Modeling Specialized Electric Power Generators, Excitation Systems and Prime Movers used by North American Utilities

Md Shamimul Islam[1] Giuseppe Laera[1] Marcelo de Castro Fernandes[1] Luigi Vanfretti[1]
Chetan Mishra[2] Kevin D. Jones[2]

[1]Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, United States,
{islamm7,vanfrl,laerag,decasm3}@rpi.edu
[2]Dominion Energy, United States, {chetan.mishra,kevin.d.jones}@dominionenergy.com

Abstract

The North American Electric Reliability Corporation (NERC) is expected to mandate model validation of power plant equipment in the near future. This will create a need to validate models for a large number of existing and future power plants. Historically, model validation of synchronous generators, excitation system, turbine governor, and other power system equipment has been conducted using diverse software platforms. As a contribution to the power system model implementation using Modelica language and validation against commercial tools, this work continues to develop power system component models and enriching the Open-Instance Power System Library (OpenIPSL). As a part of the development of OpenIPSL this paper describes the development of models used by North American utilities that follow NERC modeling requirements, including models of a synchronous generator, an excitation system, a turbine and governor using Modelica language in Dymola. The component implementation process is described and the validation of the models implemented in Modelica against PSS®E using both a single machine infinite bus (SMIB) and multi-machine system models is illustrated.

Keywords: Modelica, OpenIPSL, model validation, power systems, power grid.

1 Introduction

Precise mathematical modeling and simulation has become an integral part of different engineering tasks carried out by utilities and grid operators, as it aids in optimization of operations, planning of future expansions, and ensuring regulatory compliance. The ongoing efforts in de-carbonizaiton, rising energy costs, extreme climate events (Franke and Wiesmann 2014), etc., are challenging the resilience of power grids and may result in increased costs and may pose potential risks of disruption if not addressed timely. To address these challenges, engineers rely upon extensive computer simulations to understand, design and analyze the performance of power grids under diverse operating scenarios. In turn, these simulations enable stakeholders to identify vulnerabilities, implement improvements, and minimize risks associated with power delivery. Reliable and accessible modeling solutions of power system components and networks are essential for comprehensive analysis and decision making.

There are many different software tools available for power system modeling and simulation. The *de facto* standard tools used in the industry are domain-specific software, including PSS®E and PowerFactory, to name a few (Laera et al. 2022) concerned with power system dynamics and stability, with time-scales of tens of ms to tens of seconds. There is also software with a specific focus on ultra-fast time-scales, such as EMTP and PSCAD for analysis of electromagnetic transients and power electronic-based components. Despite meeting industry requirements and providing a vast library, there are limitations for closed-source software such as PSS®E. These include a lack of transparency regarding underlying algorithms and models, limited and less flexible modeling capabilities, limited simulation solver options, the requirement of a specific skill set for its use, rigid data format and high software cost, as well as limited community support.

To address these issues, there have been open-source initiatives to democratize research, development and overall access to alternative modeling tools in this field. Vanfretti et al. proposed the use of object-oriented equation-based modeling language Modelica to model power systems (Vanfretti et al. 2013). This started the efforts of the development of an open source power system library for power grid modeling and simulation consistent with the power industry practices and requirements, now called Open-Instance Power System Library (OpenIPSL) (Baudette et al. 2018). It is worth to note that other past efforts in power grid modeling with Modelica have been summarized in (Winkler 2017), with more recent efforts reported in (Adrien Guironnet et al. 2018) and (Bartolini, Casella, and A. Guironnet 2019). Meanwhile, outside the scope of power system dynamic modeling, other open-source initiatives that offer specific solutions to other analysis needs have emerged. These include OpenDSS(Montenegro, Dugan, and Reno 2017) for power distribution analysis, GridCal[1] and PyPSA (Brown, Hörsch, and Schlachtberger 2017; Parzen et al. 2023) etc., that allow

[1]See: https://gridcal.readthedocs.io/en/latest/

to perform studies focused on the solution of steady-state problems, while also establishing a community to support such efforts.

Outside the power grid domain, significant research efforts have taken place to model various energy systems with Modelica such as modeling of building energy system modules (Wüllhorst, Maier, et al. 2022), Heat Ventilation and Air Conditioning systems (Wüllhorst, Storek, et al. 2022), and overall modeling of buildings for energy efficiency (Wetter et al. 2014), to name a few. These efforts show that there is tremendous potential in the Modelica "approach" towards modeling of different facets of energy systems. In this context, OpenIPSL aims to contribute to this growing body of Modelica libraries providing an open source and Modelica-based resource to address energy system needs. OpenIPSL offers different power grid component models with specific characteristics, such as a synchronous generator, excitation system, turbine governor/prime mover, power system stabilizer, load models, transformers, transmission line models, control unit, and so on, enabling modeling of power system dynamic modeling for generation, transmission and distribution systems (Castro et al. 2023).

This work aims to expand the existing capabilities of OpenIPSL by introducing the modeling of three new components using Modelica language and OpenIPSL library that are available in PSS®E and are required by the North American Electric Reliability Corporation (NERC) for the modeling of real-world power plants in the United States. Targeting specific power generation facilities in the Eastern Interconnection of the United States, new models of generators (GENTPJ), excitation systems (ESURRY), and prime movers (WPIDHY) that contain unique characteristics in terms of design, response, and functionality, are presented herein.

The specific contributions of this paper are:

- A Modelica language-based implementation of new and representative power generators, excitation systems, and prime mover models. These models will be included in the future release of the open-source OpenIPSL library.

  Generator (GENTPJ): A recent synchronous machine model (Birchfield et al. 2017) allowing the representation of sub-transient saliency and containing a much more complex saturation representation than existing models. Although being used by utilities and being available in the PSS®E software, this newer model is not currently in the IEEE standards. This model addresses some of the limitations of widely used GENROU and GENSAL models.

  Excitation System (ESURRY): This model is specialized for nuclear power plants in North American Utilities.

  Turbine Governor (WPIDHY):. This is a specialized model called Woodward PID Hydro Governor (WPIDHY) for hydraulic turbines and their speed control system used by utilities in the Eastern Interconnection. It includes a hydro turbine, governor and penstock representation for the modeling of hydro power plants using Woodward governor control systems. The model includes a nonlinear gate to power relationship and a linearized turbine and penstock model.
- Benchmarking the developed models by comparing the Modelica implementation results with the industry *de facto* standard propriety software tool PSS®E.
- A discussion of challenges and a framework for future development.

The reminder of this paper is organized as follows. Section 2 describes the Modelica implementation of the new electrical machine, excitation system, and turbine & governor models. Section 3 discusses the validation approach and explains the test procedure to test each of the components in the SMIB and multi-machine system. Section 4 describes the simulation results and discusses the validation results. Finally, Section 5 concludes the paper.

2 Modelica Implementation

In modeling power system components, it is critical to have a comprehensive understanding of the model's specifications and conceptual framework for its derivation. This understanding is largely domain-specific and gives a guide in the identification of relevant equations and/or block diagrams that accurately describe the system's dynamic behavior. Once these equations and/or block diagrams have been identified, they can be used to construct a model in Modelica.

One important stage in the modeling process is to determine how to initialize the model, which involves specifying how the initial values of the model's variables should be determined. After initialization, the model should be subjected to a software-to-software validation test, in which it is compared to a reference result (Otter et al. 2022) to ensure that the model's inputs and outputs are consistent with its specifications. This test is crucial in verifying the model's accuracy and ensuring it behaves as intended.

