Hydro Power Systems: Scripting Modelica Models for Operational Studies in Education

Dietmar Winkler*, Bernt Lie

Faculty of Technology, Telemark University College, P.O. Box 203, N-3901 Porsgrunn, Norway;
*Dietmar.Winkler@hit.no

Simulation Notes Europe SNE 23(3-4), 2013, 179 - 184
DOI: 10.11128/sne.23.tn.10214
Received: April 10, 2013 (Selected SIM 2012 Postconf. Publ.);
Revised Accepted: November 15, 2013;

Abstract. Telemark University College is offering a master degree program called 'Systems and Control Engineering'. Most students of that program have a background in either electrical, mechanical, control engineering or a combination of those. Since Norway covers about 99% of its electrical energy demand using hydro-electric power plants it is natural to also educate master students in the subject of hydro power systems.
About three years ago the Telemark University Colleges started a cooperation with the Norwegian power company 'Skagerak Energi' in order to offer real-life projects for students and to establish a new teaching course for second year master students called 'Modelling and Simulation of Hydro Power Systems'. That course teaches the students the basic principles of hydro-electric power generation starting the prediction of precipitation "down" to the distribution of electrical power in the grid with other loads and consumers connected to it.
This paper presents the teaching approach we have taken so far and our evaluations of opensource tools to be used within the 'Modelling and Simulation of Hydro Power Systems' course. The evaluations were also focused on possibilities of scripting model simulations.

1 Teaching Hydro Power Systems

1.1 Overview

Teaching hydro power systems gives one the great opportunity to deliver combined knowledge of at least three major engineering domains:

- Mechanical engineering
- Electrical engineering
- Control engineering

In detail the course deals with:

- Formulation of mathematical models across different physical domains (e.g., mechanical, electrical, hydrological).
- Introduction to the object-oriented modelling language Modelica.
- Development of a simple hydro power plant model which can be extended to more complex and accurate models.
- The benefit of using object orientation when implementing such models, with special emphasis on how the model can be gradually extended.

We use the modelling language Modelica which was especially designed for models which contain components from different physical domains. The benefit of using Modelica in teaching are for example:

Openness. Students can look at the exact equation based mathematical description of physical systems.

Multi-domain nature. In Modelica it is possible to connect the different domains (e.g., electrical, mechanical, control) within one model in order to get close representation of the real physical system.

Object-orientation. This Enhancing models in a "top-down" manner is very simple. This means students can start working on simple models and increase the level of detail later on.

1.2 Using modelling and simulation in projects

After having learnt about the mathematical and physical theory of hydro power systems, students can now apply that knowledge in working on operational studies. Those studies consist normally of real-life problems which need to be solved. In the past our students have for example worked on "Modelling and Optimisation of Deviation in Hydro Power Production"[1] and "Stability Analysis of AGC in the Norwegian Energy System"[2]. In the latter example it was especially important to use scripting and optimisation tools.

2 Modelling Tool Chain used so far

2.1 The modelling language Modelica

Modelica is a unified object-oriented language for systems modelling. It is developed by the *Modelica Association* which was founded in 1996 and consists of members from industry, university and rersearch organisations.

The *Modelica Association* also develops the free and open-source *Modelica Standard Library* (MSL) which is currently at version 3.2 and consist of 1280 non-trival models and 910 functions. The MSL makes it possible to generate models of complex systems in a simple and quick manner.

Since Modelica is especially suitable for multi-domain modelling and also because of the transparency of its models we decided to base our 'Modelling and Simulation of Hydro Power Systems' course on this powerful modelling language.

2.2 The modelling tools

In the course so far, we were using the commercial modelling tool *Dymola*. There are several reasons for this. One is that our students are mainly engineering students with little programming background. Since our hydro power systems course should mainly concentrate on the modelling and simulation tasks and not so much on the programming side we needed something that the students are comfortable working with and can learn within a reasonable short period of time. Basically this means that we needed a Modelica tool that allows to edit models graphically in a drag-and-drop manner rather than doing textual programming.

Another reason was that for the course we also liked to demonstrate the real power of Modelica with detailed models of a complete hydro power system. For this task we came across the `HydroPowerLibrary` which includes such complex models and easy to use examples.

