Combining Advantages of Specialized Simulation Tools and Modelica Models using Functional Mock-up Interface (FMI)

Yongqi Sun, Stephanie Vogel, Haiko Steuer
Siemens AG, Energy Sector, Erlangen, Germany
yongqi.sun@siemens.com, vogel.stephanie@siemens.com, haiko.steuer@siemens.com

Abstract

In power plant applications, detailed transient analysis of the evaporator results in very large fluid systems, i.e. the model consists of many equations. This is because the evaporator has many tubes and the spatial discretization has to be quite fine in order to appropriately model transient evaporation processes. In such cases, due to performance and workflow reasons, we use a specialized in-house tool Dynaplant [1]. Coupling between Dynaplant and a Modelica-simulator helps to benefit in addition from the possibility of rapid development of new components in Modelica.

Here, it is shown, how the Functional Mock-up Unit (FMU) export from a Modelica model can be used to perform a co-simulation with Dynaplant and an FMU simulator. The FMU is defined via the Functional Mock-up Interface (FMI) for Model Exchange v1.0 [2]. Advantages but also restrictions and challenges are covered.

Keywords: Fluid; Co-simulation; FMU; FMI

1 Introduction

1.1 Modelica world versus Dynaplant

Modelica is the preferred modeling language for dynamic simulations within Siemens Energy [3] due to the high degree of maintainability of Modelica models. In addition, new models can rapidly be developed.

Dynaplant is highly specialized for large fluid systems (e.g. detailed evaporator models). Its performance and usability fulfills our requirements: Performance is increased by using fast water/steam property functions and taking advantage of multi-core CPUs. Usability features are data interface to steady state design tool and restart ability. However, the model library is quite restricted and the development of new Dynaplant model is time-consuming.

1.2 Use cases, where best of two worlds is needed

(A) Typically, there is already an evaporator model in Dynaplant, such that the evaporator itself can easily be simulated in Dynaplant. Now assume some urgent analysis request, which could be answered via a transient simulation of the evaporator together with some neighboring components. There is not yet any Dynaplant model of this neighboring component, but it can be rapidly developed in Modelica.

In this case, some coupling between Dynaplant and Modelica models will be helpful. Here, the process interface between Modelica and Dynaplant may consist of a mass flow rate m_flow and enthalpy h from the Dynaplant evaporator and a pressure p from the new Modelica component.

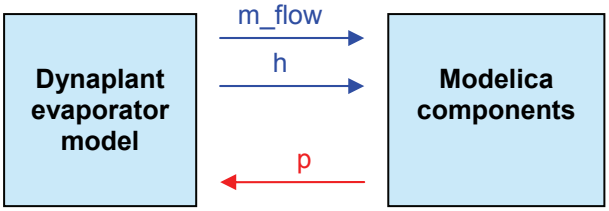

Figure 1: Typical use case (A) for coupling Dynaplant with Modelica models.

Formerly, in such a case an iterative procedure had been carried out. Here, the Dynaplant and the Modelica model are solved alternating with time tables containing the output of one model as input for the other. After several iterations hopefully a convergence is observed. This iterative procedure is very time-consuming and does not guarantee any convergence at all.

(B) The power plant block control is implemented in another in-house software SPPA-T3000 [4]. Currently, there is a T3000-Modelica parser under development generating Modelica models out of SPPA-T3000. This way, the block control could be easily modeled in Modelica. Then, the steam generator (and may be steam turbines) would be modeled in

Dynaplant and remaining parts of the power plant including block control would be modeled in Modelica. This enables the solution of the following use cases for coupling Dynaplant and Modelica models:

- Testing of SPPA-T3000 control schemes using detailed Dynaplant process models
- Dynaplant process simulation using original SPPA-T3000 control schemes

2 Routes to couple in-house tools with Modelica using FMU

In the following two sections 2.1 and 2.2, the Functional Mock-up Unit and its in-house simulator will be introduced. Sections 2.3 and 2.4 present possibilities to couple Modelica and Dynaplant models.

2.1 Introduction to FMU

The FMU is an implementation of the FMI for Model Exchange [2] definitions. We used this instead of the alternative specification (FMI for Co-Simulation), since our long-term goal is to completely include the FMU model into a single simulation environment. The intention is that dynamic system models of different software systems (including Modelica and non-Modelica tools) can generate C-Code of a dynamic system model, which can be used for simulation.

An FMU-file is a zipped archive which contains at least:

1. A small set of easy to use C-functions for all the needed model equations such as differential, algebraic and discrete equations with events. These C-functions can either be provided in source and/or binary form.
2. An xml-file containing the definition of all variables in the model and other model information needed to simulate the model.

