M. Larsson:
ObjectStab - A Modelica Library for Power System Stability Studies.
Modelica Workshop 2000 Proceedings, pp. 13-22.

Paper presented at the Modelica Workshop 2000, Oct. 23.-24., 2000, Lund, Sweden.

All papers of this workshop can be downloaded from
http://www.Modelica.org/modelica2000/proceedings.html

Workshop Program Committee:
- Peter Fritzson, PELAB, Department of Computer and Information Science, Linköping University, Sweden (chairman of the program committee).
- Martin Otter, German Aerospace Center, Institute of Robotics and Mechatronics, Oberpfaffenhofen, Germany.
- Hilding Elmqvist, Dynasim AB, Lund, Sweden.
- Hubertus Tummescheit, Department of Automatic Control, Lund University, Sweden.

Workshop Organizing Committee:
- Hubertus Tummescheit, Department of Automatic Control, Lund University, Sweden.
- Vadim Engelson, Department of Computer and Information Science, Linköping University, Sweden.

ObjectStab - A Modelica Library for Power System Stability Studies

Mats Larsson, Student Member
Dept. of Industrial Automation
Lund University
Sweden

September 20, 2000

Abstract

Traditionally the simulation of transient and voltage stability in power systems has been constrained to domain-specific tools such as Simpow, PSS/E, ETMSP and EuroStag. While being efficient and thereby able to simulate large systems, their component models are often encapsulated and difficult or impossible to examine and modify. Also, these simulators often require substantial training and are therefore unsuitable for normal classroom use. For academic and educational use, it is more important that the component modelling is transparent and flexible, and that students can quickly get started with their simulations. This paper describes a freely available power system library called ObjectStab intended for voltage and transient stability analysis and simulation written in Modelica, a general-purpose object-oriented modelling language. All component models are transparent and can easily be modified or extended. Power system topology and parameter data are entered in one-line diagram form using a graphical editor. The component library has been validated using comparative simulations with EuroStag.

1 Introduction

The simulation of power systems using special-purpose tools is now a well-established area and several commercial tools are available, e.g. Simpow, PSS/E and EuroStag. While most of these tools are computationally very efficient and reasonably user-friendly they have a closed architecture where it is very difficult or impossible to view or change most of the component models. Some tools, e.g. EuroStag, provide the possibility of modelling controllers such as governors and exciters using block diagram representation but this is often cumbersome. Moreover, these constructs do not enable the user to modify any of the generator or network models. The Power System Toolbox [1] is a set of Matlab program files capable of dynamical simulation of power systems. Here the component models are accessible, but the modification of these requires a proper understanding of the interaction of the model files in this specific environment. Graphical editing of the models is not possible. Some of the above mentioned tools have the capability of exporting a linearised representation of the system for further analysis but the full nonlinear representation remains hidden to the user. Also, most of the industrial-grade tools require substantial training before they can be used productively.

Modelica [2, 3] is an object-oriented general-purpose modelling language that is under development in an international effort to define a unified language for modelling of physical systems. Modelica supports object-oriented modelling using inheritance concepts taken from computer languages such as Simula and C++. It also supports non-causal modelling, meaning that a model's terminals do not necessarily have to be assigned an input or output role. While object-oriented concepts enable proper structuring of models, the capability of non-causal modelling makes it easy to model for example power lines that are very cumbersome to model using block-oriented languages such as Simulink.

Causality is generally not defined in electrical systems [4]. For example, when modelling a resistor, it is not evident ahead of time, whether an equation of the type

$$u = R\, i \tag{1}$$

will be needed, or one of the form:

$$i = \frac{u}{R} \tag{2}$$

It depends on the environment in which the resistor is embedded. Consequently the modelling tool should relax the causality constraint that has been imposed on the modelling equations in the past. Using causal modelling as in Simulink that does not have the capability of relaxing such constraints is illustrated in [5] where a model for transient stability simulation is described. With this approach, the network has to be modelled using the traditional admittance matrix method and the power system topology can not be visualized in the graphical editor.

In Modelica, models can be entered as ordinary or differential-algebraic equations, state-machines and block diagrams etc. Modelica also supports discrete-event constructs for hybrid modelling. Object-oriented modelling of power systems using general purpose simulation languages has previously been described in [6], where a block library was developed using the modelling language Omola. However, the Omola project is now discontinued and the experiences from this have been brought into the Modelica project.

