Object-Oriented Modeling and Simulation of Power Systems using Modelica

Inés Romero Navarro
E.T.S.I.I Cartagena
Murcia University
Spain

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

Gustaf Olsson, Member
Dept. of Industrial Automation
Lund University
Sweden

Abstract: Traditionally the simulation of transient and voltage stability in power systems has been constrained to tools developed specifically for this purpose, e.g. 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 modeling is transparent and flexible, and that the 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 modeling 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.

Keywords: Simulation, Modeling, Object oriented methods, Voltage stability, Transient stability, Multi-machine power system, Objectstab, Modelica, Dymola.

I. 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 these 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, provides the possibility of modeling 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 linearized representation of the system for further analysis but the full nonlinear representation remains hidden to the user.

A. Dymola

This paper describes a power system component library intended for use with the general-purpose tool Dymola from Dynasim AB[1], Sweden. Dymola is a general-purpose simulation tool based on the Modelica language [2]. There is already a number of Modelica libraries intended for use with Dymola for various applications domains, such as multibody systems, hydraulics, thermodynamical systems and chemical processes. There is also a library for power systems electromagnetic transients simulation under development [3].

B. Modelica

Modelica is an object-oriented general-purpose modeling language that is under development in an international effort to define a unified language for modeling of physical systems. Modelica supports object-oriented modeling using inheritance concepts taken from computer languages such as Simula and C++. It also supports non-causal modeling, meaning that a model's terminals do not

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

necessarily have to be assigned an input or output role. While object-oriented concepts enable proper structuring of models, the capability of non-causal modeling makes it easy to model for example power lines which are very cumbersome to model using block-oriented languages such as Simulink. Unfortunately, causality is generally not defined in power networks. 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 modeling tool should relax the causality constraint that has been imposed on the modeling equations in the past. Using causal modeling as in Simulink is illustrated in [4] where a model for transient stability simulation is described. With Simulink, the network has to be modeled 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 differential equations (ODE), differential-algebraic equations, state-machines and block diagrams etc. Modelica also supports discrete-event constructs for hybrid modeling. The Modelica language definition [2] and a tutorial [5] on Modelica can be found at the Modelica website[2]. In [6] a power system block library was developed for using the modelling language Omola, which similarly to Modelica enabled structuring of models using object-oriented concepts and models. However, the Omola project is now discontinued and the experiences from this have been brought into the Modelica project.

C. Contributions of the Paper

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. Power system topology and parameter data are entered in one-line diagram form using a graphical editor. The full nonlinear or linearized equation system can be exported for use as a block in Simulink simulations. The component library has been validated using comparative simulations with Eurostag [7].

[2] `http://www.modelica.org`

II. 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 excitation and prime mover control systems.
- Transmission lines in pi-link or series impedance representation.
- Reactive power compensation devices; shunt reactors, shunt capacitances and series capacitances according to [8].
- Fixed ratio transformers.
- On-load tap changing transformers (OLTC) modeled as detailed discrete models or using their corresponding continuous approximations according to [9].
- Static and dynamic loads, including induction motor and generic exponential recovery loads [10].
- Buses.
- Faulted lines and buses with fault impedance.

Standard assumptions for multi-machine transient 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 modeled according to the guidelines given in [8]. 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 using the so-called Kron's transformation [8].

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. In fact the generator with governor and exciter is the basic generator model extended by the block diagram shown in Fig. 1 and the on-load tap changing transformer is the basic transformer extended by the state-graph shown in Fig. 2. More complex exciter or turbine models can easily be entered by drawing their block diagram representations in the graphical editor.

