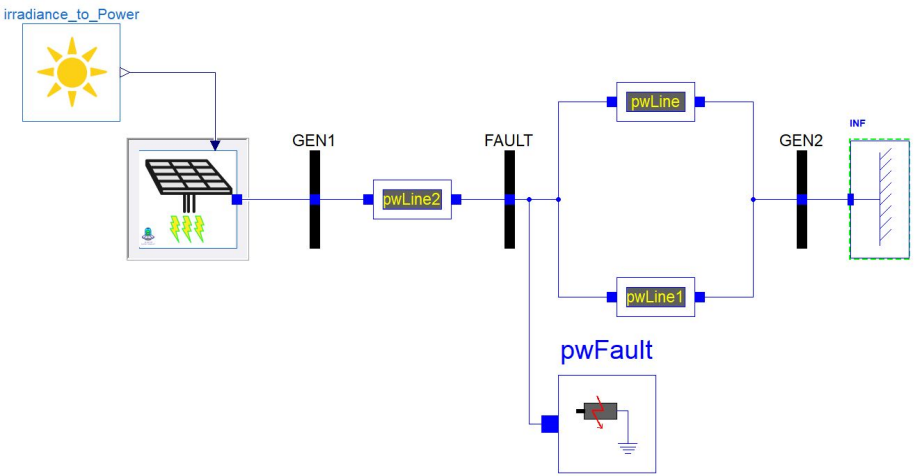
Rensselaer

ALSETlab why not change the world?®



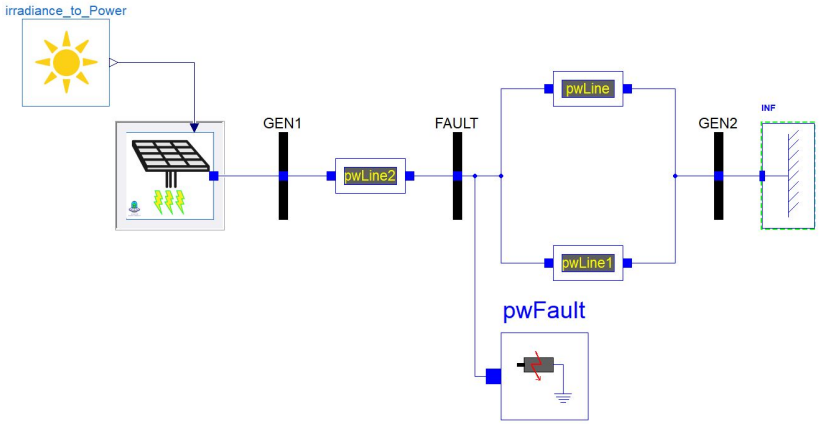
Overcoming Resistance to Change:

10 years of development of the Modelica-based OpenIPSL library for electrical power grid simulation

Dr. Luigi Vanfretti - Associate Professor - Rensselaer Polytechnic Institute, Troy, NY, USA
E-mail: luigi.vanfretti@gmail.com Web: https://alsetlab.com

Overview

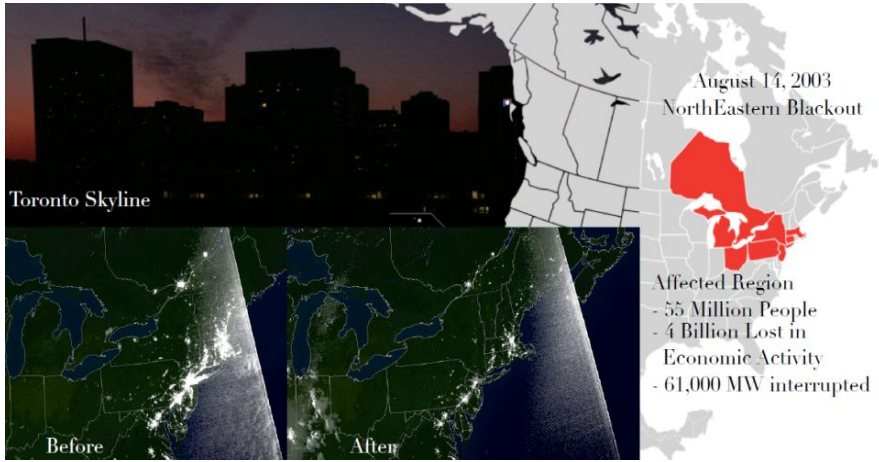
- **Motivation**
  - Need for Modeling and Simulation
  - New York's Future Grid Needs
  - Consequences of the Status-Quo
- **Introduction**
  - Modelica and OpenIPSL
  - Why OpenIPSL?
  - What is the OpenIPSL Library?
- **Model Development Process**
  - WECC Generic Renewable Energy Models: Photovoltaic (PV), Wind, and Battery Energy Storage Systems (BESS)
- **Portability & Interoperability**
  - Enabling analysis in multiple tools, to the cloud and real-time with Modelica and FMI!
- **Automating Model Verification and Model Transformation**
  - Enabling easier transition to Modelica+FMI Technologies
- **Conclusions and Future Work**



Motivation

The Need for Modeling and Simulation of the Grid

- Models have always been created for power system analysis.
  - It is a way to study each device and how many devices behave when interconnected.
- Networks increased in size and in complexity of devices.
  - Domain specific simulation tools have been adapted to deal with these issues.
- Simulation tools should provide means of anticipate any failures and insight to improve the current power system.
  - **Failure to anticipate events may result in huge costs!**
- Many examples of such events can be found in the last 20 years:
  - WECC 1996 Break-up, European Blackout (4-Nov.-2006), London (28-Aug-2003), Italy (28-Sep.-2003), Denmark/Sweden (23-Sep.-2003)



Failure!

Existing modeling and simulation (and associated) tools were unable to predict this (and other) events.

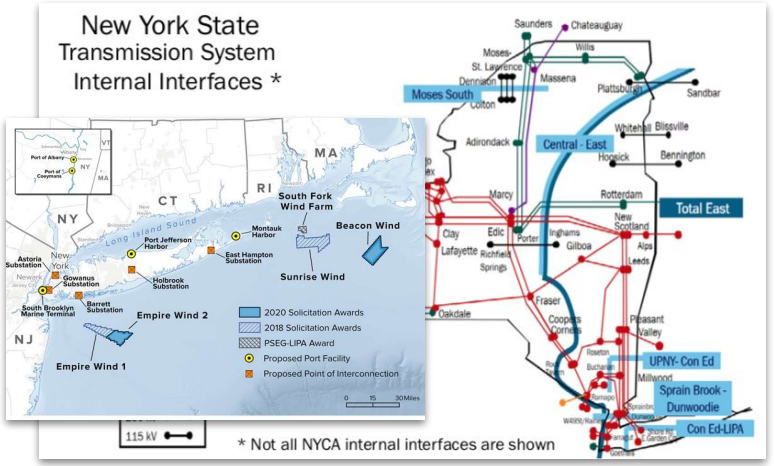
Motivation

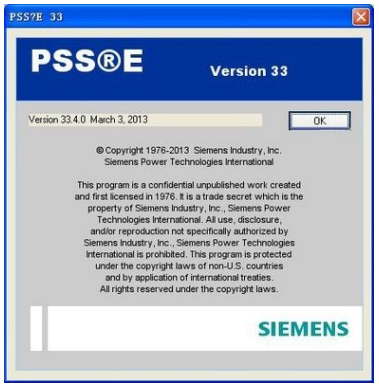
Future Energy Scenario

- Addition of large number of renewable energy sources in the grid, both utility scale and at lower voltage levels (DER)
- New modeling and simulation capabilities and tools will be required:
  - **The performance of clean energy technologies need to be properly studied before deployed!**
  - Models need to be portable, i.e. can be used in different software tools, to perform different types of analysis and support decision making.

Today's Challenges with Analysis Tools

- ***Lock-in:*** Few specific and proprietary software tools bound models to single tool, user is locked-in.
- ***Lack of interoperability***: Files retain parameter details but no information on equations
- ***Duplication and Inconsistenc***y: Many different models are created to represent the same system, difficult to obtain the same results.





.RAW

.DYR

Motivation

Consequences of Status-Quo on M&S

- ***Caveat 1:*** Not being able to understand the dynamic performance and interaction with the grid of new tech leads to unexpected consequences.
  - ***Major design flaws*** slow-down the adoption of clean energy technologies and increase their cost.
  - ***PV Example:*** Since 2011, Germany started a costly (estimate of €175 Million) retrofitting of PV inverters' firmware for more than 350,000 PVs to address over-frequency protections [1] whose impact was not studied until was too late [2].
  - ***Wind Example***: Wind turbine/farm sub-synchronous control interactions, not identified and addressed during design. During operations, the only solution to mitigate them is curtailment [3].

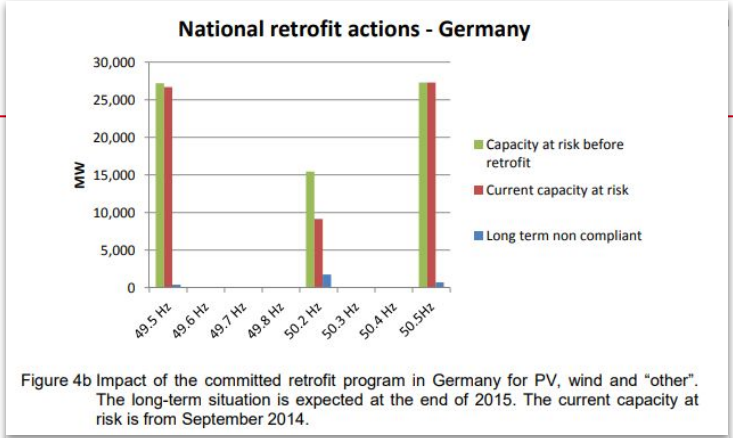
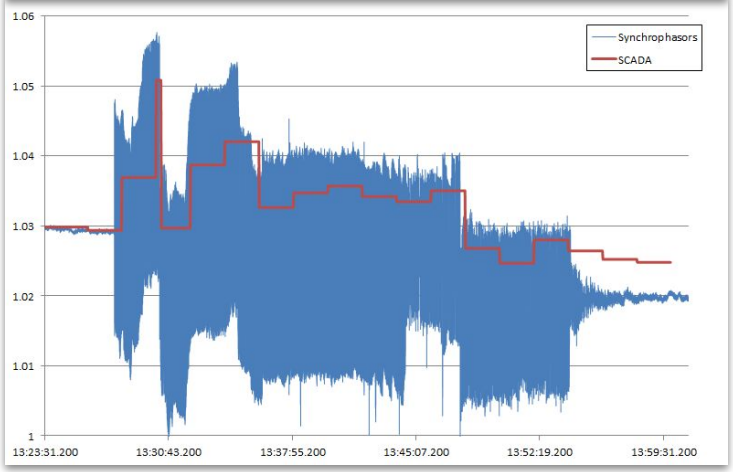
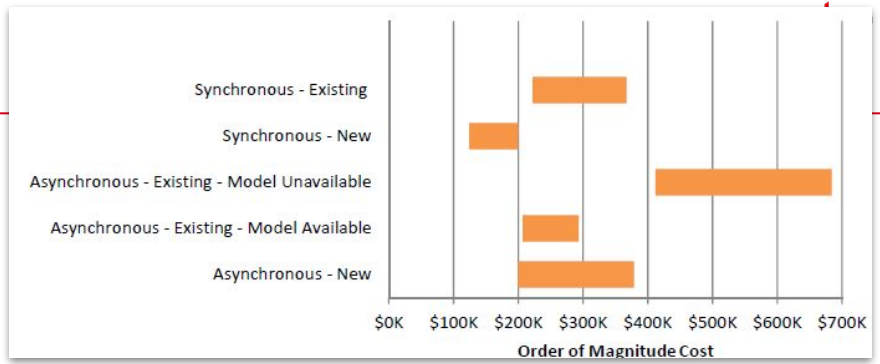


Figure 4b Impact of the committed retrofit program in Germany for PV, wind and "other". The long-term situation is expected at the end of 2015. The current capacity at risk is from September 2014.