At the time of publication, the simulation tool Dymola [7] was the only tool supporting the Modelica language although some other producers of general-purpose simulators have announced that future versions of their programs will support Modelica. There is already a number of Modelica libraries for various application domains such as; 3-dimensional multi-body systems, rotational and translational mechanics, hydraulics, magnetics, thermodynamical systems and chemical processes.

This paper describes a component library written in Modelica for the simulation of transient and voltage stability in power systems. It is designed for use by undergraduate and graduate students in the teaching of power system dynamic stability and for rapid testing of research ideas. All component models are transparent and can easily be modified or extended. Emphasis has been given to keeping the component models transparent and simple. Power system topology and parameter data are entered in one-line diagram form using a graphical editor. Using the Dymola simulator, the full hybrid nonlinear or linearities equation system can be exported for use as a block in Simulink simulations and the full range of MATLAB tools for visualization and control design can then be used. The component library has been validated using comparative simulations with EuroStag [8].

2 The Component Library

The ObjectStab library presently contains the following component models :

- Generators with constant frequency and voltage as slack or PV nodes, or using 3rd or 6th order dq-models with detailed models of excitation and governor control systems, or as quasi-steady state equivalents according to [9] .
- Transmission lines in pi-link or series impedance representation.
- Reactive power compensation devices; shunt reactors, shunt capacitances and series capacitances according to [10].
- Fixed ratio transformers.
- On-load tap changing transformers (OLTC) modelled as detailed discrete models or using their corresponding continuous approximations according to [11].
- Static and dynamic loads, including generic exponential recovery loads [12].
- Buses.
- Faulted lines and buses with fault impedance.

Standard assumptions for multi-machine power system stability simulations are made, i.e., generator stator and network time constants are neglected and voltages and currents are assumed to be sinusoidal and symmetrical. Except where other references are given, the components are modelled according to the guidelines given in [10]. Furthermore each generator's set of equations are referred to an individual dq-frame and the stator equations are related to the system (common) reference frame through the so-called Kron's transformation [10].

All models can be modified and extended through inheritance. New models or extensions of old ones can be entered as differential and algebraic equations, block diagrams or state-graphs. Voltages and currents are described by their phasor representation:

$$\mathbf{I} = i_a + j i_b \qquad (3)$$
$$\mathbf{V} = 1 + v_a + j v_b \qquad (4)$$

Using this representation a connection point for power system components can be defined by the following Modelica connector definition

```
connector Pin
  Real va;
  Real vb;
  flow Real ia;
  flow Real ib;
end Pin;
```

By definition all `Pins` use a p.u. system based on the system reference of 100 MVA, although the generator models support parameter entry on their own individual base.

2.1 Modelling example: Impedance line model

The inheritance hierarchy for the impedance transmission line model is given below:

```
partial model TwoPin
  Pin T1;
  Pin T2;
end TwoPin;
```

The definition implies that a `TwoPin` is a model with two pins named `T1` and `T2` that will be used as external connectors for the line. The `Impedance model` is defined using the `TwoPin` as a base class and thereby inherit its attributes. Furthermore equations defining the real and imaginary voltage drop on the line have to be given:

```
model Impedance
  extends Base.TwoPin;
  parameter Base.Resistance R=0.0 "Resistance";
  parameter Base.Reactance X=0.1 "Reactance";
equation
  [T1.va-T2.va; T1.vb-T2.vb] = [R, -X;
                                X,  R]*[T1.ia; T1.ib];
  [T1.ia; T1.ib] + [T2.ia; T2.ib] = [0; 0];
end Impedance;
```