In summary, modeling a system requires a clear understanding of its specifications and conceptual framework, identification of relevant equations and/or block diagrams, construction of the model in Modelica, initialization, and validation through software-to-software testing.

In this article, the process summarized above is applied to three components used for the modeling of power generation systems in the Eastern Interconnection of the US. Namely, the components are: a generator GENTPJ (as GENTPJU1 in PSS®E), an excitation system ESURRY, and a turbine governor WPIDHY. The components have been modeled using Modelica language and verified against PSS®E. The model is presented as block diagrams in PSS®E's manual, which is used along with other literature to start the implementation in Modelica using developed components available in Modelica Standard Li-

DOI 10.3384/ecp204653

brary and OpenIPSL. These components were already developed and modeled in PSS®E.

2.1 Synchronous Machine (GENTPJ)

The Western Electricity Coordinating Council (WECC) has historically used the GENROU model to simulate round-rotor synchronous generators (used in steam- and gas-turbine power plants) and the GENSAL model to simulate salient-pole synchronous generators (used in hydropower plants). In recent years, WECC has begun to use the GENTPF model for round-rotor generators and a modified version of the GENTPF model called GENTPJ for salient-pole generators (Pourbeik et al. 2016). The GENTPJ model was first developed by (J. Undrill 2012) by introducing a new parameter K_{is}, which is a scalar multiplier of total stator current. Finally, NERC issued a "Modeling Notification(North American Electric Reliability Corporation 2018)," in November 18, 2016, recommending *to use the GENTPJ model for new modeling of salient pole generators and future (re)verification of salient pole generator models*.

Using GENTPJ, each synchronous machine is modeled in its rotor reference frame, i.e., rotating at the speed of the rotor. The electric source is represented by equations of the flux behavior in orthogonal dq-axes. When the system containing the generator model is subjected to a disturbance this results in an imbalance between the power generated and consumed, which in turn results in a speed deviation from its synchronous reference at the machine. Therefore, all of the machine variables are transferred to the synchronous reference frame.

The synchronous machine is implemented using Modelica in Dymola software by using the equations listed in (Pourbeik et al. 2016; Olive 1968; J. M. Undrill 1969). The GENTPJ model is similar to the GENROU, GENSAL, and to GENTPF, except the saturation function uses the K_{is} term. More details regarding the saturation function in the original 2007 and current 2012 specification are discussed in a presentation by BC Hydro (Cui 2022). The reader is refereed to (Zhang et al. 2015) for a discussion on the Modelica implementation of GENROU and GENSAL included in OpenIPSL. The equations that are used to implement the GENTPJ model using Modelica and the OpenIPSL library are listed in the Appendix. The meaning of the symbols are specified to (Schulz 1975; Kundur 1994), and they are discussed in (Pourbeik et al. 2016) and in the Modelica implementation within the annotations and comments in the Appendix.

Finally, it should be noted that when $K_{is} = 0$, GENTPJ can be used to represent the WECC Type F generator model, GENTPF.

2.2 Excitation System (ESURRY)

The excitation system is an essential component of a power generator, providing the necessary voltage/current to excite the generator's field winding. The ESURRY model can be seen as a modified version of the IEEE Type

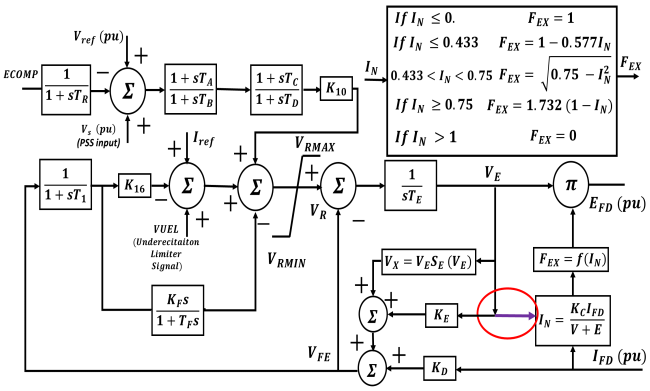


Figure 1. Block diagram of ESURRY from PSS®E (PTI 2017), corrected by adding the purple arrow circled in red.

AC1A ("IEEE Recommended Practice for Excitation System Models for Power System Stability Studies" 2016) excitation system that was developed in PSS®E, and it is used to model the specialized excitation system in synchronous machines at nuclear power plants.

The ESURRY model, as shown in Figure 1, consists of a non-controlled rectifier and an alternator. The model has three inputs: the generator terminal voltage, *ECOMP*, generator field current, I_{FD}, and an input for power system stabilizer (PSS) signal, V_S. The model's output is the exciter field voltage, E_{FD} , which is connected to the synchronous machine.

The non-controlled rectifier in the ESURRY model is responsible for rectifying the generator's output voltage and producing a DC voltage that is applied to the exciter's field winding. The alternator generates a small AC voltage that controls the rectifier's firing angle.

The ESURRY model also includes an input signal to couple a power system stabilizer (PSS), which dampens the generator's response to disturbances. The input signal provided by the PSS modulates the exciter's output voltage in response to system frequency or electrical load changes. This exciter model is implemented in Modelica using a block diagram, however, the implementation was challenging as detailed information or documentation to the different functions within the blocks were not available. During this study, a connection error was identified in the PSS®E reference manual that is corrected in this article with its location marked in Figure 1 in red[2], and the corrected diagram is shown in Figure 2. Note that the initialization equations are implemented in the text layer within the `initial equation` section of the model and therefore are omitted in the Figure.

2.3 Turbine Governor (WPIDHY)

The primary frequency control of synchronous machines is a critical function of a power plant. The governor achieves this by adjusting the mechanical power output

[2]The error is a missing input signal V_E used in the computation of I_N in the lower right corner of the diagram. The correction is shown as a purple arrow, added to the diagram taken from the manual.

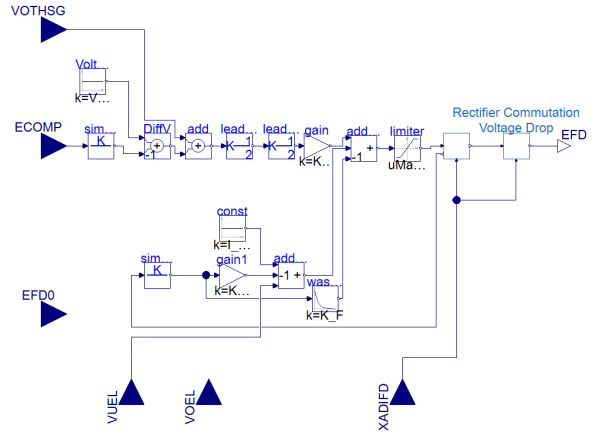


Figure 2. Implementation of ESURRY in Modelica using the OpenIPSL.

