Guidelines and Use Cases for Power Systems Dynamic Modeling and Model Verification using Modelica and OpenIPSL

Giuseppe Laera[1], Luigi Vanfretti[1], Marcelo de Castro Fernandes[1], Sergio A. Dorado-Rojas[1], Fernando Fachini[1], Chetan Mishra[2], Kevin D. Jones[2], R. Matthew Gardner[2], Hubertus Tummescheit[3], Stéphane Velut[3], Ricardo J. Galarza[4]

[1]Rensselaer Polytechnic Institute, [2]Dominion Energy, [3]Modelon, [4]PSM Consulting, Inc.

American Modelica Conference 2022 | October 26–28, 2022

- Introduction
- Motivations
- Contributions
- Guidelines
  - Template Models for Modeling and Validation
  - Model Implementation Guide
  - Model Validation Guide
- Use cases
  - Power System Stabilizer PSS2A model
  - IEEE 421.5 2005 DC4B Excitation System model
  - Examples of interoperability
- Conclusions and Future Work

- Goal of the paper: to formalize the process of power systems dynamic modeling and model verification using the Modelica language and OpenIPSL
- Provide a formal description of the steps for complete re-implementation of power systems components of closed-source commercial software (PSS®E)
- Challenging key use cases are presented to highlight the value of the proposed approaches for model implementation and validation
- Show the advantages of self-documenting nature of object-oriented equation-based modeling offered by Modelica compared to existing modeling tools (not always transparent in the dynamic behavior description)
- Emphasizing the value of the open-access standardized Modelica specification as a key enabler of open-access standards-based interoperability:
  - Models can be re-utilized in multiple Modelica-standard-compliant tools without re-implementation

- Modeling of power systems has always been fundamental for the design, operation and planning of electric networks

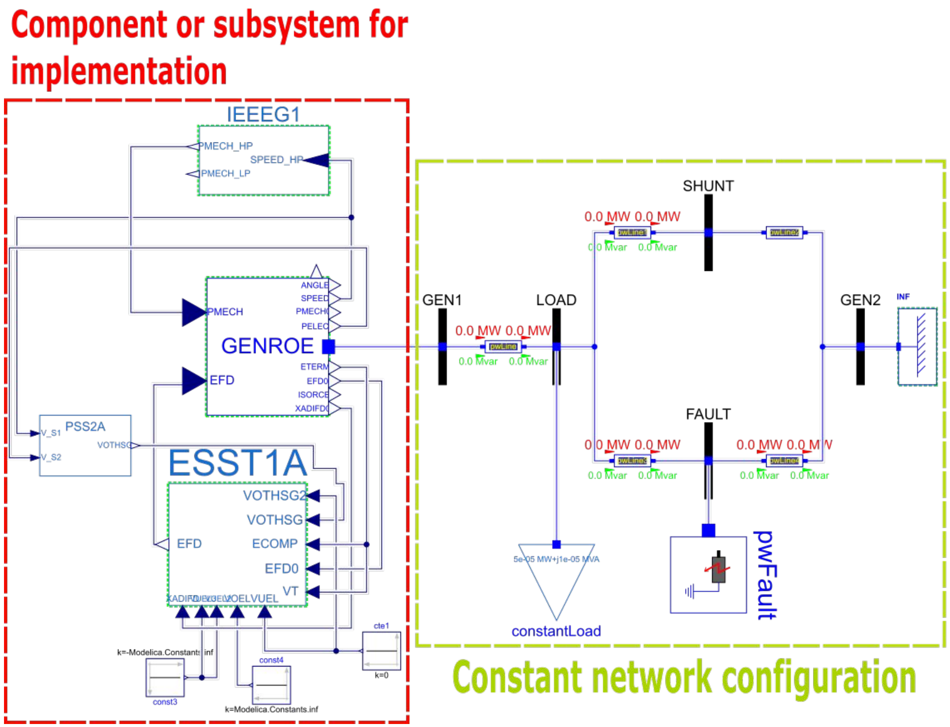
- Over the last decades several software tools for studies and analyses of electric power systems have been developed
  - User needs to be an expert of their functionalities to be productive
  - Each tool has its own "data format" → difficult to share models and data between tools

- Inconsistency of dynamic simulation results between different simulation platforms
  - Need to re-implement models in each simulation platform (tremendous costs)

- Idea to remove ambiguity in power systems modeling by using the object-oriented equation-based modeling language Modelica and the OpenIPSL library (maintained and expanded based on the concepts of regression testing and Continuous Integration)

- To formalize the model implementation approach used for component model development in Modelica to meet the requirements of the power industry
- To formalize the software-to-software model validation process for Modelica model validation against reference domain-specific tools, illustrated with the *de facto* standard in the power industry, PSS®E software
- To propose an approach to validate sub-system model components without the need to define entire system models in Modelica by replaying the reference tool simulation results into the Modelica model
- To illustrate the proposed approach with challenging implementation and validation use cases
- Synthesizing the know-how into guidelines which fully elicit the model implementation and validation process using flowcharts

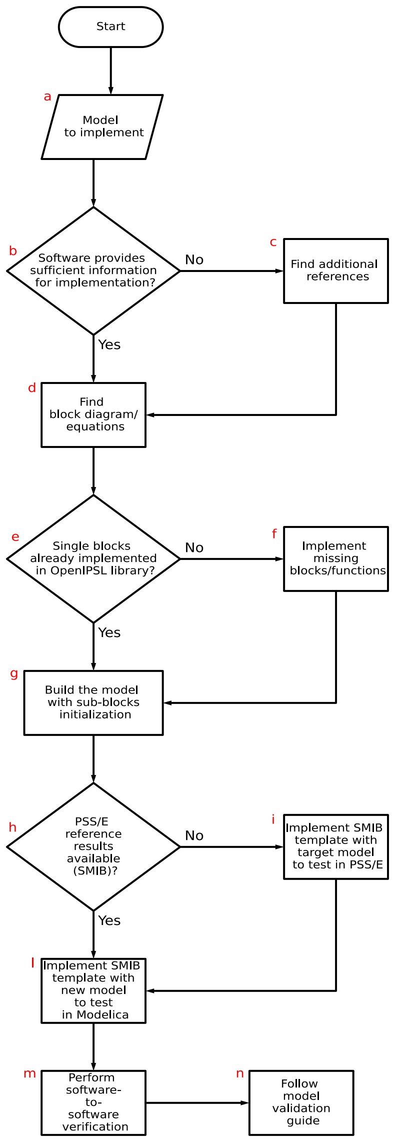
- Basic example of electric power system network is the SMIB used to model a power plant with its controls connected to the rest of the grid through transmission lines and substations represented by buses