2.2 Modelling example: 3rd order Generator Model

The dynamic generator models inherit from the `OnePin` class and its declarations are split into three classes in order to simplify the implementation of new generator models. The class `DetGen` defines the swing equation and the coordinate transformations as follows:

```
partial model Partials.DetGen
  extends Base.OnePin;
  parameter Base.Voltage V0=1;
  parameter Base.ActivePower Pg0=1;
  parameter Base.VoltageAngle theta0=0;
  parameter Boolean isSlack=true;
  parameter Base.ApparentPower Sbase=100;
  parameter Base.InertiaConstant H=6;
  parameter Base.DampingCoefficient D=0;
  Base.AngularVelocity w(start=1);
  Base.VoltageAngle delta;
  Base.MechanicalPower Pm;
  Base.VoltageAmplitude Efd;
  Base.Current id,iq;
  Base.Voltage vd,vq;
  Base.Current Iarm=sqrt(id^2 + iq^2);
  Base.ActivePower Pe;
equation
  //   swing equations
  der(w) = 1/(2*H)*(Pm-Pe-D/Base.ws*(w-Base.wref));
  der(delta) = Base.ws*(w - Base.wref);
  // Kron's transformation
  -[T.ia; T.ib] = [-sin(delta), cos(delta);
                   cos(delta), sin(delta)]*[id; iq];
  // Load-flow initialization equation
  if initial() then
    if isSlack then
      1 + T.va = V0*cos(theta0);
      T.vb = V0*sin(theta0);
    else
      V = V0;
      Pg = Pg0;
    end if;
  else
    [1 + T.va; T.vb] = [-sin(delta), cos(delta);
                        cos(delta), sin(delta)]*[vd; vq];
  end if;
end DetGen;
```

During the initialization which is equivalent to a load-flow calculation, generators are represented as either PV-nodes or Slack nodes [10] depending on the attribute `isSlack`. It is declared as a partial class, i.e. non-instantiable, since it lacks equations for the stator and EMF equations that are dependent on the type of generator model used. For the 3rd order generator model with a single transient EMF in the q-axis the definition is as follows:

```
model ObjectStab.Generators.Partials.DetGen3
  extends DetGen;
  parameter Base.Resistance ra=0;
  parameter Base.Reactance xd=0.8948;
  parameter Base.Reactance xq=0.84;
  parameter Base.Reactance xdp=0.30;
  parameter Base.Time Td0p=7;
  Base.Voltage Eqp(start=1);
equation
  // Transient EMF equation
  Td0p*der(Eqp) = Efd - Eqp + id*(xd - xdp);
  // stator equations
  vd = -ra*id - xq*iq;
  vq = Eqp + xdp*id - ra*iq;
  // electrical power
  Pe = Eqp*iq + (xdp - xq)*id*iq;
end DetGen3;
```

Also `DetGen3` is declared as a partial class since it does not contain equations for the governor and excitation system. These are provided in the instantiable class `GovExc3rdGen` which contains models of the governor and excitation system.

```
model GovExc3rdGen
  extends Partials.DetGen3;
  replaceable Controllers.ConstPm Gov;
  replaceable Controllers.ConstEfd Exc;
```

```
equation
  // connection equations for governor and exciter
  Gov.u = w;
  Pm = Gov.y;
  Exc.u1 = sqrt(vd^2 + vq^2);
  Exc.u2 = w - Base.wref;
  Efd = Exc.y;
  // initialization
  when initial() then
    reinit(w, Base.wref);
    reinit(delta, atan2(T.vb-ra*T.ib-xq*T.ia,
                        1.0+T.va-ra*T.ia+xq*T.ib));
    reinit(id, -(-sin(delta)*T.ia+cos(delta)*T.ib));
    reinit(iq, -(cos(delta)*T.ia+sin(delta)*T.ib));
    reinit(Eqp,(Pg+ra*(id^2+iq^2)-id*iq*xdp+id*iq*xq)/iq);
    reinit(Gov.Pm0,Pg + ra*(id^2 + iq^2));
    reinit(Exc.Ef0,(Pg+ra*(id^2+iq^2)+id*iq*xq-id*xd*iq)/iq);
    reinit(Exc.Vref, V + Exc.Ef0/Exc.K);
  end when;
end GovExc3rdGen;
```

This model also contains equations for the initialization of the generator dynamic states from the values given by the `Pin` variables. The values of the `Pin` variables are automatically calculated in the load-flow that precedes every simulation. Since the Governor and Exciter models have been declared using the modifier `replaceable`, they can be replaced for models taken from a controller library or for user-defined models in instantiations of the `GovExc3rdGen` class.