Voltages and currents are described by their phasor representation:

$$\mathbf{I} = i_a + j i_b \tag{3}$$

$$\mathbf{V} = 1 + v_a + j v_b \tag{4}$$

Using this representation a connection point can be defined by the following Modelica connector definition

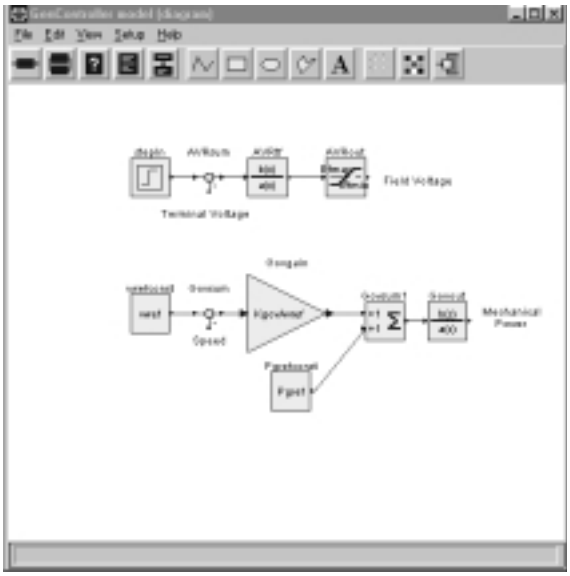

Fig. 1. Block diagram modeling of a governor and an exciter.

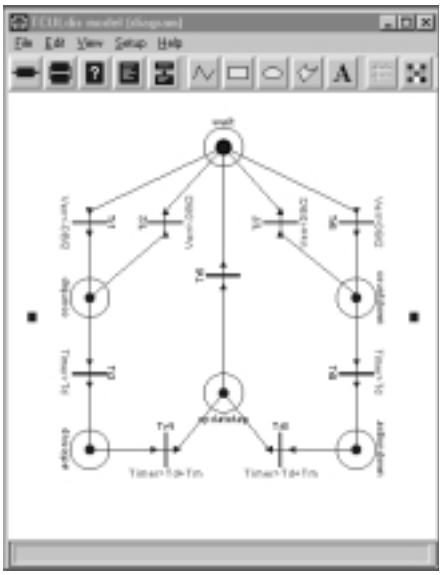

Fig. 2. State machine modeling of OLTC controllers.

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

The equations generated by the connection of two pins called `Pin1` and `Pin2` will be:

```
Pin1.va = Pin2.va
Pin1.vb = Pin2.vb

Pin1.ia + Pin2.ia = 0
Pin1.ib + Pin2.ib = 0
```

The first pair of equations implies that the voltages of two connected Pins will be equal and the second pair that the sum of all currents flowing into a point will be zero.

A. Modeling example: Pi-link line model

The inheritance hierarchy for the Pi-link transmission line model is given below:

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

The definition implies that a `TwoPin` is a model which has two pins named `T1` and `T2` that will be used as external connectors for the Pi-link. The Pi-link model is defined using the `TwoPin` as a base class and thereby inherit its attributes:

```
model Pilink
  extends TwoPin;
  parameter Real R=0.0, X=0.1;
  parameter Real G=0.0, B=0.1;
  Impedance Imp (R=R, X=X);
  Admittance T1Adm (G=G/2, B=B/2);
  Admittance T2Adm (G=G/2, B=B/2);
  Ground GT1, Ground GT2;
  equation connect(Imp.T1, T1);
  equation connect(Imp.T2, T2);
  equation connect(T1Adm.T2, T2);
  equation connect(T1Adm.T1, GT2.T);
  equation connect(GT1.T, T2Adm.T1);
  equation connect(T2Adm.T2, Imp.T1);
end Pilink;
```

The admittance model is defined as follows:

```
model Admittance
  extends TwoPin;
  parameter Real G=0.0;
  parameter Real B=0.1;
equation
  T1.ia = (T1.va - T2.va)*G - (T1.vb - T2.vb)*B;
  T1.ib = (T1.va - T2.va)*B + (T1.vb - T2.vb)*G;
  T1.ia + T2.ia = 0;
  T1.ib + T2.ib = 0;
end Admittance;
```

The impedance model is analogous to the admittance model. It would have been equivalent to define the pi-link model by drawing the system shown Fig. 3 using the graphical editor and the blocks from the ObjectStab library. The textual representation of the model would then be automatically generated. Similarly to the `TwoPin` definition there is a `OnePin` for the various components with just one connector, such as generators and loads.

