Coupling Model Exchange FMUs for Aggregated Simulation by Open Source Tools

Pukashawar Pannu[1] Christian Andersson[1,2] Claus Führer[1] Johan Åkesson[2]

[1]Centre for Mathematical Sciences, Lund University, Sweden
[2]Modelon AB, Sweden

Abstract

The Functional Mock-up Interface standard allows to generate stand-alone sub-systems which can be simulated and verified individually. In this paper we present a design of a model aggregation which allows to simulate several Functional Mock-up Units as a coupled model. The formulation is based on Assimulo as a numerical integration environment. Assimulo problem classes are extended to a class for aggregated problems which collects information provided by the Functional Mock-up Units through the tool PyFMI together with Python based problem classes defined by Assimulo. This allows to set-up test environments of complex models composed of several sub-systems.

Keywords: FMI, Jacobian, Algebraic loops, Events, Model Exchange 2.0, Assimulo

1 Introduction

The Functional Mock-up Interface (FMI) (Blochwitz et al., 2012) has gained momentum in simulation of dynamical systems and in exchanging dynamic simulation models between tools. The standard has proven to be highly successful as it fills a gap where there were costly custom integrations before. The open source tools PyFMI [1] together with Assimulo (Andersson et al., 2015) provide a solid foundation for performing simulations and experiments on single Functional Mock-up Units (FMUs).

A key feature that is currently lacking is the ability to easily simulate coupled systems and thus fully taking advantage of the standard.

In this article, an extension to the open-source tools PyFMI and Assimulo is presented that allows for simulation of coupled model exchange FMUs following the FMI 2.0 standard. The extension enables coupling of FMUs and models written directly in Python to a so-called aggregated model.

[1] `http://www.pyfmi.org` PyFMI - Version 2.1. Accessed, 2015-05-18

The dynamical models considered here can be described as,

$$\dot{\bar{x}} = \bar{f}(\bar{x}, \bar{u}) \tag{1a}$$
$$\bar{y} = \bar{g}(\bar{x}, \bar{u}) \tag{1b}$$

where $\bar{x}$ represents the states, $\bar{u}$ the input signal and $\bar{y}$ the output, consistent with the FMI.

Commonly, a full system model is represented by several stand-alone sub-systems coupled together by coupling equations to a model for a global system. This results in the following general system description,

$$\dot{x} = f(x, u, w) \tag{2a}$$
$$y = g(x, u, w) \tag{2b}$$
$$u = c(y, w) \tag{2c}$$

where x represents the combined states from the separate models. The local inputs for the ith model, $\bar{u}^{[i]}$, has here been separated into two vectors, $\bar{u}^{[i]} = [\hat{u}^{[i]}, \hat{w}^{[i]}]$, and subsequently combined into the global vectors (for N models), $u = [\hat{u}^{[1]}, \ldots, \hat{u}^{[N]}]$ and $w = [\hat{w}^{[1]}, \ldots, \hat{w}^{[N]}]$. This as to separate between inputs determined by the coupling, u, and external inputs acting on the coupled system, w. In general the external inputs can not only influence the model behaviour directly but also the coupling, Eq (2c), which is highlighted in Section 3.

When solving a coupled system, an approach is co-simulation as is explored in (Andersson, 2013) where the systems have their own integrator and the focus is on communication between systems. In this paper however, the focus is on coupling model exchange FMUs under a single solver.

2 Concept

The idea is to take N coupled sub-systems, either FMUs or Python models, and aggregate them into a single system and treating the final full system as any other model. In order to facilitate the general description of a sub-system, as is defined in FMI, for the aggregated system,

care needs to be considered in how for example the Jacobian is defined. The Jacobian is a necessity when using implicit methods for solving the resulting system and is discussed in Section 2.3. Additionally, the events for each sub-system and external events need to be considered, discussed in Section 2.2, as well as algebraic loops which can occur due to the coupling, Section 2.4.