- This small network also defines the model to standardize the testing of different device models implemented in Modelica and having PSS®E models as reference

- The modeling implementation process is illustrated in this flow chart

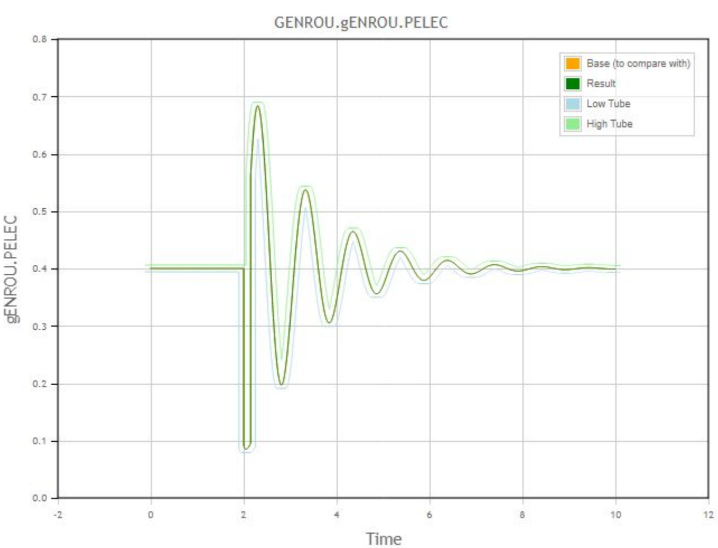
- While the approach is generic, the process depicted considers PSS®E the reference software tool

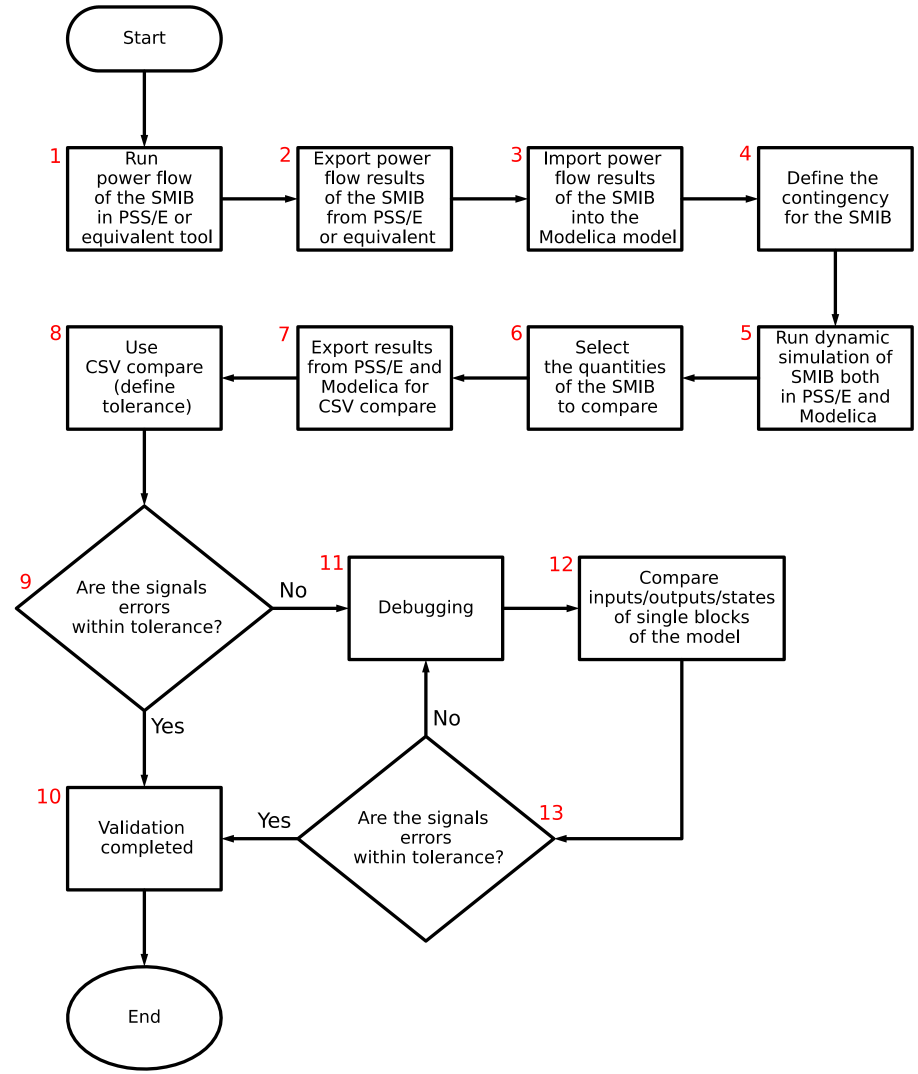


ALSET lab

- The model validation process is illustrated in this flow chart and it comes after the process of model implementation of the previous slide

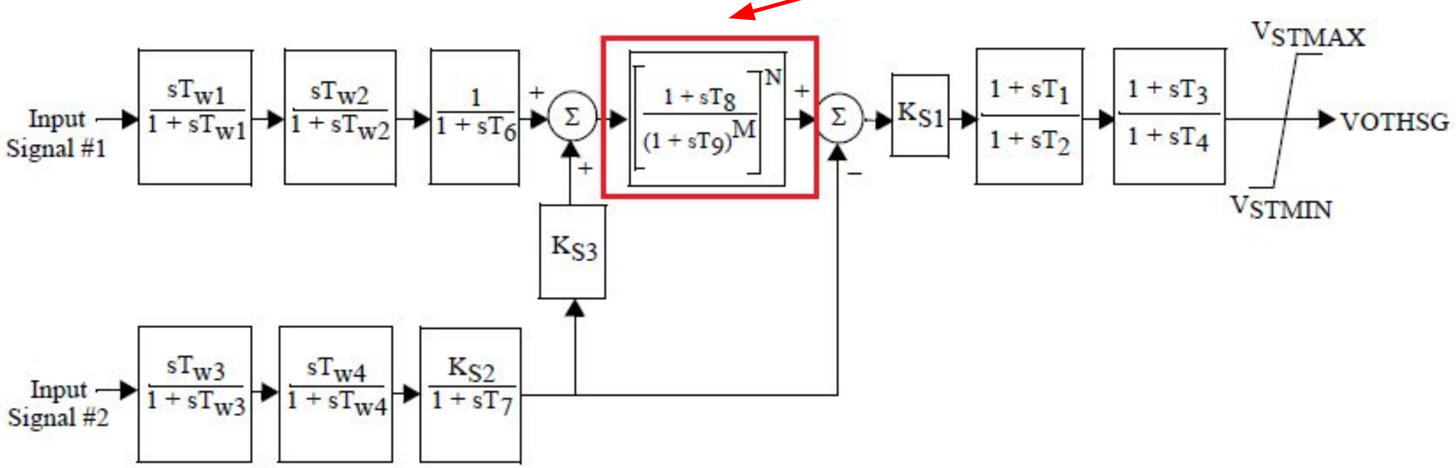
- Example with CSV compare tool (8)





