

Open-Source Distribution Transient Modeling with Modelica

T. A. Short, *Senior Member, IEEE*

Abstract—Modelica is a relatively new multi-domain simulation language that is well suited to modeling power system transients. This paper gives an introduction to Modelica for distribution system applications with some examples. Overhead line models are given as an example of using Modelica for developing components. An interface to Modelica simulations through R, an open-source processing environment, allows for plotting, parameter sweeps, component optimization, and Monte Carlo analysis.

Index Terms—Power distribution, modeling, EMTP, transients.

I. INTRODUCTION

Modelica is an object-oriented modeling language. In this paper, we will discuss using Modelica for modeling transients on power systems. Unlike various electromagnetic transients programs (EMTP's) that are available, Modelica is not designed specifically for power systems. It has true multi-domain capabilities and can model electrical, mechanical, and thermal systems. The Modelica language allows easy re-use of components because of its object-oriented design. The Modelica language and its usage is similar to VHDL-AMS (see McDermott [1] for power system usage of VHDL-AMS).

Modelica is a modeling language. Several modeling tools implement Modelica. Commercial tools include Dymola, MathModelica, MOSILAB, and SimulationX. Two open-source tools are also available: OpenModelica [2] and Scicos [3]. Many of the tools provide graphical interfaces for modeling components. Fig. 1 shows a simple circuit modeled in Dymola. Two books are available on simulating with Modelica [4, 5].

Modelica models describe differential, algebraic, and discrete equations. Simulation tools include these solvers. Different integration methods may be offered; DASSL by Petzold [6] is commonly used as the default.

A key feature of Modelica is that models are specified and connected like they are represented physically. This is referred to as non-causal or acausal modeling. The modeling tool processes Modelica models and identifies state variables automatically. This simplifies development of models.

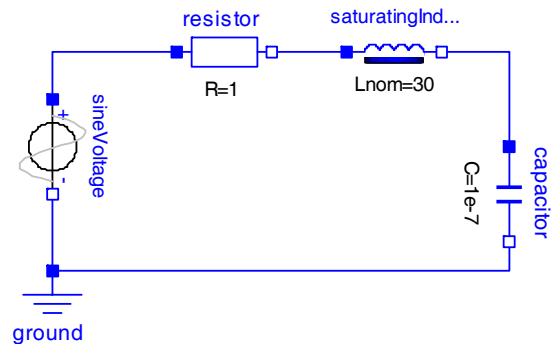


Fig. 1. Example ferroresonant circuit in Dymola.

The Modelica language is defined by the non-profit Modelica Association (www.modelica.org). The Modelica Association develops the specifications for Modelica and also develops the Modelica Standard Library. This library contains components for modeling mechanical, thermal, electrical, electronic, hydraulic, and control systems. The standard library is available under a liberal open-source license. Additional add-on libraries are also hosted at modelica.org; many of them are open source.

II. POWER SYSTEM MODELING

The Modelica Standard Library contains several components for power system modeling. The Electrical.MultiPhase subsystem includes multiphase components for circuit elements like capacitors, transformers, voltage sources, and more. The Electrical.Machines subsystem includes models for several ac, dc, and induction machines.

One of the earliest Modelica libraries for power-system modeling was the ObjectStab library by Larsson [7]. ObjectStab implements a transient stability solution in Modelica.

The most comprehensive Modelica power-system library to date is the open-source SPOT library developed by Wiesmann [8]. SPOT includes models for many power system components, including various machines, transformers, lines, loads, and inverters. It also includes arc models, circuit breakers, and fault models. The SPOT library has different simulation modes: it has a transient solution mode where the full dynamics are modeled as well as a steady-state simulation mode which is more like a transient stability solution (it

assumes a synchronous reference frame).

Other power-system related work includes PQLib, a library for power quality with models for drives by Kalaschnikow [9]. Svensson [10] has also modeled distributed generation, including renewable energy. Models for generation, power electronics, and control were used to analyze distributed energy operation and control. Schoder et al. [11] have used Modelica to model shipboard power systems.

Several groups have implemented libraries for a wide range of applications that are complementary to power-system modeling. These include several spice-type circuit libraries [12], fuel-cell libraries [13], hybrid-electric vehicle models [14], magnetic actuators [15], and batteries and supercapacitors [16]. Other possibly useful libraries include those for fluid flow in pipes and various mechanical libraries. Many of these are open source.

