FMUtoolbox for MATLAB/Simulink

Jan Glos

FMUTOOLBOX FOR MATLAB/SIMULINK

Jan Glos
Doctoral Degree Programme (1), FEEC BUT
E-mail: xglosj00@stud.feec.vutbr.cz

Supervised by: Pavel Václavek
E-mail: vaclavek@feec.vutbr.cz

Abstract: Acausal (physical) modeling can be advantageously used to construct dynamic models, which are legible and easily modifiable for user. Even highly complex models can be easily assembled and the model remains clear. The most widespread language for acausal modeling is Modelica. Unfortunately Matlab/Simulink does not support Modelica language and that is why FMUtoolbox for Matlab/Simulink was established. It adds possibility to simulate Functional Mock-Up Units (exported Modelica models) to Matlab/Simulink environment. FMUtoolbox was practically proved during verification of control algorithms for stepper motor and in the design of heat pump control.

Keywords: Modelica, Functional Mock-up Interface, FMI, Functional Mock-up Unit, FMU, Matlab, Simulink, Model Exchange, Co-Simulation

1 INTRODUCTION

Modelica is an object-oriented language designed for modeling heterogeneous physical systems [1]. It is intended for multi-domain modeling and aspires to be general-purpose modeling tool [2]. It provides large amount of free libraries for modeling and simulation of electrical, magnetic, mechanic, thermal and other physical systems. Moreover there are lots of commercial libraries for special purposes (like refrigeration, engines, batteries, pneumatics etc.). The user can also create own library using existing models or create completely new one with own developed models, which can be build using equations or by connecting existing objects.

Modelica language uses acausal modeling, which brings significant advantages for user. Compared to causal description (like in Simulink using integrators, gains etc.) the acausal approach provides better legibility of model, resulting in less chance of mistakes [3]. Also the model can be easily modified or extended.

Functional Mock-up Interface (FMI) describes standardized way to exchange models between modeling and simulation tools [4]. It is supported by most of Modelica tools (Dymola, OpenModelica etc.). Matlab/Simulink supports neither Modelica language nor this standard and that's why FMUtoolbox for Matlab/Simulink was established. It adds functionality for FMU simulation into Matlab/Simulink environment, what brings the possibility of using Modelica models in Matlab/Simulink.

Very first version of FMUtoolbox was presented in my previous article [5]. This paper describes the second version of the toolbox, whose most important improvements include support of FMI 2.0, support of Co-Simulation, notable simulation speed-up, native Matlab installer and others.

2 MODELICA LANGUAGE

This chapter briefly describes the Modelica language, in depth as needed in the following chapters. More details about Modelica language can be found in [1], [6], [7], [8] and others.

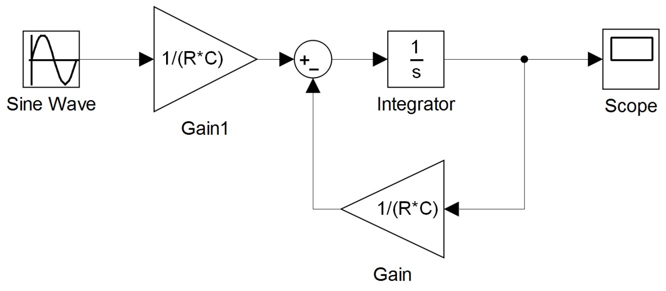
Every Modelica model consists of three basic sections. The first of them is placed behind keyword `model` (with the model name) and contains model variables, inheritance and others. The second section of the model contains initial equations, which are used to compute initial conditions for state variables. The third part of the model includes model equations and is prefaced by keyword `equation`. There can be both the real equations (like $a+b-c=0$) and connections of blocks (like `connect(objA.output, objB.input)`). The model is terminated with keyword `end` with the name of the model.

A basic example of simple Modelica model is presented below.