3 Extending the Library

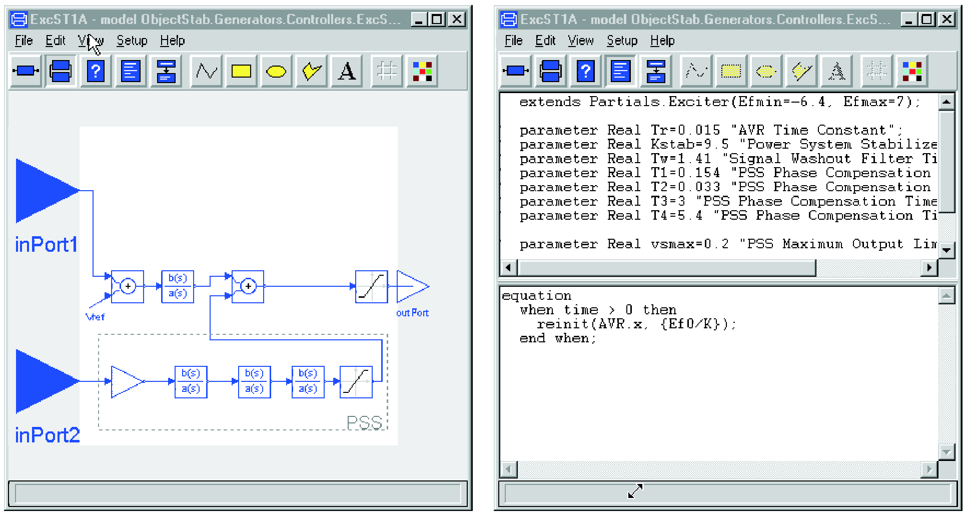
The goal has been to provide a framework with the basic models necessary for power system stability studies. However, such a library can never be complete and users can create their own models inheriting from the (partial) base classes and providing the equations given below:

- Generic current injector models `OnePin` and `TwoPin` - equations for each of the real and imaginary currents of each connector (`Pin`).
- Loads - equations for active and reactive load.
- Generators - equations for active and reactive production.
- Governor, Exciter - Generator control systems, equations for the mechanical shaft power or the field voltage must be given.

The equations in user-defined model can be defined by block-diagrams, state-graphs or differential or algebraic equations. Internal dynamic states in user-defined models should be initialized from their terminal variables, similarly to the generator model in Sec. 2.2.

3.1 Extension Example: Exciter IEEE ST1A [13]

New exciter models can be constructed by creating a new class that inherits from the partial class `Exciter`. This partial class contains the declarations necessary for communication with the generator models and calculation of the initial values of the field voltage and voltage reference. The block diagram can then be drawn using components from the block libraries as shown in Figure 1, or alternatively, the corresponding differential or algebraic equations can be entered in the equation window of the new model. The design of new governor system is analogous. Note that also equations for dynamic blocks which at the start of the simulation have non-zero output must be given as shown in Figure 1.


ExcST1A - model ObjectStab.Generators.Controllers.ExcS...
File Edit View Setup Help
inPort1
inPort2
Vref
outPort
PSS
b(s)
a(s)

```
  extends Partials.Exciter(Efmin=-6.4, Efmax=7);

  parameter Real Tr=0.015 "AVR Time Constant";
  parameter Real Kstab=9.5 "Power System Stabilize
  parameter Real Tw=1.41 "Signal Washout Filter Ti
  parameter Real T1=0.154 "PSS Phase Compensation
  parameter Real T2=0.033 "PSS Phase Compensation
  parameter Real T3=3 "PSS Phase Compensation Time
  parameter Real T4=5.4 "PSS Phase Compensation Ti

  parameter Real vsmax=0.2 "PSS Maximum Output Lim
```

```
equation
  when time > 0 then
    reinit(AVR.x, {Ef0/K});
  end when;
```

Figure 1: Diagram and equation windows for the IEEE ST1A Excitation system model [13].