- The reference software tool PSS®E comes with several manuals to help understanding the implementation and behavior of the different components present in its libraries

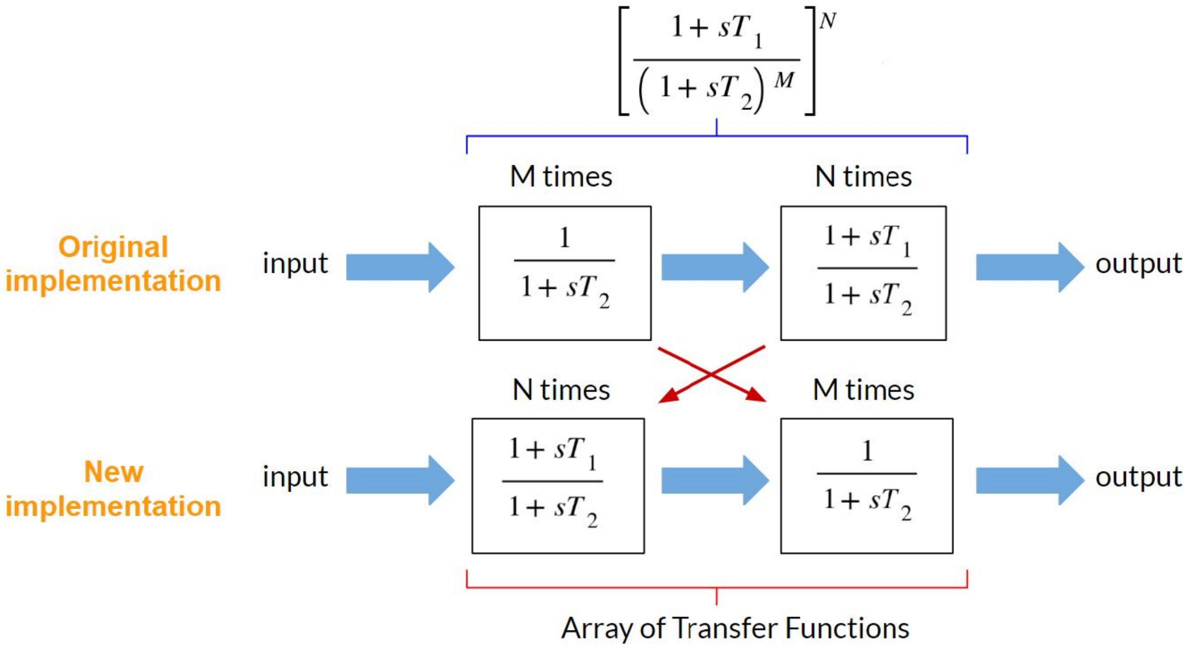
- In some cases, the documentation is insufficient like, for example, about the ramp tracking filter included in the model of PSS2A



PSS2A model from PSS®E manuals

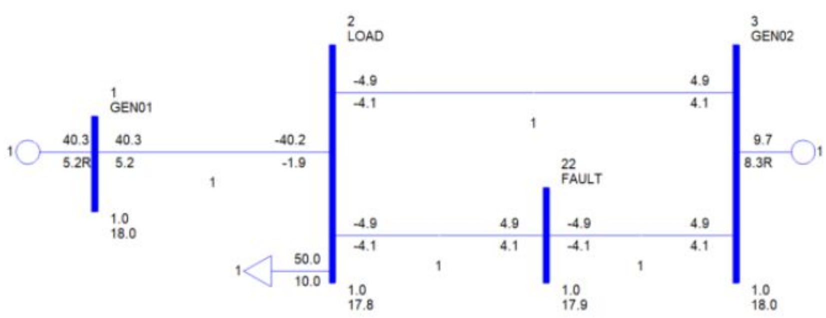
- The initial implementation of the ramp tracking filter in Modelica revealed to be not accurate compared to PSS®E implementation → additional investigations

- PSS®E documentation does not offer more details
  - Idea of analysing the continuous time trajectories of the states of this block available from the simulation of a SMIB with PSS2A in PSS®E

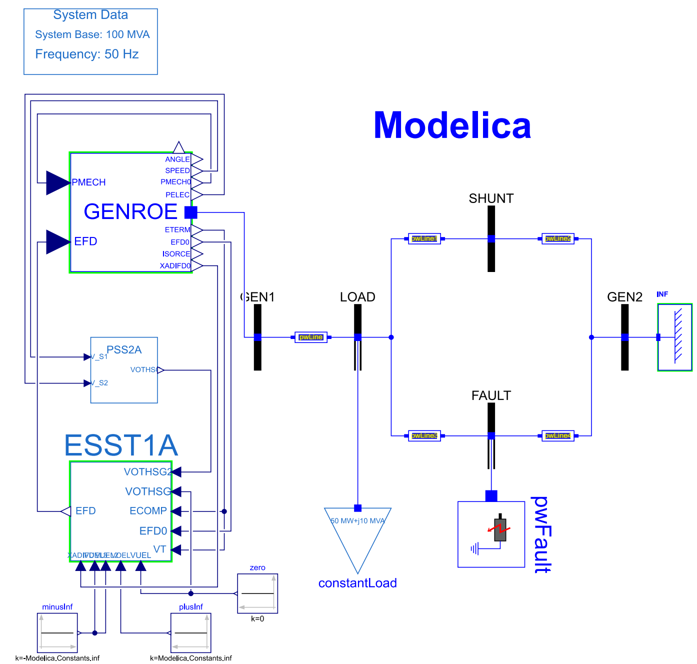


ALSET lab

- Modelica language is object-oriented and it allows for the creation of arrays of transfer functions to build the model of the ramp tracking filter
- Transparent and self-documented implementation of the model without leaving any uncertainty about the dynamics of the component
- The new implementation of the ramp tracking filter has been introduced in the PSS2A model and tested in a SMIB

