FMUtoolbox Cross Check Implementation and FMI Standard Compliance Results

Jan Glos

Accepted manuscript

J. Glos, "FMUtoolbox Cross Check Implementation and FMI Standard Compliance Results". In Proceedings of the 24th Conference STUDENT EEICT 2018. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2018. pp. 403-407. ISBN: 978-80-214-5614-3.

FMUTOOLBOX CROSS CHECK IMPLEMENTATION AND FMI STANDARD COMPLIANCE RESULTS

Jan Glos
Doctoral Degree Programme (3), FEEC BUT
E-mail: xglosj00@stud.feec.vutbr.cz

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

Abstract: FMI Cross Check is very important extension of FMI standard, as it defines rules for FMU compatibility verification of exporting and importing tools. This paper provides a brief overview of implementation of Cross Check for FMUtoolbox and reports results of executed check of this tool.

Keywords: FMU, FMI, Functional Mock-Up Interface, Functional Mock-Up Unit, Modelica, Cross Check, Matlab, Simulink

1 INTRODUCTION

FMUtoolbox is a tool which enables simulation of Functional Mock-Up Units (FMU) in Matlab/ Simulink environment [2]. It can be beneficially used for importing and simulating both the Model Exchange and Co-Simulation FMUs in Matlab and Simulink. A screenshot of FMUtoolbox graphical user interface is shown in Fig. 1, which is one possible way to control the FMUtoolbox. Also Matlab command line control of this toolbox is possible and this feature was used for Cross Check implementation, which is described within this paper.

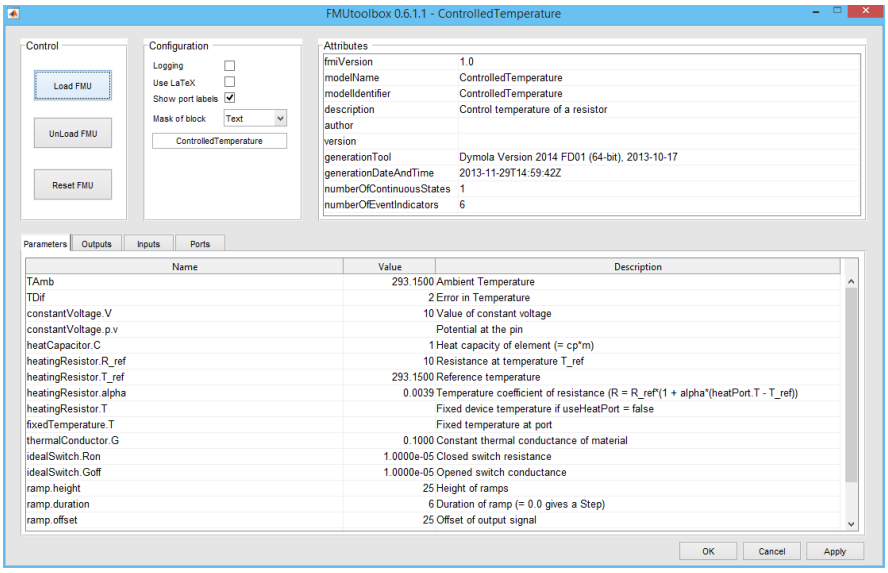

Figure 1: FMUtoolbox graphical user interface

FMUtoolbox was created with the aim to be compatible with most of FMU exporting tools. The compatibility should be ensured by FMI standard, which strictly defines the FMU exchange format. Nevertheless, during FMUtoolbox development a lot of issues and inaccuracies were found, especially

in FMUs exported by other vendor tools. That is the reason, why automatic FMI Cross Check was implemented. It allows bulk check of set of FMUs, which can be for example downloaded from FMI Standard repository (test FMUs from different tools).

Cross Check was used as verification tool for proving the FMUtoolbox compatibility with FMI standard and to demonstrate FMUtoolbox applicability for FMU simulation in Matlab Simulink.

2 CROSS CHECK RULES

FMI Cross Check rules were defined in [3] in order to establish a standardized approach of FMU compatibility testing. The reason for that was poor FMU compatibility between different vendor tools, what damaged reputation of FMI standard. The goal of Cross Check is improving FMU conformance with FMI standard, helping vendors to detect and fix the possible FMU incompatibilities and improving FMI reputation.

In [3] there are eleven Cross Check rules, which are written comprehensibly and it should be easy to fulfill them. These rules were followed during FMUtoolbox Cross Check implementation (see next section).

Improvements of FMI Cross Check and thus FMU compatibility are continuously ongoing. The issues with FMU compatibility were discussed in [1] and reference FMUs were proposed to improve FMI standard compliance for both the FMU exporting and importing tools.

3 IMPLEMENTATION

FMUtoolbox provides both the Simulink GUI interface and also pure Matlab interface (using Matlab command line). Using the Matlab interface it was possible to prepare automatic Cross Check execution, which is beneficial for evaluating compatibility of multiple FMUs at once.