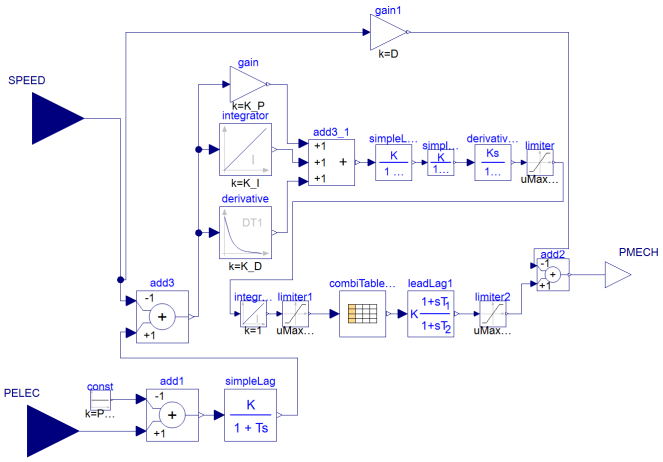


Figure 4. Implementation of WPIDHY in Modelica using the OpenIPSL

of the turbine in response to changes in electrical load or disturbances on the system.

The WPIDHY model, as shown in Figure 3, considers the rotor speed deviation $\Delta\omega$, electric power P_{ELEC}, and reference power P_{REF} as input signals. These signals are used to determine the mechanical power $PMECH$ to be applied to the generator. The model uses a proportional-integral-derivative (PID) control scheme. The PID controller compares the generator's speed with a predefined reference and adjusts the mechanical power output to maintain the desired frequency. Similar to ESURRY, the Modelica implementation was carried out using a block diagram, as shown in Figure 4. Note that the initialization equations were implemented in the text layer within the `initial equation` section of the model and therefore are omitted in Figure 4.

In Figure 3 after the integrator, a graph shows the relationship between gate position $(X-axis)$ and the turbine governor's corresponding output per unit $(Y-axis)$. Each data point signifies a unique mapping between the gate's position and power output. In the graph, the turbine generates no power starting at the zero gate position $(MBASE = 0)$. As the gate opens towards its maximum extent (one), the turbine reaches its peak capacity and produces the maximum possible per unit output $(MBASE = P_3)$. Between these two points lies a range of gate positions and their accompanying power outputs.

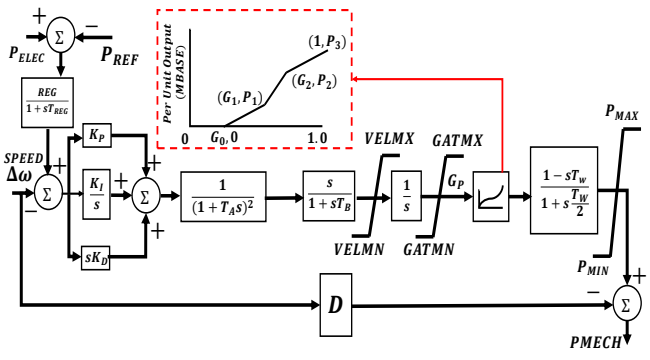


Figure 3. Block diagram of WPIDHY from PSS®E (PTI 2017).

The curve shows an exponential rise in power output between the zero gate position and G_1 gate position. At Gate Position G_1, the power output rises sharply before leveling off around Gate Position G_2. Beyond Gate Position G_2, minor variations in gate position yield only minimal gains in power output. Finally, the curve approaches the maximum power output $(MBASE = P_3)$ as the gate approaches its whole opening (one).

This graph serves multiple analysis purposes:

- It enables analysis of the minimum gate position needed to generate electricity G_1.
- It identifies the optimal gate position for maximum power output G_2.
- It suggests appropriate gate settings based on desired power output levels between G_1 and G_2.

Overall, understanding this graph is necessary for the effective modeling of hydro turbines with non-linear gate-to-power relationships, as in the case of WPIDHY.

3 Model Validation

3.1 Model Validation Procedure

In this paper, we followed the model validation approach according to (Laera et al. 2022), which is summarized as follows:

- Obtain the steady state computation results of a power flow solution in PSS®E.
- Export the results and provide them as initial guess values to solve the initialization problem of the corresponding SMIB in the Modelica-compliant software tool.
- Define the scenario for the dynamic simulation in both tools and run a dynamic simulation of the SMIB or other test system in both software.

DOI 10.3384/ecp204653

- Choose the quantities to compare and export them in the appropriate format to be used in another tool, for example, CSV Compare (`https://github.com/modelica-tools/csv-compare`) or funnel (`https://github.com/lbl-srg/funnel`).
- Use a tool (e.g. CSV Compare or funnel) to quantify the discrepancies between the simulation software tools after defining an acceptable tolerance level.
- The validation is complete if the errors between the quantities to compare are within the tolerance band.
- If the errors are more significant than the defined tolerance, then more model debugging is required.
- Compare the implemented model's sub-component inputs, outputs, and states with the analogous signals from the SMIB in PSS®E.
- Continue the iterative process until the signal difference is lower than the tolerance.

In the sequel, the model validation procedure described above is applied to perform a software-to-software validation of the models implemented in Modelica against PSS®E using two types of test systems.

3.2 Test in SMIB Models

The Single-Machine Infinite-Bus (SMIB) model is a simplified representation of a power network typically used to analyze the interconnection of generation facilities to the rest of the grid. The SMIB model includes one generator, one infinite bus, several transmission lines and/or transformers. Figures 5, 6 and 7 show the implementation of the test system, one for each of the components under test, all of which are located on the left side of the Figures, close to bus GEN1.

In Figures 5, 6 and 7, the infinite bus is modeled with a GENCLS machine with constant voltage and high inertia. It is connected to bus GEN2 to represent the interconnection to a stiff network. In contrast, the generator connected at bus GEN1 is composed of synchronous machine: GENTPJ in Figure 5, GENROU in Figure and GENSAL in Figure 7. When validating ESURRY, in Figure 6, this fast static excitation system model is added to the GENROU model. Meanwhile, to validate WPIDHY, this turbine and governor model is added to a GENSAL model.

To verify the simulation, the power network is perturbed during the simulation. To this end a three-phase-to-ground fault is applied on bus FAULT to the models in Figures 5, 6 and 7.

3.3 Test using a Real Power Plant Model

To further validate the developed models (GENTPJ, ESURRY, and WPIDHY), we perform the validation test of a power plant composed by two generators, connected to a similar network as used before, as shown in Figure 8. This simple model is representative of a specific generation station in the Eastern Interconnection of the US. According to Figure, this test system has two generation units, each

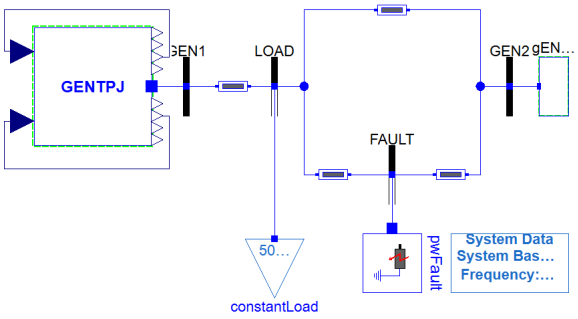


Figure 5. Implementation of the SMIB system to test GENTPJ in Modelica using the OpenIPSL.