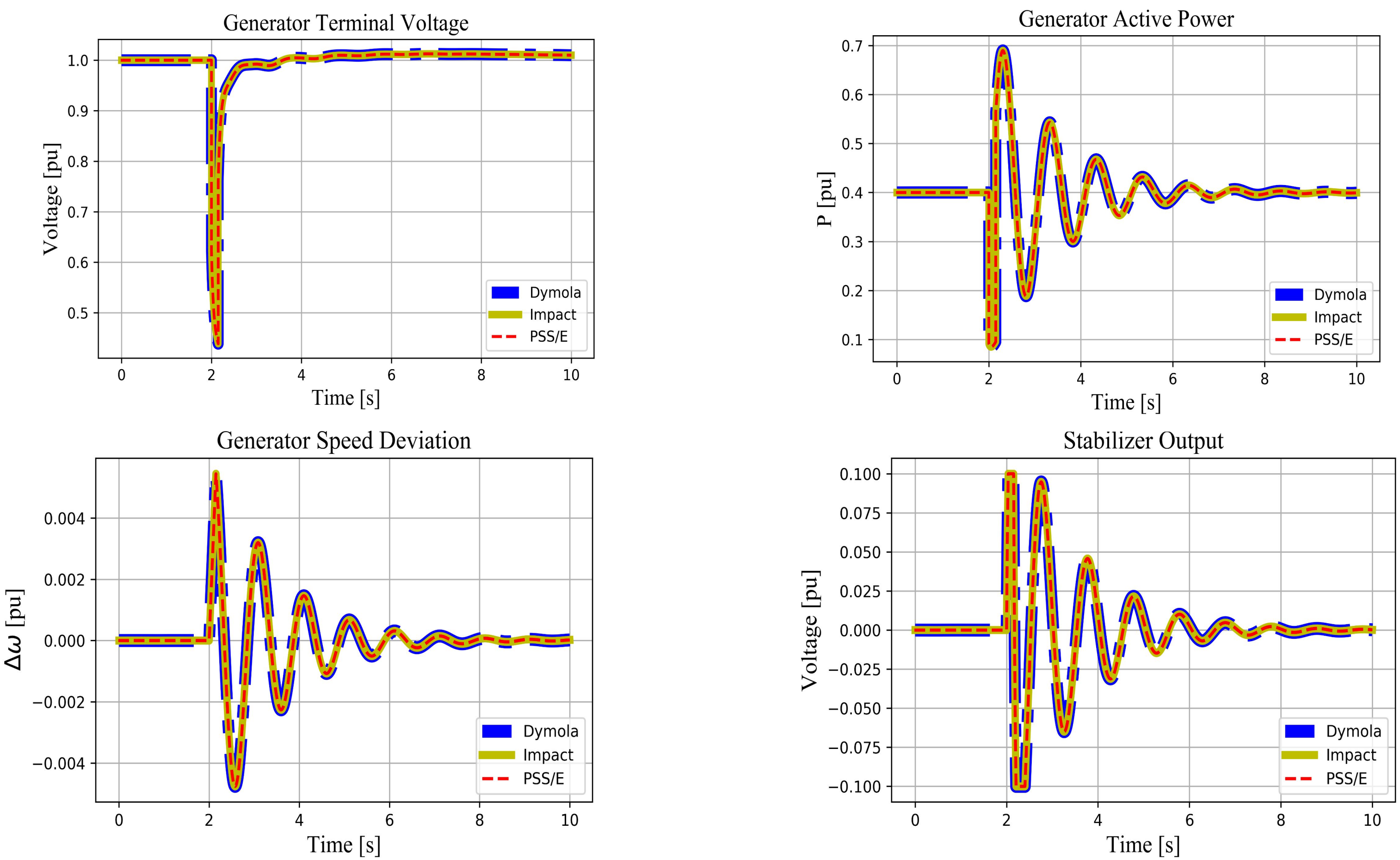


PSS/E

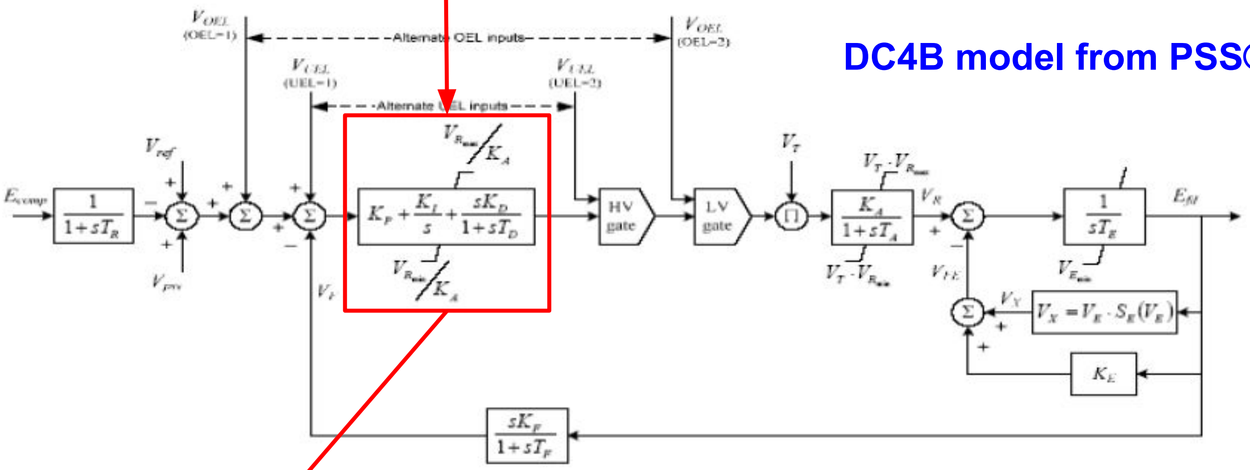


Power System Stabilizer (PSS2A) model (4/4)

