u$ $u = if\ t < 2\ then\ 0\ else\ if\ 4 \leq t\ then\ 2\ else\ 4$	y
y	$i \cdot R_{Part2} = x - y,$ $-i = \partial x/dt,\ x(0) = 2$	i

The DAE index of the second FMU is one. But the smaller R_{Part2} becomes the “closer” the index gets to two, which occurs if R_{Part2} is zero. Therefore, difficulties in the coupled simulation arises, if R_{Part2} is small. The corresponding result with $R_{Part2} = 0.001$ obtained by the generating tool without partitioning is shown in Figure 11.

DOI 10.3384/ecp15118865

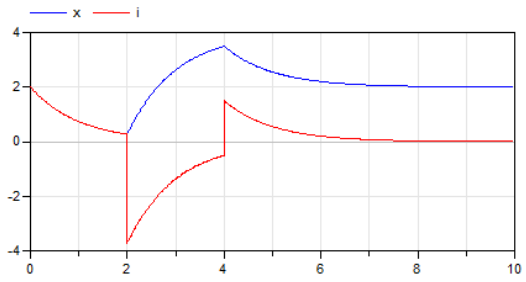

Figure 11. Reference solution Resistor-Capacitor-Circuit

For this example GSM fails. GS1 produces an extremely increasing result. The result of Newton's method is not quite exact due to a large communication step size of 0.1 (Figure 12). If the step size is reduced to 0.001 the reference values are met, c.f. Figure 13. There are further investigations necessary to identify the influence of a "nearly" high index to properties of the coupled simulation.

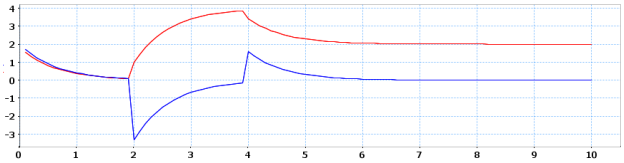

Figure 12. Result using NRM, step size 0.1

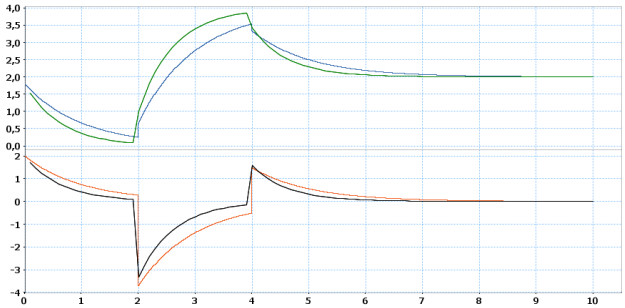

Figure 13. Comparison NRM, step sizes 0.1 and 0.001, separated into two diagrams, *x(t)* above, *i(t)* below.

5.5 Recommendations

The experience collected in checking test examples so far can be summarized in the following recommendations.

Essential for successful co-simulation is an intelligent tearing:

- Closely interacting parts should not be separated. Tearing is recommended at points where a signal flow direction is clearly recognized. The coupling variable should be output variable at that FMU which calculates it "significantly". If the propagation direction of a variable changes during simulation it should not become a coupling variable.
- An output coupling variable of an FMU should have no influence to the input variables of the same FMU. At least such reactions should be delayed.
- Sometimes it is essential to transfer both the value and the derivative(s) of coupling variables. This should be checked at each coupling variable, e.g. using test simulations.

Often these recommendations regarding tearing cannot be followed, since FMUs for components may be predetermined by specialized simulation tools. In such cases it is advantageous to revise the definition of interfaces.

Furthermore, a suitable choice of master algorithm parameters is important:

- The communication step size should be small because of numerical reasons, and large because of performance. The estimation of time constants, and test simulations can help to choose a reasonable step size. If time events (changing of discrete variables) are known they should coincidence with end points of communication intervals.
- Cycles in the connection graph should be handled using a small communication step size.
- If the GSM diverges both the tearing and the definition of the direction of coupling variables should be checked.

The following points are necessary to improve the master algorithms of the test tool:

- Both the calling sequence of the FMUs and cycles in the connection graph should be defined automatically.
- It should be possible to apply different algorithms to cycles. Sometimes an algorithm should be changed during simulation.
- Variable communication step size should be introduced.
- More algorithmic parameters for the coupling methods should be introduce to adapt algorithms to given co-simulation tasks. Otherwise such adaptions should be done as automatically as possible to unburden the user.

6 Conclusion

For co-simulation of two or more FMUs three basic algorithms were described. These obvious algorithms can be a starting point for developing further coupling algorithms. Some ideas are presented.

It has to be further examined which other existing coupling algorithms can be described with the notation introduced in chapter 3, e.g. the asynchronous method (Petridis et al, 2008; Petridis, 2013). Additionally it has to be checked if other algorithms can be implemented with the existing FMI standard.

The algorithms were implemented in a master tool for testing. This master tool supports FMI 1.0 as well as FMI 2.0. It is desirable to implement further and modified algorithms which can be optimal adapted to given simulation tasks.

Many small coupling examples were developed which address special issues. These examples should be extended in future. The examples show the different behavior of the basic algorithms. The Newton-Raphson method turns out to be one of the most powerful algorithms, but its performance is usually bad. Therefore, improved algorithms should be developed.

Generally, the master tool should become more „intelligent“. It is to investigate whether internal information (the method used, the actual internal step size, numerical properties, …) of the slave simulation within an FMU should be transferred to the master. This could help the master algorithms to be adapted better. Otherwise, a master should be capable of acting without any FMU-internal information. Both directions seem to be needed.

The presented examples are a good starting point for the FMI cross checking, because until now only single FMU are tested. With the presented examples in combination with the FMI Test Library (Otter, 2014) the coupling and with this the capability of master algorithms can be tested.

Acknowledgements

The authors are much obliged to Prof. Martin Arnold, Halle, as well as to Dr. Tom Schierz, Gilching, for any cooperation.

References

Jens Bastian, Christoph Clauss, Susann Wolf, Peter Schneider. Master for CoSimulation Using FMI. 8th International Modelica Conference, Dresden, March 20-22, 2011.

Christian Bertsch, Elmar Ahle, Ulrich Schulmeister. The Functional Mockup Interface – seen from an industrial perspective. 10th International Modelica Conference, March 10-12, Lund, Sweden, pp. 27-31, 2014.

FMI project website, https://www.fmi-standard.org/

Martin Otter. Modelica FMI Test Library. In: Tutorial: Functional Mockup Interface 2.0 and HiL Applications of the International Modelica Conference, Lund, Sweden, 2014

Kosmas Petridis. Synchrone und asynchrone Verfahren zur gekoppelten Simulation mechatronischer Systeme. VDI Verlag, 2013.

Kosmas Petridis, Andreas Klein, Michael Beitelschmidt. Asynchronous method for the coupled simulation of mechatronic systems. In: PAMM Volume 8 (2008) Nr. 1

Tom Schierz. Modulare Zeitintegration gekoppelter Differentialgleichungssysteme in der technischen Simulation. Fortschr.-Ber. VDI Reihe 20 Nr. 447. Düsseldorf: VDI Verlag 2013.

Tom Schierz, Martin Arnold and Christoph Clauß. Co-simulation with communication step size control in an FMI compatible master algorithm. In: Proceedings of the 9th International Modelica Conference, Munich, Germany, 2012.

Hubert Schwetlick. Numerische Lösungen nichtlinearer Gleichungen. Deutscher Verlag der Wissenschaften, Berlin, 1979, und R. Oldenbourg Verlag München, Wien, 1979.