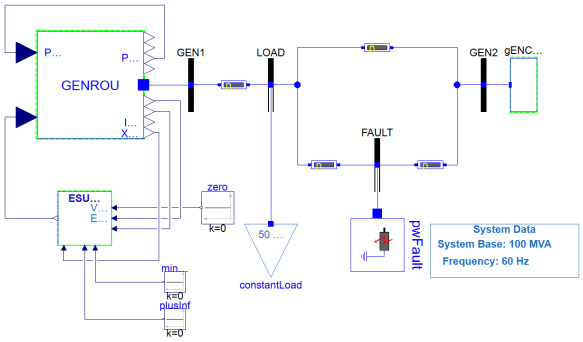


Figure 6. Implementation of the SMIB system to test ESURRY in Modelica using the OpenIPSL.

consisting of a machine, an excitation system, a turbine governor, and a PSS. In Figure 8, the generators are connected to two buses, and then through a transformer, they are connected to the SMIB network we discussed earlier.

4 Results

To perform software-to-software validation of the newly implemented components in Modelica, we performed simulations in a simple SMIB network and multi-machine test system in both Dymola and PSS®E. In all simulation scenarios, a three-phase bus fault is applied to the FAULT bus at $t = 2\,s$ and cleared at $t = 2.15\,s$.

In the case of the unit test models in Figures 5, 6 and 7, simulation results are shown in Figures 9, 10, and 11, respectively. These results show the successful validation of individual components that are implemented for this work, i.e., GENTPJ, ESURRY and WPIDHY. These figures show that the Modelica implementation can produce the same results as PSS®E. Here we can see the generator terminal voltage, exciter field voltage, and mechanical power of the turbine governor for the models GENTPJ, ESURRY, and WPIDHY perfectly match the PSS®E for both steady state and dynamic response.

In the case of the real power plant model, Figures 12, 13, and 14 depict the Dymola vs. PSS®E validation results through the steady state and dynamic response of generator terminal voltage, active, reactive power, speed deviation, and exciter field voltage for the system in Figure. 8. Similar to the SMIB test system, the response

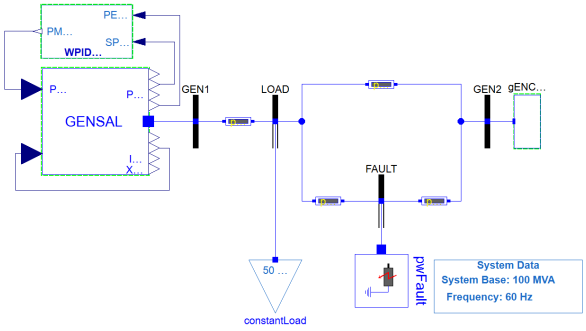


Figure 7. Implementation of the SMIB system to test WPIDHY in Modelica using the OpenIPSL.

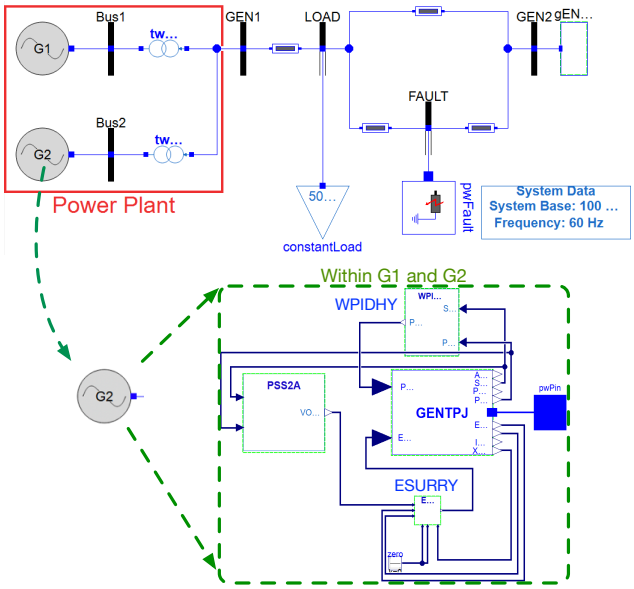


Figure 8. Implementation of the real world power plant model in Modelica using the OpenIPSL.

match that of PSS®E response; however, a small mismatch starts during the fault period. From these figures, we can see the mismatch starts disappearing at $t = 4.5s$, and after that, the dynamic response is the same in both Dymola and PSS®E. This difference is attributed to the differences on how the PSS®E handles the equations during the fault event, which are unknown to the authors.

5 Conclusion and Future Work

The detailed implementation of new power system component models used by North American utilities using the Modelica language, the OpenIPSL, and Dymola has been documented in this article. This paper summarizes the Modelica implementation of three power system components: a round rotor synchronous machine, an excitation system, and a turbine & governor system. Moreover, the implemented components were tested through using both the Dymola software and the PSS®E software for three simple test unit power system models (i.e., the SMIB) and multi-machine test system model representative of a power plant in the Eastern Interconnection of the US. Finally, the simulation results obtained using Dymola were compared against PSS®E. According to the results discussed in this paper, the Modelica implementation of power systems components performs similar to those of PSS®E. These models need to be tested in various test model setups to solve any possible issues that have not yet appeared during this work. Future work includes performing further simulation experiments to detect any remaining issues, performing validation and finally integrating the newly developed models into a future release of OpenIPSL.

Acknowledgements

This material is based upon work supported in part by Dominion Energy, in part by the National Science Foundation Award No. 2231677, and in part by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) under the Advanced Manufacturing Office, Award Number DE-EE0009139.

References

Bartolini, A., F. Casella, and A. Guironnet (2019-09). "Towards Pan-European Power Grid Modelling in Modelica: Design Principles and a Prototype for a Reference Power System Library". In: *Proceedings of the 13th International Modelica Conference*. Linköping Electronic Conference Proceedings 157:64. Regensburg, Germany: Modelica Association and Linköping University Electronic Press, pp. 628–636. ISBN: 978-91-7929-027-6. DOI: 10.3384/ecp19157627.

Baudette, Maxime et al. (2018). "OpenIPSL: Open-instance power system libraryupdate 1.5 to iTesla power systems library (iPSL): A modelica library for phasor time-domain simulations". In: *SoftwareX* 7, pp. 34–36.

Birchfield, Adam B et al. (2017). *Impact of Synchronous Generator Model GENTPJ on System Dynamics*. URL: https://adambirchfield.com/cv/gm2017_paper.pdf.

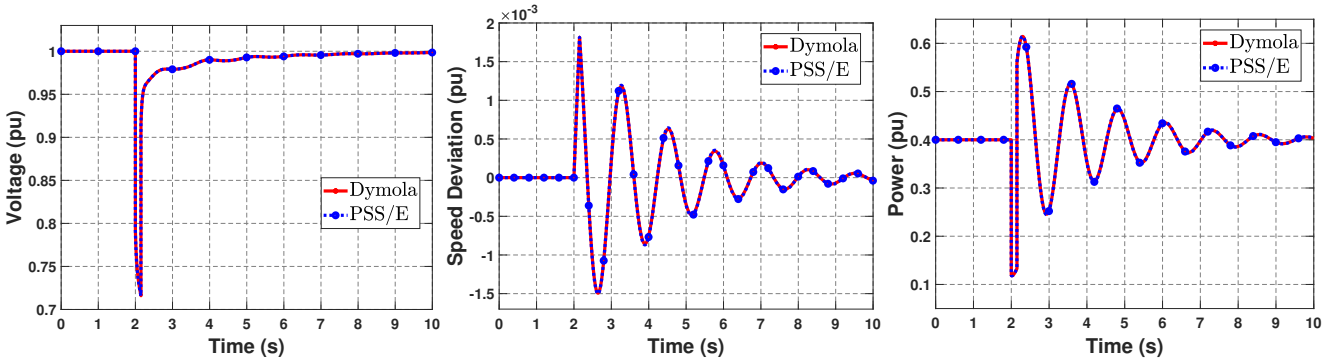
Brown, Tom, Jonas Hörsch, and David Schlachtberger (2017). "PyPSA: Python for power system analysis". In: *arXiv preprint arXiv:1707.09913*.

