



D4.2.11 - Prototype for multi-mode DAE systems in PySimulator

WP 4.2 Prototypes of tool support for multi-mode DAE systems

WP 4 Systems with multiple operating modes

MODRIO (11004)

Version 1.0

Date 21/12/2015

Authors

Andreas Pfeiffer

DLR

Executive Summary

The Functional Mock-Up Interface Standard FMI 2.0 was developed within the MODRIO project. Originally it was planned to include multi-mode support, that is to handle Differential Algebraic Equations (DAEs) with dynamically changing continuous-time states. However, there is currently only a prototype of Dymola available for this feature, but no FMI extensions have been yet proposed.

Therefore, this document shows the interface between the open source simulator "PySimulator" and the Functional Mock-up Interface 2.0. It is demonstrated that FMUs according to FMI version 2.0 can properly be simulated by PySimulator. Simulation results for some examples are compared to results generated by Dymola.

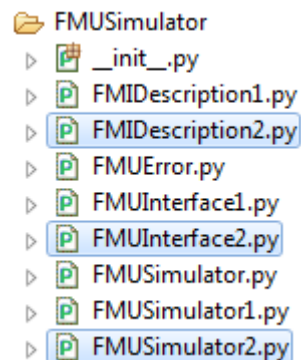
Contents

Executive Summary	2
Contents	2
1. PySimulator and FMUs	3
2. Model Description and Interface for FMI 2.0 in PySimulator.....	3
3. FMU Simulator in PySimulator	5
3.1. FMI 2.0 for Model Exchange	5
3.2. FMI 2.0 for Co-Simulation	5
4. Tests by Examples.....	7
5. References	8

1. PySimulator and FMUs

The simulation and analysis environment PySimulator [1] is a Python based open source tool (www.pysimulator.org) for simulating different model types (currently Functional Mockup Units, as well as Modelica models processed with the simulator plugins for Dymola, OpenModelica and SimulationX), plotting result variables and applying simulation result analysis tools. Until end of the year 2014 Functional Mock-Up Units (FMUs) according to the Functional Mock-Up Interface (FMI) specification 1.0 for Model Exchange were supported by PySimulator. These FMUs can be loaded into PySimulator, they can be simulated by different numerical integration algorithms and the results are stored in the Modelica Association Times Series File format (MTSF) based on HDF5 (Hierarchical Data Format version 5), see [2] for details. In PySimulator MTSF files can be loaded, the results can be plotted and according analysis plugins of PySimulator can be applied to these results.