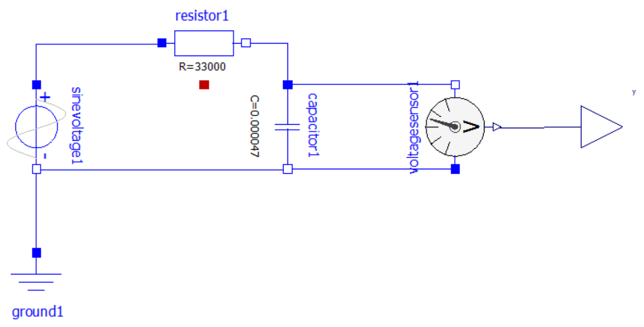
```
model simpleExample
    Real x "State variable";
    parameter Real x0 = 5 "initial value of x";
initial equation
    x = x0;
equation
    0 = der(x) + 3*x;
end simpleExample;
```

The name of the model is `simpleExample` and using parameter `x0` and initial equation the start value of `x` can be set.

Figures 1a and 1b illustrate the main difference between causal and acausal modeling, especially Simulink and Modelica. Simulink model in Fig. 1a is strictly based on differential equations and it is not very legible for user. Conversely the model in Modelica (Fig. 1b) is based on objects and their connections, so the model is legible and it can reflect the reality.



(a) in Simulink



(b) in Modelica

Figure 1: RC circuit model

3 FMI 2.0

FMI 2.0 is the second major version of the standard defined to normalize Model Exchange and Co-Simulation between Modelica (and other) modeling and simulation tools [4]. In this version standards for Model Exchange and Co-Simulation were merged, which simplifies the solution needed for handling model import, export and Co-Simulation.

Almost every Modelica tool is capable to export a model into Functional Mock-up Unit (FMU), a special file type consisting of model description in XML file and model equations as source code or binary form (dynamic-link library or shared object). There might be additional files packed into FMU (like tables, maps and images). In fact, FMU is a folder with defined structure, which is packed into zip and then renamed to have an extension `.fmu`.

In comparison with the previous version multiple changes were performed. As mentioned above, Model Exchange and Co-Simulation are now based on a common core and only necessary parts are different. Regarding model description clarification of some terms was accomplished (especially model variable properties `causality` and `variability`). Further an item `ModelStructure` was added, which describes model outputs, derivatives and initial unknown. It is an advantageous improvement, for example in version 1.0 it was needed to find outputs between all model variables (sometimes thousands and more). Moreover items describing FMU capabilities were added (`ModelExchange` and `CoSimulation`). Regarding Application Programming Interface (API) several improvements were made. For example possibility to reset FMU can be mentioned, hereafter logging categories and other improvements.

4 REALIZATION OF FMUTOOLBOX

FMUtoolbox is a tool allowing simulation of Modelica models (exported into FMU format) in Matlab/Simulink environment. This tool was created by me and is continuously improved and perfected.

New version of FMUtoolbox had to be created due to new version of FMI standard, performance issues and because of the need for Co-Simulation support. FMUtoolbox 1.0 was created previously and described in my last article [5]. The first version provided basic FMU simulation capabilities for Matlab/Simulink.

The core of FMUtoolbox 2.0 is derived from the previous version. It is based on Matlab class, which provides methods and data storage for single FMU. Class methods are responsible for loading FMU into Matlab, loading XML file with model description, partly for user interface and others.

FMUtoolbox 2.0 provides user interface in two ways - graphical user interface (GUI) for FMU usage in Simulink and command line interface for simulation using Matlab command line.