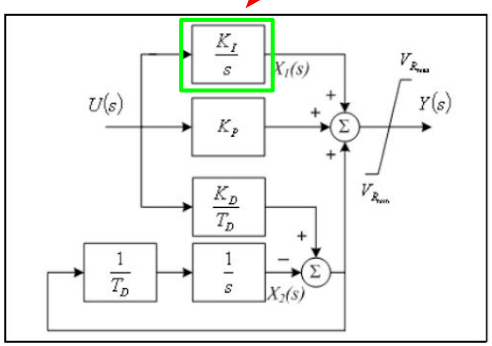
- Software-to-software validation (3-phase fault to ground applied at bus FAULT at t=2s for 0.15s)



- The challenge of the implementation of this component was represented by the integrator block inside the PID with non-windup limits block

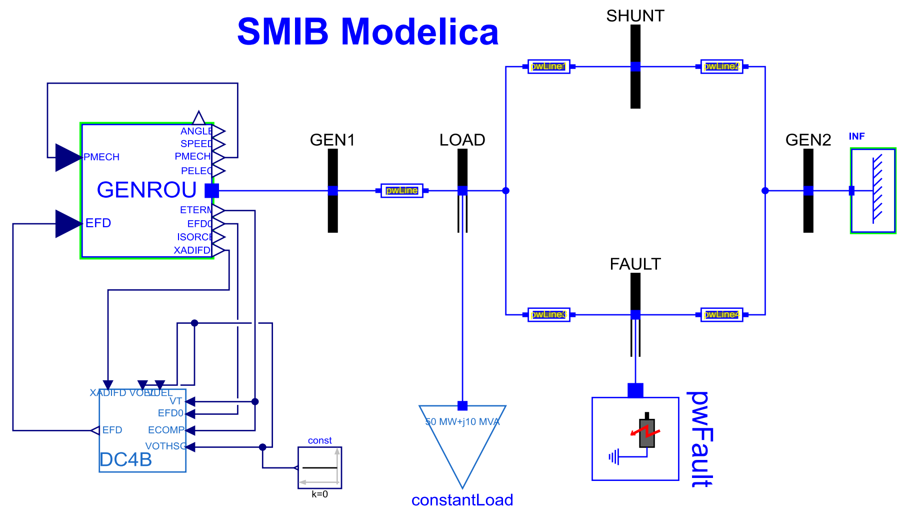


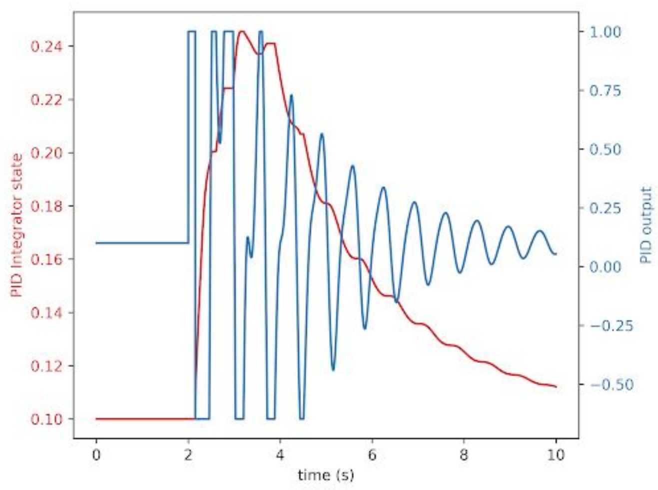
DC4B model from PSS®E manuals



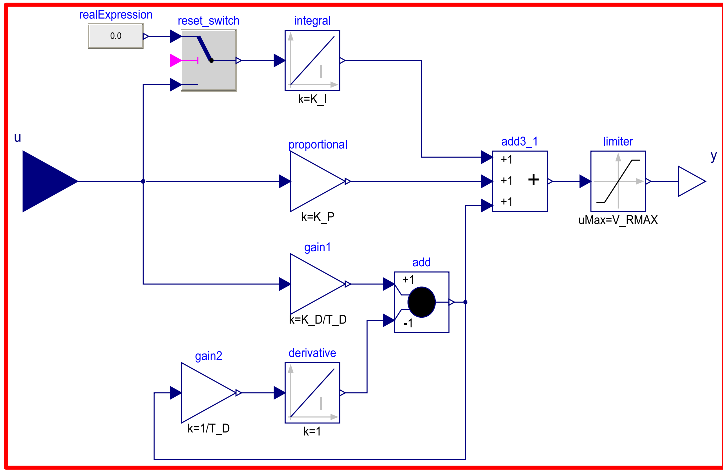
PID model from PSS®E manuals

- To find the right representation of the dynamics of the PID block it has been necessary to check the state of the integrator
- Observing the output of the PID block together with the state of the integrator of the PID during a dynamic simulation in PSS®E of the SMIB including the DC4B model
- Scenario is a 3-phase fault applied at bus FAULT at t=2s for 0.15s