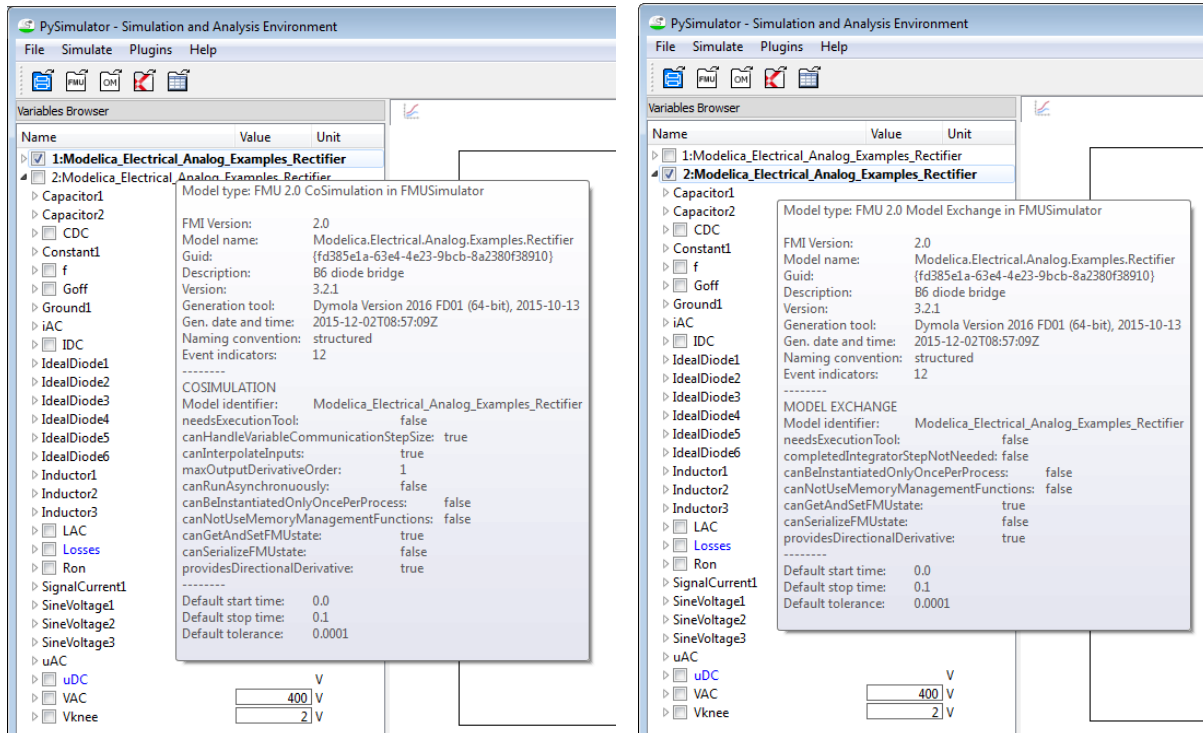
Within MODRIO PySimulator is extended to support FMI 2.0 both for Model Exchange and Co-Simulation. PySimulator has several simulator plugins to run simulations of different types of models. The FMU Simulator is one of these simulator plugins and it has been partially redesigned to distinguish between FMUs for FMI 1.0, FMI 2.0, for Model Exchange and for Co-Simulation. According to the FMI specification version 1.0 and 2.0 there are two versions of the Python modules for FMI description, FMU interface and FMU Simulator:



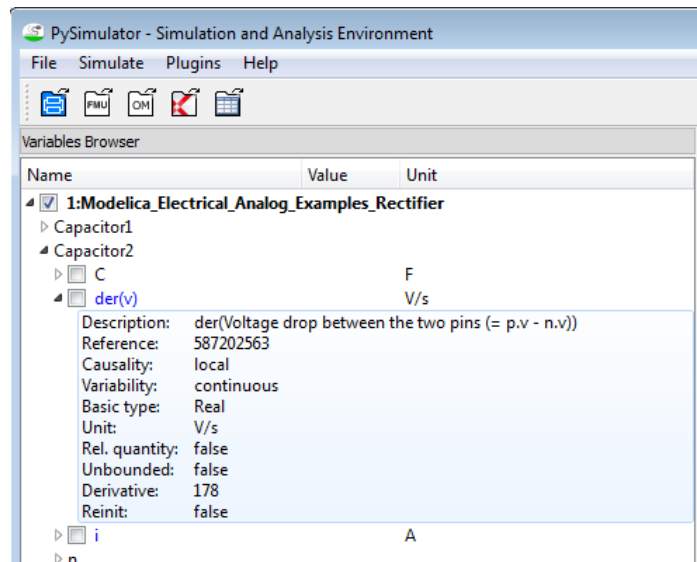
In Section 2 of this document the FMI description and the interface implementation are discussed whereas in Section 3 an overview is given how the simulation of FMUs version 2.0 is realized in PySimulator and how the user interface looks like. In Section 4 several automated tests are run to demonstrate the quality of FMU simulations of PySimulator.