[1] Jens. C. Boemer, et al, "Overview of German Grid Issues and Retrofit of Photovoltaic Power Plants in Germany for the Prevention of Frequency Stability Problems in Abnormal System Conditions of the ENTSO-E Region Continental Europe," 1st International Workshop on Integration of Solar Power into Power Systems, 24 October 2011, Aarhus, Denmark.
[2] ENTSO-E, "Dispersed Generation Impact on CE Region Security - Dynamic Study." 2014. Online: here
[3] L. Vanfretti, M. Baudette, and A. White, "Monitoring and control of renewable energy sources using synchronized phasor measurements," Book Chapter, in Renewable Energy Integration: Practical Management of Variability, Uncertainty and Flexibility in Power Grids, Editor: Lawrence E. Jones, Elsevier, 2014. Online: here

Motivation



Consequences of Status-Quo on M&S

- ***Caveat 2:*** Time consuming and expensive process to develop and use models for new technology:
  - Costs for new model development can reach over $600,000.00 per model [1].
  - ***Lack of interoperability:*** models have to be implemented for specific software tools, increasing costs for porting and analysis.

Proposed Solution

- ***Open Access*** (no pay wall) ***Standard*** (developed and maintained) computer and human readable ***modeling language*** to define models
  - Provide separation between the model's mathematical description and the numerical solver/tool
  - Avoids vendor lock-in: many tools can support the standard.

[1] Lasantha Perera and Winodh Jayewardene, "EMT and RMS Model Requirements" AECOM Australia Pty Ltd. Prepared for the Australian Energy Market Commission, June 2017.

Modelica for Grid Modeling

Introduction - What is Modelica & OpenIPSL?

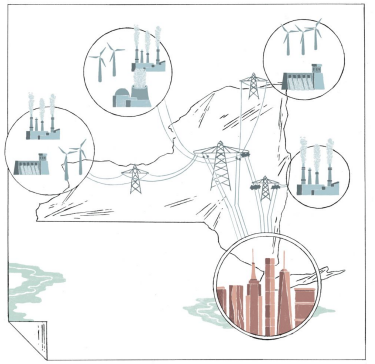
Modelica

- Non-proprietary, object-oriented, equation-based ***modeling language*** for cyber physical systems .
- Open access (no paywall) & standardized language specification (link), maintained by the Modelica Association
- Open source Modelica Standard Library with more than 1,600 components models.
- *Supported by 9 tools natively*, both proprietary (Dymola, Modelon Impact, etc.) and Open Source (OpenModelica)
- A vast number of proprietary and open-source Modelica Libraries

OpenIPSL

- OpenIPSL is an open-source Modelica library for power systems that:
  - Contains a vast number of power system components for phasor time domain modeling and simulation of power systems (transmission and distribution)
  - Several models have been verified against a number of reference tools (PSS/E, PSAT).
- OpenIPSL enables:
  - Unambiguous model exchange, use of model in Modelica-compliant tools, and export with FMI.
  - Formal mathematical description, no discretization w.r.t. specific integration method.
  - Separation of models from tools and solvers.
  - Using Dymola, as fast* as PSS/E (link).

Introduction - Modelica and Power Systems

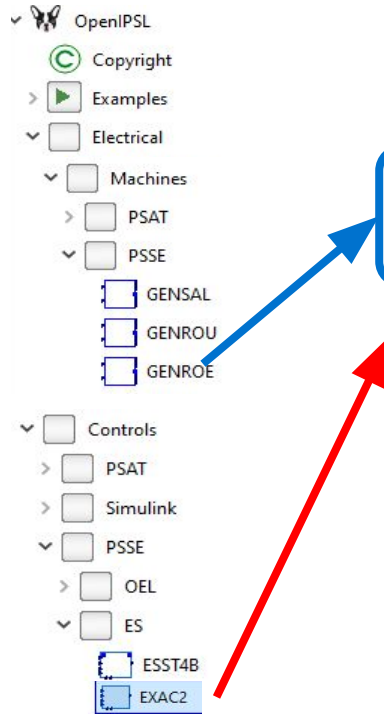


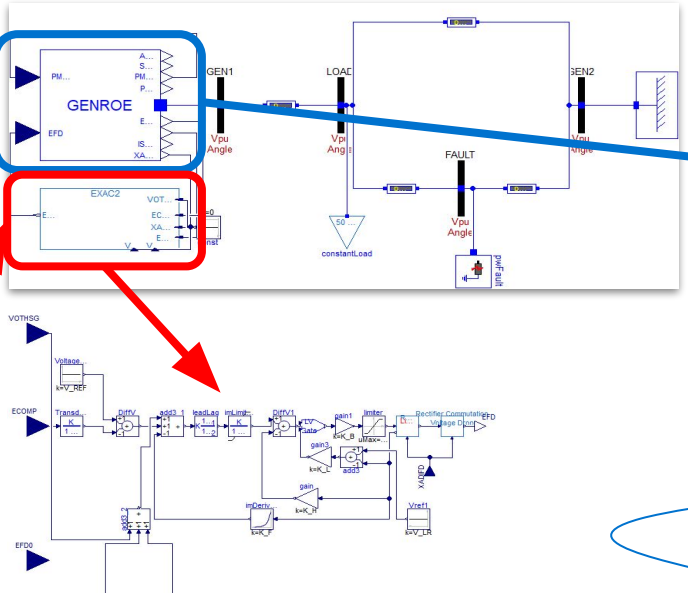
Previous and Related Efforts

- Modelica for power systems *was first attempted* in the early 2000's (Wiesma Modelica 2000) - "electromagnetic transient (EMT) modeling" approach.
  - SPOT (Weissman, EPL-Modelon) and its close relative PowerSystems supports multiple modeling approaches –i.e. 3phase, steady-state, "transient stability", etc.
- Electro-mechanical modeling or "transient stability" modeling:
  - Involves electro-mechanical dynamics, and neglects (very) fast transients
  - For system-wide analysis, easier to simulate/analyze - domain specific tools approach
- ObjectStab (Larsson, 2002; Winkler, 2015) adopts "transient stability" modeling.
- The PEGASE EU project (2011) developed a small library of components in Scilab, which where ported to proper Modelica in the FP7 iTesla project (2012-2016).
- The iPSL - iTesla Power Systems Library (Vanfretti et al, Modelica 2014, SoftwareX 2016), was released during 2015. Most models validated against typical power system tools (e.g. PSS/E)
- **OpenIPSL: Open-Instance Power Systems Library is an evolution of iPSL, with major improvements, since 2016.**

OpenIPSL will be covered in this this presentation.

Introduction - OpenIPSL Library and Example





equation

//Interfacing outputs with the internal variables

$$XADIFD = XadIfd$$
$$ISORCE = XadIfd$$
$$EFD0 = efd0$$
$$PMECH0 = pm0$$
$$\frac{d\,Epq}{d\,t} = \frac{1}{Tpd0}\left(EFD - XadIfd\right)$$
$$\frac{d\,Epd}{d\,t} = \frac{1}{Tpq0}\left(-1\right)XaqIlq$$
$$\frac{d\,PSIkd}{d\,t} = \frac{1}{Tppd0}\left(Epq - PSIkd - \left(Xpd - Xl\right)id\right)$$
$$\frac{d\,PSIkq}{d\,t} = \frac{1}{Tppq0}\left(Epd - PSIkq + \left(Xpq - Xl\right)iq\right)$$
$$Te = PSId \cdot iq - PSIq \cdot id$$
$$PSId = PSIppd - Xppd \cdot id$$
$$PSIq = \left(-PSIppq\right) - Xppq \cdot iq$$
$$PSIppd = Epq \cdot K3d + PSIkd \cdot K4d$$
$$-PSIppq = \left(-Epd \cdot K3q\right) - PSIkq \cdot K4q$$
$$PSIpp = \sqrt{PSIppd \cdot PSIppd + PSIppq \cdot PSIppq}$$
$$XadIfd = K1d \cdot \left(Epq - PSIkd - \left(Xpd - Xl\right)id\right) + Epq$$
$$XaqIlq = K1q \cdot \left(Epd - PSIkq + \left(Xpq - Xl\right)iq\right) + Epd$$

//change sign for PSIppq 3/3

$$ud = \left(-PSIq\right) - R_a \cdot id$$
$$uq = PSId - R_a \cdot iq$$

//flow

end GENROE;

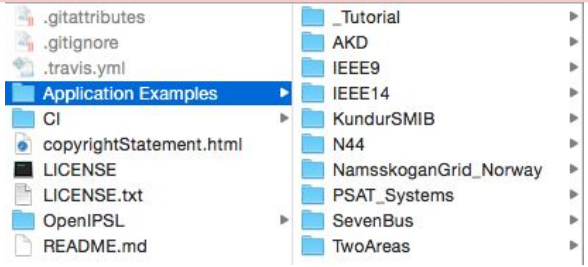
$$+ id \cdot \left(Xd - Xpd\right) + SE\_exp\left(PSIpp, S10, S12, 1, 1.2\right)PSIppd$$
$$- iq \cdot \left(Xq - Xpq\right) - \frac{SE\_exp\left(PSIpp, S10, S12, 1, 1.2\right)\left(-1\right)PSIppq \cdot \left(Xq - Xl\right)}{Xd - Xl}$$

Note: Modelica uses object-orientation, so the swing equations are not shown because they are inherited from a base class, as they are the same for all models.

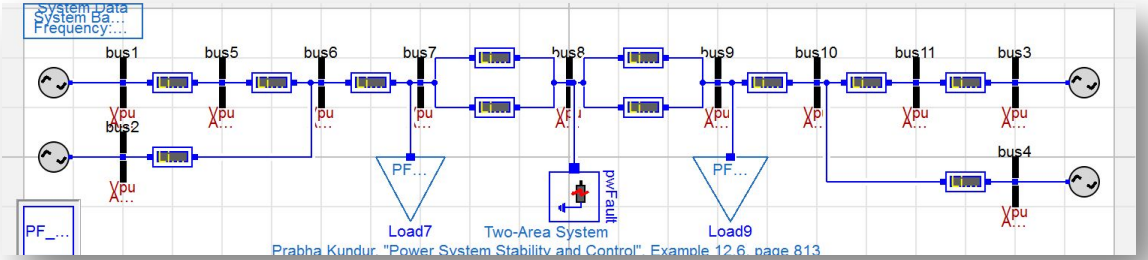
Introduction - OpenIPSL “Application Examples”

ALSETlab

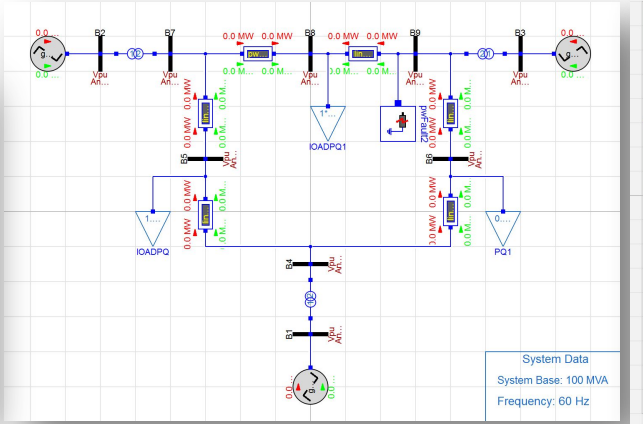
Many Application Examples Developed!!!