III. A SIMPLE FERRORESONANCE EXAMPLE

This section discusses a simple representation of a ferroresonant circuit with some capacitance in series with a nonlinear inductance. Fig. 1 shows how the model can be entered graphically. The Modelica code for this example is shown below. The code was modified somewhat for readability from the original that Dymola generated from its graphical interface, but not much. It is generally quite readable. The Dymola results for the capacitor voltage for this model are in Fig. 2.

```
model SimpleFerro "Ferroresonance example"
  import A = Modelica.Electrical.Analog;
  A.Sources.SineVoltage
    sineVoltage(V=7200, freqHz=60);
  A.Basic.Ground ground;
  A.Basic.Resistor resistor(R=1);
  A.Basic.SaturatingInductor
    saturatingInductor(Inom=.8, Lnom=30);
  A.Basic.Capacitor capacitor(C=1e-7);
equation
  connect(ground.p, sineVoltage.n);
  connect(sineVoltage.p, resistor.p);
  connect(resistor.n, saturatingInductor.p);
  connect(saturatingInductor.n, capacitor.p);
  connect(capacitor.n, ground.p);
end SimpleFerro;
```

Most Modelica models have a section where the components and other variables are defined and an equation section that describes the interactions of the model. In this simple case, the equation section just specifies how components are connected together. The function `connect` is used to “connect” connectors. Connectors are an important part of Modelica. In this example, these are electrical connectors that are defined such that the voltages of connected connectors are set to be equal, and currents in and out of these connectors are set to sum to zero. Modelica supports many types of connectors, including multiphase electrical connectors, mechanical shafts, and hydraulic lines, and it is possible to add additional connector types.

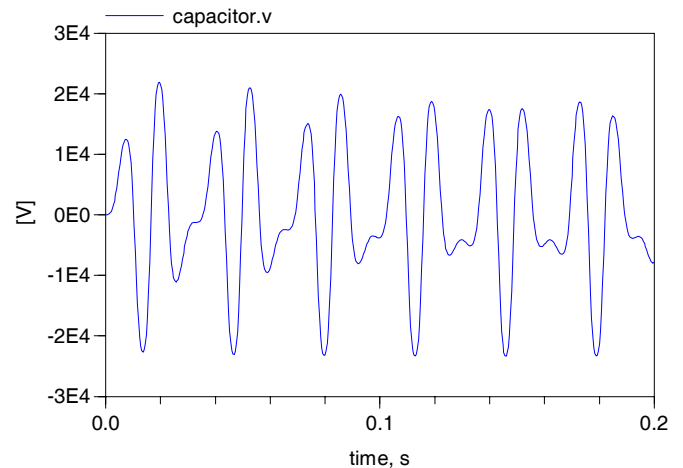


Fig. 2. Capacitor voltage for the ferroresonance example.

IV. SURGE ARRESTERS

In many focused transient modeling tools, adding custom models is difficult, and added models may run more slowly than built-in models. Adding models is a strength of Modelica, and added models run as fast as built-in models as almost all built-in models are themselves defined in Modelica. Modelica allows components to be grouped in packages and subpackages. As an illustration of model building, we will look at models for surge arresters (the Modelica Standard Library does not have a built-in model for surge arresters). A simple exponential model of an arrester can be defined as follows:

```
model Exponential "Simple arrester model"
  extends
    Modelica.Electrical.Analog.Interfaces.OnePort;
  parameter Modelica.SIunits.Voltage
    Vdischarge(final min=0) = 30.E3
    "10-kA discharge voltage";
  parameter Real Exponent(final min=1) = 26
    "Exponent coefficient";
equation
  i = sign(v) * (v / Vdischarge)^Exponent *
    10000;
  annotation (...);
end Exponential;
```

Beyond some housekeeping code, the main part of the code is one line specifying the voltage-current relationship in the equation section. Models can also be developed by piecing together components. See Fig. 3 for an implementation of an arrester model that was developed by an IEEE working group [17]. The model is composed of resistors, inductors, a capacitor, and nonlinear resistors A0 and A1 which are each nonlinear resistor models that interpolate on a log scale between a table of voltages and currents.