2. Model Description and Interface for FMI 2.0 in PySimulator

When loading an FMU into PySimulator in a first step the Extensible Markup Language (XML) file (`modelDescription.xml`) of the FMU is read to get the FMI version (1.0 or 2.0) of the FMU. According to the version the full content of the XML file is read and analyzed in the module `FMIDescription<x>.py`. All the information is stored in a Python class to be available when handling with the FMU. In PySimulator's GUI many top level properties of the FMU are displayed when moving with the mouse pointer over the model name in the variables browser – see the following figures for an FMU 2.0 for Co-Simulation (left one) and for Model Exchange (right one):



On the level of each variable declared in the XML file and visible in the variables browser the following properties of the variable are displayed:



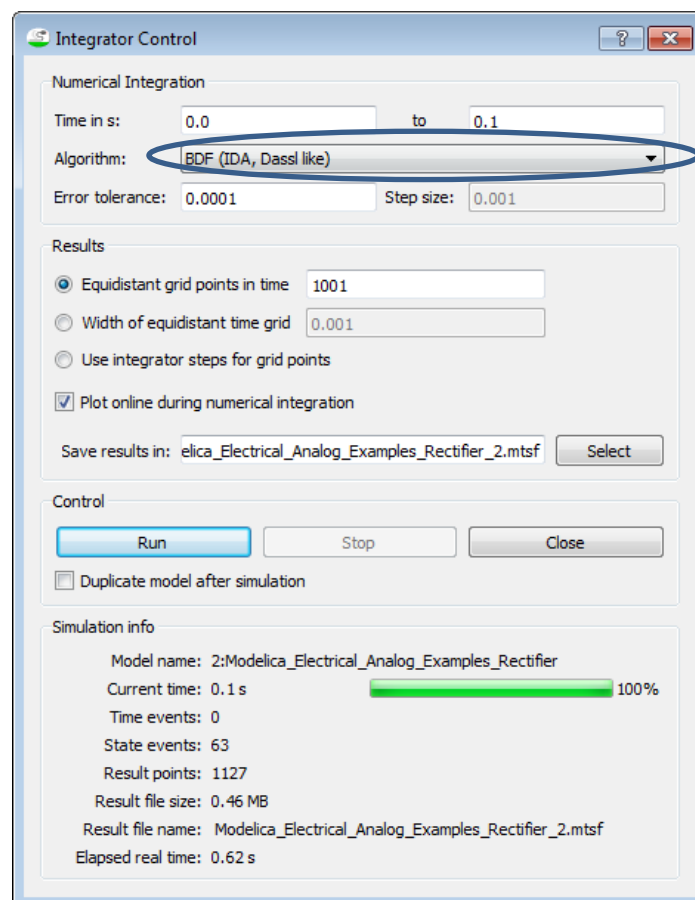
For each FMI function of the loaded FMU a separate Python function is available to be called by the FMU Simulator. The FMU has to contain a dynamic link library for Windows or a shared library for Linux in order to be properly interfaced by PySimulator.

3. FMU Simulator in PySimulator

Each of the loaded FMUs in PySimulator can be simulated by the FMU Simulator. The FMU Simulator is a simulator plugin in PySimulator that has been extended to support FMUs with FMI version 2.0. PySimulator's GUI to control the numerical integration is the same for FMUs for Model Exchange and for Co-Simulation.

3.1. FMI 2.0 for Model Exchange

For FMUs for Model Exchange different integration algorithms can be selected. There are variable step size solvers as well as fixed step size solvers. Start time, stop time and the time grid of the results have to be specified, the name of the result file can be selected and during the simulation process some information about the progress is displayed:

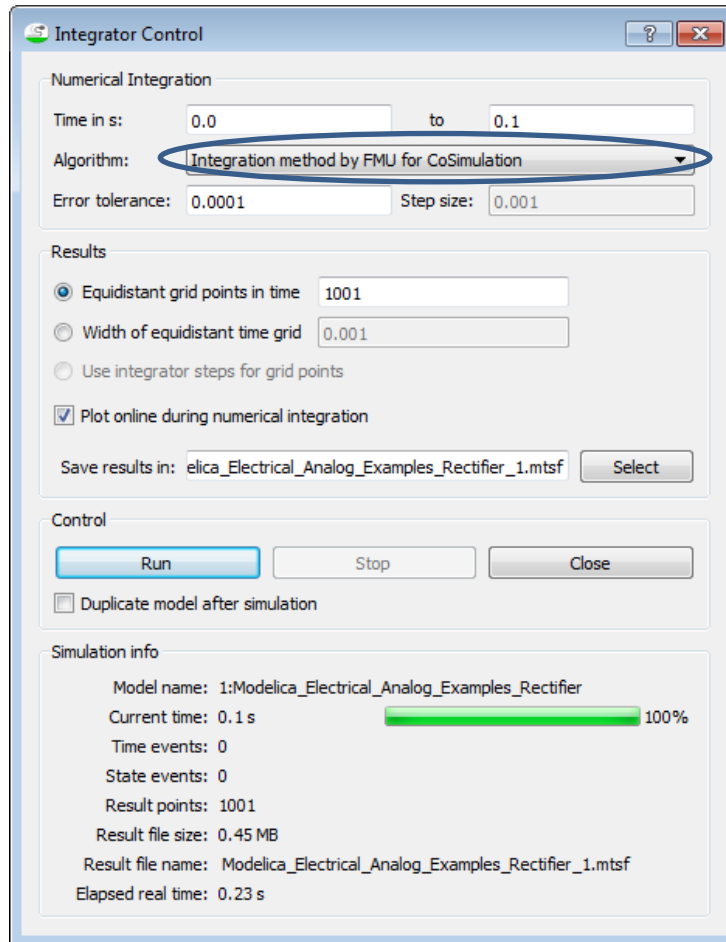


The results are efficiently stored in an HDF5 file with a special structure according to time series, see [2] for the detailed specification.

3.2. FMI 2.0 for Co-Simulation

FMUs for Co-Simulation without inputs are correctly simulated by PySimulator. If the FMU has inputs, then they are set equal to zero. It is planned to implement some kind of file based input handling in the future. The current general restriction is that only one single FMU can be simulated, but not coupled ones.

The integration control GUI is the same as for the integration of FMUs for Model Exchange. Of course, the integration algorithm cannot be selected for FMUs for Co-Simulation, because it is contained in the FMU:



Integrator Control

Numerical Integration

Time in s: 0.0 to 0.1

Algorithm: Integration method by FMU for CoSimulation

Error tolerance: 0.0001 Step size: 0.001

Results

☒ Equidistant grid points in time 1001

☐ Width of equidistant time grid 0.001

☐ Use integrator steps for grid points

☒ Plot online during numerical integration

Save results in: elica_Electrical_Analog_Examples_Rectifier_1.mtsf Select

Control

Run Stop Close

☐ Duplicate model after simulation

Simulation info

Model name: 1:Modelica_Electrical_Analog_Examples_Rectifier

Current time: 0.1 s 100%

Time events: 0

State events: 0

Result points: 1001

Result file size: 0.45 MB

Result file name: Modelica_Electrical_Analog_Examples_Rectifier_1.mtsf

Elapsed real time: 0.23 s

4. Tests by Examples

For the following tests four Modelica models of the Modelica Standard library are selected:

- `Modelica.Electrical.Analog.Examples.Rectifier`
- `Modelica.Fluid.Examples.HeatExchanger.HeatExchangerSimulation`
- `Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.fullRobot`
- `Modelica.Thermal.FluidHeatFlow.Examples.ParallelPumpDropOut`

These models are exported by Dymola 2016 FD01 to an FMU for Model Exchange and to an FMU for Co-Simulation (each FMI version 2.0). The according FMUs are simulated by PySimulator and as reference they are imported and simulated in Dymola 2016 FD01.

To compare the results between the Dymola simulation and the PySimulator simulation of the FMUs, PySimulator's Regression Testing plugin [3] is applied. The plugin compares all variables in a result file with a reference solution in another result file and documents the difference in an HTML report.