The most important changes were performed on Simulink library, blocks and S-function. Originally Level 2 Matlab S-Function was used to provide the interface between Matlab and FMU (additionally there was a dynamic-link library or shared object between Matlab and FMU to provide needed functionality). Actual solution employs Level 2 C S-Function as the interface between Matlab and FMU. No other DLL or SO is necessary, all needed functionality is formed inside C S-function (or is included into it). This solution brought significant simulation speed-up and simplification of communication between Matlab/Simulink and FMU. Actually there are four C S-Functions created (Fig. 2), two of them are for FMI 1.0 (the first one for Model Exchange, the second for Co-Simulation) and two for FMI 2.0 (the first one for Model Exchange, the second for Co-Simulation). This division ensures the possibility to simulate any combination of FMUs concurrently - for example several the same FMUs with different parameters, one Model Exchange FMU 1.0 with Co-Simulation FMU 2.0 and so on.

One of the most important improvement is added support for Co-Simulation. In some cases Co-

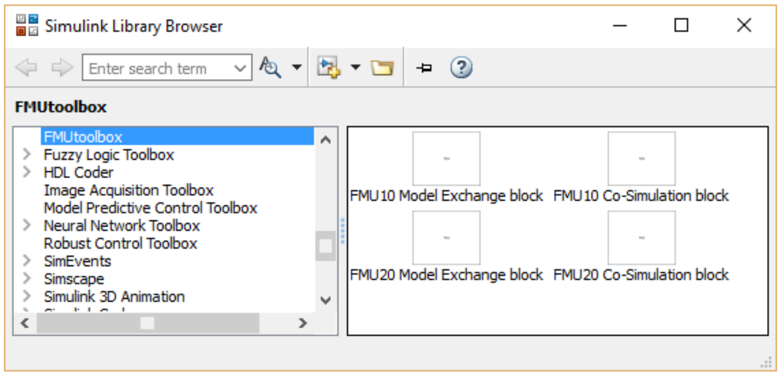
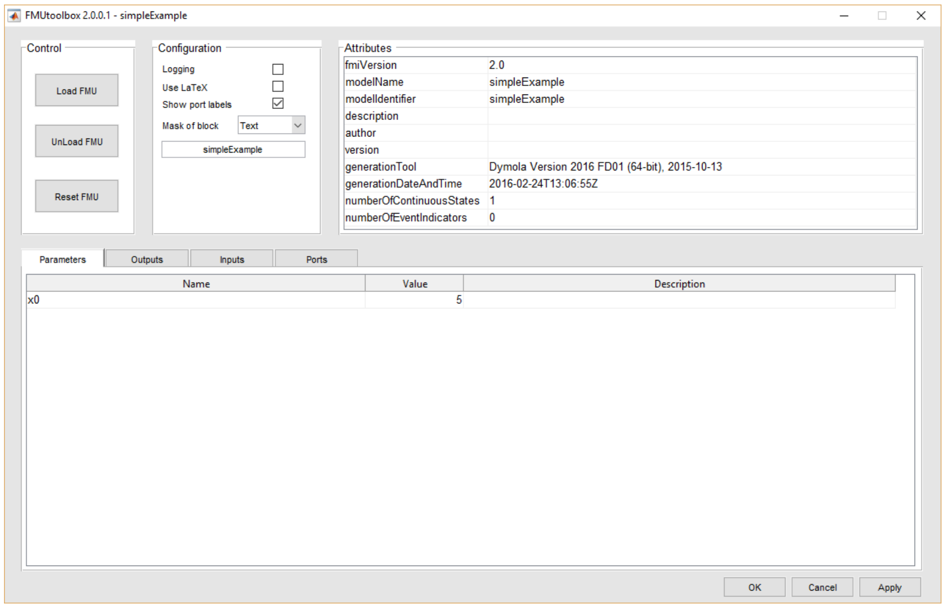


Figure 2: FMUtoolbox library for Simulink



Configuration

Logging ☐
Use LaTeX ☐
Show port labels ☑
Mask of block: Text
simpleExample

Attributes

fmiVersion	2.0
modelName	simpleExample
modelIdentifier	simpleExample
description	
author	
version	
generationTool	Dymola Version 2016 FD01 (64-bit), 2015-10-13
generationDateAndTime	2016-02-24T13:06:55Z
numberOfContinuousStates	1
numberOfEventIndicators	0

Name	Value	Description
x0	5	

Figure 3: FMUtoolbox GUI

Simulation is more suitable as it can bring simulation speed-up or increased accuracy. There are two possible ways to perform Co-Simulation - communication between different tools and packing solver into FMU. The second option was chosen because of the compatibility reasons and because the communication between tools is not addressed in FMI 2.0 standard. It was found that for some models Dymola solvers are more appropriate than Matlab solvers - in those cases Co-Simulation is very useful.