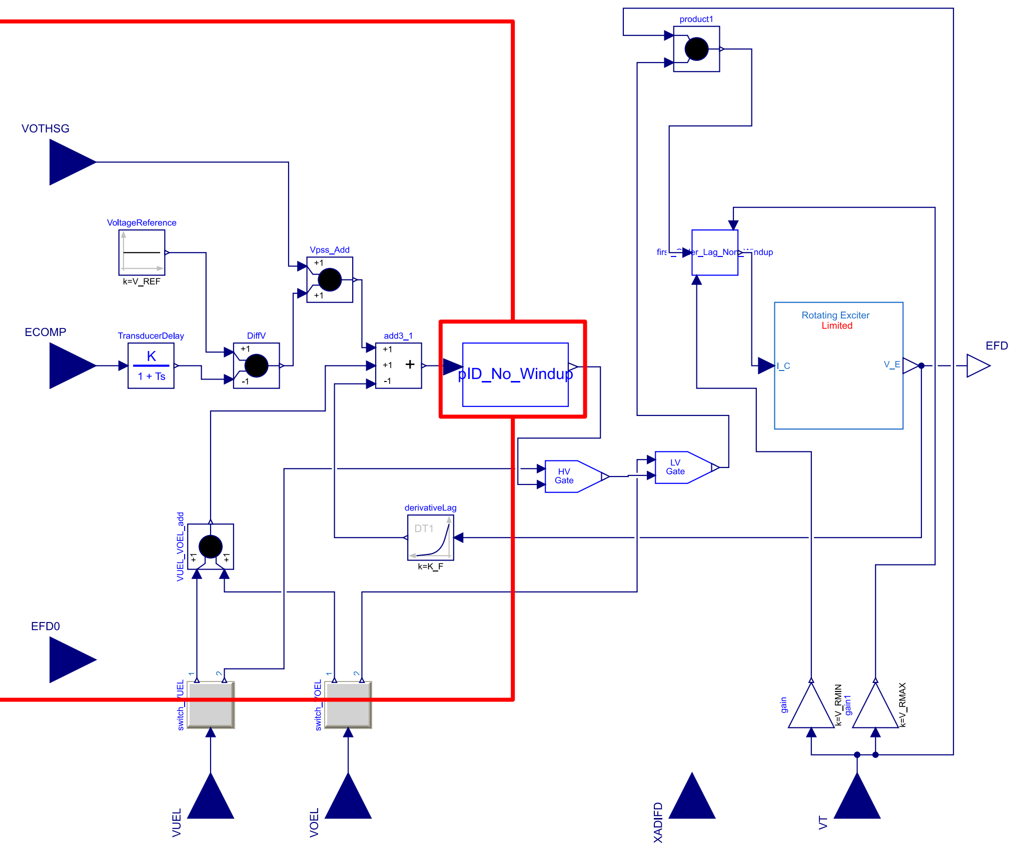


IEEE 421.5 2005 DC4B Excitation System model (3/6)



PID Modelica (Block digram)

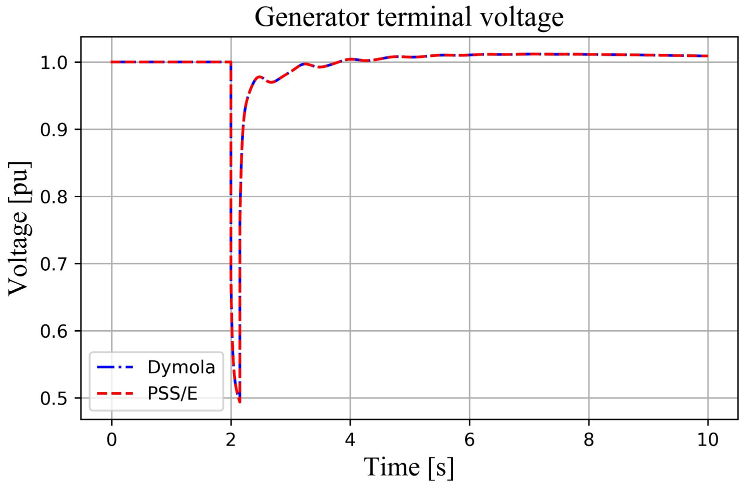
DC4B Modelica

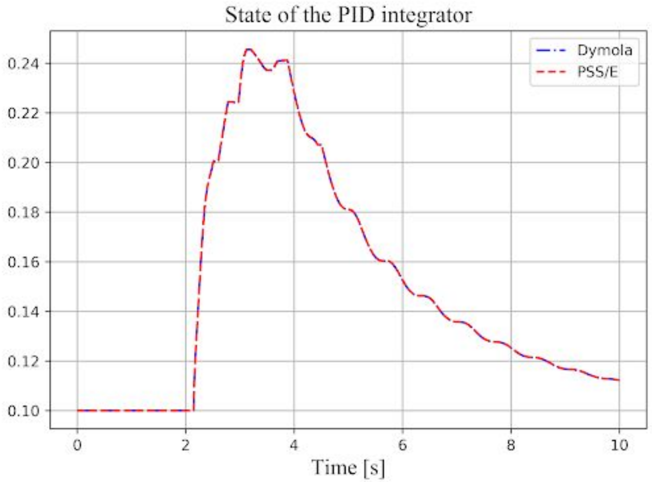


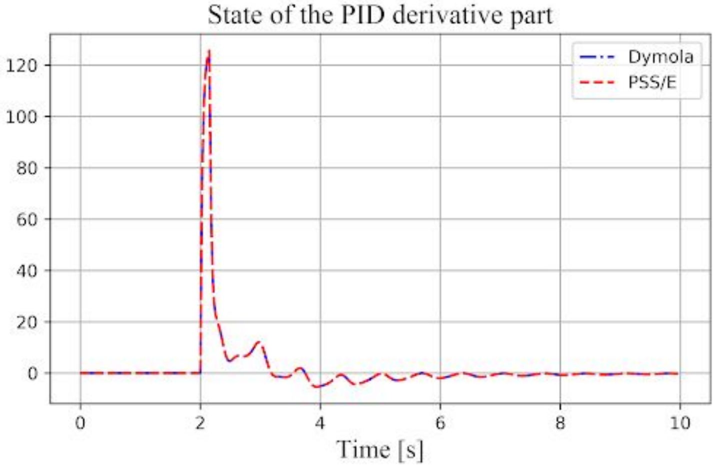
PID Modelica (Text layer)

```
model PID_No_Windup
  import      Modelica.Units.SI;
  parameter SI.PerUnit K_P "Voltage regulator proportional gain (pu)";
  parameter SI.TimeAging K_I "Voltage regulator integral gain (pu)";
  parameter SI.PerUnit K_D "Voltage regulator derivative gain (pu)";
  parameter SI.Time T_D "Voltage regulator derivative channel time constant (sec)";
  parameter SI.PerUnit V_RMAX "Maximum regulator output (pu)";
  parameter SI.PerUnit V_RMIN "Minimum regulator output (pu)";
  parameter SI.PerUnit K_A "Voltage regulator gain (pu)";
  parameter Real VR0;
  parameter Real VT0;

equation
  reset_switch.u2 =
    if (abs(V_RMAX - y) <= Modelica.Constants.eps and der(integral.y)>0) then true
    else if (abs(V_RMIN - y) <= Modelica.Constants.eps and der(integral.y)<0) then true
    else false;

end PID_No_Windup;
```

- Software-to-software validation (3 phase fault to ground applied at bus FAULT at t=2s for 0.15s)