For the FMUs for Model Exchange the regression report is as follows:

Regression Report

Given Error Tolerance: 5e-3

Disk space of all used result files: 35.3 MB

Total number of compared files: 5 against 5 baseline files

Total number of Compared Variables: 3593 (3593 passed, 0 failed)

Disk space of full report directory: 0.1 MB

Generated: 11:20AM on December 21, 2015 by PySimulator

Time Taken: 00h:00m:13s

[Legend](#)

Result Files	Status	Dymola	FMUSimulator
	4 / 0		4 passed / 0 failed
	100%		100% passed
		Baseline	
Modelica.Electrical.Analog.Examples.Rectifier	1 / 0	204	192 / 180 [4.7e-5]
Modelica.Fluid.Examples.HeatExchanger.HeatExchangerSimulation	1 / 0	4372	4232 / 311 [5.2e-6]
Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.fullRobot	1 / 0	5610	5538 / 2813 [5.0e-4]
Modelica.Thermal.FluidHeatFlow.Examples.ParallelPumpDropOut	1 / 0	301	295 / 289 [8.4e-5]

The results of the simulations in Dymola and by PySimulator are identical within the specified error tolerance 5e-3 (the greatest error is 5e-4 for the `fullRobot` model). For the models `HeatExchangerSimulation` and the `fullRobot` only a part of all variables are compared due to different naming conventions for vector-valued variables in Dymola and PySimulator.

In the same manner the regression report of FMUs for Co-Simulation looks like:

Regression Report

Given Error Tolerance: 5e-3

Disk space of all used result files: 33.9 MB

Total number of compared files: 5 against 5 baseline files

Total number of Compared Variables: 3593 (3593 passed, 0 failed)

Disk space of full report directory: 1.0 MB

Generated: 11:19AM on December 21, 2015 by PySimulator

Time Taken: 00h:00m:13s

Legend

Result Files	Status	Dymola	FMUSimulator
	4 / 0		4 passed / 0 failed
	100%		100% passed
		Baseline	
Modelica.Electrical.Analog.Examples.Rectifier	1 / 0	210	192 / 180 [2.4e-3]
Modelica.Fluid.Examples.HeatExchanger.HeatExchangerSimulation	1 / 0	4282	4232 / 311 [2.5e-5]
Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.fullRobot	1 / 0	5584	5538 / 2813 [4.1e-3]
Modelica.Thermal.FluidHeatFlow.Examples.ParallelPumpDropOut	1 / 0	306	295 / 289 [2.5e-17]

Here, the maximum error between the compared result signals is 4.1e-3 for the model `fullRobot`. General, there are some deviations in the results of Dymola and PySimulator because of different handling of discrete variables. But despite of these differences the analyzed results coincide very well.

In summary, the tests show that PySimulator can load and simulate FMUs with FMI version 2.0. The results are in line with simulation results generated by Dymola shown by several examples.

5. References

- [1] A. Pfeiffer, M. Hellerer, S. Hartweg, M. Otter and M. Reiner: *PySimulator – A Simulation and Analysis Environment in Python with Plugin Infrastructure*. In: Proceedings of the 9th International Modelica Conference, p. 523-536, 03.-05. Sep. 2012, Munich, Germany.
- [2] A. Pfeiffer, I. Bausch-Gall and M. Otter: *Proposal for a Standard Time Series File Format in HDF5*. In: Proceedings of the 9th International Modelica Conference, p. 495-506, 03.-05. Sep. 2012, Munich, Germany.
- [3] A. Asghar, A. Pfeiffer, A. Palanisamy, A. Mengist, M. Sjölund, A. Pop and P. Fritzson: *Automatic Regression Testing of Simulation Models and Concept for Simulation of Connected FMUs in PySimulator*. In: Proceedings of the 11th International Modelica Conference, p. 671-679, 21.-23. Sep. 2015, Versailles, France.