Now, looking at N sub-systems,

$$\dot{\bar{x}}_1^{[1]} = \bar{f}_1^{[1]}(\bar{x}_1^{[1]}, \hat{u}_1^{[1]}, \hat{w}_1^{[1]}) \tag{3a}$$
$$\bar{y}_1^{[1]} = \bar{g}_1^{[1]}(\bar{x}_1^{[1]}, \hat{u}_1^{[1]}, \hat{w}_1^{[1]}) \tag{3b}$$
$$\bar{u}_1^{[1]} = \bar{c}_1^{[1]}(\bar{y}^{[1]}, \hat{w}_1^{[1]}) \tag{3c}$$
$$\vdots$$
$$\dot{\bar{x}}_N^{[N]} = \bar{f}_N^{[N]}(\bar{x}_N^{[N]}, \hat{u}_N^{[N]}, \hat{w}_N^{[N]}) \tag{4a}$$
$$\bar{y}_N^{[N]} = \bar{g}_N^{[N]}(\bar{x}_N^{[N]}, \hat{u}_N^{[N]}, \hat{w}_N^{[N]}) \tag{4b}$$
$$\bar{u}_N^{[N]} = \bar{c}_N^{[N]}(\bar{y}^{[N]}, \hat{w}_N^{[N]}) \tag{4c}$$

and the resulting aggregated system,

$$\dot{x} = f(x,u,w) = \begin{bmatrix} \bar{f}_1^{[1]}(\bar{x}_1^{[1]}, \hat{u}_1^{[1]}, \hat{w}_1^{[1]}) \\ \vdots \\ \bar{f}_1^{[N]}(\bar{x}_1^{[N]}, \hat{u}_1^{[N]}, \hat{w}_1^{[N]}) \end{bmatrix} \tag{5a}$$
$$y = g(x,u,w) = \begin{bmatrix} \bar{g}_N^{[1]}(\bar{x}_N^{[1]}, \hat{u}_N^{[1]}, \hat{w}_N^{[1]}) \\ \vdots \\ \bar{g}_N^{[N]}(\bar{x}_N^{[N]}, \hat{u}_N^{[N]}, \hat{w}_N^{[N]}) \end{bmatrix} \tag{5b}$$
$$u = c(y,w) = \begin{bmatrix} \bar{c}_1^{[1]}(\bar{y}^{[1]}, \hat{w}_1^{[1]}) \\ \vdots \\ \bar{c}_N^{[N]}(\bar{y}^{[N]}, \hat{w}_N^{[N]}) \end{bmatrix} \tag{5c}$$

The vectors x, y, u and w of the aggregated system are defined as:

$$x = \begin{bmatrix} \bar{x}_1^{[1]} \\ \vdots \\ \bar{x}_N^{[N]} \end{bmatrix}, \quad y = \begin{bmatrix} \bar{y}_1^{[1]} \\ \vdots \\ \bar{y}_N^{[N]} \end{bmatrix}, \quad u = \begin{bmatrix} \hat{u}_1^{[1]} \\ \vdots \\ \hat{u}_n^{[N]} \end{bmatrix}, w \quad = \begin{bmatrix} \hat{w}_1^{[1]} \\ \vdots \\ \hat{w}_N^{[N]} \end{bmatrix}$$

2.1 Aggregated Problem

Using the open-source tools PyFMI together with Assimulo, an FMU can be accessed from Python together with being solved using solvers available in Assimulo. With this in mind two Assimulo problem classes have been worked on. One that creates an input/output problem structure called `ExplicitProblemModel`. The other aggregates several FMUs, or `ExplicitProblemModels`, to one large problem that can be integrated using one of Assimulos available solvers, called `AggregatedProblem`. For simplicity an already existing problem class, `ExplicitProblem`, was extended to handle the aggregation. To define an aggregated problem class some basic data is required:

- Aggregated states.
- RHS (Right-Hand-Side) function of aggregation.
- Coupling handling.

Through PyFMI there exists already a wrapper interface that can load an FMU ME 2.0 and integrate it using Assimulo. When instantiating the `AggregatedProblem` class a list of FMUs is provided from which the initial states are easily accessible and aggregated,

```
for model in models:
    aggregated_x0 aggregate model.x0
```

The crucial part of the aggregated problem class is how to handle the right hand side function. The first major difference between an aggregated problem and an Assimulo problem is the presence of couplings. For each call to the RHS, coupling terms must be up to date. The condition can be satisfied by updating the coupling relations within the RHS-function.