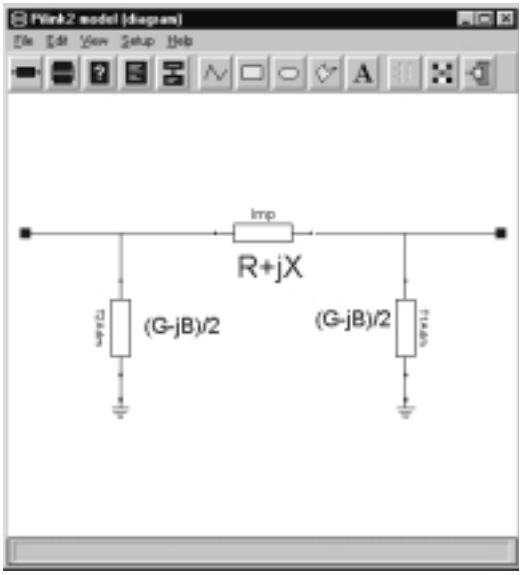

Fig. 3. Graphical representation of the pi-link line model.

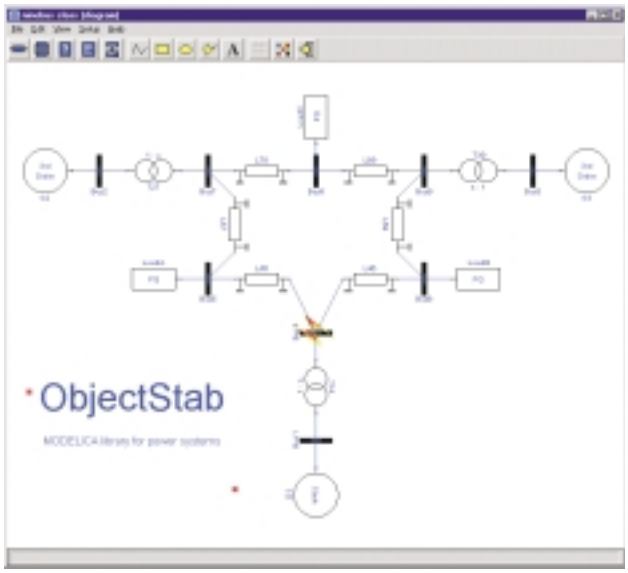

Fig. 4. The nine-bus test system from [11].

This example illustrates the principle of inheritance where new classes inherit the attributes of their parent classes, and how different components can be connected using electrical connectors called Pins. Class definitions and documentation for the other models can be found on the ObjectStab website[3].

III. Case Study: Nine bus system

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

1. Create a new model window.
2. Draw the desired power system in the model window using the graphical model editor. New components can be 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. Solve for the initial state of the system using the interactive initial value solver. By default, this tool solves for the stationary point on the basis of the various reference values and other parameters given. Parameter values and the variables to solve for can be changed interactively. For example, the tool can be used to solve for given load voltages instead of generator voltages which is the default.
5. Simulate the system.

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

As an alternative to steps 1-2, the model can be entered in textual format. If the system is built using the graphical editor, parameter values are entered using dialog boxes and the corresponding textual representation is automatically generated. Fig. 4 shows the graphical display of the nine bus system [11] used for validation of the library. Fig. 5 shows simulation results obtained using the ObjectStab library and results from simulation results obtained using Eurostag [7]. At simulation time 1 s, a change of the setpoint voltages of generator G2 and G3 to 1.15 p.u. occurs and, at simulation time 10 s, a bus-ground fault with fault resistance 0.1 p.u. occurs. The fault clears at simulation time 11 s. From the figure we can se that there is a perfect agreement between the results of the two programs.