Castro, Marcelo de et al. (2023). "Version [OpenIPSL 2.0.0] - [iTesla Power Systems Library (iPSL): A Modelica library for phasor time-domain simulations]". In: *SoftwareX* 21. DOI: https://doi.org/10.1016/j.softx.2022.101277. URL: https://www.sciencedirect.com/science/article/pii/S2352711022001959.

Cui, Philip (2022). *GENTPJ Model Saturation Function. WECC MVS Meeting*. URL: https://nerc.com/comm/pc/nercmodelingnotifications/use%20of%20gentpj%20generator%20model.pdf.

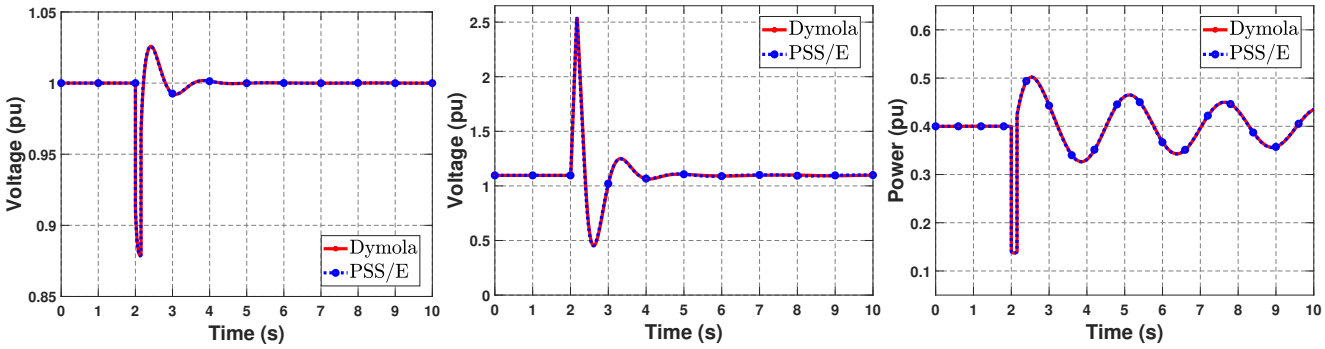
Franke, Rüdiger and Hansjürg Wiesmann (2014). "Flexible modeling of electrical power systems–the Modelica PowerSystems library". In: *Proceedings of the 10th International Modelica Conference*. Linköping Electronic Conference Proceedings 96:54. Lund, Sweden: Linköping University Electronic Press, pp. 515–522. ISBN: 978-91-7519-380-9. DOI: 10.3384/ecp14096515.

Guironnet, Adrien et al. (2018-10). "Towards an Open-Source Solution using Modelica for Time-Domain Simulation of Power Systems". In: *2018 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*. DOI: 10.



(a) Terminal voltage at bus GEN1. **(b)** Speed deviation at bus GEN1. **(c)** Active power at bus GEN1.

Figure 9. Generator terminal voltage, speed deviation, and active power of system in Figure 5.



(a) Terminal voltage at bus GEN1. **(b)** Exciter field voltage at bus GEN1. **(c)** Active power at bus GEN1.

Figure 10. Generator terminal voltage, exciter field voltage, and active power of system in Figure 6.

1109/isgteurope.2018.8571872. URL: https://doi.org/10.1109%2Fisgteurope.2018.8571872.

"IEEE Recommended Practice for Excitation System Models for Power System Stability Studies" (2016). In: *IEEE Std 421.5-2016 (Revision of IEEE Std 421.5-2005)*, pp. 1–207. DOI: 10.1109/IEEESTD.2016.7553421.

Kundur, Prabha (1994). *Power System Stability and Control*. McGraw-Hill.

Laera, Giuseppe et al. (2022). "Guidelines and Use Cases for Power Systems Dynamic Modeling and Model Verification using Modelica and OpenIPSL". In: *Proceedings of the American Modelica Conference 2022*. Dallas, Texas, USA, pp. 146–157. DOI: 10.3384/ECP21186146.

Montenegro, Davis, Roger C Dugan, and Matthew J Reno (2017). "Open source tools for high performance quasi-static-time-series simulation using parallel processing". In: *2017 IEEE 44th Photovoltaic Specialist Conference (PVSC)*. IEEE, pp. 3055–3060.

North American Electric Reliability Corporation (2018). *Modeling Notification. Use of GENTPJ Generator Model*. URL: https://nerc.com/comm/pc/nercmodelingnotifications/use%20of%20gentpj%20generator%20model.pdf.

Olive, David W. (1968). "Digital Simulation of Synchronous Machine Transients". In: *IEEE Transactions on Power Apparatus and Systems* PAS-87.8, pp. 1669–1675. DOI: 10.1109/TPAS.1968.292127.

Otter, Martin et al. (2022). "Towards Modelica Models with Credibility Information". In: *Electronics* 11.17. ISSN: 2079-9292. DOI: 10.3390/electronics11172728. URL: https://www.mdpi.com/2079-9292/11/17/2728.

Parzen, Maximilian et al. (2023). "PyPSA-Earth. A new global open energy system optimization model demonstrated in Africa". In: *Applied Energy* 341, p. 121096.

Pourbeik, Pouyan et al. (2016-10). "Modeling of synchronous generators in power system studies". In: *CIGRE Science & Engineering* 6, pp. 21–32.

PTI, Siemens (2017). *PSS/E 34.2.0 Model Library*. Siemens Power Technologies International. Schenectady, NY, USA.

Schulz, Richard P (1975). "Synchronous machine modeling". In: *IEEE Symposium on Adequacy and Philosophy of Modeling: Dynamic System Performance*, pp. 24–28.

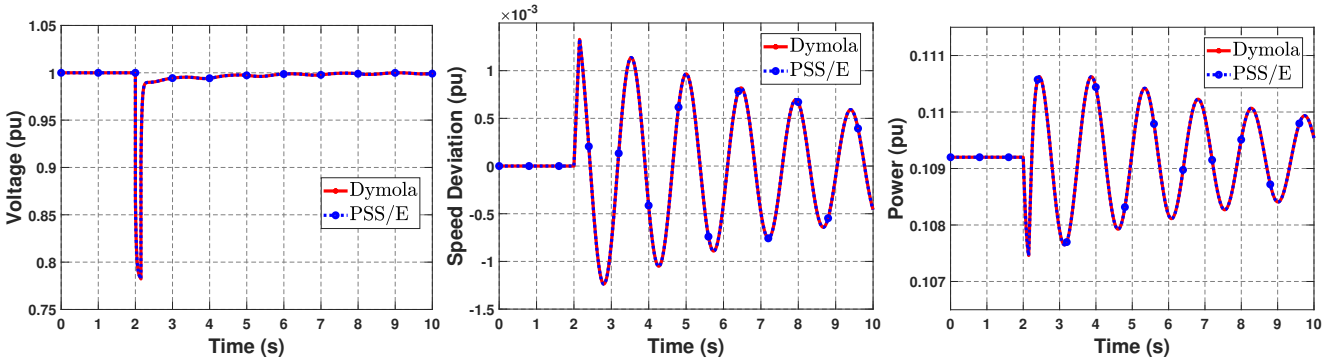
Undrill, John (2012). *The Gentpj Model*. Western Electricity Coordinating Council. URL: https://www.wecc.org/Reliability/gentpj-typej-model-specification.pdf.