Since the separate problem classes already have an RHS-function structure, computing the RHS-function of the aggregated system is simply to call the RHS-function of each sub-system,

```
set_connections()
for model in models:
    aggregated_rhs aggregate model.rhs
```

Coupling handling is done in *set_connections*(). For simple cases when for example system A input $u_2^{[A]}$ needs inputs from system B output $y_4^{[B]}$, the function simply sets $u_2^{[A]} = y_4^{[B]}$. However, this is not always the case which is further discussed in Section 2.4.

For implicit solvers a Jacobian is required and must be provided by `AggregatedProblem`. More advanced models require `AggregatedProblem` to take into account events and algebraic loops. The three mentioned topics are affected by aggregation and are discussed in the following sections.

2.2 Events

Many models include discontinuities. One way of integrating such systems is by using events (Eich-Soelner and Führer, 1998) which requires that a set of event indicators are monitored during the integration. The integration is interrupted when conditions on the event indi-

DOI 10.3384/ecp15118903

cators are violated and the event (discontinuity) is appropriately handled, finally the integration is restarted. Most of the solvers available in Assimulo have this functionality (Fredriksson et al., 2014).

The aggregated problem can access event indicators of an FMU. When asked for event indicators by an Assimulo solver the `AggregatedProblem` combines all event indicators from the sub-systems and hands them to the solver. Once an event has been detected and the integration stopped, the problem class identifies the triggered event and calls the corresponding sub-system's event handling.

Another type of events in FMUs are time events, which are known at the start of a simulation. They split up the integration into segments by setting up end times for the simulation at which point an event is handled (Andersson, 2013). This could be, for instance, a force periodically applied to the system. It is up to the aggregated system to search through all sub-systems for the closest time event to define the end time of the next integration segment and handle the event.

From the Assimulo problem design it is simple to add events to a problem. For the aggregated problem, adding of events would be to add external events to a coupled system. Events can not only be provided through the sub-systems but also through how the system is coupled. Consider a pendulum with no knowledge of its surroundings. Now, in the system model the pendulum is positioned such that its degree of freedom is limited by for instance positioning close to a wall. The limitation can be considered as an external event that needs to be taken into account. In the problem formulation this is easily done by providing extra sets of event indicators for the integrator to monitor.

2.3 Jacobian

When solving an ODE the Jacobian can be explicitly provided or numerically approximated. For an uncoupled input/output system where the inputs are only time dependent the Jacobian, $\frac{\partial \bar{f}}{\partial \bar{x}}$, is computed. When looking at a coupled system the dynamic changes. Due to coupling some input terms are state dependent instead of time dependent as in the uncoupled case. Consider the coupled system,

$$\dot{x} = f(x,u,w) \tag{6a}$$
$$y = g(x,u,w) \tag{6b}$$
$$u = c(y,w) \tag{6c}$$

Inserting Eq (6c) into Eq (6a) and Eq (6b) gives:

$$\dot{x} = f(x,c(y),w) \tag{7a}$$
$$y = g(x,c(y),w) \tag{7b}$$

Differentiating Eq (7a) with respect to x yields:

$$J = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial c}\frac{\partial c}{\partial y}\frac{\partial y}{\partial x} \tag{8}$$

The term $\frac{\partial y}{\partial x}$ is found by differentiating Eq (7b):

$$\frac{\partial y}{\partial x} = \frac{\partial g}{\partial x} + \frac{\partial g}{\partial c}\frac{\partial c}{\partial y}\frac{\partial y}{\partial x} \tag{9}$$

Solving for $\frac{\partial y}{\partial x}$ gives:

$$\frac{\partial y}{\partial x} = \left(I - \frac{\partial g}{\partial c}\frac{\partial c}{\partial y}\right)^{-1}\frac{\partial g}{\partial x} \tag{10}$$

Resulting in the Jacobian:

$$J = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial c}\frac{\partial c}{\partial y}\left(I - \frac{\partial g}{\partial c}\frac{\partial c}{\partial y}\right)^{-1}\frac{\partial g}{\partial x} \tag{11}$$

For the system to be solvable there is necessary condition that $(I - \frac{\partial g}{\partial c}\frac{\partial c}{\partial y})$ is non singular. The $\frac{\partial g}{\partial c}$ term handles the coupling relations and $\frac{\partial c}{\partial y}$ the sub-system feed-through terms.