IV. Further Work

The Dymola tool translates the object-description into a set of differential-algebraic (DA) equations on the form

$$F(t, \dot{x}, x, w) = 0 \tag{5}$$

where x, and w are the dynamic and algebraic state vectors respectively, and t the current time. In solving this system the Dymola tool translates the DA form to the ordinary differential equation form

$$\dot{x} \quad = \quad f(t, x) \tag{6}$$

This reduction eliminates the algebraic states. Some of the algebraic states can be eliminated symbolically and the remainder must be solved for by an iterative solver in each evaluation of f. Note that network equations are linear when formulated in terms of voltages and currents

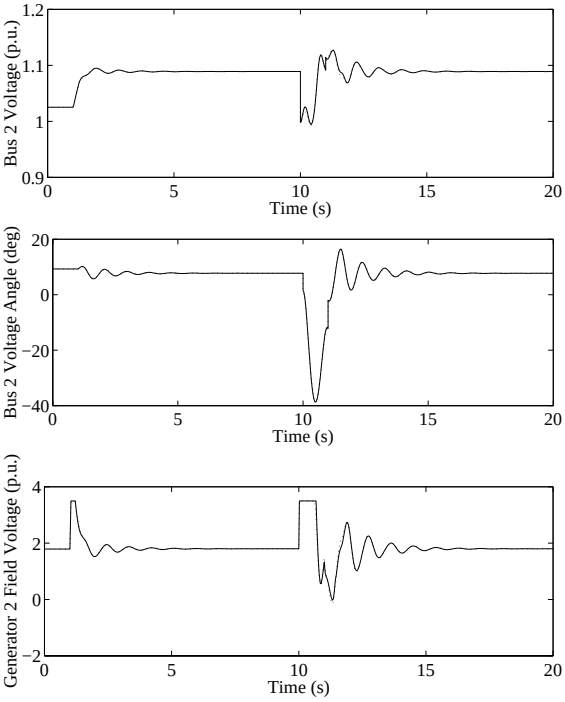

Fig. 5. Simulation results obtained using the ObjectStab (solid lines) library and Eurostag (dotted lines).

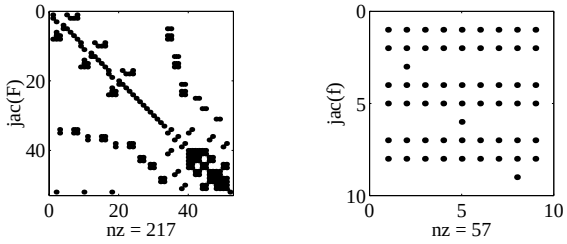

Fig. 6. Sparsity plots of the Jacobian of Eqs. (5) and (6) for the nine bus test system. The number 'nz' corresponds to the number of non-zero entries in the Jacobian matrix.

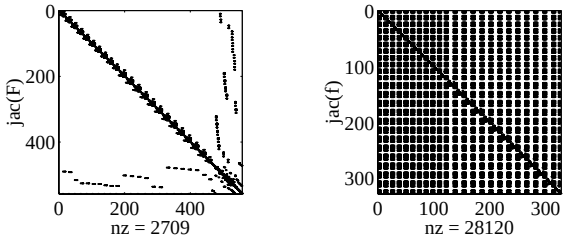

Fig. 7. Sparsity plots of the Jacobian of Eqs. (5) and (6) for the Nordic32 twenty-three machine, thirty-two bus system. The number 'nz' corresponds to the number of non-zero entries in the Jacobian matrix.

and the algebraic states corresponding to voltages can be eliminated symbolically without iterative solving. For small systems this reduces the computational complexity and enables the use of simple integration algorithms. However, an important aspect is that, for a power system model, the Jacobian of (5) is sparse whereas the Jacobian of (6) is not as shown in Figs. 6-7. The number of non-zero elements in the Jacobian can be used as a rough measure of the computational effort necessary to solve the equation system. For the small system the manipulation reduces the number of non-zero elements, whereas for the large system transformation is counter-productive and increases the number. Integration algorithms working directly on the DAE form using sparse-matrix manipulation methods may be included in later versions of Dymola, and will then vastly reduce simulation times for large systems.

With large systems using the dynamic 3rd or 6th order generator models there are some problems of convergence of the initial value solver, which cannot easily find the initial states for the machine speeds and angles. Normally, e.g. in the commercial simulators Eurostag and PSS/E, the dynamic states are initialized from a solved load-flow case. Since the dynamics of the systems are not considered in the load-flow solution, the calculated initial state vector may be an infeasible starting point e.g. due to load dynamics or generator current limitations that were not properly modeled in the load-flow. The proper way of calculating the initial state is to use the static equivalents (with the derivative terms set to zero) of the differential equations, while all limits are still enforced. Presently, there is no construct in Modelica that enables the use of alternative models by the initial value solver, but this may be included in a later version of the Modelica definition [2].