The Cross Check scripts require path to folder with downloaded FMUs (whose compatibility with FMUtoolbox should be verified). The script expects the same foder structure as used in FMI standard SVN repository
`https://trac.fmi-standard.org/browser/branches/public/Test_FMUs`. The script goes through the directory structure with different vendors and for each FMU it prepares data for simulation (provided inputs, parameters etc.), executes the simulation and then processes the simulation results.

By using a FMUclass constructor an empty object instance with name “myFMU” is created. Subsequently a FMU is loaded (e.g. the zip file is unpacked, xml file is read and other preparation tasks are performed).

```
1 myFMU = FMUclass(fmuFile,"myFMU");
2 myFMU.loadFMU;
```

The settings of simulation should be provided by exported FMU vendor in file `{FMUName}_ref.opt`, which is parsed for every FMU and the extracted data are then used for FMU simulation settings. The function `setSimulationParameters` was prepared to allow detailed settings of simulation parameters.

```
1 setSimulationParameters ( startT, stopT, solverType,...
2                           Solver, fixedStepSize,...
3                           minStepSize, maxStepSize,...
4                           relTol, absTol)
```

For each FMU a set of input values can be used (not obligatory, depends on particular model in FMU). These values are also parsed before simulation and passed to FMUclass instance to be used during simulation.

```
myFMU.setInputValues(inputValues);
```

A large set of FMUs (and their vendors) do not mark variables as outputs, what would lead to no output data in FMUtoolbox (as it saves only the data required explicitly required to keep the simulation as fast as possible). To solve this issue a function `setModelVariableAsOutput` was introduced. It allows to mark a model variable as output based on provided variable name and thus the variable values are logged for further use.

```
myFMU.setModelVariableAsOutput("J1.phi",true);
```

Function `simulate` is intended for starting the simulation. It prepares the Simulink model for FMU simulation by inserting FMU Simulink block, Scopes and From Workspace blocks, sets up and connects all the blocks and then starts the simulation.

```
myFMU.simulate();
```

The resulting data from FMU outputs are logged into `FMUclass` property `savedSimData` and then used to compare against reference result data from FMU exporting tool. These reference data should be provided inside `{FMUName}_cc.csv` file.

For each FMU multiple outputs are automatically generated. First of all and the most important, the file indicating the overall result of check of that FMU (“passed”, “rejected” or “failed”) is created. Secondly, a comma-separated value (CSV) file with time and value series is exported for later investigations. Lastly, a comparison of reference and simulated outputs are printed into a figure and then exported into graphic file.

The outputs described above were prepared to strictly comply with Cross Check rules.

4 CROSS CHECK RESULTS

In Fig. 2 there is an example of result of FMUtoolbox Cross Check. A FMU Coupled Clutches (generated by Dymola) was selected to demonstrate the results of Cross Check. Both the reference and the simulated outputs values are printed into graphs for four different output variables. It is obvious that simulation results correspond to the reference model solution. There is only very small inaccuracy, which can be caused by different sampling periods during simulations.

In Table 1 there is an example of CSV output file (also the result of Cross Check of combination of FMUtoolbox and FMU Coupled Clutches). This file is automatically exported after FMU Cross Check execution using FMUtoolbox.

In Table 2 there is a summary of FMUtoolbox Cross Check. The first number (green) of each cell represents a number of successfully imported FMU of given tool and platform, the second number (yellow) stands for number of rejected FMUs and the last number (red) represents a number of failed FMUs. In addition to the standard we added a value “N/A”, which means that for this combination of tool and platform there is no test FMU available.

The platforms `win64` and `win32` are marked green, as the criterion for status “Available” is fulfilled (at least for three tools three different FMUs were successfully imported). The platform `linux64` is not marked, as there were not enough test FMUs available in FMI repository to perform a detailed check.