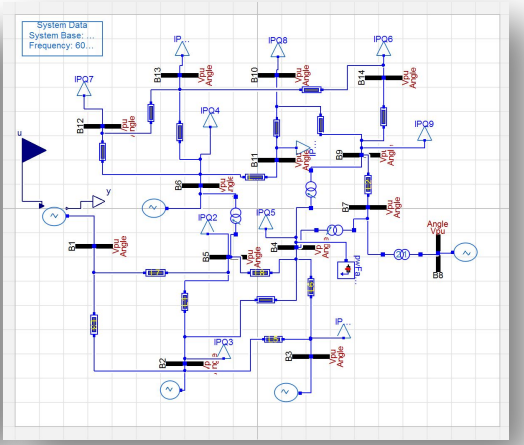
Klein-Rogers-Kundur 2-Area 4-Machine System



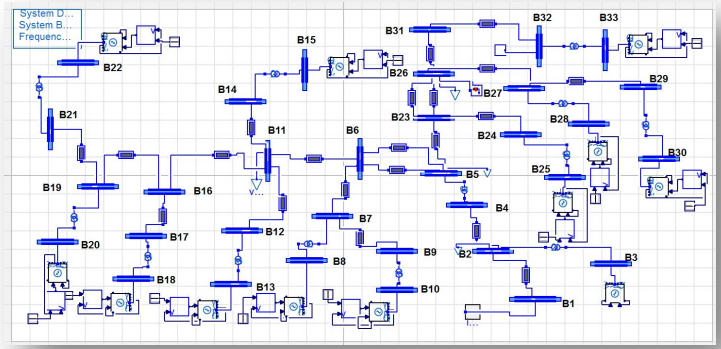
IEEE 9 Bus



IEEE 14 Bus



Namsskogan Distribution Network



Introduction - Addressing Resistance to Change

Why another library for power systems?

- Because the previous approaches have largely failed in gaining adoption.
- They had not addressed issues related to "resistance to change", albeit, they have only been identified relatively recently (Vanfretti et al, Modelica 2014).

- **Overcoming Resistance to Change:**

  - Make the recipient's "resistance" and the agent sense-making public part of the discourse for change:

    - Users of conventional power system tools are skeptical about any other tools different to the one they use (or develop), and are averse about new technologies (slow on the uptake)
    - Face the "your baby is ugly" syndrome

  - Establish a strategy and objectives, and mobilize action.
    - Change agents contribute (+/-) to address resistance through actions and interactions.
    - How?

Introduction - Resistance to Change

From management research:

- Change agent: those responsible of identifying the need for change, creating a vision, and making it happen.
- Change agent-centric view: resistance is an irrational and dysfunctional reaction of change recipients.
- Self-serving and self-fulfilling label:
  - Given by change agents attempting to make sense of recipients' reactions.
- Change agents contribute to the occurrence of the reactions labeled as resistance through actions/inactions:
  - Breach of agreement
  - Failure to gain/restore trust
- Lack of rationally coherent strategies and objectives.

Resistance to change:

- A function of the quality of the relationship between change agents and recipients
  - Change agents are active participants and contributors to the "resistance to change"
- Change agent has important contributions to make to the relationship:
  - Finding agreement
  - Establishing trust
  - Keeping it real - overly optimistic promises may appear deceptive or misleading.
  - Legitimizing change
  - Mobilizing action
- **Overcoming resistance:** agents effectively managing the agent-recipient relationship:
  - Establish a strategy and objectives, and mobilize action.
  - **Make the recipient's "resistance" and the agent sense-making public part of the discourse for change.**

Jeffrey D. Ford, Laurie W. Ford, and Angelo D'Amelio, 2008: Resistance to Change: The Rest of the Story. AMR, 33, 362–377, https://doi.org/10.5465/amr.2008.31193235

Introduction - Evolving Strategies

(1) Do not impose the use of a specific simulation environment (software tool), instead,

(2) Propose a new objective:
Use of an ***open-access*** **common human and computer-readable mathematical "description":** use of Modelica for unambiguous model exchange.

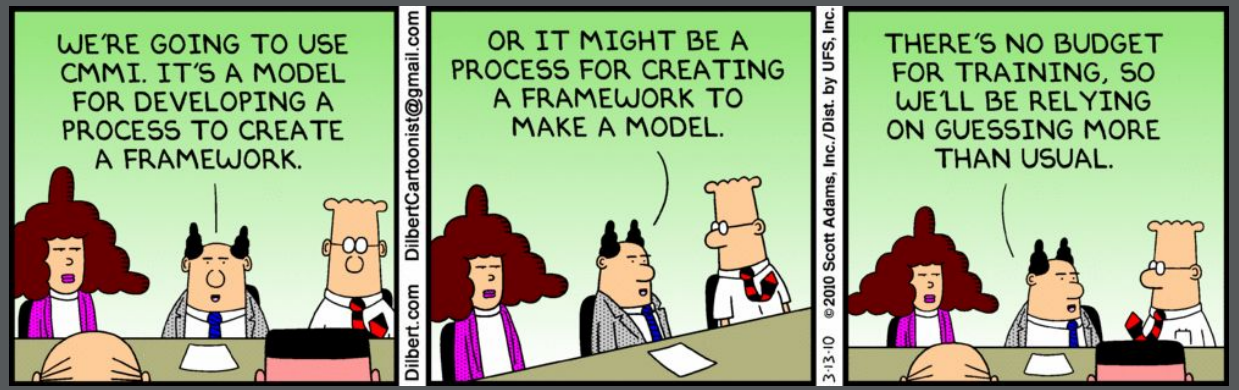
(3) Decrease of avoidance forces

- SW-to-SW validation gives quantitatively a similar answer than domain specific tools.
- Accuracy (w.r.t. to *de* facto tools) more important than performance
- Support technology adoption through adequate training (e.g. I developed / teach a course on Modelica at my university where I train our devs on the language itself).

(4) Use Multiple Resources for Legitimizing Change

- Give stakeholders the opportunity to participate in the effort via open-source development approach.
- Recognize development needs for technology adoption and gaps → develop necessary tools to easily transition to new technology, e.g. model transformation tooling; work with technology vendors in improving performance, user friendliness, etc.

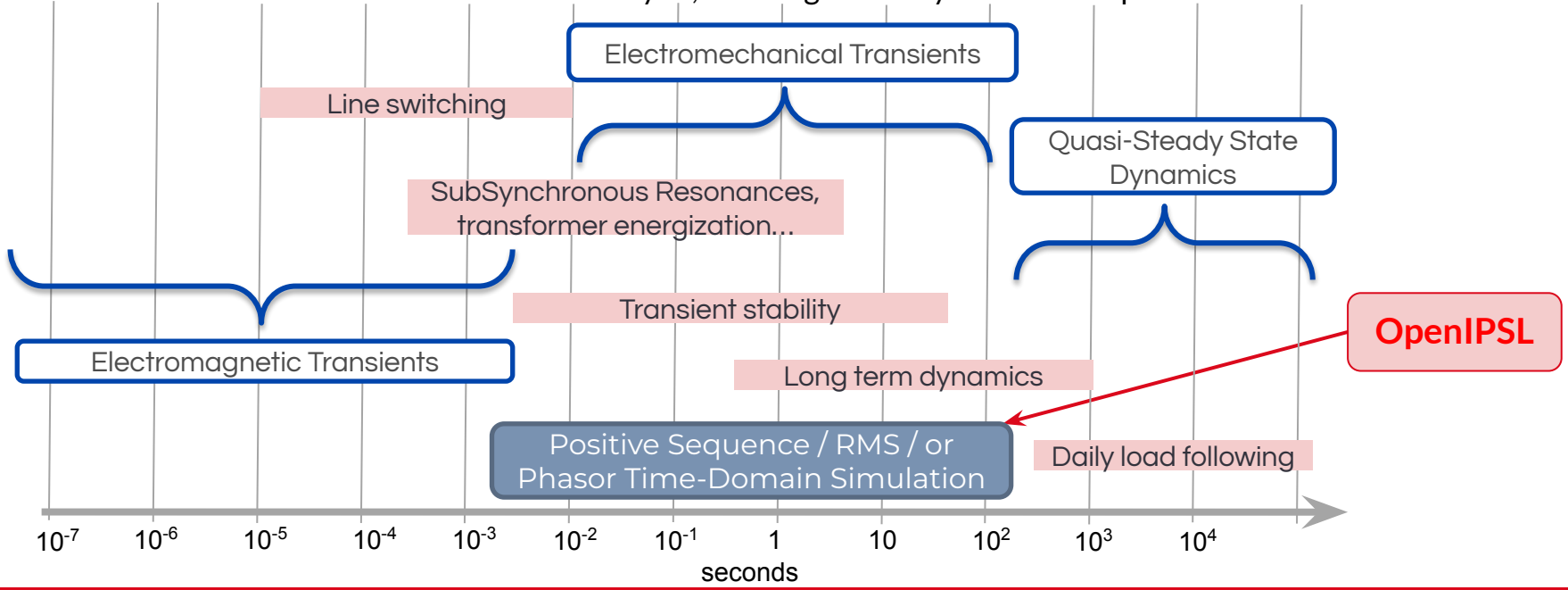
Model Development Process



Example Model Development: WECC Generic RE Models

Conceptual Background

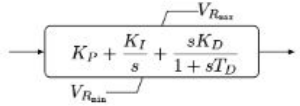
- The new RE models implemented in the OpenIPSL Library are based on the WECC Modeling and Validation Work Group guidelines [1, 2, 3]
- They include solar Photovoltaic (PV) plants, Wind plants, and Battery Energy Storage Systems (BESS).
- Are intended for electromechanical transient analysis, meaning that they are used for phasor simulation.



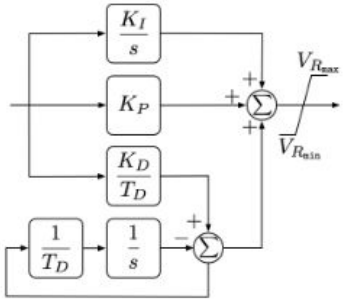
[1] Council, Western Electricity Coordinating. "WECC wind power plant dynamic modeling guide." Western Electricity Coordinating Council Modeling and Validation Work Group: Salt Lake City, ST, USA (2014).
[2] WECC Renewable Energy Modeling Task Force. "Solar Photovoltaic Power Plant Modeling and Validation Guideline. 2019."
[3] WECC Renewable Energy Modeling Task Force. "WECC battery storage dynamic modeling guideline." Western Electricity Coordinating Council Modeling and Validation Work Group, Nov (2016).

Implementation of Building Blocks

- Even simple blocks such as a limited output PID can have different implementations
  - When **only** block diagrams are used, the model can be ambiguous
  - Additional information is not always available or transparent
  - Different implementations can lead to different results

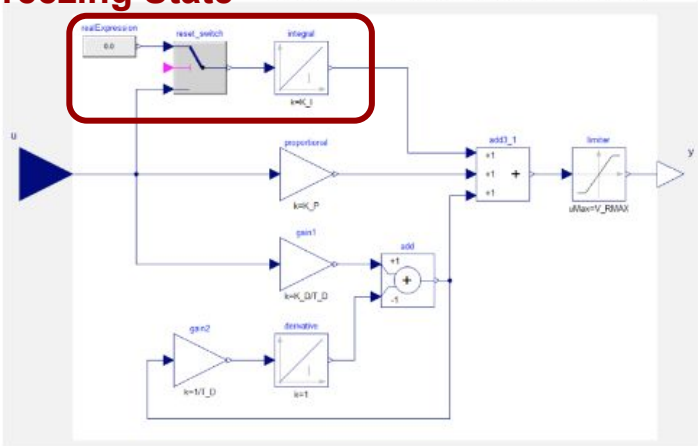


(a) Block diagram presenting an implementation for a limited output PID.