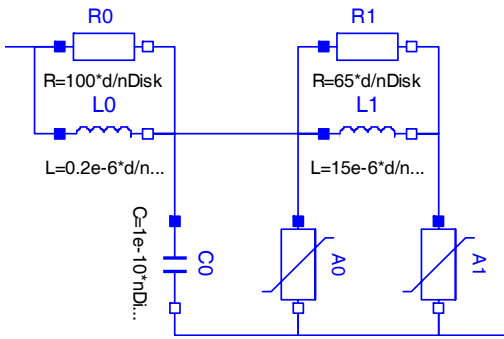


Fig. 3. IEEE surge arrester model.

V. LINE MODELS

Power system line models are key to many types of distribution and transmission modeling scenarios. Modelica does not have the equivalent of EMTP's line constants routine, and built-in line models are rudimentary. Modelica does have the numeric processing capability to implement line constants.

A set of models has been developed for overhead lines. These include series impedance, pi, and lossless and lossy modal parameter models for overhead lines. All are derived from a partial model that calculates the impedance and admittance matrices from line parameters. Modelica has eigen-value and other matrix functions, which are needed for the line constants estimations. The series impedance model is very simple, with almost a one-line equation section as follows. The `der` function indicates the derivative. Modelica allows vectorized equations; each of the voltages and currents shown below are vectors with one element for each conductor.

$$L * \text{der}(p.i) + R * p.i = p.v - n.v;$$

The modal line model is not much more complicated. The surge impedances and modal delays are calculated from the impedance and admittance matrices. The core part of the lossless modal model equation section is:

```

for i in 1:cond.nc loop
  Imode_in_p[i] = Imode_out_p[i] -
    delay(Imode_out_n[i], delays[i]);
  Imode_in_n[i] = Imode_out_n[i] -
    delay(Imode_out_p[i], delays[i]);
  Vmode_p[i] / Zsurge[i] = Imode_out_p[i] +
    delay(Imode_out_n[i], delays[i]);
  Vmode_n[i] / Zsurge[i] = Imode_out_n[i] +
    delay(Imode_out_p[i], delays[i]);
end for;

```

Dymola automatically creates the user interface from the component specifications. For example, the input parameters for an overhead pi model are given in Fig. 4. Several conductor configuration are possible. Fig. 5 shows an example for a four-conductor system common for distribution circuits.

Fig. 6 shows an example built in Dymola for a four-conductor ungrounded system with a line-to-neutral fault at the end. Fig. 7 shows the fault current from this simulation.

4-Conductor Overhead Line Configuration	▶	Conductor configuration
60	▶	Hz Frequency, Hz
100	▶	Ohm.m Earth resistivity, ohm-m
10000	▶	m Line length, m
10	▶	Number of segments

Fig. 4. General overhead line parameters.

Parameters		
phase1	1	▶ Phase
x1	1	▶ m X position
y1	10	▶ m Y position
cond1	AAC 500 kcmil	▶ Conductor
phase2	2	▶ Phase
x2	0	▶ m X position
y2	10	▶ m Y position
cond2	AAC 500 kcmil	▶ Conductor
phase3	3	▶ Phase
x3	-1	▶ m X position
y3	10	▶ m Y position
cond3	AAC 500 kcmil	▶ Conductor
phase4	4	▶ Phase
x4	0	▶ m X position
y4	8	▶ m Y position
cond4	AAC 2/0	▶ Conductor

Fig. 5. Parameters of a four-conductor line.

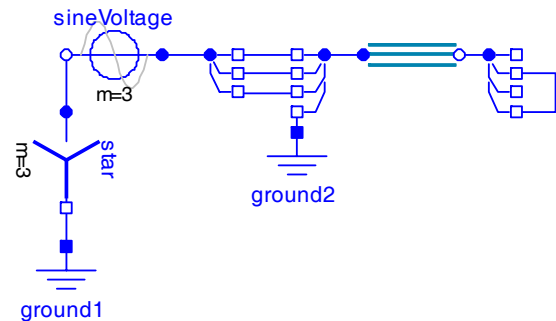


Fig. 6. Four-conductor line model example with a line-to-neutral fault.