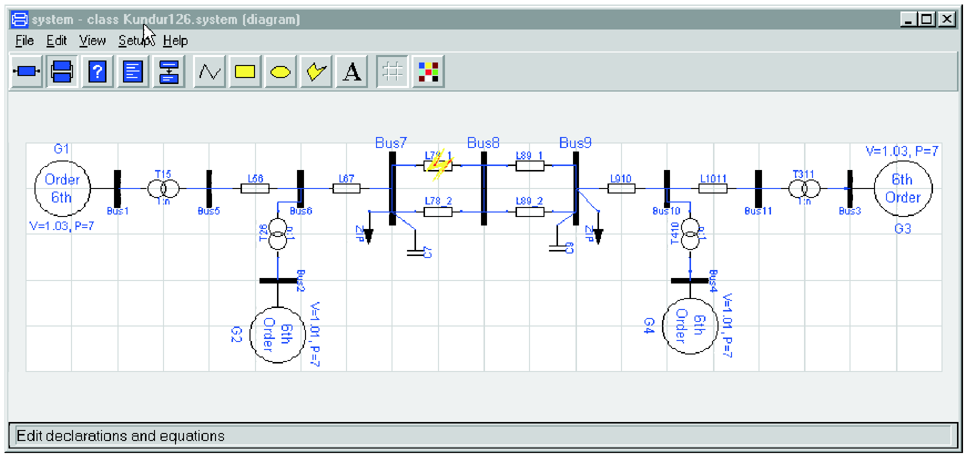
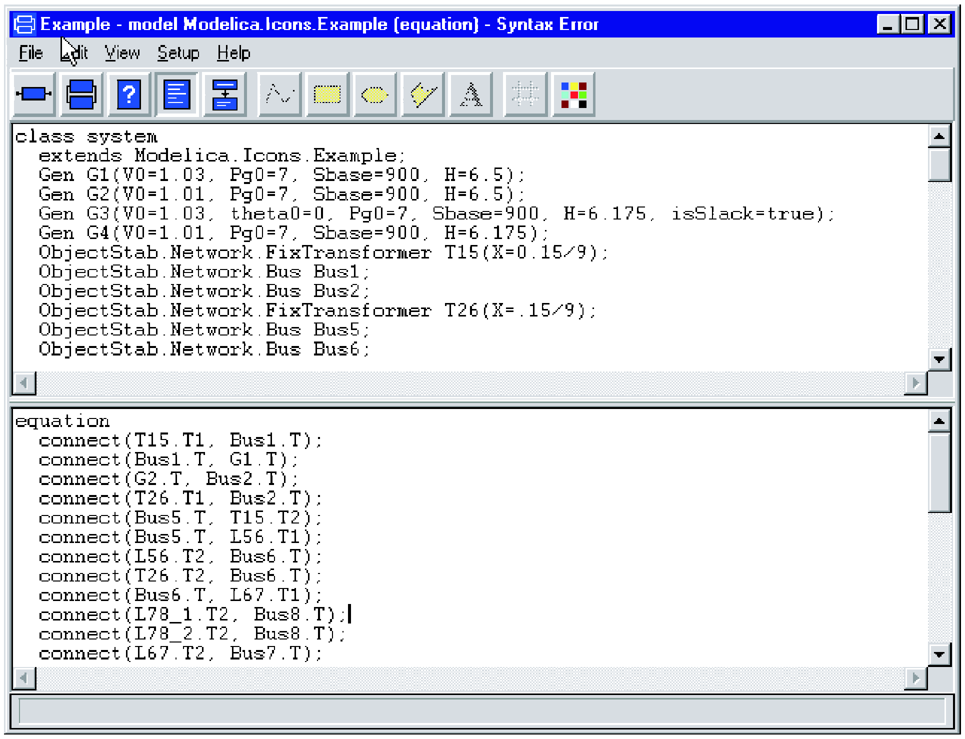


Figure 2: The four-generator test system from [13].


Example - model Modelica.Icons.Example (equation) - Syntax Error
File
Edit
View
Setup
Help

```
class system
  extends Modelica.Icons.Example;
  Gen G1(V0=1.03, Pg0=7, Sbase=900, H=6.5);
  Gen G2(V0=1.01, Pg0=7, Sbase=900, H=6.5);
  Gen G3(V0=1.03, theta0=0, Pg0=7, Sbase=900, H=6.175, isSlack=true);
  Gen G4(V0=1.01, Pg0=7, Sbase=900, H=6.175);
  ObjectStab.Network.FixTransformer T15(X=0.15/9);
  ObjectStab.Network.Bus Bus1;
  ObjectStab.Network.Bus Bus2;
  ObjectStab.Network.FixTransformer T26(X=.15/9);
  ObjectStab.Network.Bus Bus5;
  ObjectStab.Network.Bus Bus6;
```

```
equation
  connect(T15.T1, Bus1.T);
  connect(Bus1.T, G1.T);
  connect(G2.T, Bus2.T);
  connect(T26.T1, Bus2.T);
  connect(Bus5.T, T15.T2);
  connect(Bus5.T, L56.T1);
  connect(L56.T2, Bus6.T);
  connect(T26.T2, Bus6.T);
  connect(Bus6.T, L67.T1);
  connect(L78_1.T2, Bus8.T);
  connect(L78_2.T2, Bus8.T);
  connect(L67.T2, Bus7.T);
```

Figure 3: Textual representation of the four-generator test system.