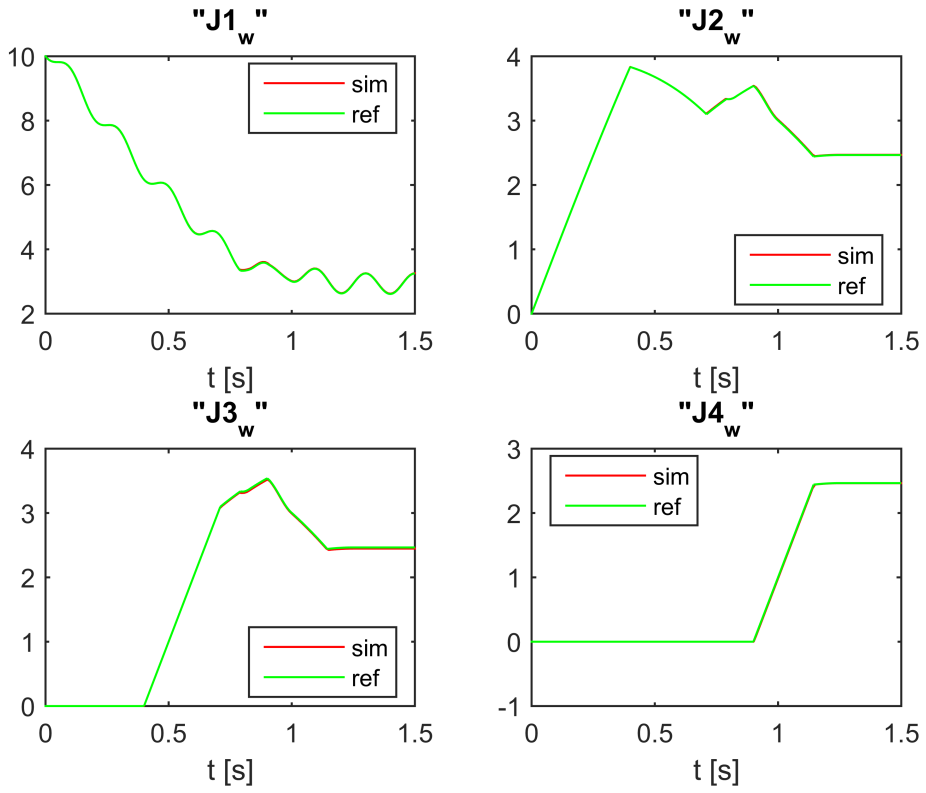

Figure 2: FMUtoolbox Cross Check result for FMU CoupledClutches

Quite a lot of FMUs were rejected due to different reasons. For example, FMUs from CATIA had no variables marked as output (so it wasn't clear, which variables should be compared), FMUs from ControlBuild have incompatible format of XML file inside FMU and a lot of test FMUs were not properly completed with required metadata (exported by JModelica.org, OPTIMICA_Studio, LMS_Virtual.Lab_Motion etc.).

5 CONCLUSION

This paper provides an overview of FMUtoolbox FMI Cross Check implementation and the verification results. As the FMI standard is intended for cooperation of different modeling and simulation tools, it is crucial to ensure the compatibility of FMUs between exporting and importing tools.

However experiences show that some vendors have even problems to provide the test FMUs in required format with all the needed metadata. On the other hand some tools provides both the FMUs and the metadata, which strictly comply to the Cross Check specification.

The compatibility of FMUtoolbox and several FMU exporting tools was examined. The overall result is that FMUtoolbox is available for FMU import for `win32` and `win64` platforms and for platform `linux64` the tests were not conclusive. However the overall status of the FMU compatibility is not satisfactory from my point of view, since quite large number of FMUs were rejected or the simulation failed. From the overall perspective the Cross Check is a very useful tool for FMU compatibility assessment and it has the potential to gradually improve the FMU compatibility between different modelling and simulation tools.

Table 1: Result data of Cross Check of FMU CoupledClutches

time	J1_w	J2_w	J3_w	J4_w
0	10	0	0	0
0.01	9.915592	0.099998	-5.00E-16	-4.99E-16
0.02	9.860831	0.199981	-9.99E-16	-9.98E-16
0.03	9.831307	0.299933	5.55E-17	5.74E-17
0.04	9.82014	0.399839	-3.89E-16	-3.86E-16
0.05	9.818661	0.499683	1.11E-15	1.11E-15
0.06	9.817257	0.599449	2.11E-15	2.11E-15
⋮	⋮	⋮	⋮	⋮

Table 2: FMUtoolbox Cross Check result summary

platform	win64	win32	linux64
AMESim	2 0 0	3 0 0	3 0 0
CATIA	0 3 0	0 3 0	N/A
ControlBuild	0 3 0	0 3 0	N/A
Dymola	7 0 0	7 0 0	N/A
FMIToolbox_MATLAB	N/A	0 3 0	N/A
JModelica.org	N/A	4 0 0	0 3 0
LMS_Virtual.Lab_Motion	N/A	0 1 0	N/A
MapleSim	3 0 0	3 0 0	0 2 0
OpenModelica	N/A	3 0 0	N/A
OPTIMICA_Studio	N/A	0 3 0	N/A
Silver	N/A	1 2 0	N/A
SimulationX	4 0 1	4 0 1	N/A

ACKNOWLEDGEMENT

The completion of this paper was made possible by the grant No. FEKT-S-17-4234 - „Industry 4.0 in automation and cybernetics” financially supported by the Internal science fund of Brno University of Technology.

REFERENCES

[1] Christian Bertsch, Awad Mukbil, and Andreas Junghanns. Improving Interoperability of FMI-supporting Tools with Reference FMUs. In *Proceedings of the 12th International Modelica Conference*, pages 533–540, Prague, 2017. Linköping University Electronic Press. Available from: http://www.ep.liu.se/ecp/article.asp?issue=132&article=60, doi:10.3384/ecp17132533.

[2] Jan Glos. FMUtoolbox for Matlab/Simulink. In *Proceedings of the 22nd Conference STUDENT EEICT 2016*, pages 426–430, Brno, 2016. Vysoké ucení technické v Brne, Fakulta elektrotechniky a komunikacních technologií.

[3] Andreas JUNGHANNS. FMI Cross Check: How to Improve FMI Compliance. [Online], 2014. Available from: https://svn.fmi-standard.org/fmi/branches/public/CrossCheck_Results/FMI_Cross_Check_Rules_v3.1_2015_07.pdf.