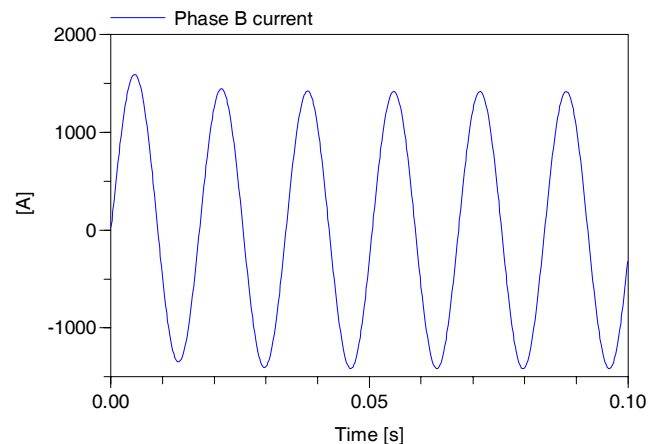


Fig. 7. Fault current on phase B.

Current plans are to further develop the line models discussed in this paper as open-source modules as an add-on to the Modelica Standard Library and/or as an addition to the SPOT library. The design of Modelica allows easy additions to this library. For example, if a routine was developed to derive the series impedance and shunt admittance of cables, these could be plugged right into existing models to produce pi models and modal models.

VI. BATCH PROCESSING AND OPTIMIZATION

Modelica tools normally compile a model into C. This allows Modelica simulations to approach speeds of domain-specific, targeted simulation systems like EMTP. Both OpenModelica and Dymola compile to C and then to an executable. The resulting executable reads model inputs and simulation parameters from a file and then outputs the simulation results to a file. This processing cycle allows for simulation control from external systems. For example, Dymola has an interface where Matlab can control the simulation and read the results. Dymola stores results in a Matlab-compatible *.mat file.

In this section, we describe a similar interface using R, an open-source processing environment based on the S language [18]. R is a command-line language. A simple use for the Dymola interface looks like:

```
results = dymSimulate(C = 20e-9)
plot(results, "capacitor.v",
      "saturatingInductor.v")
```

The `dymSimulate` command reads in the default simulation file specification (dsin.txt for Dymola) and writes out a replacement with the capacitance changed from the default value to 20 nF. Then, the simulation is run, and the results are loaded in the R variable called results.

This functionality can be used for parameter sweeps, component optimization, Monte Carlo analysis, bootstrapping variability, and control algorithm tuning, all of which are suited for a system like R. The following example shows a parameter sweep for the ferroresonance example in Fig. 1. The capacitance is varied, and the peak capacitor voltage is recorded. The results of this script are in Fig. 8.

```
C = seq(from=1,to=50,length=20)*1e-9
for (i in 1:length(C)) {
  results = dymSimulate(C = C[i])
  voltage = results$capacitor.p.v
  peakVoltage[i] = max(voltage)
}
plot(C*1e9, peakVoltage/1000,
     xlab="Capacitance, nF",
     ylab="Peak voltage, kV",
     type="b", pch=20)
```

A similar interface has also been created for OpenModelica with a similar calling sequence as the Dymola interface.

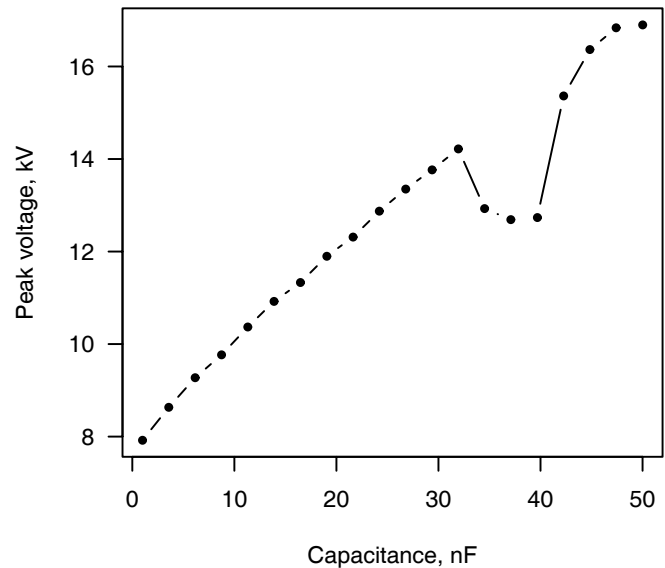


Fig. 8. Variation of peak capacitor voltage with the capacitance.

Because tools compile Modelica to C, it is possible to use the translated model in other applications. This makes it possible to make simplified web-based or Visual Basic-type interfaces to transient simulations. Modelica also defines an interface to external routines written in C or Fortran, so it is easy for Modelica models to link to numeric or other procedures written in other languages.