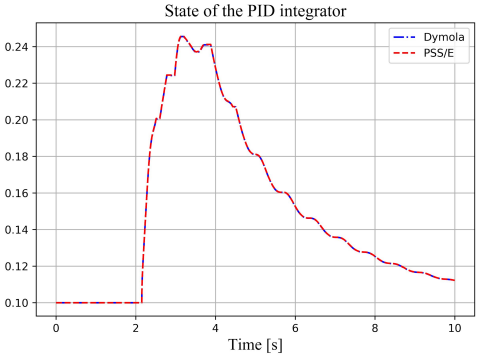


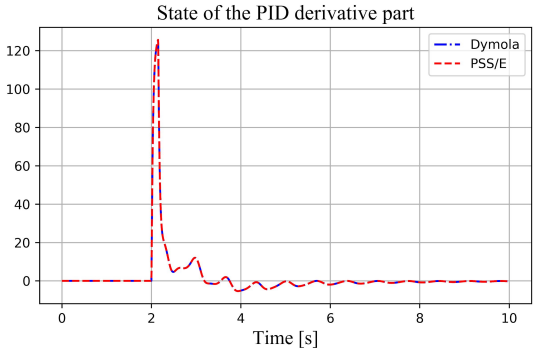
(b) Block diagram presenting an alternative and detailed implementation for the same limited output PID.

Freezing State



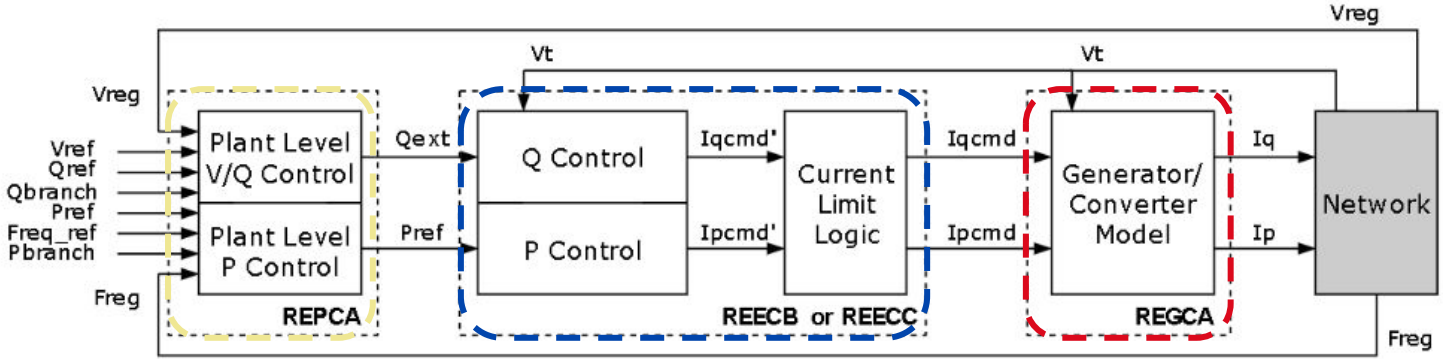
(c) Modelica implementation of the limited output PID block.





The WECC based renewable energy sources have three main components:

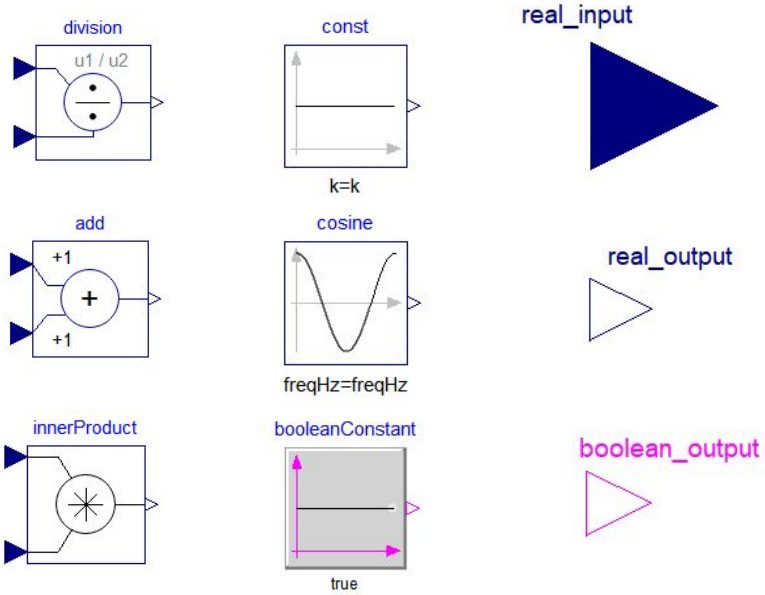
- **REGCA module:** represents the current injection model for renewable sources.
- **REEC module family:** represents the electrical controls of the electrical inverters. The electrical controller is connected to the REGCA module in a close loop connection, meaning that it acts on active and reactive power based on the output voltage and power from the REGCA module.
- **REPCA module:** represents the renewable plant controller. It establishes the controlling scheme for a conglomerate of renewable sources.



WECC Block Diagram of the RES modules

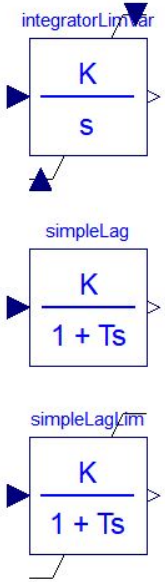
How do we build these components using the Modelica language and the OpenIPSL library?

Some components from the MSL



The MSL does include various integrator and transfer function blocks, however, due to how PSS/E harcodes the behavior those basic functions, they do not match what PSS/E does.

Some components from the OpenIPSL Library

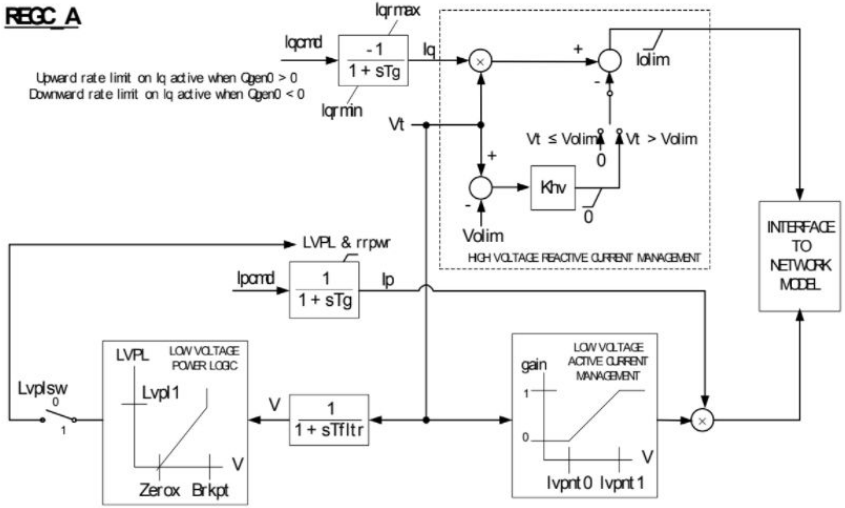


Electrical interface to couple different OpenIPSL components.

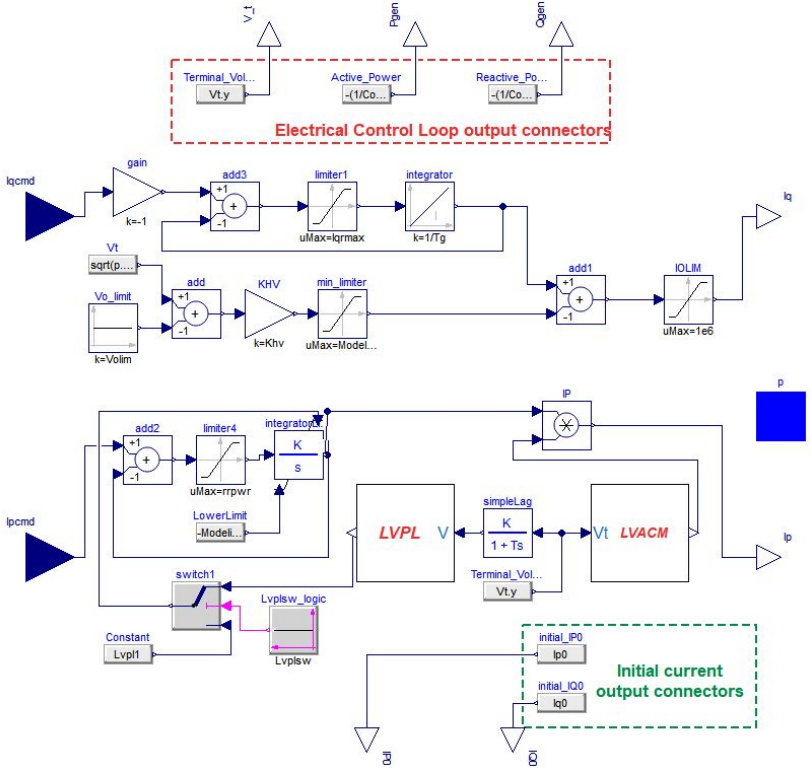
These are “specialized” blocks able to reproduce the integrators’ behavior in PSS/E.