The main features of FMU are:

1. It is possible to utilize several instances of a model and to connect models hierarchically together.
2. A model is independent of the target simulator because it does not use a simulator specific header file as in other approaches.
3. An FMU may either be self-integrating or require an external solver to perform numerical integration.

Assume an FMU without self-integration. For coupling this FMU with another simulator (e.g. Dynaplant), one may either connect an FMU simulator with the other simulator (co-simulation) or add the FMU equations directly into the equation system of the other simulator (single model, single solver). The FMU simulator needed for co-simulation is described in the next section. The remaining two sections cover both approaches mentioned.

2.2 Sadida: An Interface to FMU

Sadida is our in-house FMU simulator. It contains solvers for integrating a given FMU. The model equations from the FMU and the time integration solver are wrapped together into several methods of a simulator class. The most important methods are:

- Setup: Allocate memory for the FMU model, and initialize the integrator.
- SetVariable
- GetVariable
- InitializeModel: Computes the initial state corresponding to the FMU initial equations.
- ReAllocateModel
- Run: Perform time integration with required time span and time step size.

Using Dymola 7.4 [5] with code export option, it is possible to export a FMU from a Modelica model. Our in-house FMU simulator Sadida then enables us to integrate any Modelica model. The methods of this FMU simulator can be called by Dynaplant.

2.3 Co-simulation: FMU simulator included in Dynaplant

The first approach uses co-simulation in order to couple the FMU model with the Dynaplant model. Usually, a co-simulation is controlled by a tool like TISC [6] organizing data exchange between several simulators. In contrast, here, another kind of co-simulation is done where Dynaplant (one of the simulation partners) takes control of the co-simulation:

Dynaplant calls methods from the FMU simulator in order to manage data exchange and provides time step sizes for both simulation partners.

The co-simulation is prepared using the following procedure:

- Create an instance of the Sadida FMU simulator class. The following steps are done using methods from this FMU simulator.
- Import FMU model
- Modify FMU model parameters
- Set input variables *u(0)* of FMU model from Dynaplant's initial state

- Initialize the FMU model
- Get initial values of output variables *y(0)* from the FMU model

A Dynaplant integration of the time interval $t_n ... t_{n+1}$ is carried out using constant output $y(t_n)$ from Sadida. After each such Dynaplant time step, the following FMU simulator methods are called by Dynaplant via Sadida:

- Set input variables $u(t_{n+1})$ from Dynaplant.
- Integrate FMU model over time interval $t_n ... t_{n+1}$ using constant $u(t_{n+1})$.
- Get output variables $y(t_{n+1})$ from FMU model

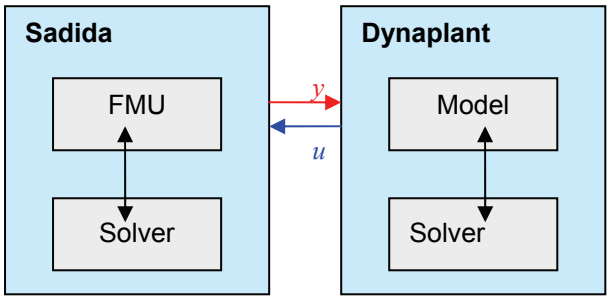

Figure 2: Co-simulation: The FMU model is solved by the FMU simulator Sadida, the Dynaplant model is solved by Dynaplant. Between both solvers, data exchange takes place.

This way, a co-simulation of both Sadida and Dynaplant model can be performed. The interaction of models and solvers is shown in Figure 2. The timing is illustrated in Figure 3. It leads to time discretization errors.

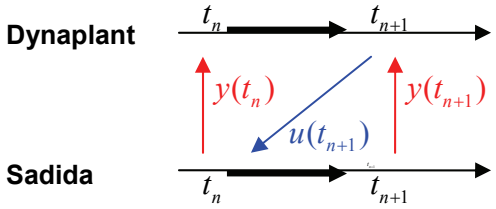

Figure 3: Timing of both Dynaplant and Sadida solvers. First Dynaplant integrates a time step $t_n ... t_{n+1}$ assuming constant $y(t_n)$ from Sadida. Then, Sadida integrates this time interval using constant $u(t_{n+1})$ from Dynaplant. This results in time discretization errors.

2.4 Next step: FMU equations included in Dynaplant (single solver)

As shown in Figure 4, the next step will go beyond co-simulation. Rather than solving the FMU model with a second simulator, the FMU equations will directly be added to the Dynaplant equation system and solved with the Dynaplant solver. This way, there will be no time discretization errors any more.

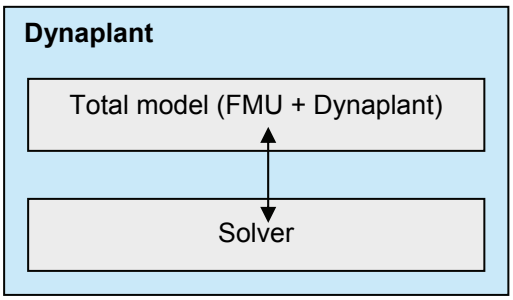

Figure 4: Next step: FMU equations are added to the Dynaplant model and the total model is solved by a single solver.