4 Case Study: Four Generator Test System

The following steps must be carried out to simulate a power system using the ObjectStab library and the Dymola simulator:

1. Create a new model window.
2. Draw the power system in the model window as shown in Figure 2 using the graphical model editor. New components are created by dragging them from a library window into the model window. By double-clicking on a component in the model window, a dialog box appears and the parameters can be entered.
3. Compile the model by choosing ’Translate’ in the ’File’ menu of the model editor.
4. Simulate the system by choosing ’Simulate’ from the ’Simulation’ menu of the simulator window. If desired, parameter values can be adjusted using a dialog box prior to starting the simulation.

As an alternative to steps 1-2, the model can be entered in the textual format illustrated in Figure 3. If the system is built using the graphical editor, parameter values are entered using dialog boxes and the corresponding textual representation is automatically generated. Figure 2 shows the graphical display of the four generator test system [13] used for validation of the library. Figure 4 shows simulation results obtained using the ObjectStab library and results from simulation results obtained using EuroStag [8]. At simulation time 2 s, a ground fault occurs close to bus 7. The faulted line is disconnected at both ends at 2.07 s, and later at 3.5 s the fault has cleared and the line is reconnected. From the figure we can se that there is a near-perfect agreement between the results of the two programs.

5 Further Work

The current version (4.1) of the Dymola simulator does not exploit the sparse structure of the differential-algebraic equation system resulting from a large power system model. The simulation of large systems is therefore very slow with Dymola compared to domain-specific

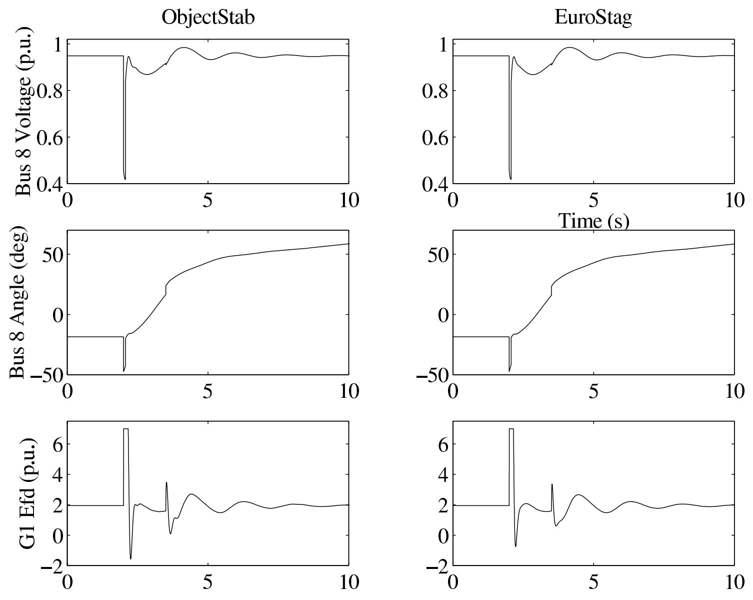


Figure 4: Simulation results obtained using the ObjectStab library and EuroStag.

tools such as PSS/E or EuroStag. As an example, the a simulation of a ground fault with the four node system take about 10 s with EuroStag and 15 s with Dymola on a 150 MHz SUN UltraSparc which is comparable. The corresponding times for the same simulation with the Nordic 32 test system [14] which contains 41 buses and 23 generators, are 20 s with EuroStag and 1330 s with Dymola - a factor 66 slower. Integration algorithms using sparse matrix techniques may be included in a later version of Dymola and will hopefully make the simulation times comparable with industrial-grade tools even for large systems. Currently, the generator and transformer models do not contain saturation modelling. This may be included in a later version of the library.

6 How To Get ObjectStab Running

The ObjectStab library was constructed for use with the simulation tool Dymola, which runs on Microsoft Windows as well as SUN Solaris and Linux platforms. In addition to the components models, the library includes models of the four generator test system used in this paper and the Nordic 32 test system from [14]. A demo version of Dymola for Windows can be downloaded from the Dynasim website[1]. The library itself can be downloaded from the ObjectStab website[2], and is free for educational use.