2.3 Example from the `HydroPowerLibrary`

A typical example that the students model in the end of the course is a complete hydro power system as depicted in Figure 1 consisting of:

- Reservoir
- Waterway
- Turbine with turbine regulator
- Generator
- Power grid

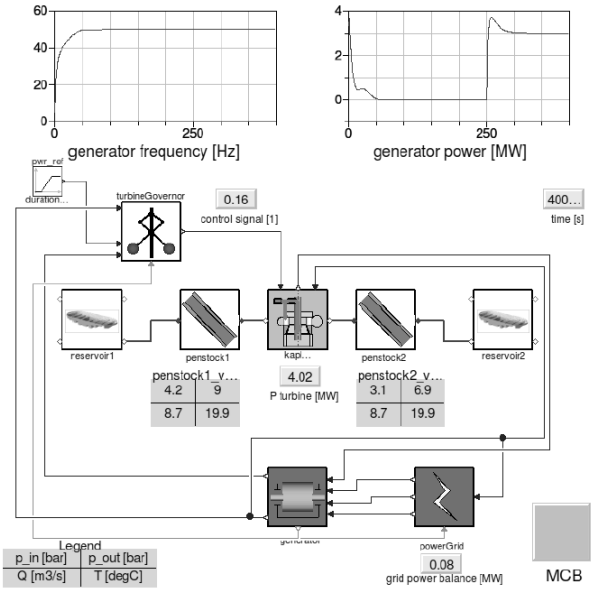

Figure 1: Screenshot from a `HydroPowerLibrary` example modelled in Dymola.

With such a model one can investigate the process of synchronising a generator that is driven by water turbine to the grid and then look at the power balance.

There are a lot of interesting aspects that the students can look into. E.g.,

- How aggressive should the turbine controller be?
- When can the electrical connection between the electrical generator and the electrical grid be made?
- What happens when it comes sudenly to a load change on the electrical grid?

And those are just some of the many scenarios that one can simulate with this model. One thing all of the different simulation scenarios have in common though is that one would like to automate the simulations with variations of some parameters, i.e., doing parameter sweeps.

2.4 Drawbacks of the commercial tools

Especially the automation of several simulations is something where *Dymola* was kind of weak or cumbersom to use. Also at this point the engineering students begin to see why it might be necessary and more convenient to be able to use a scripting language.

One of the most powerful scripting languages is *Python*. Unfortunately, the tool *Dymola* provides neither a convenient to use own scripting language nor does it provide a direct interface for Python. That is why we started to look at alternatives.

Another drawback, we as an academic institution see, is that students should be learning to use tools that they will also be able to use after they finished their degree at our university college. This might be a kind of moral aspect but a valid one none-the-less since many of our students come from countries where they basically can not afford to buy a licence (even when working at a company). It is also important for the students to be able to reproduce the results of their project and study work without restrictions after they have left higher education.

3 Going Open-Source in Modelling and Simulation

Using Modelica as an open modelling language is only the first step. We now looked into open-source tools that allows us to create the models in a convenient way, execute the simulation and do post-processing and optimisation. The most advanced open-source Modelica modelling and simulation tools are currently *OpenModelica* and *JModelica.org*.

First we looked at *OpenModelica* which already provides a graphical editor called “OMEdit”. Unfortunately that editor did not appear to be all that stable at the time of writing so we concentrated more on the script interface. Here *OpenModelica* provides the possibility to use *MetaModelica*, a special language that was developed not just for scripting but also programming the compiler itself [3]. As of version 1.8.1, *OpenModelica* also provides a beta version of Python Scripting.

JModelica.org is heavily reliant on Python and the whole simulation routine is controlled by using Python. Also does *JModelica.org* use the FMI standard [4] that offers the possibility to use exported models from other simulation tools. Thus we decided to start testing *JModelica.org* at first and wait with *OpenModelica* until the Python interface has become more mature.

Though not yet feature complete when it comes to the Modelica Language Specification [5] both tools are already powerful enough to simulate hydro power systems. However the remaining part of the paper shall present the experiences we made with *JModelica.org*.

3.1 Simplifying the models

The first thing we tried was exporting a `HydroPowerLibrary` model as a FMU and then later importing this into *JModelica.org*. Unfortunately this was not possible and we concluded this was possibly caused by either a nonstandard export on the one side or a not fully implemented import functionality on the other side.

However we continued with loading a simple `HydroPowerLibrary` model directly. Again this failed because of lacking support of certain functions used in the `HydroPowerLibrary` model.

In the end we decided to build a very simplified model that represents the functionality of a hydro power system consisting of a turbine and generator equivalent that is controlled by a turbine controller and is that then synchronised with the grid.

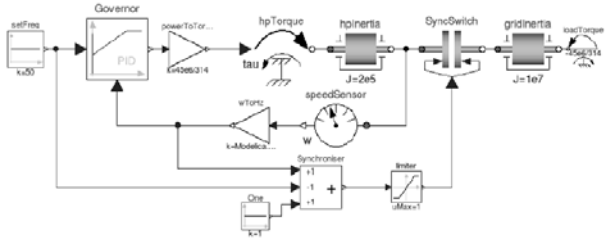

Figure 2: Screenshot of a simple system.