With FMI 2.0 models have an option to provide directional derivatives. In case they are provided `AggregatedProblem` uses directional derivatives to approximate the aggregated Jacobian matrix. If directional derivatives are unavailable a forward difference scheme is applied. The same applies for non-FMI models.

2.4 Algebraic Loops

When a system contains feed-through, i.e. when the partial derivative of Eq (6b) with respect to u is not the zero matrix, then, in general, an equation system needs to be solved to maintain consistent input and output values satisfying,

$$y = g(x,u,w) \tag{12a}$$
$$u = c(y,w). \tag{12b}$$

By rewriting Eq (12a) to,

$$y - g(x,c(y,w),w) = 0 \tag{13}$$

the algebraic loop can be solved by an iterative method. `AggregatedProblem` creates a residual function of the left-hand-side of Eq (13) and uses the `Kinsol` solver in Assimulo to solve the problem. `Kinsol` is a nonlinear algebraic equation solver, part of the `SUNDIALS` suite (Hindmarsh et al., 2005). When the outputs are known, Eq (12b) is used to update the inputs.

2.5 Workflow

The simulation flow of coupled systems using the aggregated problem class and an Assimulo solver is illustrated in Figure 1. The simulation flow is essentially equivalent to that of simulating an ODE with Assimulo, however, some nodes are affected by aggregation and these are coloured blue in the figure.

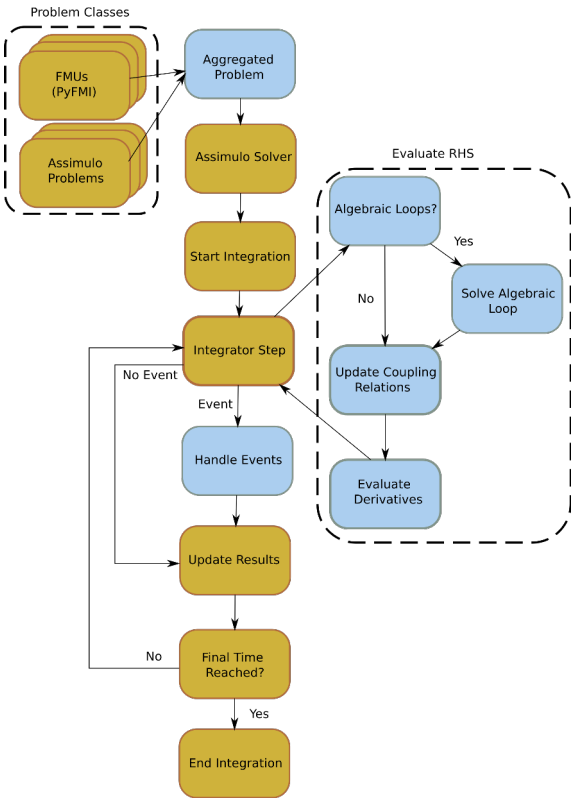

Figure 1. Assimulo simulation flow of coupled systems using FMUs, Assimulo problems and the aggregated problem class. The blue color represents nodes that are affected by aggregation.

3 Examples

In this section, the proposed framework is demonstrated. The ability to couple model exchange FMUs is shown together with coupling of FMUs with models directly defined in Python. Additionally, simulation of coupled models with externally defined events is demonstrated.

3.1 Coupled Pendula

This example demonstrates how two FMUs, each describing a pendulum, are coupled to an aggregated model. The full system consists of two pendula coupled by a force and excited by two inputs acting on the pivots.

The pendulum, with mass 1 kg and length 1 m, is described by,

$$\dot{\bar{x}}_1 = \bar{x}_3 \tag{14a}$$
$$\dot{\bar{x}}_2 = \bar{x}_4 \tag{14b}$$
$$\dot{\bar{x}}_3 = \bar{u}_1 - 2\bar{x}_1\lambda + \bar{u}_2 \tag{14c}$$
$$\dot{\bar{x}}_4 = -g - 2\bar{x}_2\lambda + \bar{u}_3 \tag{14d}$$
$$0 = \bar{x}_1^2 + \bar{x}_2^2 - 1 \tag{14e}$$
$$\bar{y}_1 = \bar{x}_1 \tag{14f}$$
$$\bar{y}_2 = \bar{x}_2 \tag{14g}$$

where $\bar{x}_1, \bar{x}_2$ are positions and $\bar{x}_3, \bar{x}_4$ velocities relative to the pendulum's pivot. The inputs are forces, $\bar{u}_2$ and $\bar{u}_3$, acting on the body's center and an acceleration, $\bar{u}_1$ due to a forced motion of the pivot. The outputs, $\bar{y}$, are the positions.