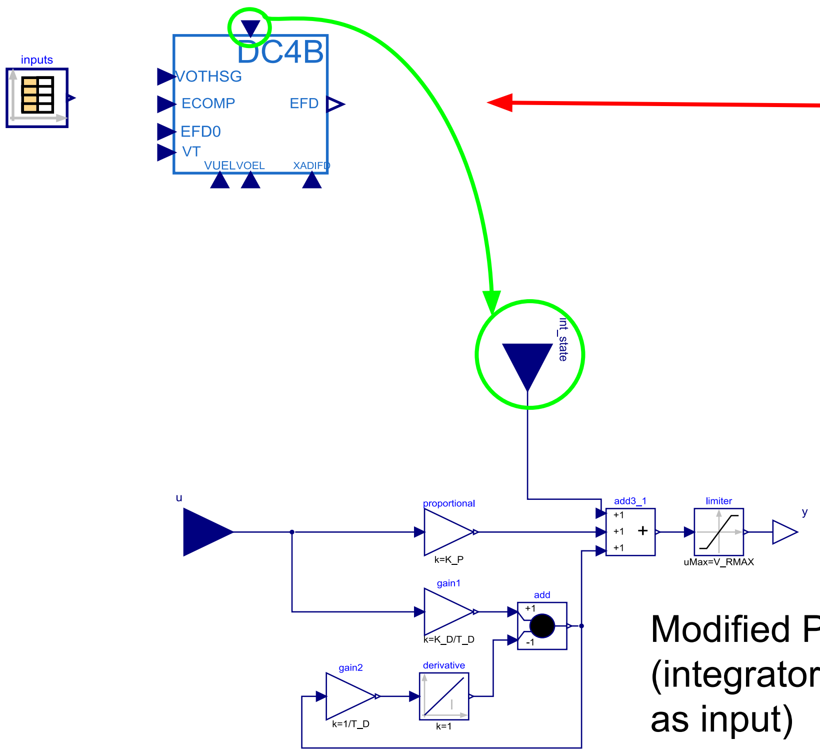






- An alternative approach for the validation of the DC4B excitation system:
  - Simulating only the block representing the DC4B component driven by external input signals collected from the reference SMIB network in PSS®E

DC4B test system (Block Diagram)

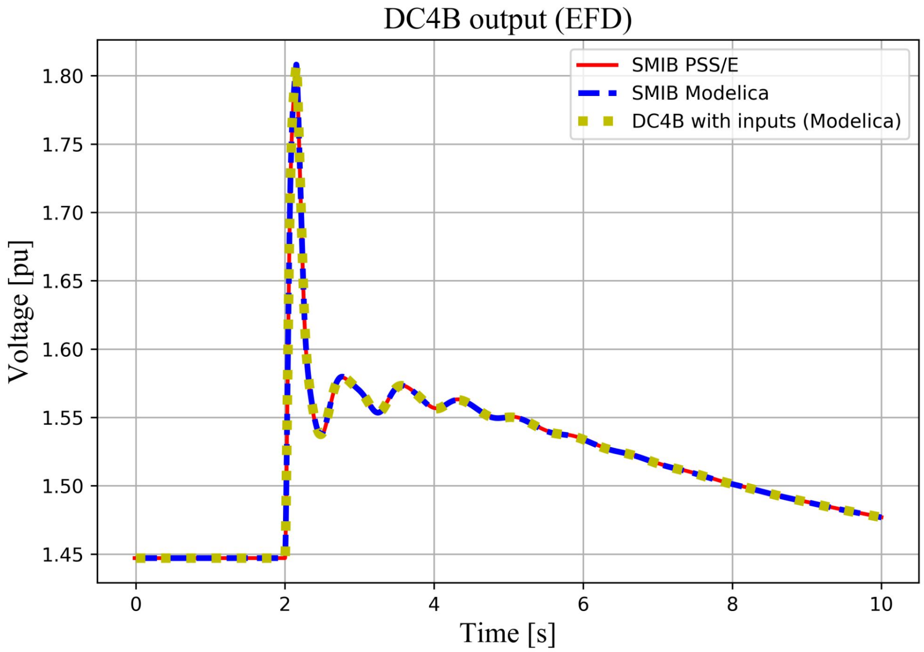


Modified PID
(integrator state trajectory as input)

DC4B test system (Text layer)

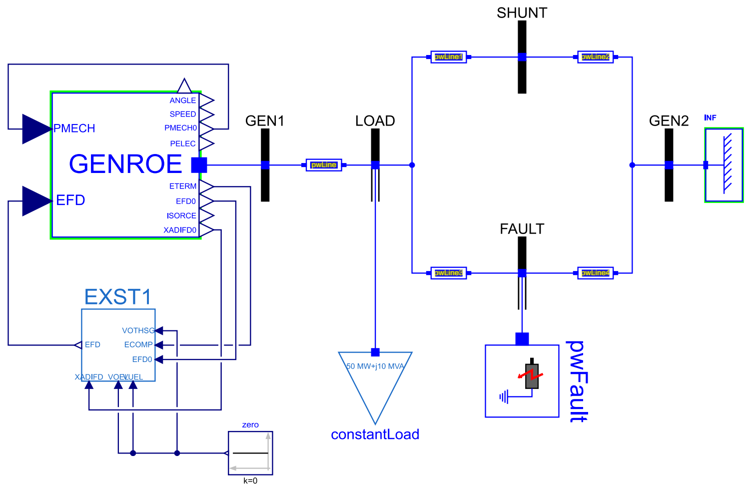
```
model DC4B_state_input_test

equation
  dC4B_state_input.VOTHSG = inputs.y[1];
  dC4B_state_input.ECOMP = inputs.y[2];
  dC4B_state_input.EFD0 = inputs.y[3];
  dC4B_state_input.VUEL = inputs.y[4];
  dC4B_state_input.VOEL = inputs.y[5];
  dC4B_state_input.XADIFD = inputs.y[6];
  dC4B_state_input.VT = inputs.y[7];
  dC4B_state_input.int_state = inputs.y[8];

end DC4B_state_input_test;
```