The `SimpleSystem` is depicted in Figure 2. The idea for this model is that one can look at the turbine and generator unit as torque source `hpTorque` that is used to accelerate their inertia `hpInertia`. The before an electrical generator can be connected with the electrical grid it needs to be synchronised. The process of synchronisation consists for several prerequisites:

- Same direction of rotation
- Same voltage level
- Same frequency

Now the simple model can only be used to simulate the frequency difference and the direction of rotation, i.e., the run-up of the generator. But this is actually sufficient for quite a lot of case studies.
When we only look at the active power balance then we can think of the electrical grid as a large inertia `gridInteria`. If the generated power and the load power are in balance then the grid inertia rotating at a constant frequency of 50Hz. Any electrical load can be represented via the `loadTorque` that can be calculated by:

$$T_{el} = \frac{P_{el}}{\omega^*} - 4\pi \qquad (1)$$

where T_{el} stands for the electrial torque, P_{el} for the electrical power and ω^* for the specific rotational velocity (depending on the number of poles in an electrical generator the angular velocity can vary and needs to be taken into account when calculating the rotatonal energy)

The last central component in the SimpleSystem is the synchronisation switch which is represented by a mechanical clutch `SyncSwitch` which closes when the frequencies of the generator and the grid are near enough. In this case we are starting to close the "switch" when the frequencies are within 1Hz of each other.

3.2 Simulation with *JModelica.org*

The simplified system from Figure 2 could almost be loaded into *JModelica.org*. The only thing that we needed to fix was that *JModelica.org* did not cope with some of the more advanced initialisation options used in the clutch model but not actually needed in our case.
Error messages that we needed to fix were:

```
The binding expression of the variable in-
itType does not match the declared type of
the variable
```

and

```
String variables are not supported
```

The simple solution was to simply remove those parts from the models used from the *Modelica Standard Library*. This is best achieved by doing as so called "save total" of the model and then manipulating the used models there. The following script will then generate a successful simulation of the `SimpleSystem` in *JModelica.org:*

```
# Import the function for compilation
# of models and the FMUModel class
from pymodelica import compile_fmu
from pyfmi import FMUModel

# Import the plotting library
import matplotlib.pyplot as plt

# Define model file name and class name
mofile = 'SimpleSystemTotal.mo'
model_name = 'SimpleSystem'

# Compile model
fmu_name = compile_fmu(model_name,mofile)
```

```
# Load model
grid = FMUModel(fmu_name)

# Simulate the model
res = grid.simulate(final_time=600)
f_gen = res['wToHz.y']
f_grid = res['gridInertia.w']
t = res['time']

# Generating the Plot
plt.figure(1)
plt.title('Synchronising a generator')
plt.ylabel('Frequency [Hz]')
plt.xlabel('Time [s]')
plt.plot(t, f_gen, t, f_grid)
plt.grid()
plt.show()
```

and the resulting plot can be seen in Figure 3.

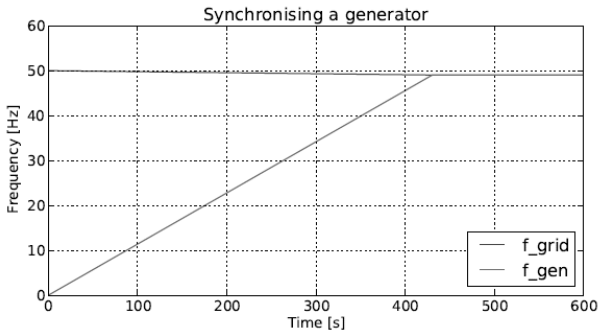

Figure 3: Simulation result from *JModelica.org*

3.3 Scripting and Optimisation

Now that we can run a simulation an extension for doing a parameter sweep can be easily achieved. It follows a variant of the previous simulation script only this time we run several simulations after each other in order to see the effect of having different hydro plant powers available (in the range of 40*MW*…140*MW*):

```
# Import the function for compilation
# of models and the FMUModel class
from pymodelica import compile_fmu
from pyfmi import FMUModel

# Import the plotting library
import matplotlib.pyplot as plt

# Import numpy
import numpy as np

# Define model file name and class name
mofile = 'SimpleSystemTotal.mo'
model_name = 'SimpleSystem'

# Compile model
fmu_name = compile_fmu(model_name,mofile)

# Load model
grid = FMUModel(fmu_name)

# Define initial conditions
p_var = 10
p_min = 40e6
p_max = 140e6
turbine_gain =
np.linspace(p_min,p_max,p_var)/
(2*np.math.pi*50)

# Setup of plot
plt.figure(1)
plt.hold(True)
plt.title('Synchronising a generator')
plt.ylabel('Frequency [Hz]')
plt.xlabel('Time [s]')

# Running the different simulations
for i in range(p_var):
   # Set initial conditions in model
   grid.set('turbineGain',turbine_gain[i])
   # Simulate
   res = grid.simulate(final_time=600)
   # Get Simulation result
   f_gen = res['wToHz.y']
   f_grid = res['gridInertia.w']/
   (2*np.math.pi)
   t = res['time']
   plt.plot(t, f_gen, t, f_grid)
   plt.grid()
plt.show()
```