In order to couple two pendula, $i = [1,2]$, the input vector is split for each pendulum into external excitations and inputs determined by the coupling,

$$\bar{u}^{[i]} = [\underbrace{\bar{u}_1^{[i]}}_{\hat{w}^{[i]}}, \underbrace{\bar{u}_2^{[i]}, \bar{u}_3^{[i]}}_{\hat{u}^{[i]}}]. \tag{15}$$

The two pendula are coupled by a linear spring which is determined by the equation, $u = c(y,w)$,

$$\begin{bmatrix} \hat{u}_2^{[1]} \\ \hat{u}_3^{[1]} \\ \hat{u}_2^{[2]} \\ \hat{u}_3^{[2]} \end{bmatrix} = k \underbrace{\begin{bmatrix} \bar{y}_1^{[1]} - a + w_1 - (\bar{y}_1^{[2]} - b - w_2) \\ \bar{y}_2^{[1]} - \bar{y}_2^{[2]} \\ -(\bar{y}_1^{[1]} - a + w_1 - (\bar{y}_1^{[2]} - b - w_2)) \\ -(\bar{y}_2^{[1]} - \bar{y}_2^{[2]}) \end{bmatrix}}_{=:\rho} \tag{16}$$

where k is the stiffness ratio. Variable a represents the pivot points x-coordinate of the left pendulum and b the point of the right pendulum. The external input vector is,

$$w = [w_1, w_2, \hat{w}^{[1]}, \hat{w}^{[2]}]. \tag{17}$$

The setup is shown in Figure 2. As previously mentioned, it is necessary to include the external inputs into the coupling as is made evident in this example. Note also, that in this example $\hat{w}^{[i]}$ has to be chosen as $\ddot{w}^{[i]}$.

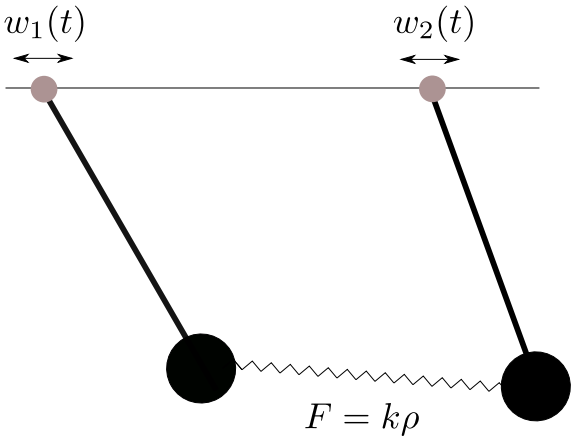

Figure 2. Two pendulums coupled via a spring.

The pendulum is modelled in the Modelica language and using the open-source tool JModelica.org (Åkesson et al., 2010) the Modelica model is compiled into an FMU. The tool is responsible for transforming the pendulum which is described as a DAE of index 3 into an ODE that FMI supports.

The aggregated system was integrated using Assimulo `CVode` solver with tolerances atol = rtol = 10^{-8} for 5

DOI 10.3384/ecp15118903

seconds and the Jacobian approximated with forward differences. Initital conditions for the system,

$$\bar{x}_1^{[1]} = 1 \tag{18}$$
$$\bar{x}_2^{[1]} = 0 \tag{19}$$
$$\bar{x}_1^{[2]} = -1 \tag{20}$$
$$\bar{x}_2^{[2]} = 0 \tag{21}$$

note that the initial conditions are from the reference point of each pendulums pivot. The pivot points are located at $(-2,0)$ for the left pendulum and $(2,0)$ for the pendulum to the right. As external forces acting on the pivots the $sin(t)$ function was chosen. Stiffness ratio of the spring is set to $k = 1.0$ N/m.