VII. SUMMARY

Modelica is well suited for modeling power system transients. A number of different groups have used Modelica for power system modeling. Ease of building models makes it a good candidate for continued growth. Many of the most successful open-source software systems make it easy for users to become developers. This can be seen with R, now with over 1000 add-on packages available. Because Modelica is open, all models can be inspected and reused. As the user progresses and builds additional components and subcomponents, libraries of these can be useful to others. Modelica is also a good research tool for many of these same reasons.

REFERENCES

- [1] T. McDermott, R. Juchem, and D. Devarajan, "Distribution Feeder and Induction Motor Modeling with VHDL-AMS," *PES TD 2005/2006*, pp. 141-146, 2006.
- [2] P. Fritzson, P. Aronsson, H. Lundvall, K. Nyström, A. Pop, L. Saldamli, and D. Broman, "The OpenModelica Modeling, Simulation, and Development Environment," *Proceedings of the 46th Conference on Simulation and Modeling*, pp. 83-90, 2005.
- [3] M. Nikoukhah, "Modeling and simulation of differential equations in Scicos," presented at Modelica Conference, 2006.
- [4] M. Tiller, *Introduction to Physical Modeling With Modelica*: Kluwer Academic Pub, 2001.
- [5] P. Fritzson, *Principles of Object-Oriented Modeling and Simulation with Modelica*: Wiley-IEEE, 2004.

- [6] L. Petzold, "Description of DASSL: a differential/algebraic system solver," *10. international mathematics and computers simulation congress on systems simulation and scientific computation*, vol. 9, 1982.
- [7] M. Larsson, "ObjectStab-an educational tool for power system stability studies," *Power Systems, IEEE Transactions on*, vol. 19, pp. 56-63, 2004.
- [8] H. J. Wiesmann, "SPOT, Simulator of Power System Transients Library," 2007. <http://www.modelica.org/libraries/spot>.
- [9] S. Kalaschnikow, "PQLib A Modelica Library for Power Quality analysis in Networks," *Proceedings of the 2nd International Modelica Conference*, 2002.
- [10] J. Svensson, "Active Distributed Power Systems," Lund University, 2006. <http://www.iea.lth.se/publications/Theses/LTH-IEA-1050.pdf>.
- [11] K. Schoder and A. Feliachi, "Object-Oriented Modeling and Simulation of AC/DC Systems," presented at IEEE Power Engineering Society General Meeting, 2007.
- [12] F. E. Cellier, C. Clauß, and A. Urquía, "Electronic Circuit Modeling and Simulation in Modelica," presented at EUROSIM, Ljubljana, Slovenia, 2007.
- [13] M. Rubio, A. Urquía, L. González, D. Guinea, and S. Dormido, "FuelCellLib -- A Modelica Library for Modeling of Fuel Cells," presented at 4th International Modelica Conference, 2005.
- [14] J. Hellgren, "Modelling of Hybrid Electric Vehicles in Modelica for Virtual Prototyping," presented at 2nd International Modelica Conference, 2002.
- [15] T. Bödrieh and T. Roschke, "A Magnetic Library for Modelica," presented at 4th International Modelica Conference, 2005.
- [16] E. Surewaard, E. Karden, and M. Tiller, "Advanced Electric Storage System Modeling in Modelica," presented at 3rd International Modelica Conference, 2003.
- [17] IEEE Working Group 3.4.11, "Modeling of Metal Oxide Surge Arresters," *IEEE Transactions on Power Delivery*, vol. 7, pp. 302-9, 1992.
- [18] R Development Core Team, *R: A Language and Environment for Statistical Computing*, 2007. ISBN 3-900051-07-0, <http://www.R-project.org>.

Tom A. Short is a part of an EPRI office in Saratoga County, NY. Before joining EPRI in 2000, he worked for Power Technologies, Inc. for ten years. Mr. Short has a Master's of Science degree in Electrical Engineering from Montana State University (1990). Mr. Short authored the *Electric Power Distribution Handbook* (CRC Press, 2004). In addition, he led the development of IEEE Std. 1410-1997, *Improving the Lightning Performance of Electric Power Overhead Distribution Lines* as the working group chair. For this effort, he was awarded the 2002 IEEE PES Technical Committee Distinguished Service Award. He developed the open-source Rpad engineering analysis interface (www.rpad.org) that has been used mainly for analyzing utility reliability databases.