WECC Generic Renewable Energy Models - PV, Wind, and BESS Model Configuration

Block Diagram and OpenIPSL implementation of the Generator/Converter

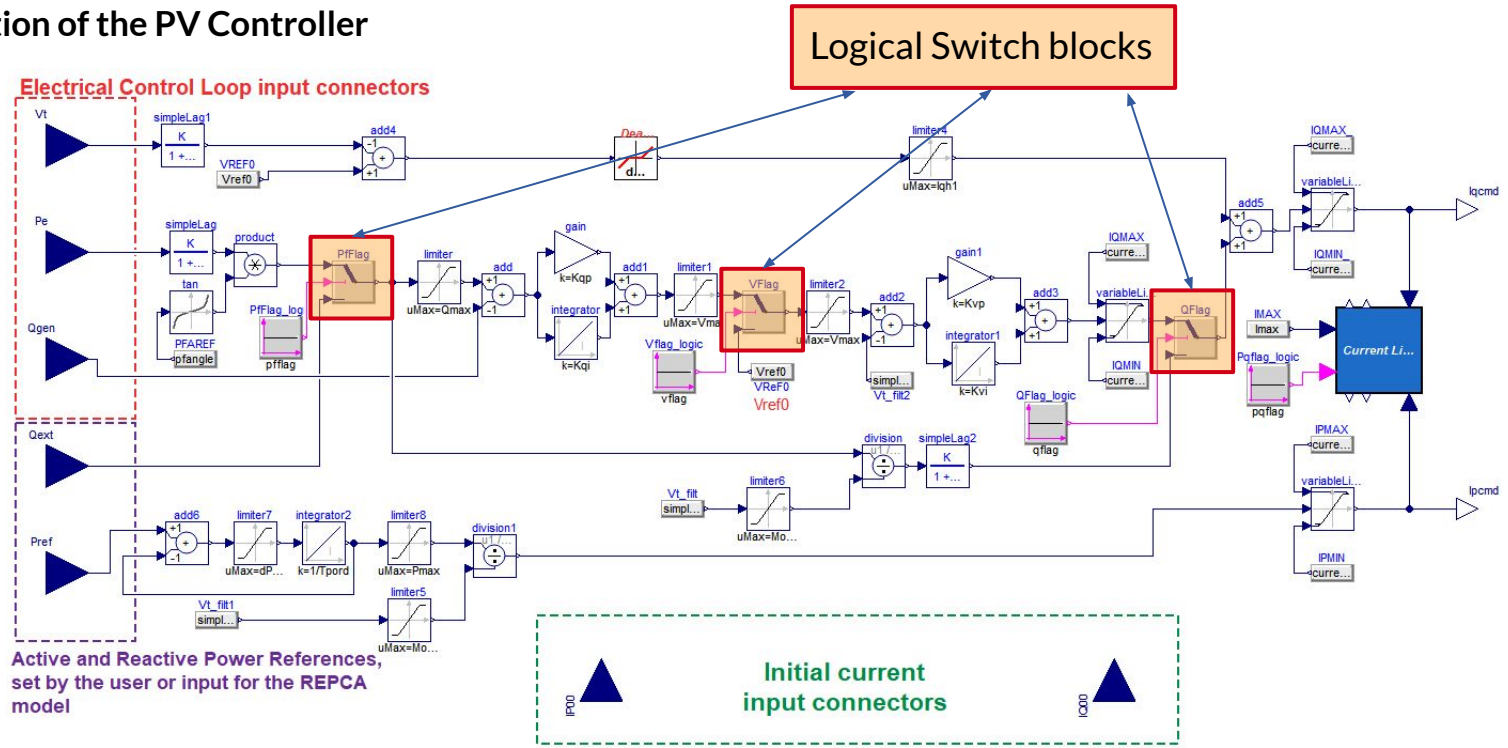


Block Diagram of the REGCA component



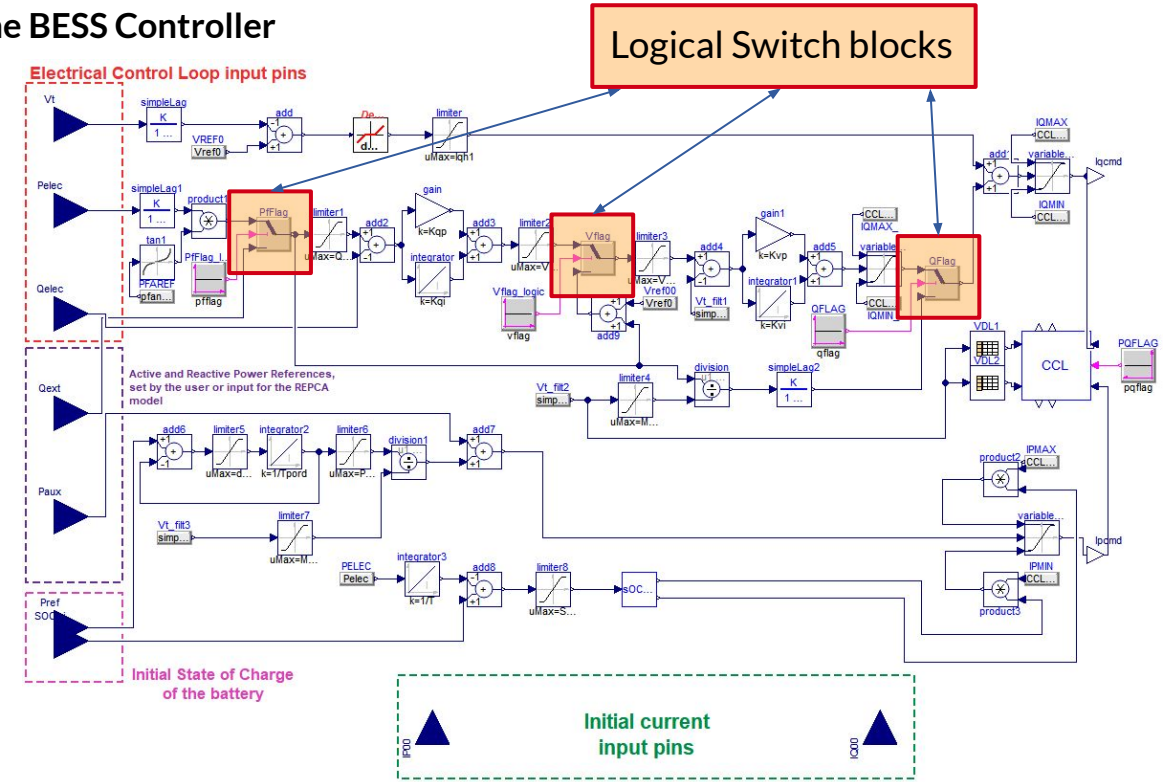
REGCA Module implemented in OpenIPSL

Implementation of the PV Controller



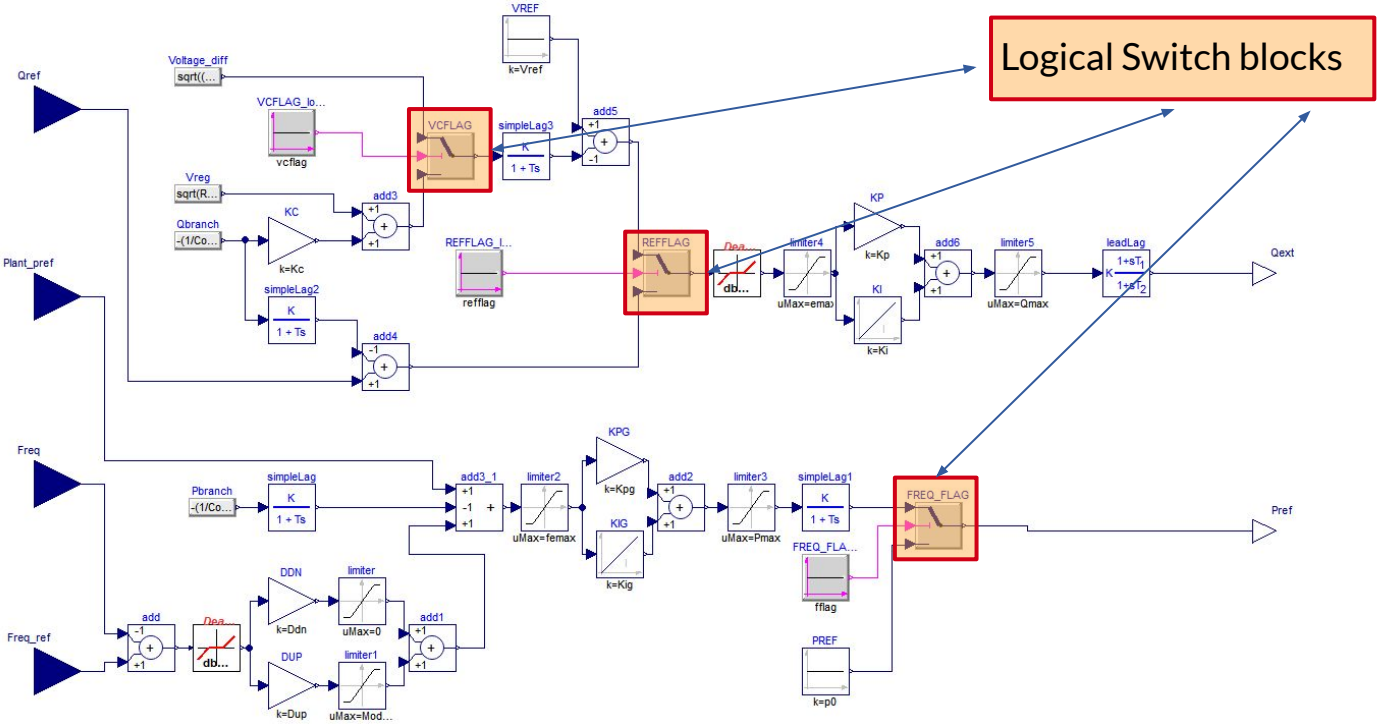
REECB Module using the OpenIPSL Library

Implementation of the BESS Controller



REECC Module using the OpenIPSL Library

Implementation of the Plant Controller

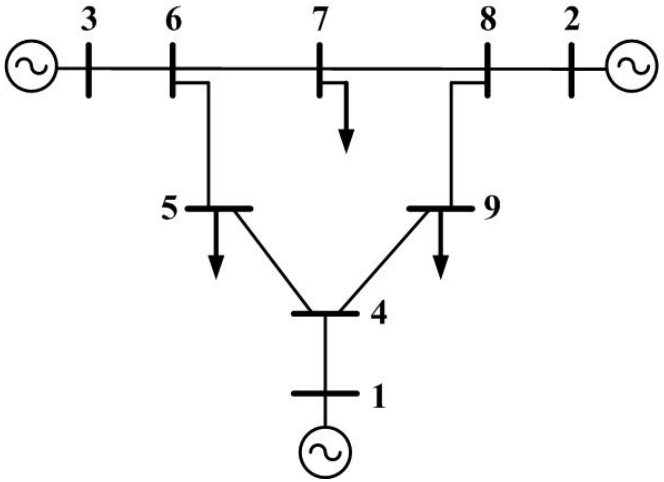


REPCA Module using the OpenIPSL Library

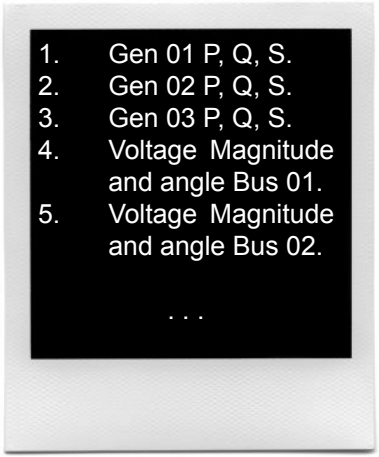
How do we provide initial conditions to the models?

- Providing a good initial guess for the current source model and its controllers is key in guaranteeing that the system, as a whole, will successfully initialize to a valid steady state.
- During initialization, Modelica tools attempt to find a solution to the entire set of Differential and Algebraic Equations (DAEs), at t=0, which will be used to start the simulation.
- In order to do so, the user must provide the following guess values: generator's base apparent power, active power, base power, reactive power, voltage magnitude and angle.