As reference a monolithic model of the system was created in Modelica and simulated in Dymola (Dassault Systèmes, 2016) using the solver `Dassl` with tolerance tol $= 10^{-12}$. Error of both pendulums x, y positions is presented in Figure 3 in log-scale.

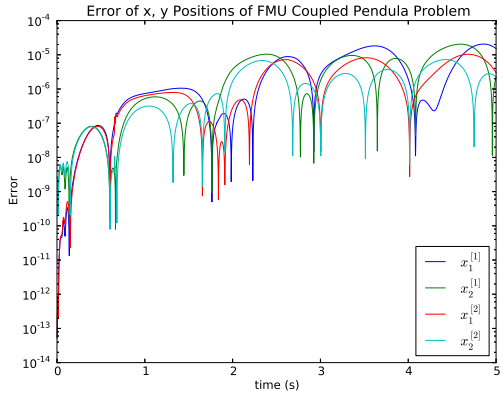

Figure 3. Error of x, y positions of aggregated coupled pendulum described in Section 3.1, simulated with CVode with atol $=$ rtol $= 10^{-8}$ for 5 seconds. $x_1^{[i]}$ denotes the x-coordinate and $x_2^{[i]}$ the y of model i. Model [1] is the pendulum to the left and [2] the one to the right.

3.2 Coupled Pendula with Different Model Types

Three aggregated systems of coupled pendulas were modelled and compared to a monolithic reference model. The first system built by two FMU models modelled in Modelica and compiled with JModelica.org. The second by two Assimulo models and the third with the left pendulum as an Assimulo model and the right pendulum as an FMU. For this example the pendulums were modelled as ODEs in polar coordinates with unit mass and length,

$$\dot{\bar{x}}_1 = \bar{x}_2 \tag{22a}$$
$$\dot{\bar{x}}_2 = (-g + \bar{u}_3)sin(\bar{x}_1) + (\bar{u}_2 + \bar{u}_1)cos(\bar{x}_1) \tag{22b}$$
$$\bar{y}_1 = \bar{x}_1 \tag{22c}$$

where g is gravitational acceleration, $\bar{x}_1$ is angular displacement with respect to the pivot point, $\bar{x}_2$ angular velocity. The inputs $\bar{u}_2$ and $\bar{u}_3$ are forces acting on the bob horizontally and vertically respectively. $\bar{u}_1$ is an input of acceleration due to a forced motion of the pivot. The output, $\bar{y}_1$, is the angular displacement.

Similarly to the example described in Section 3.1 the input vector is split into external excitations and inputs by coupling.

$$\bar{u}^{[i]} = [\underbrace{\bar{u}_1^{[i]}}_{\hat{w}^{[i]}}, \underbrace{\bar{u}_2^{[i]}, \bar{u}_3^{[i]}}_{\hat{u}^{[i]}}]. \tag{23}$$

The linear spring coupling the two pendulas is determined by,

$$\begin{bmatrix} \hat{u}_2^{[1]} \\ \hat{u}_3^{[1]} \\ \hat{u}_2^{[2]} \\ \hat{u}_3^{[2]} \end{bmatrix} = k \underbrace{\begin{bmatrix} (sin(\bar{y}_1^{[1]}) - a + w_1) - (sin(\bar{y}_1^{[2]}) - b - w_2) \\ (-cos(\bar{y}_1^{[1]}) - (-cos(\bar{y}_1^{[2]})) \\ -((sin(\bar{y}_1^{[1]}) - a - w_1) - (sin(\bar{y}_1^{[2]}) - b - w_2)) \\ -((-cos(\bar{y}_1^{[1]}) - (-cos(\bar{y}_1^{[2]}))) \end{bmatrix}}_{=:\rho} \tag{24}$$

where k is the stiffness ratio. Variables a and b represent the pivot points x-coordinate for the left-hand-side and right-hand-side pendulas respectively. The external input vector is,

$$u_1 = [w_1, w_2, \hat{w}^{[1]}, \hat{w^{[2]}}]. \tag{25}$$

As with example in Section 3.1, $\hat{w}^{[i]}$ has to be chosen as $\ddot{w}^{[i]}$.