Undrill, John M. (1969). "Structure in the Computation of Power-System Nonlinear Dynamical Response". In: *IEEE Transactions on Power Apparatus and Systems* PAS-88.1, pp. 1–6. DOI: 10.1109/TPAS.1969.292330.

Vanfretti, Luigi et al. (2013). "Unambiguous power system dynamic modeling and simulation using Modelica tools". In: *2013 IEEE Power & Energy Society General Meeting*. IEEE, pp. 1–5. DOI: 10.1109/PESMG.2013.6672476.

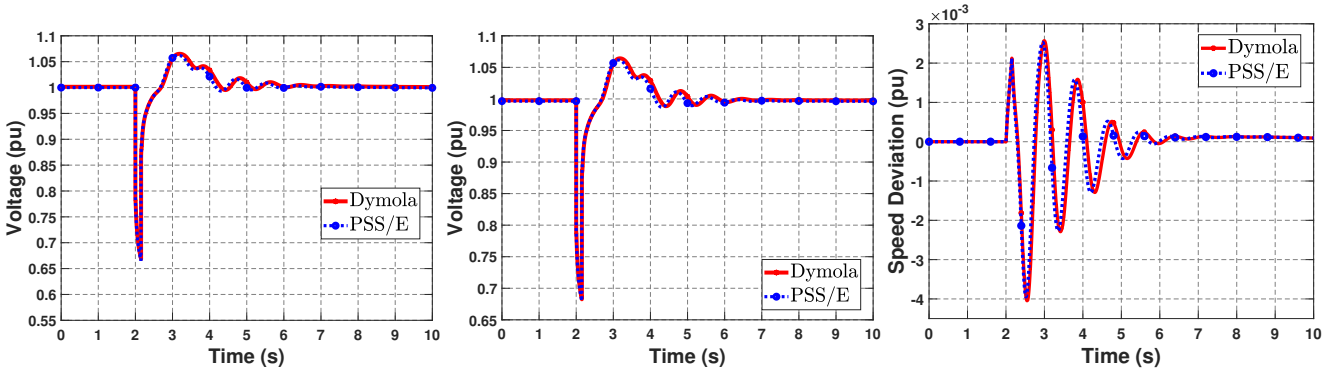
Wetter, Michael et al. (2014). "Modelica Buildings library". In: *Journal of Building Performance Simulation* 7.4, pp. 253–270. DOI: 10.1080/19401493.2013.765506. URL: https://doi.org/10.1080/19401493.2013.765506.

Winkler, Dietmar (2017). "Electrical Power System Modelling in Modelica - Comparing Open-source Library Options". In: *Proceedings of the 58th Conference on Simulation and Modelling (SIMS 58)*, pp. 263–270.



(a) Terminal voltage at bus GEN1. **(b)** Speed deviation at bus GEN1. **(c)** Governor's mech. power at bus GEN1.

Figure 11. Generator terminal voltage, speed deviation, and mechanical power of system in Figure 7.



(a) Terminal voltage at Bus01. **(b)** Terminal voltage at Bus02. **(c)** Speed deviation at Bus01.

Figure 12. Terminal voltage and speed deviation system in Figure 8.

Wüllhorst, Fabian, Laura Maier, et al. (2022). "BESMod-A Modelica Library providing Building Energy System Modules". In: *Modelica Conferences*, pp. 9–18.

Wüllhorst, Fabian, Thomas Storek, et al. (2022). "AixCaliBuHA: Automated calibration of building and HVAC systems". In: *Journal of Open Source Software* 7.72, p. 3861.

Zhang, Mengjia et al. (2015-10). "Modelica Implementation and Software-to-Software Validation of Power System Component Models Commonly used by Nordic TSOs for Dynamic Simulations". In: *Proceedings of the 56th Conference on Simulation and Modelling (SIMS 56)*. Linköping Electronic Conference Proceedings 119 (2015). Linköping University, Sweden: Linköping University Electronic Press, pp. 105–112. ISBN: 978-91-7685-900-1. DOI: 10.3384/ecp15119105.

Appendix

GENTPJ Mathematical Model

The derivation of the dynamic equations of GENTPJ are described in detail in (Pourbeik et al. 2016), they are summarized as follows.

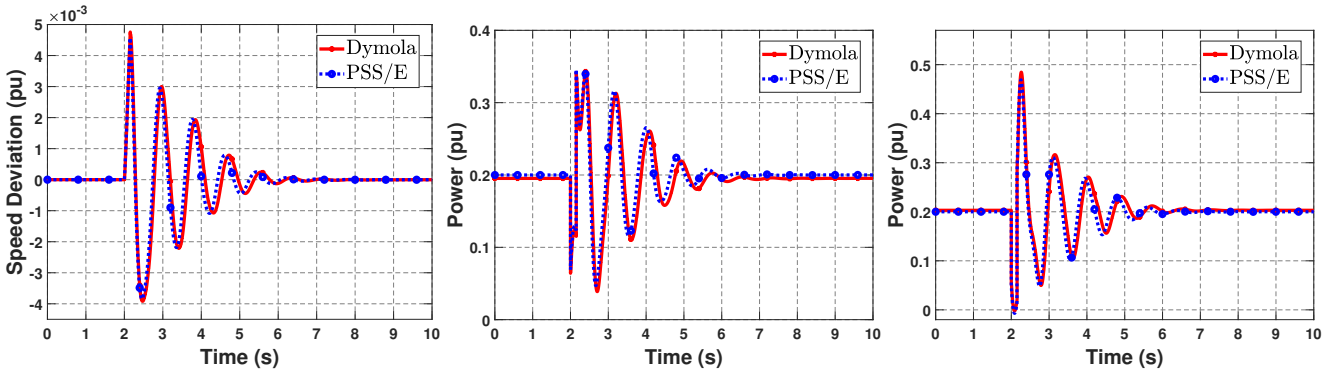
The differential equations are,

$$\frac{dE'_q}{dt} = \left[E_{fd} - (1+S_d)E_{q1}\right]\frac{1}{T'_{do}},$$
$$\frac{dE'_d}{dt} = -(1+S_q)\frac{E_{d1}}{T'_{qo}},$$
$$\frac{dE''_q}{dt} = -(1+S_d)\left(\frac{X'_d - X''_d}{X_d - X''_d}\right)\frac{E_{q2}}{T''_{do}},$$
$$\frac{dE''_d}{dt} = -(1+S_q)\left(\frac{X'_q - X''_q}{X_q - X''_q}\right)\frac{E_{d2}}{T''_{qo}}.$$