Using this code we will get a plot like shown in Figure 4 where the different rising graphs represent the frequencies of the accelerated turbinegenerator unit. For example can one see that the starting power of $P_{gen} = 40W$ is in this case not enough to bring back the grid frequency to 50Hz.

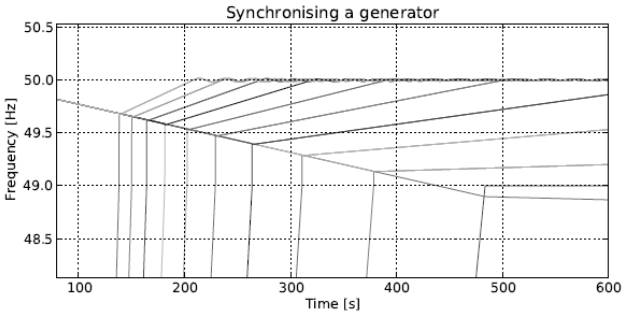

Figure 4: Simulation result of a simulation sweep with varying $P_{gen} = [40 \dots 140]MW$.

4 Conclusion

Our study has shown that is possible to simulate Hydro Power Systems with open-source tools that also allow for convenient scripting. However the there is still room for improvement both, on the compiler side in order to support more Modelica models, especially from the Modelica Standard Library. The other thing that is actually still lacking (but in development) in *JModelica.org* is a graphical editor. Without such a tool it will be hard to convince engineering students of the benefits and possibilities of Modelica and its rich modelling potentials. The scripting itself is thanks to Python very easy and quick to learn and the produced plots are thanks to `Matplotlib` also more advanced aswhat *Dymola* would be able to produce. To be honest, the open-source tools are not quite mature enough to allow us to completely switch our courses away from the commercial solutions we are currently using. But at least for student projects (i.e., where students can invest more time and energy) those offer a very interesting alternative and we are definitely continuing the evaluation as the tools keep improving all the time.

Acknowledgement

This contribution is a post-conference publication from SIMS 2012 Conference (53rd SIMS Conference, Reykjavík, Iceland, October 4 - 6, 2012). The contribution is a (partly) modified publication from the paper published in the Proceedings of SIMS 2012, published by Orkustofnun, National Energy Authority Iceland, ISBN: 978-9979-68-318-6, electronically available at http://www.scansims.org/sims2012/SIMS2012.pdf.

References

[1] Winkler D, Thoresen HM, Andreassen I, Perera MAS, Sharefi BR. Modelling and Optimisation of Deviation in Hydro Power Production. In: Modelica Association and Technische Universiät Dresden. *Proceedings of the 8th International Modelica Conference*; 2011 March 20022; Dresden, Germany. Doi: 10.3384/ecp1106318

[2] Andreassen I, Winkler D. Stability Analysis of AGC in the Norwegian Energy System. In: Dahlquist E, editor. *SIMS 2011. Proceedings of the 52nd Scandinavian Conference on Simulation and Modeling*; 2011 Sep 29030; Västerås, Sweden. Scandinavian Simulation Society, Mälardalen University; c2011. P.133–143

[3] Pop A, Fritzson P. Metamodelica: A unified equation-based semantical and mathematical modeling language. In: Lightfoot D and Szyperski C, editors. Modular Programming Languages, Lecture Notes in Computer Science, 4228. Berlin/Heidelberg: 2006. P.211– 229.

[4] Blochowitz T, Otter M, Arnold M, Bausch C, Clauß C, Elmqvist H, Junghanns A, Mauss J, Monteiro M, Neidhold T, Neumerkel D, Olsson H, Peetz J, Wolf S and Atego Systems Gmbh, Qtronic Berlin, Fraunhofer Scai, St. Augustin. The functional mockup interface for tool independent exchange of simulation models. In: *Proceedings of the 8th International Modelica Conference*, 2011 March 20022; Dresden, Germany. Linköping Electronic Conference Proceedings; c2011. P.105– 114

[5] Modelica Association, Modelica – *A Unified Object-Oriented Language for Physical Systems Modeling – Language Specification*, version 3.3 ed., 2012 Sep 05.