Initial conditions for the aggregated system were chosen for the pendulas to mirror each other with angles $\frac{\pi}{2}$ and $-\frac{\pi}{2}$ for the left and right pendulas and zero initial angular velocity.

$$\bar{x}_1^{[1]} = \frac{\pi}{2} \tag{26a}$$
$$\bar{x}_2^{[1]} = 0 \tag{26b}$$
$$\bar{x}_1^{[2]} = -\frac{\pi}{2} \tag{26c}$$
$$\bar{x}_2^{[2]} = 0 \tag{26d}$$

As external force exciting the pendula pivots a $sin(t)$ signal was chosen and the springs stiffness ratio $k = 1.0$ N/m.

The aggregated system was integrated using the `CVode` solver in the Assimulo package with tolerances, atol $=$ rtol $= 10^{-8}$ for a time of 5 seconds and the Jacobian approximated using forward differences. Results were then compared to a reference where the coupled pendulas were modelled as a monolithic system in Modelica and simulated with `Dassl` in Dymola using tolerance tol $= 10^{-12}$. Figure 4 shows the error in log-scale of the angle $\bar{x}_1^{[1]}$ of all three systems compared to the control. The same plot for angle $\bar{x}_1^{[2]}$ is shown in Figure 5.

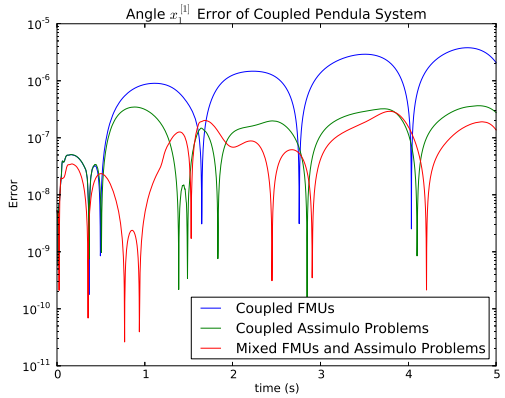

Figure 4. Error of angle $x_1^{[1]}$ of aggregated FMU, Assimulo and mixed systems, simulated for 5 seconds with tolerances atol = rtol = 10^{-8} with CVode solver in Assimulo package.

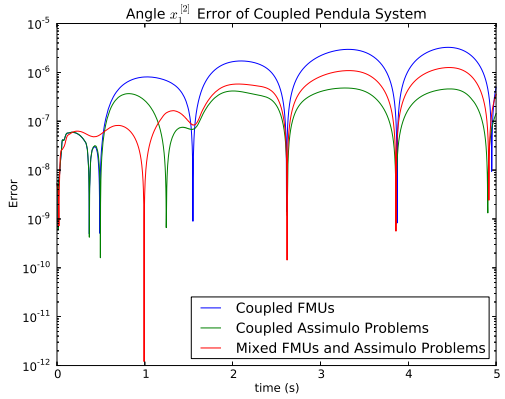

Figure 5. Error of angle $x_1^{[2]}$ of aggregated FMU, Assimulo and mixed systems, simulated for 5 seconds with tolerances atol = rtol = 10^{-8} with CVode solver in Assimulo package.

3.3 Coupled Pendula Impact on Wall

The aggregated system constructed with two FMUs in Section 3.1 is here reused with the addition of discontinuities as two walls are externally placed in the path of the two pendulums' swinging motion. One wall for each pendulum. This as to highlight the possiblity of externally adding state events to the coupled problem. Also the external forces acting on the pivots have been removed.

When the bob hits the wall a discontinuity occurs. This is handled by defining event indicators that trigger an event when the impact occurs. Event indicators are defined as zero-crossings as,

$$event^{[i]} = wall^{[i]} - \bar{x}_1^{[i]} \tag{27}$$

where $wall^{[i]}$ is the x-coordinate of the wall blocking pendulum $[i]$. The impact itself is elastic and the event handling is done by simply reversing the velocity of the bob. For the pendulum to the left a wall is placed directly below the pivot point and the impact occurs when