The algebraic equations for the terminal voltage in dq-axis are given by

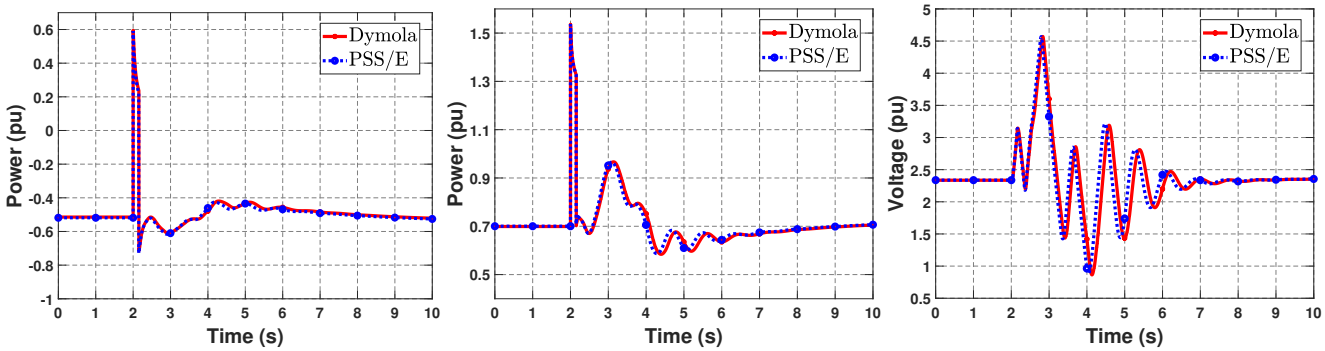
$$V_q = E_{q1} + E_{q2} - I_q r_a - I_d\left(\frac{X_d - X_l}{1+s_d} + X_l\right),$$
$$V_d = E_{d1} + E_{d2} - I_d r_a + I_q\left(\frac{X_q - X_l}{1+S_q} + X_l\right).$$

Meanwhile the auxialiry equations for the dq-axis voltage behind the transient and sub-transient impedances are de-



(a) Speed deviation at Bus02. **(b)** Active power at Bus01. **(c)** Active power at Bus02.

Figure 13. The system's speed deviation and active power in Figure 8.



(a) Reactive power at Bus01. **(b)** Reactive power at Bus02. **(c)** Field voltage at Bus02.

Figure 14. Reactive power and field voltage of system in Figure 8.

fine, for the q-axis, as

$$E_{q1} = E''_q - E_{q2} + I_d \left(\frac{X_d - X''_d}{1 + S_d} \right),$$

$$E_{q2} = \left[E''_q - E'_q + I_d \left(\frac{X'_d - X''_d}{1 + S_d} \right) \right] \left(\frac{X_d - X''_d}{X'_d - X''_d} \right),$$

and for the d-axis:

$$E_{d1} = E''_d - E_{d2} - I_q \left(\frac{X_q - X''_q}{1 + S_q} \right),$$

$$E_{d2} = \left[E''_d - E'_d - I_q \left(\frac{X'_q - X''_q}{1 + S_q} \right) \right] \left(\frac{X_q - X''_q}{X'_q - X''_q} \right).$$

Finally, the terminal voltage magnitude with leakage and the saturation is defined as:

$$E_l = \sqrt{(V_q + I_q r_a + I_d X_l)^2 + (V_d + I_d r_a - I_q X_l)^2}$$

$$S_d = f_S(E_l) \text{ and } S_q = \frac{X_q}{X_d} f_s(E_l).$$

where $f_s(E_l)$ saturation function introduced by quadratic open-circuit.

GENTPJ Modelica Implementation

Listing 1 presents the Modelica implementation of GENTPJ. Observe that in addition to dependencies to the Modelica Standard Library, there are important dependencies to the OpenIPSL. It is worth to note that the swing equations for the model are inherited from the base machine class `OpenIPSL.Machines.PSSE.baseMachine`, and therefore are not displayed within the listing below.

Listing 1. GENTPJ Partial Modelica Implementation

```
model GENTPJ "WECC Type J GENERATOR: ROUND
    ROTOR WITH SATURATION ON BOTH AXES."
  extends Icons.VerifiedModel;
  //Import of dependencies
  import Complex;
  import Modelica.ComplexMath.arg;
  import Modelica.ComplexMath.real;
  import Modelica.ComplexMath.imag;
  import Modelica.ComplexMath.abs;
  import Modelica.ComplexMath.conj;
  import Modelica.ComplexMath.fromPolar;
  import Modelica.ComplexMath.j;
  import OpenIPSL.NonElectrical.Functions.
      SE;

  extends BaseClasses.baseMachine(
    XADIFD(start=efd0),
    delta(start=delta0, fixed=true),
    id(start=id0),
    iq(start=iq0),
    Te(start=pm0),
    ud(start=ud0),
    uq(start=uq0));
```

```
  //Machine parameters
  parameter Types.PerUnit Xpq "q-axis
      transient reactance ";
  parameter Types.Time Tpq0 "q-axis
      transient open-circuit time constant"
      ;
  parameter Types.PerUnit Kis "Current
      multiplier for saturation calculation
      ";
  Types.PerUnit Epd(start=Epd0) "d-axis
      voltage behind transient reactance ";
  Types.PerUnit Epq(start=Epq0) "q-axis
      voltage behind transient reactance ";

  // Machine variable start values
  Types.PerUnit Eq1(start=Eq10);
  Types.PerUnit Eq2(start=Eq20);
  Types.PerUnit Ed1(start=Ed10);
  Types.PerUnit Ed2(start=Ed20);
  Types.PerUnit Xppdsat(start=Xppdsat0);
  Types.PerUnit Xppqsat(start=Xppqsat0);
  Types.PerUnit dsat(start=dsat0);
  Types.PerUnit qsat(start=qsat0);

  //State variables
  Types.PerUnit PSId(start=PSId0) "d-axis
      flux linkage ";
  Types.PerUnit PSIq(start=PSIq0) "q-axis
      flux linkage ";
  Types.PerUnit PSIppd(start=PSIppd0) "d-
      axis subtransient flux linkage ";
  Types.PerUnit PSIppq(start=PSIppq0) "q-
      axis subtransient flux linkage ";
  Types.PerUnit PSIpp "Air-gap flux ";
  Types.PerUnit XadIfd(start=efd0) "d-axis
      machine field current ";

protected
  parameter Complex Zs=R_a + j*Xppqsat0 "
      Equivalent impedance";
  parameter Complex VT=v_0*cos(angle_0) + j
      *v_0*sin(angle_0) "Complex terminal
      voltage";
  parameter Complex S=p0 + j*q0 "Complex
      power on machine base";
  parameter Complex It=real(S/VT) - j*imag(
      S/VT) "Complex current, machine base"
      ;
  parameter Complex Is=real(It + VT/Zs) + j
      *imag(It + VT/Zs) "Equivalent
      internal current source";
  parameter Complex PSIpp0=real(Zs*Is) + j
      *(imag(Zs*Is) - id0*(Xppqsat0-
      Xppdsat0))  "Sub-transient flux
      linkage in stator reference frame";
  parameter Types.Angle ang_PSIpp0=arg(
      PSIpp0) "flux angle";
  parameter Types.Angle ang_It=arg(It) "
      current angle";
  parameter Types.Angle ang_PSIpp0andIt=
      ang_PSIpp0 - ang_It "angle difference
      ";
  parameter Types.PerUnit abs_PSIpp0=abs(
      PSIpp0) "magnitude of sub-transient
      flux linkage";
  parameter Complex Z = R_a+j*Xl;
  parameter Complex PSIag= real(VT+Z*It) +
      j*(imag(VT+Z*It));
  parameter Real dsat0=1+SE(
      (abs(PSIag)+Kis*sqrt(id0*id0+iq0*iq0)
          ),
      S10,
      S12,
      1,
      1.2) "To include saturation during
          initialization";
  parameter Real qsat0=1+(Xq/Xd)*SE(
      (abs(PSIag)+Kis*sqrt(id0*id0+iq0*iq0)
          ),
      S10,
      S12,
      1,
      1.2) "To include saturation during
          initialization";
  parameter Real a=(abs(PSIag))*dsat0;
  parameter Real b=(It.re^2 + It.im^2)
      ^0.5*(Xppdsat0-Xq);
  //Initializion rotor angle position
  parameter Types.Angle delta0 = ang_PSIpp0
       + atan(b*cos(ang_PSIpp0andIt)/(b*sin
      (ang_PSIpp0andIt) - a))  "initial
      rotor angle in radians";
  parameter Complex DQ_dq=cos(delta0) - j*
      sin(delta0)  "Parks transformation,
      from stator to rotor reference frame"
      ;
  parameter Complex PSIpp0_dq=PSIpp0*DQ_dq
      "Flux linkage in rotor reference
      frame";
  parameter Complex I_dq=conj(It*DQ_dq); //
      "The terminal current in rotor
      reference frame"
  parameter Types.PerUnit PSIppq0=imag(
      PSIpp0_dq) "q-axis component of the
      sub-transient flux linkage";
  parameter Types.PerUnit PSIppd0=real(
      PSIpp0_dq)  "d-axis component of the
      sub-transient flux linkage";
  //Initialization of current and voltage
      components in rotor reference frame (
      dq-axes).
  parameter Types.PerUnit iq0=real(I_dq) "q
      -axis component of initial current";
  parameter Types.PerUnit id0=imag(I_dq) "d
      -axis component of initial current";
  parameter Types.PerUnit ud0=(-PSIq0) -
      R_a*id0 "d-axis component of initial
      voltage";
  parameter Types.PerUnit uq0=PSId0 - R_a*
      iq0 "q-axis component of initial
      voltage";
  //Initialization current and voltage
      components in synchronous reference
      frame.
  parameter Types.PerUnit vr0=v_0*cos(
      angle_0) "Real component of initial
      terminal voltage";
  parameter Types.PerUnit vi0=v_0*sin(
      angle_0) "Imaginary component of
      initial terminal voltage";
```