Instant picture of the power system through the lens of a Power Flow solution (static solver)

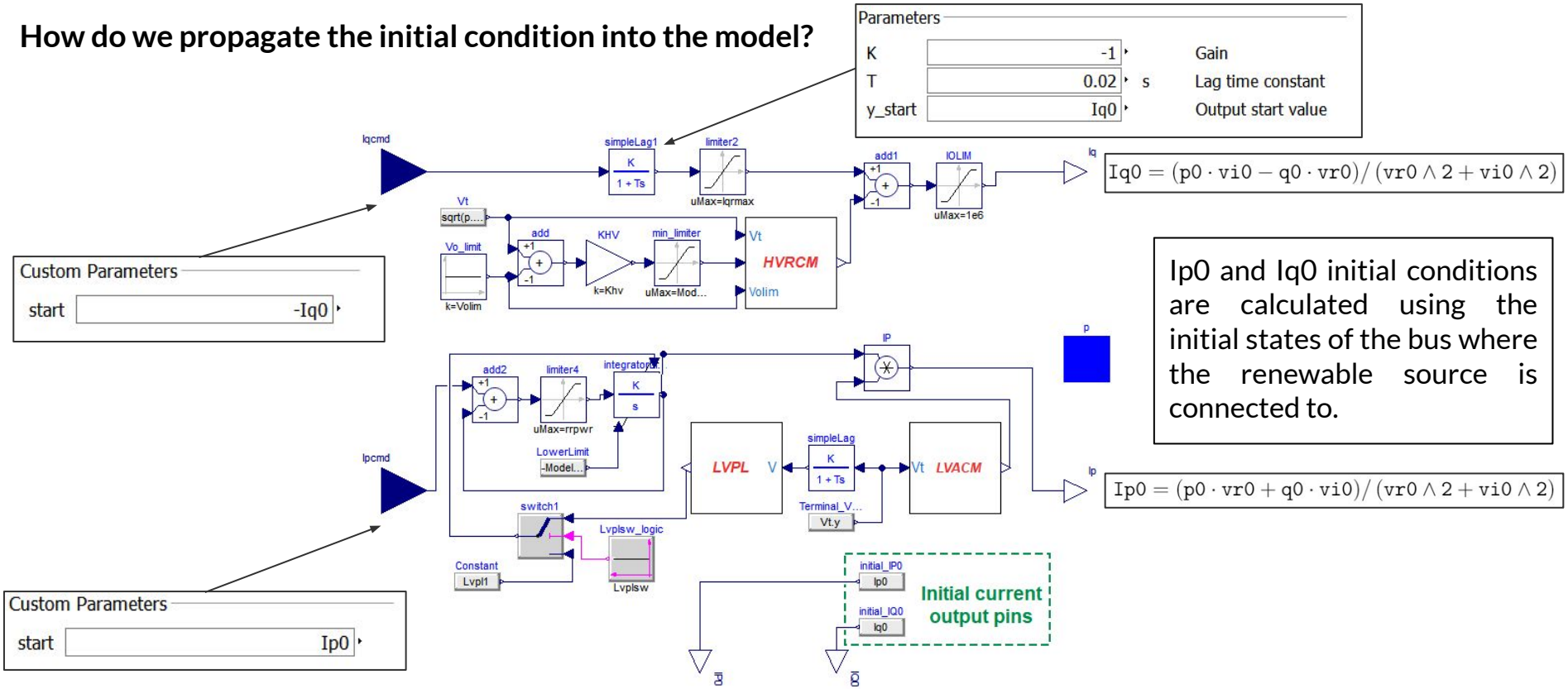


Model of the physical system



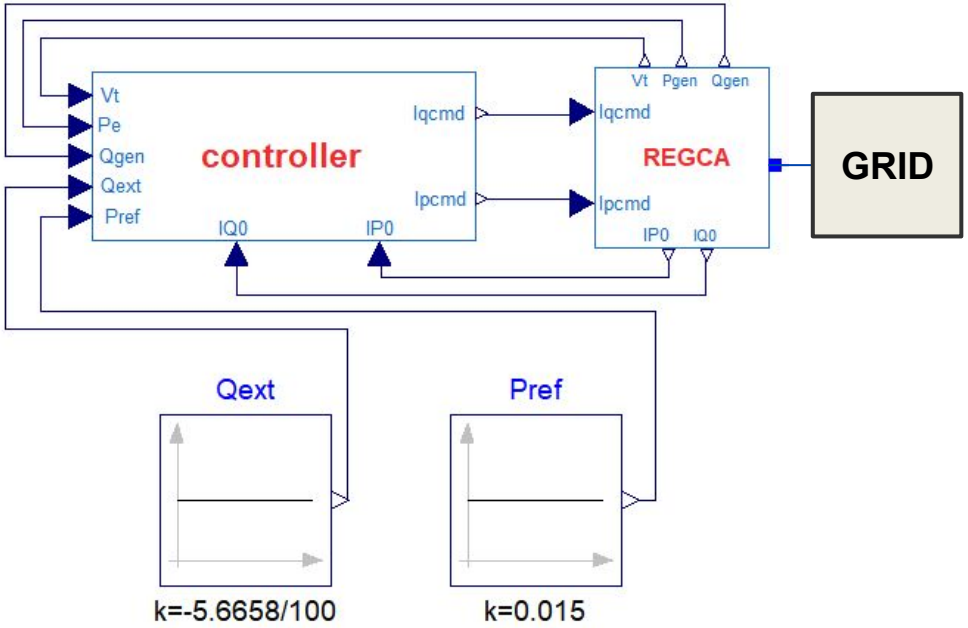
Initial Guess for the Algebraic States

How do we propagate the initial condition into the model?



Ip0 and Iq0 initial conditions are calculated using the initial states of the bus where the renewable source is connected to.

Example - PV source with electrical controller

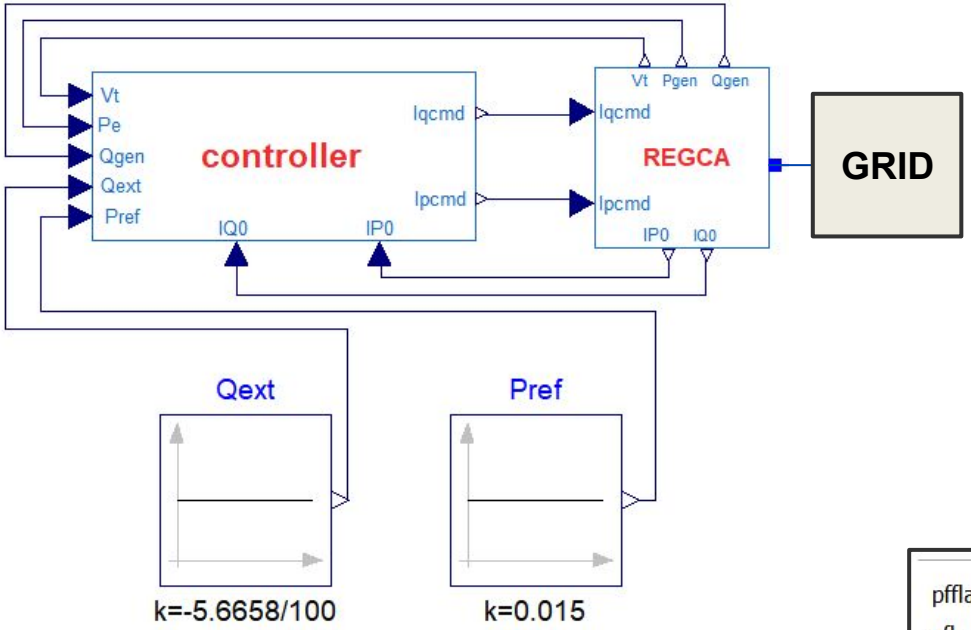


REGCA + REECB Modules

In the electrical control module, additional reactive control options are available:

- **Constant power factor (PF)**, based on the generator PF in the solved power flow case
- **Constant reactive power**, based either on the equivalent generator reactive power in the solved power flow case or from the plant controller

Example - PV source with electrical controller



REGCA + REECB Modules

Possible Switch Configurations for the PV model with the Electrical Controller (REECB)

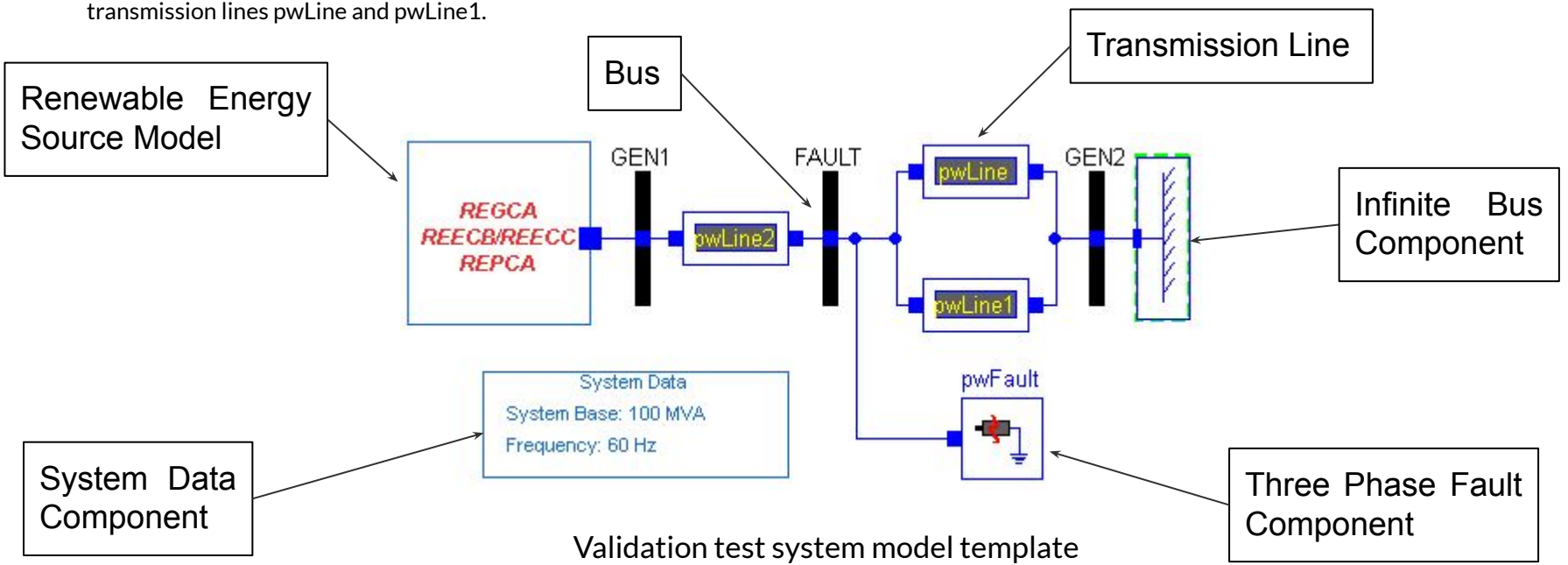
Mode of Operation	Blocks Necessary	PfFlag	Vflag	Qflag	PqFlag
Constant Local Power Factor Control	REGCA + REECB	True	N/A	False	N/A
Constant Reactive Power (Q) Control	REGCA + REECB	False	N/A	False	N/A
Local Voltage (V) Control	REGCA + REECB	False	False	True	N/A
Local Coordinated Q/V Control	REGCA + REECB	False	True	True	N/A

Control Tab in the REECB controller module

pfflag	false	Constant Q (False) or PF (True) local control
vflag	true	Local Q (False) or voltage control (True)
qflag	false	Bypass (False) or engage (True) inner voltage regulator loop
pqflag	false	Priority to reactive current (False) or active current (True)

Validation Process - Comparing the simulation results against Siemens PTI PSS@E

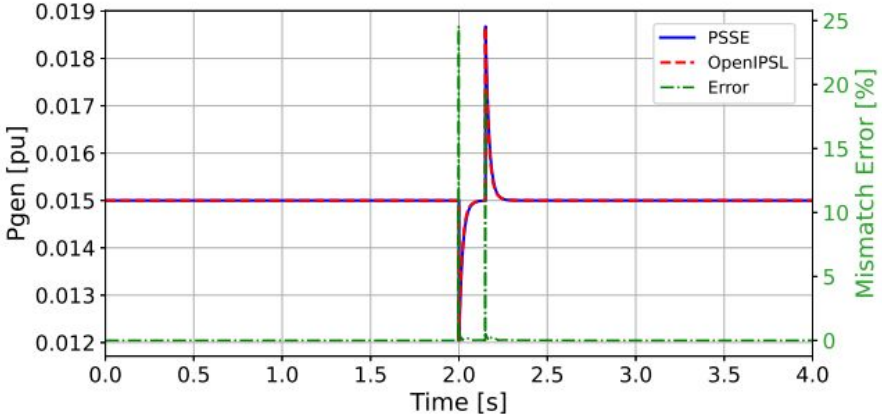
- Model validation was carried out by comparing simulation results obtained from running simulations in Dymola, a simulation tool for Modelica models, and comparing the output results against the solution from PSS®E.
- The `pwFault` component represents the three phase to ground fault in a test system model and is used to generate a transient in the system to verify the models' response.
- The voltage magnitude at the **FAULT** bus will decrease as a result of the current splitting its flow between the fault to ground and the transmission lines pwLine and pwLine1.