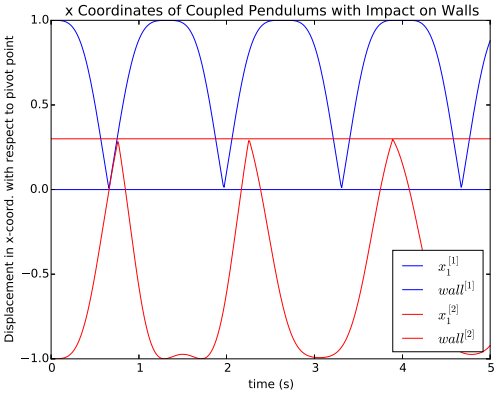

Figure 6. Shows the x-coordinate displacement, with respect to their own pivots, of the pendulums [1], to the left, and [2], to the right, has they hit a wall. The horizontal lines represent walls blocking each pendulums path.

the bobs x-coordinate reaches $\bar{x}_1^{[1]} = 0$ with respect to its pivot. The right-hand-side pendulum wall is placed slightly to the right of its pivot and the bob impacts the wall when its x-coordinate reaches $\bar{x}_1^{[2]} = 0.3$ with respect to its pivot. Initial conditions and parameters are the same as for the example described in Section 3.1.

The aggregated system was integrated with the `CVode` solver with tolerances atol = rtol = 10^{-8} for 5 seconds. Figure 6 shows the x-coordinate displacement with respect to each pendulums pivot. The two horizontal lines represent each pendulums respective walls.

4 Conclusion

In this paper, a framework has been presented for simulation of coupled systems by aggregation. Care needs to be taken when a coupled system contains feed-through as an equation system needs to be solved in order to compute the derivatives of the system. This puts a condition on the sub-system feed-through terms that also presents itself when computing the Jacobian.

The sub-system events are handled by aggregation. A benefit of this approach is that events from all subsystems together with external events can be monitored at once and handled through the aggregated system. Example described in Section 3.3 shows that external events can be added to an aggregated coupled system.

The FMI has all functionality needed to carry out the presented scheme. By combining the discussed ideas with Assimulo and allowing direct coupling of FMUs and Python based problems one gets a flexible and powerful environment for solving coupled dynamical problems.

DOI 10.3384/ecp15118903

References

Christian Andersson. *A Software Framework for Implementation and Evaluation of Co-Simulation Algorithms*. Licentiate thesis, Centre for Mathematical Sciences, Lund University, Lund, Sweden, 2013.

Christian Andersson, Claus Führer, and Johan Åkesson. Assimulo: A unified framework for ode solvers. *Math. Comput. Simulat.*, 2015. doi:10.1016/j.matcom.2015.04.007. In press.

Torsten Blochwitz, Martin Otter, Johan Åkesson, Martin Arnold, Christoph Clauss, Hilding Elmqvist, Markus Friedrich, Andreas Junghanns, Jakob Mauss, Dietmar Neumerkel, Hans Olsson, and Antoine Viel. Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. In *In 9th International Modelica Conference 2012*. Modelica Association, 2012.

Dassault Systèmes. Dymola - Multi-Engineering Modeling and Simulation - Version 2016. `http://www.dymola.com/`, 2016. Accessed: 2015-08-01.

Edda Eich-Soelner and Claus Führer. *Numerical Methods in Multibody Dynamics*. European Consortium for Mathematics in Industry (ECMI). Teubner, 1998. ISBN 3-519-02601-5.

Emil Fredriksson, Christian Andersson, and Johan Åkesson. Discontinuities handled with events in Assimulo. In Hubertus Tummescheit and Karl-Erik Årzén, editors, *Proceedings of the 10th International Modelica Conference*, number 96 in Linköping Electronic Conference Proceedings, pages 827–836. Linköping University Electronic Press, Linköpings universitet, 2014. URL `http://dx.doi.org/10.3384/ECP14096827`.

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 Trans. Math. Softw.*, 31(3):363–396, September 2005. ISSN 0098-3500. doi:10.1145/1089014.1089020.

Johan Åkesson, Karl-Erik Årzén, Magnus Gäfvert, Tove Bergdahl, and Hubertus Tummescheit. Modeling and optimization with Optimica and JModelica.org—languages and tools for solving large-scale dynamic optimization problem. *Comput. Chem. Eng.*, 34(11):1737–1749, November 2010. doi:http://dx.doi.org/10.1016/j.compchemeng.2009.11.011.