This approach is just an outlook with many open questions. Especially the following Dynaplant features have to be clarified:

- Fixed Jacobian structure (using potential Jacobian non-zeros)
- Restart ability
- Nominal values dependent on type of variable (e.g. fluid pressure)
- Less sophisticated event handling

3 Co-simulation Restrictions and Challenges

So far the co-simulation approach for coupling Dynaplant and Modelica models is realized. In this chapter we share our experiences including restrictions and challenges.

3.1 Restrictions of co-simulation based on FMU

Due to the time discretization errors caused by the exchanged variables, time intervals $t_n ... t_{n+1}$ should not be too large. Dynaplant uses time step adjustments. The resulting time steps have to be restricted to an appropriate upper limit: $t_{n+1} - t_n \leq \Delta t_{\max}$.

The following restriction is Dymola specific but motivates an extension of the FMI: An FMU exported by Dymola 7.4 without "code export option" will result in an initialization failure on a PC without Dymola. If there are license restrictions in the FMU, it would be better to include license check methods to the FMI rather than computational methods returning FALSE.

A rule deserving notice is that, after the model has been initialized, no parameter or constant vari-

able of the model allows its value to be changed. ReAllocateModel must be called if there is need changing parameter. The values of the other variables can be changed at any time step.

3.2 Challenges

In this section some problems with the co-simulation approach are mentioned.

Consistent initialization: In order to obtain a consistent initial state of the FMU model, the inputs u(0) from Dynaplant's initial state have to be set first (by the way: there was a similar bug in Dymola 7.4 for which a patch is available). A consistent initialization of both FMU and Dynaplant model in total is not yet realized, since the output of the FMU initialization may influence Dynaplant's initial state.

Restart ability: One of the advantages of Dynaplant is its ability to load the final state of a former simulation as initial state of another simulation. Until now, the final state of the FMU is not saved for this purpose, such that the initial equations of the FMU may be adapted for a restart.

Support of more than one FMU: Until now only one FMU is supported. It will be possible to enable several FMUs by creating a simulator instance for each single FMU. However, since there have not yet been a use case for this feature, it is not yet resolved.

Fluid connector interface: At the co-simulation interface, there is no fluid connector like in the Modelica Standard Library. Instead real inputs and outputs are used, assuming no flow reversal.

Accessibility of FMU variables for output: Until now, not all variables of the FMU can be seen in the Dynaplant output. FMU variables which shall enter the Dynaplant output have to be selected individually. A tree view of all FMU variables would be more usable.

3.3 First experiences

With the help of the co-simulation between FMU and in-house simulator, a clear improvement of the productivity was reached in comparison to the iterative procedure (by factor 5). Up to now the co-simulation seems to be robust, indeed, an easy Modelica model (with few states) was used that individually simulated also has no convergence problems.

A disadvantage is absolutely that the FMU is like a black box and it is very difficult to identify errors in the FMU model. Usability will be enhanced when all FMU variables are accessible in the Dynaplant output.

4 Conclusions

Coupling of Modelica models with in-house tools combines advantages of both specialized simulation tools and flexible Modelica models. The FMU export of the Modelica model is helpful, since it contains standard interfaces to exchange information with the model.

In our co-simulation approach, an FMU simulator is used to integrate the FMU model. Dynaplant takes control over the time step adjustments and uses set, run and get methods to exchange variables with the FMU simulator. This approach is realized and reveals promising advantages.

Next step will go beyond co-simulation: It will be even better to use the FMU gathering its equations in order to add them to the equation system of the specialized simulation tool. In contrast to co-simulation, the total model will be integrated by a single solver.

Tools importing FMUs should be enabled to generate meaningful error messages in case of license failures. Therefore we propose to add some license check method to the FMI.

Support by German Ministry BMBF (BMBF Förderkennzeichen: 01IS09029C) within the ITEA project OPENPROD [7] is gratefully acknowledged.

References

[1] K. Link, H. Steuer, A. Butterlin, Deficiencies of Modelica and its simulation environments for large fluid systems, Proceedings 7th International Modelica Conference, Como, Italy, Sep 20-22, 2009

[2] Functional Mock-up Interface for Model Exchange FMI_for_ModelExchange_v1.0.pdf (Jan 26, 2010)

[3] www.energy.siemens.com

[4] www.energy.siemens.com/hq/en/automation/power-generation/sppa-t3000.htm

[5] Dymola 7.4

[6] TISC from www.tlk-thermo.com

[7] www.openprod.org