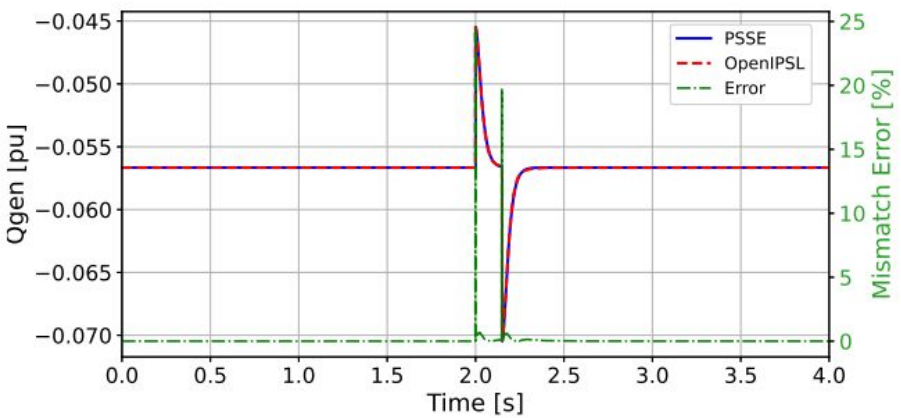
Validation test system model template

Validation Test 1 — Constant Local Reactive Power Control for PV model

- The control scheme in the first test utilizes the REGCA and REECB modules.
- The flag configuration for this particular test is:
  pfflag = false, vflag = true or false, qflag = false, and pqflag = true or false.



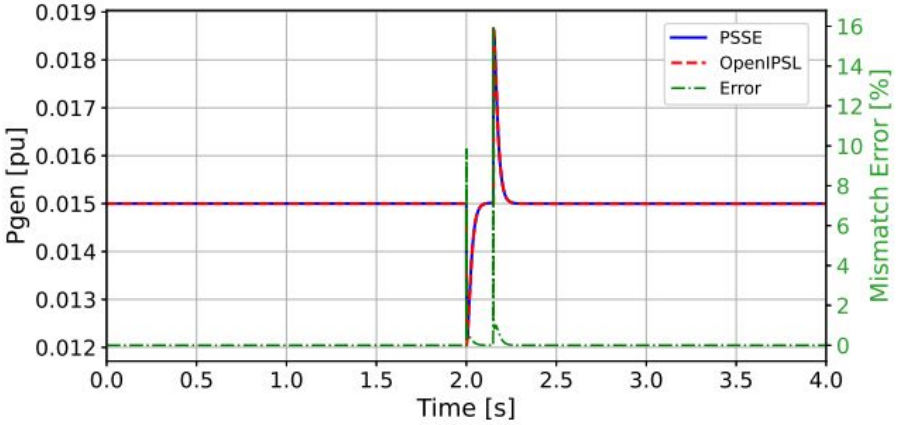
(a) $Pgen$ RMSE $= 6.75 \times 10^{-5}$.



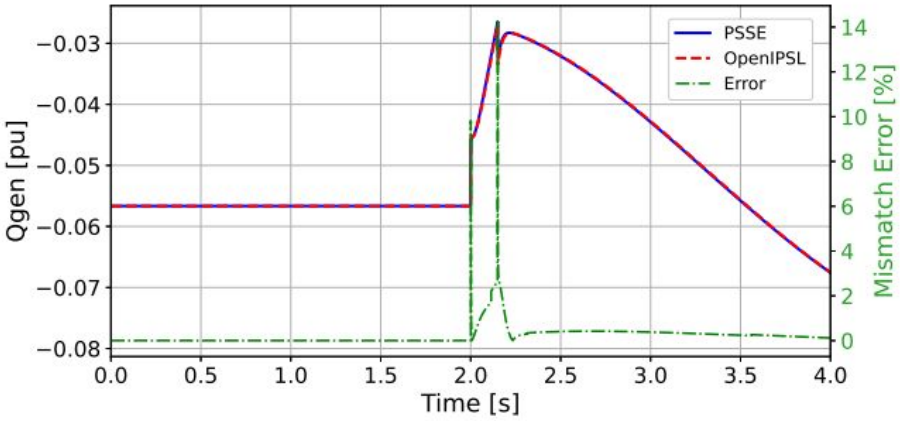
(b) $Qgen$ RMSE $= 2.56 \times 10^{-4}$.

Validation Test 2 — Plant Level Reactive Power Control + Local Coordinated Voltage/Reactive Power Control for BESS model:

- The control scheme in the second test utilizes REGCA, REECC, and the REPCA modules.
- The flag configuration in the electrical controller for this particular test is: pfflag = false, vflag = true, qflag = true, and pqflag = true or false.
- The flag configuration in the plant controller is: vcflag = true, refflag = false, and fflag = false.



(a) $Pgen$ RMSE $= 3.89 \times 10^{-5}$.

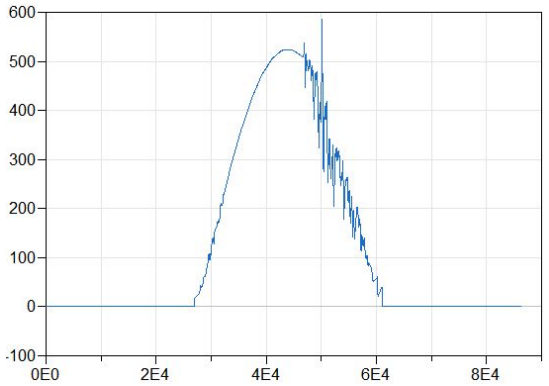


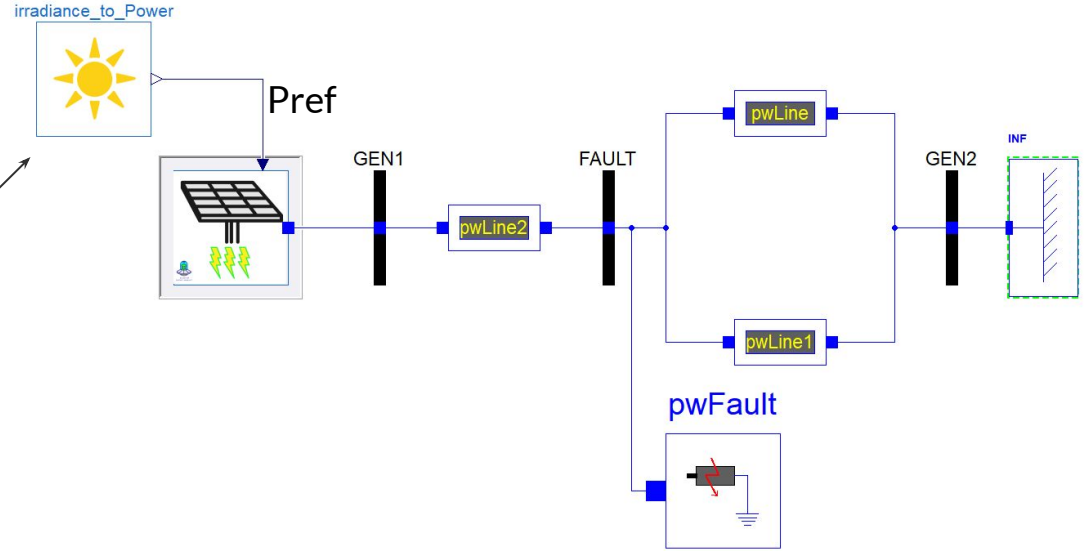
(b) $Qgen$ RMSE $= 1.49 \times 10^{-4}$.

Additional capabilities within Modelica tools - PV model with irradiance input

- Average irradiance over a large PV plant can change appreciably during the span of a typical dynamic simulation (up to 30 seconds).
- By default, the WECC generic models assume a fixed reference generator output in the solved power flow case. The original models in PSSE are limited to a fixed active power reference.
- With the models implemented in Modelica, it is now possible to vary the irradiance input!

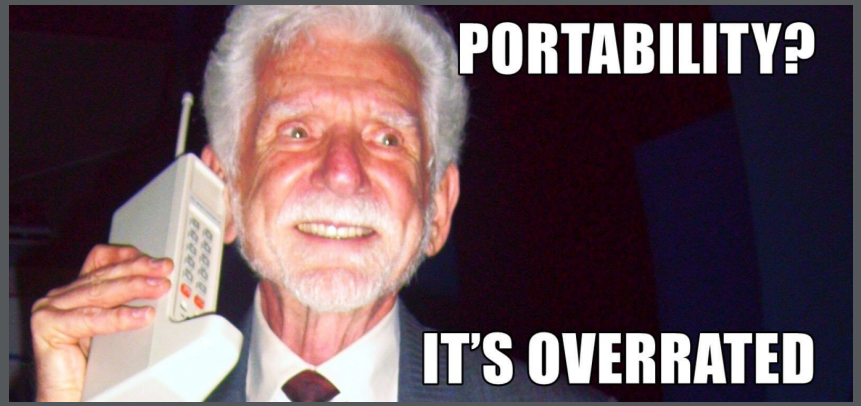
Irradiance Table for a span of 24 hours

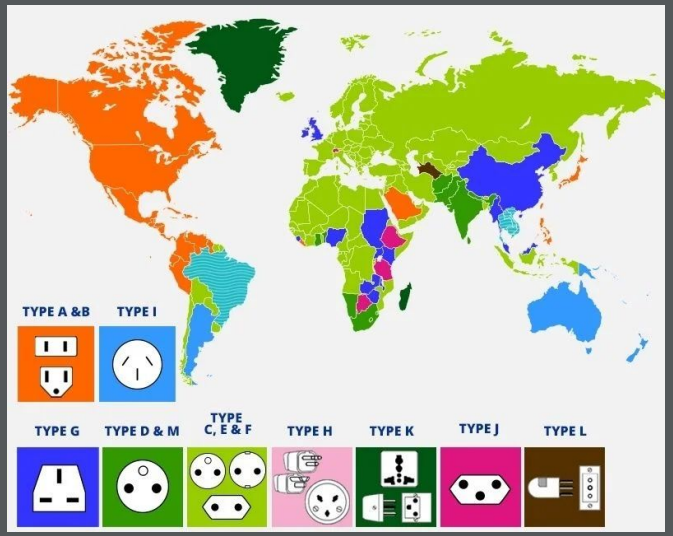




F. Fachini, **L. Vanfretti**, M. de Castro, and T. Bogodorova, and G. Laera "Modeling and Validation of Renewable Energy Sources in the OpenIPSL Modelica Library," 47th Annual Conference of the IEEE Industrial Electronics Society (IECON), October 13-16, 2021.