7 Conclusions

A component library called ObjectStab for power system transient and voltage stability simulations has been presented. Using the library, students can quickly enter their power system in one-line diagram form using a graphical editor, without having to type cryptic data files in text format or entering parameter data in countless dialog boxes. The library has an open structure and all models can be modified or extended using various Modelica constructs, such as state machines, block diagrams or plain algebraic or differential equations. The models also have reasonable default parameter values defined such that students can play around with the models even before they have full understanding of the meaning of all system parameters. It is primarily intended as an educational tool, which can be used to experiment with and observe the effect of the different levels of modelling or trying out new types of controllers.

[1] http://www.dynasim.se/

[2] http://www.iea.lth.se/~ielmatsl/ObjectStab/

Because of some drawbacks described in Sec. 5 it can not yet compete with special-purpose tools such as EuroStag in terms of computational efficiency, and is therefore most suitable for analysis of small systems.

8 Acknowledgement

Parts of the library are based on models developed in a master's thesis [15]. A special thank goes to Inés Romero Navarro for her dedicated work in the masters project. Also the contributions from Professor Gustaf Olsson and the financial support from the Sydkraft Research Foundation is gratefully acknowledged. Thanks also to the development crew at Dynasim AB, especially Hans Olsson and Sven-Erik Mattsson for modelling help and quick bug fixes.

References

[1] J H Chow and K W Cheung, "A toolbox for power system dynamics and control engineering education and research", *IEEE Transactions on Power Systems*, vol. 7, no. 4, pp. 1559–64, November 1992. 1

[2] Modelica Design Group, *Modelica - A Unified Object-Oriented Language for Physical Systems Modeling, Language Specification*, 1999, http://www.modelica.org/documents.shtml. 1

[3] Modelica Design Group, *Modelica - A Unified Object-Oriented Language for Physical Systems Modeling, Tutorial and Rationale*, 1999, http://www.modelica.org/current/modelicarational12rev.pdf. 1

[4] K J Åström, H Elmqvist, and S E Mattson, "Evolution of continuous-time modeling and simulation", in *The 12th European Simulation Multiconference, ESM'98, June 16-19, 1998, Manchester, UK*, 1998. 2

[5] T Hiyama, Y Fujimoto, and J Hayashi, "Matlab/Simulink based transient stability simulation of electric power systems", in *Power Engineering Society 1999 Winter Meeting*. 1999, vol. 1, pp. 249–53, IEEE. 2

[6] S-E Mattsson, "Modelling of power systems in omola for transient stability studies", in *IEEE Symposium on Computer-Aided Control System Design*, 1992. 2

[7] H Elmqvist, D Brück, and M Otter, *Dymola, Users Manual*, Dynasim AB, Sweden, 2000, WWW Document: http://www.dynasim.se. 2

[8] J Deuse and M Stubbe, "Dynamic simulation of voltage collapses", *IEEE Transactions on Power Systems*, vol. 8, no. 3, pp. 894–904, August 1993. 2, 7

[9] T Van Cutsem and C Vournas, *Voltage Stability of Electric Power Systems*, Power Electronics and Power Systems Series. Kluwer Academic Publishers, 1998. 2

[10] J Machowski, J W Bialek, and J R Bumby, *Power System Dynamics and Stability*, John Wiley, New York, 1997. 2, 3, 4

[11] P W Sauer and M A Pai, "A comparison of discrete vs. continuous dynamic models of tap-changing-under-load transformers", in *Proceedings of NSF/ECC Workshop on Bulk power System Voltage Phenomena - III : Voltage Stability, Security and Control*, 1994. 2

[12] D Karlsson and D J Hill, "Modelling and identification of nonlinear dynamic loads in power systems", *IEEE Transactions on Power Systems*, vol. 9, no. 1, pp. 157–163, February 1994. 2

[13] P Kundur, *Power System Stability and Control*, Power System Engineering Series. McGraw-Hill, New York, 1994. 5, 6, 7

[14] CIGRE Task Force 38.02.08, "Long term dynamics phase II", Tech. Rep., CIGRE, 1995. 8

[15] I Romero Navarro, "Object-oriented modelling and simulating of power systems using dymola", Master's thesis, Department of Industrial Electrical Engineering and Automation, Lund University, June 1999. 9