Presently, it is required to have at least one slack node which acts as a angle reference for all the generators. Since the slack node acts as an infinite bus that keeps the reference system frequency constant, the library can not be used to study frequency stability. Extended models for study of frequency stability will be included in a later version of the ObjectStab library.

V. 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 UNIX platforms. In addition to the components models, the library includes models of the nine bus test system used in this paper and the IEEE 30 bus test system. A demo version of Dymola can be downloaded from the Dynasim website[4]. The library itself can be downloaded from the ObjectStab website[5], and is free for educational use.

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

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

VI. 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. Because of some drawbacks described in Sec. IV 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.

VII. Acknowledgement

The financial support from the Sydkraft Research Foundation is gratefully acknowledged. The authors would also like to thank the development crew at Dynasim AB, especially Hans Olsson and Sven-Erik Mattsson for modelling help and quick bug fixes.

VIII. 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.

[2] Modelica Design Group, *Modelica - A Unified Object-Oriented Language for Physical Systems Modeling, Language Specification*, 1999, `http://www.modelica.org/current/modelicaspec12norev.pdf`.

[3] Modelica Design Group, *Minutes from 11th design meeting, Helsinki, April 15-17*, 1998, `http://www.modelica.org/History/Minutes/min11.html`.

[4] T. Hiyama, Y. Fujimoto, and J. Hayashi, "Matlab/Simulink based transient stability simulation of electric power systems", in *IEEE Power Engineering Society. 1999 Winter Meeting (Cat. No.99CH36233). IEEE, Piscataway, NJ, USA; 1999; 2 vol. xxiii+1340 pp. p.249-53 vol.1.*

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

[6] S-E. Mattsson, "Modelling of power systems in Omola for transient stability studies", in *IEEE Symposium on Computer-Aided Control System Design, CACSD '92, March 17-19, Napa, California, USA.*

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

[8] J. Machowski, J.W. Bialek, and J.R. Bumby, *Power System Dynamics and Stability*, Number ISBN 0-471-97174. Wiley, 1993.

[9] 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*. Davos, Switzerland, 1994.

[10] 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.

[11] P.M. Anderson and A.A. Fouad, *Power System Control and Stability*, Number ISBN 0-7803-1029-2. IEEE Press, second edition, 1994.

IX. Biographies

Inés Romero Navarro was born in Murcia, Spain on December 19, 1974. She got the degree of Industrial Technical Engineer in Electrical and Electronic systems in 1996 in The ETSITI and the degree of High Industrial Engineer in 1999 in the ETSII, Cartagena, with a M.Sc. in Power Systems in Lund University with an exchange program (1998/1999). Her work was based on Object-Oriented Modelling and Simulation of Power Systems using Dymola.

Mats Larsson was born in Halmstad, Sweden on June 6, 1969. He graduated with a M.Sc. in Computer Science in 1993 and a technical licentiate degree in Industrial Automation in 1997 from Lund University. He is presently a postgraduate student with Lund University. His main research interests are: Modeling and Simulation, Power Systems Operation and Control, with a special emphasis on reactive power management and voltage collapse. He was previously employed at WM Data Ellips AB as a R&D engineer in the Power System Control field.

Gustaf Olsson got his licentiat in engineering physics from the Royal Institute of Technology in Stockholm, Sweden. He was appointed assistant professor in 1967 and in 1970 associate professor in automatic control at the Lund Institute of Technology. In 1978 he was awarded the docent degree. Since 1987 he is professor and head of the Industrial Automation Department at the Lund University. He has held visiting positions at several universities and industries, including Univ. of California, Los Angeles, California, Univ. of Houston, Texas, Kyoto University, Japan, University of Wisconsin, Madison, University of Queensland, Brisbane Australia, and various companies in the USA. He has worked with automation problems related to nuclear reactor operation, power transmission as well as process and manufacturing control. He has published three books, contributed to another six books and published some 90 papers.