Another important change covers installation of FMUtoolbox. In previous version the installers were prepared for each platform (Windows, Linux, OS X) separately according to different possibilities of these platforms. For FMUtoolbox 2.0, standard shared installation procedure for all platforms was introduced. It uses Matlab Add-Ons to install FMUtoolbox.

Regarding graphical user interface (GUI) some performance improvements were realized and the GUI was adjusted for FMI 2.0 usage. In Fig. 3 GUI with loaded FMU is shown - it is an example Modelica model from section 2 exported into FMU and then loaded into Matlab. The value of parameter `x0` can be adjusted before simulating the model.

5 CONCLUSION

In this paper FMUtoolbox for Matlab/Simulink was described, especially the second version of this tool. This version brought support of FMI 2.0, Co-Simulation and some other features. Main differences between version FMI 2.0 and FMI 1.0 were described as well as differences between FMUtoolbox 1.0 and 2.0.

Most importantly, the toolbox has been verified in practical use and it was found satisfactory for the validation of control algorithms. The first successful use was in the design of control loop for hybrid stepper motor (HSM). The accurate model of HSM was constructed in Modelica language and then it was exported into FMU. Then the model was loaded into Matlab/Simulink using FMUtoolbox and the testing of control algorithms could be done. The model contained both the HSM and the power circuit, so the simulations results were similar to behaviour of the real system.

FMUtoolbox was also used during designing heat pump control. As usually the model of heat pump was formed using Modelica language and then it was loaded into Matlab/Simulink for further use. In this case co-simulation was advantageously employed, because Dymola solvers were found to be much more faster and accurate when used with heat pump Modelica model. Moreover it is almost impossible to construct a heat pump model in Matlab/Simulink environment and conversely Dymola is not so suitable for the design of control algorithms as Matlab.

According to the above examples it is evident that FMUtoolbox is a suitable tool for Modelica models simulation in Matlab/Simulink environment.

ACKNOWLEDGEMENT

The completion of this paper was made possible by the grant No. FEKT-S-14-2429 - „The research of new control methods, measurement procedures and intelligent instruments in automation" financially supported by the Internal science fund of Brno University of Technology.

REFERENCES

[1] Modelica Association, "Modelica - a unified object-oriented language for systems modeling," 2014.

[2] S. E. Mattsson, H. Elmqvist, and M. Otter, "Physical system modeling with Modelica," *Control Engineering Practice*, vol. 6, no. 4, pp. 501 – 510, 1998.

[3] J. Kofránek, M. Mateják, P. Privitzer, and M. Tribula, "Causal or acausal modelling: Labour for humans or labour for machines," *Technical Computing Prague*, 2008.

[4] MODELISAR consortium and Modelica Association Project "FMI", "Functional Mock-up interface for Model Exchange and Co-Simulation," 2014.

[5] J. Glos, "Využití modelů v jazyce Modelica v prostředí Matlab-Simulink," in *Proceedings of the 21st Conference STUDENT EEICT 2015*, (Brno), pp. 310–312, Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2015.

[6] P. A. Fritzson, *Principles of object-oriented modeling and simulation with Modelica 2.1*. New York: Wiley-Interscience, c2004.

[7] M. Tiller, *Modelica by Example*. 2014.

[8] M. Tiller, *Introduction to physical modeling with Modelica*. Boston: Kluwer Academic Publishers, c2001.