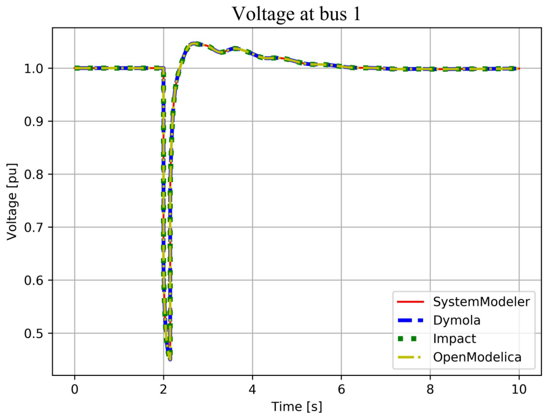
- Software-to-software validation (3-phase fault to ground applied at bus FAULT at t=2s for 0.15s)

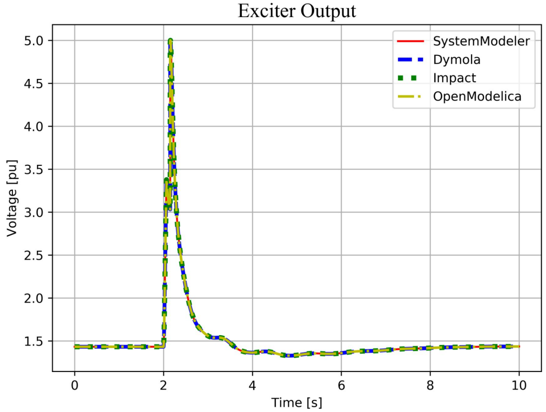


- The output of DC4B for the SMIB in Modelica, the corresponding SMIB in PSS®E and the new test system from the previous slide has been plotted

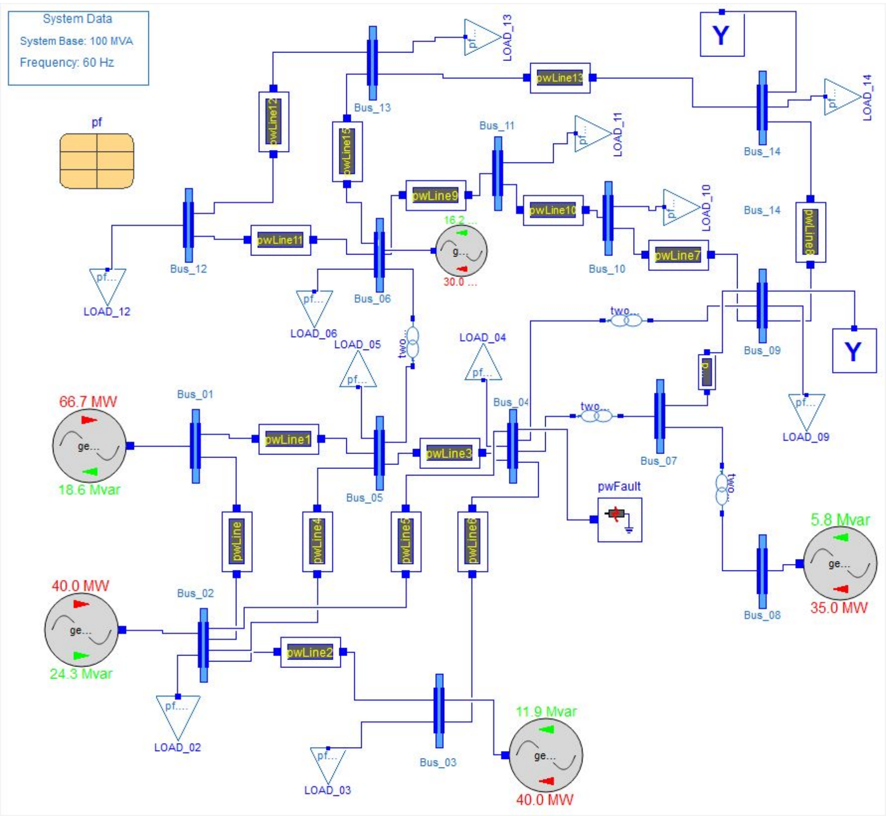
- Additional value of implementing and validating models with the Modelica language: **portability** (tool-specific re-implementation can be avoided)
- Example of a SMIB with the exciter EXST1 from OpenIPSL library

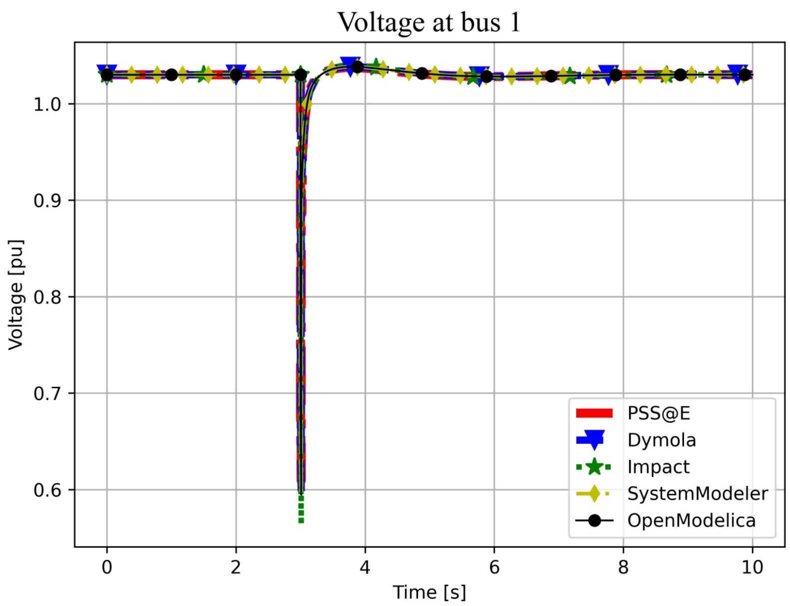






- Another example is the network model IEEE14





- 3-phase fault to ground (Z = R+jX = (0.01+j0.02)pu) at Bus 4 at time t=3s for 0.01s

- This paper formalizes and illustrates a procedure to implement power system models in Modelica with a software-to-software validation methodology
- The described steps are important for developing and maintaining a Modelica library using the concepts of regression testing and continuous integration
- The guidelines illustrated in this paper are indispensable for the initial debugging of new models addressing all possible challenges of the time consuming re-implementation process in a systematic way
- Challenging key use cases were provided to give an idea of the difficulties faced when developing power system models from a reference software tool
- Once the results of an initial validation are satisfactory then the software-to-software validation process can be automated using scripts to test different scenarios in the different simulation platforms (future work)

Thank you for your attention!

Rensselaer

why not change the world?®