DOI 10.3384/ecp204653

```
  parameter Types.PerUnit ir0=-CoB*(p0*vr0
      + q0*vi0)/(vr0^2 + vi0^2) "Real
      component of initial armature current
       (system base)";
  parameter Types.PerUnit ii0=-CoB*(p0*vi0
      - q0*vr0)/(vr0^2 + vi0^2) "Imaginary
      component of initial armature current
       (system base)";
  //Initialization mechanical power and
      field voltage.
  parameter Types.PerUnit pm0=p0 + R_a*iq0*
      iq0 + R_a*id0*id0 "Initial mechanical
       power (machine base)";
  parameter Types.PerUnit efd0= dsat0*Eq10
      "Initial field voltage magnitude";
  parameter Types.PerUnit Epq0= PSIppd0 +
      id0*(Xpd-Xppd)/dsat0;
  parameter Types.PerUnit Epd0= -PSIppq0 -
      iq0*(Xpq-Xppq)/qsat0;
  parameter Types.PerUnit Eq10= ((-1)*
      PSIppd0*(Xd-Xpd) + Epq0*(Xd-Xppd))/(
      Xpd-Xppd);
  parameter Types.PerUnit Ed10= (PSIppq0*(
      Xq-Xpq)+Epd0*(Xq-Xppq))/(Xpq-Xppq);
  parameter Types.PerUnit Eq20= (PSIppd0-
      Epq0+id0*((Xpd-Xppd)/dsat0))*((Xd-
      Xppd)/(Xpd-Xppd));
  parameter Types.PerUnit Ed20= (-PSIppq0-
      Epd0-iq0*((Xpq-Xppq)/qsat0))*((Xq-
      Xppq)/(Xpq-Xppq));
  //Initialize remaining variables:
  parameter Types.PerUnit Xppdsat0=((Xppd-
      Xl)/dsat0)+Xl;
  parameter Types.PerUnit Xppqsat0=((Xppq-
      Xl)/qsat0)+Xl;
  parameter Types.PerUnit PSId0=PSIppd0 -
      Xppdsat0*id0;
  parameter Types.PerUnit PSIq0=PSIppq0 -
      Xppqsat0*iq0;
  // Constants
  parameter Real CoB=M_b/S_b  "Constant to
      change from system base to machine
      base";

initial equation
  der(Epd) = 0;
  der(Epq) = 0;
  der(PSIppd) = 0;
  der(PSIppq) = 0;

equation
  //Interfacing outputs with the internal
      variables
  XADIFD = XadIfd;
  ISORCE = XadIfd;
  EFD0 = efd0;
  PMECH0 = pm0;
  // Differential equations
  der(Epq) = (1/Tpd0)*(EFD - XadIfd);
  der(Epd) = (1/Tpq0)*(-1)*qsat*Ed1;
  der(PSIppd) = -(dsat)*((Xpd-Xppd)/(Xd-
      Xppd))*(Eq2/Tppd0);
  der(PSIppq) = (qsat)*((Xpq-Xppq)/(Xq-Xppq
      ))*(Ed2/Tppq0);
  Te = PSId*iq - PSIq*id;
  // Unsaturated air-gap flux
  PSIpp = sqrt((uq+iq*R_a+id*Xl)*(uq+iq*R_a
      +id*Xl)+(ud+id*R_a-iq*Xl)*(ud+id*R_a-
      iq*Xl));
  // Saturation on d-axis
  dsat=1+SE(
      ((PSIpp+Kis*sqrt(id*id+iq*iq))),
      S10,
      S12,
      1,
      1.2);
  // Saturation on q-axis
  qsat=1+(Xq/Xd)*SE(
      ((PSIpp+Kis*sqrt(id*id+iq*iq))),
      S10,
      S12,
      1,
      1.2);
  // Auxiliary Equations
  Eq1= ((-1)*PSIppd*(Xd-Xpd) + Epq*(Xd-Xppd
      ))/(Xpd-Xppd);
  Ed1= (PSIppq*(Xq-Xpq)+Epd*(Xq-Xppq))/(Xpq
      -Xppq);
  Eq2= (PSIppd-Epq+id*((Xpd-Xppd)/dsat))*((
      Xd-Xppd)/(Xpd-Xppd));
  Ed2=-(Epd+PSIppq)*((Xq-Xppq)/(Xpq-Xppq))-
      iq*((Xq-Xppq)/qsat);
  // Field Current
  XadIfd = dsat*Eq1;
  // Flux and saturated inductances
  Xppdsat=((Xppd-Xl)/dsat)+Xl;
  Xppqsat=((Xppq-Xl)/qsat)+Xl;
  PSId=PSIppd - Xppdsat*id;
  PSIq=PSIppq - Xppqsat*iq;
  // Terminal voltage
  ud = (-PSIq) - R_a*id;
  uq = PSId - R_a*iq;
end GENTPJ;
```