Portability and Interoperability





Modeling for Portability & Interoperability

Portable and Interoperable Models with Open Standards

- Assess the notable features that explain why Modelica is suitable for portability:
  - Object-oriented equation-based
    - Ideal for complete model description rather than just parameter tables
  - Use different Modelica-compliant tools:
    - Dymola, OMEdit, SystemModeler, Impact
  - Compliant with Functional Mock-up Interface standard
    - Models can be exported to different tools!
- Make models publicly available!

Methods

- Assess the OpenIPSL in different modelica compliant tools and computing environments
- Assess the simulation of Modelica models exported as FMUs in different tools
- Create and release repositories with models

Models in OpenIPSL must work with maximum number of Modelica - compliant tools for interoperability

OpenIPSL models that are exported with FMUs must work in different simulation tools and environments

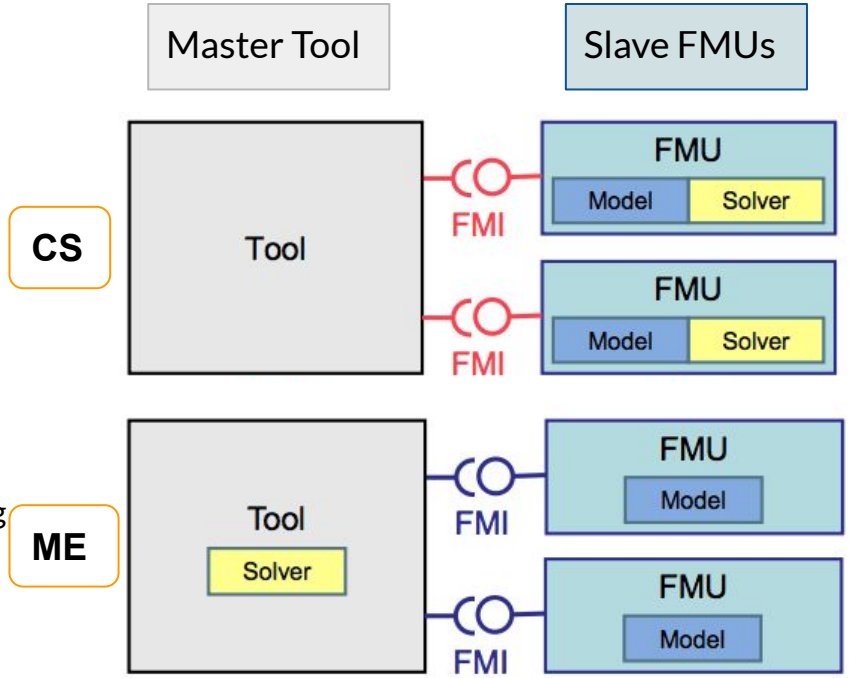
Modeling for Portability & Interoperability

fmi Functional Mock-Up Interface

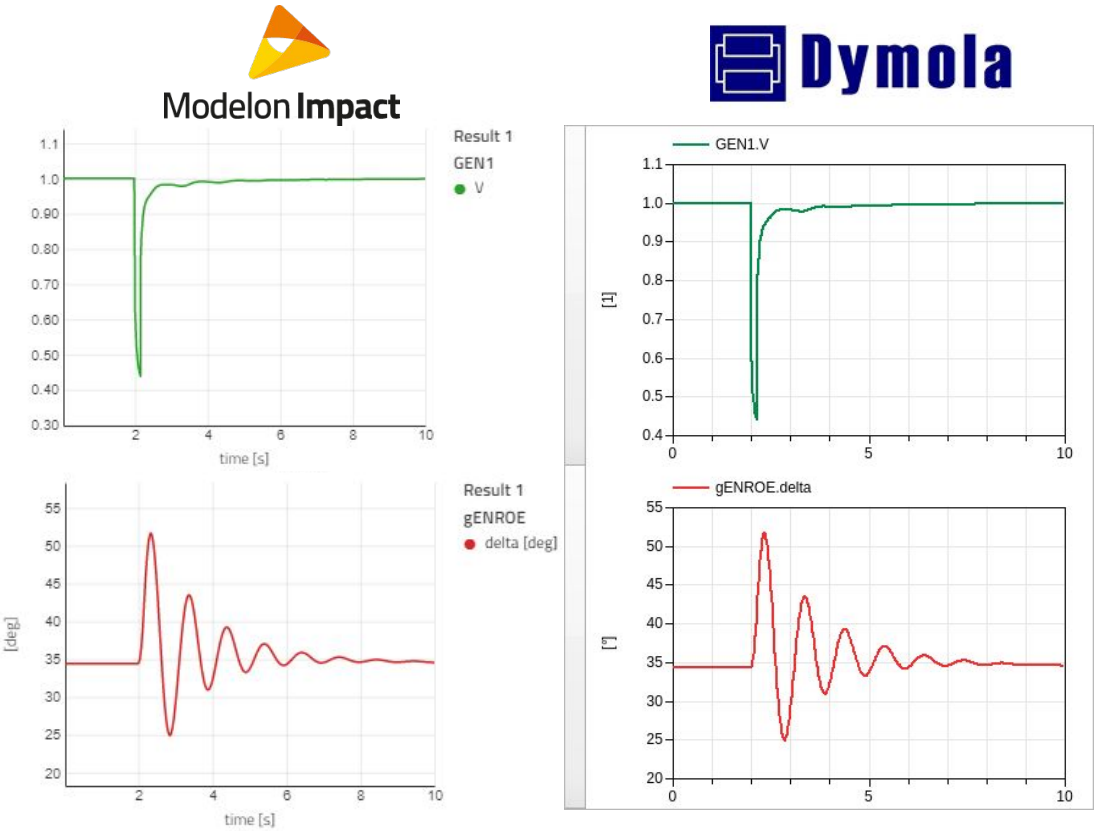
- FMI is an open access standard, also from the Modelica Association.
- It defines a container and an interface to ***exchange dynamic models*** using a combination of XML files, binaries and/or (source) C code zipped into a *single file, called a Functional Mock-up Unit (FMU) or .fmu.*

- **Supported by simulation 100+ tools!**

- FMI supports model export in two modes Co-Simulation (CS) and Model Exchange (ME)
  - With a Model Exchange FMU, the numerical solver is supplied by the importing tool. The solver in the importing tool will determine what time steps to use, and how to compute the states at the next time step.
  - With a Co-Simulation FMU, the numerical solver is embedded and supplied by the exporting tool. The importing tool sets the inputs, tells the FMU to step forward a given time, and then reads the outputs



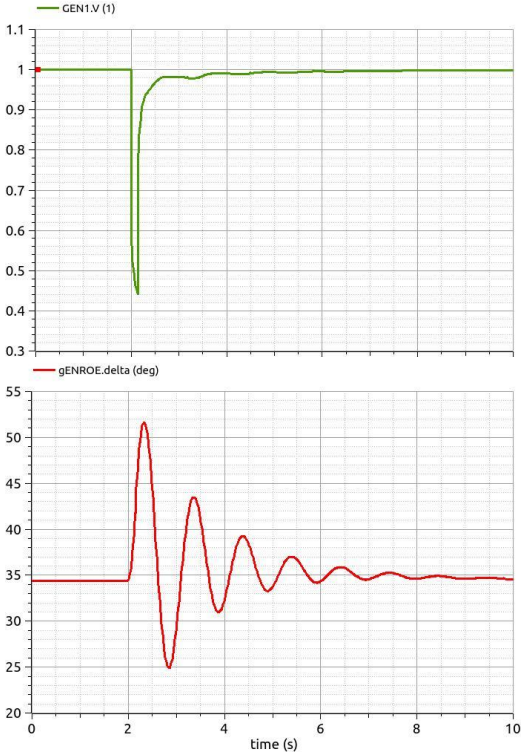
Interoperability Between Modelica Tools (1/2)



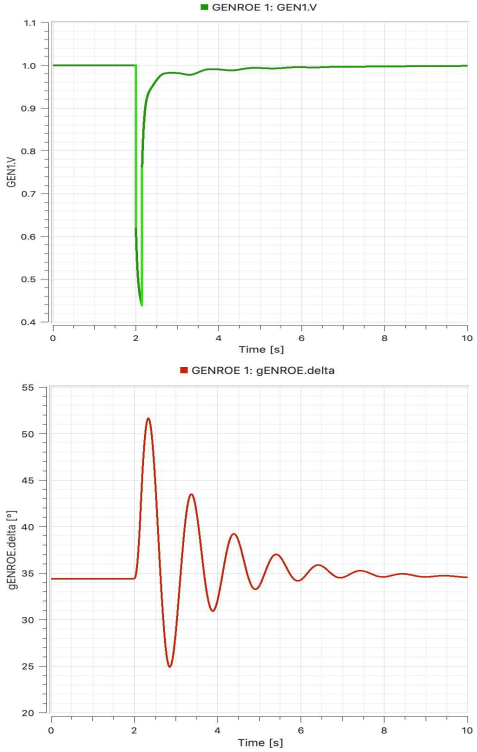
- OpenIPSL Library must maximize its compatibility with different Modelica-compliant software tools
- Library must be independent from tool
- Allows user to select their preferred tool to perform a study

Interoperability Between Modelica Tools (2/2)

OMEdit (OpenModelica)



Wolfram SystemModeler

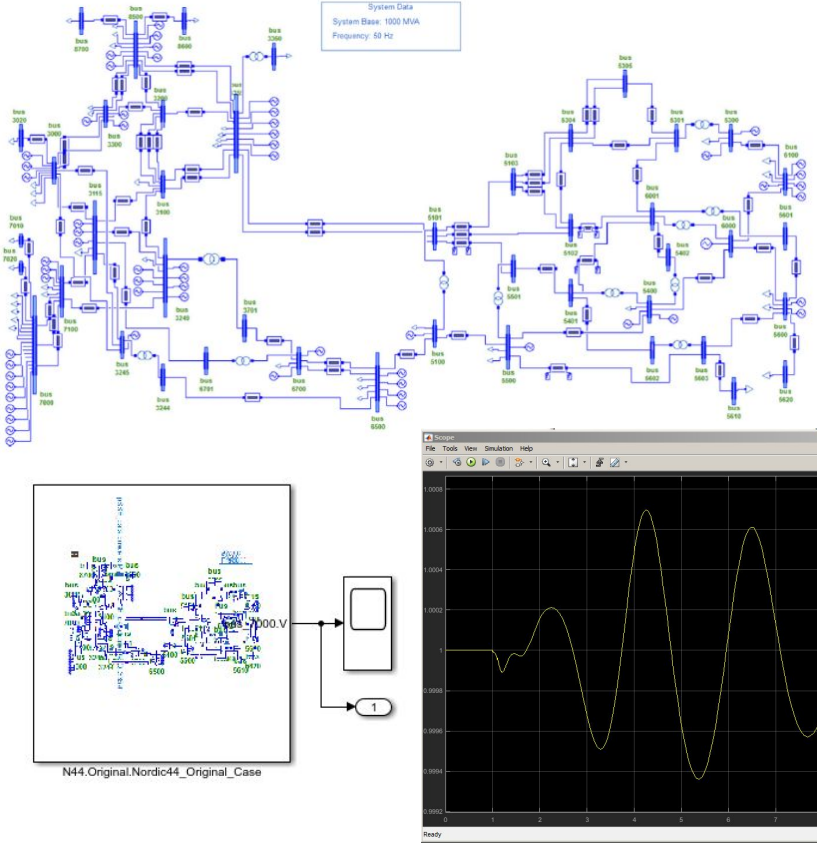


- OpenIPSL Library must maximize its compatibility with different Modelica-compliant software tools
- Library must be independent from tool
- Allows user to select their preferred tool to perform a study

OpenIPSL and FMUs for Interoperable Simulation

Recomposing models for FMI-Export

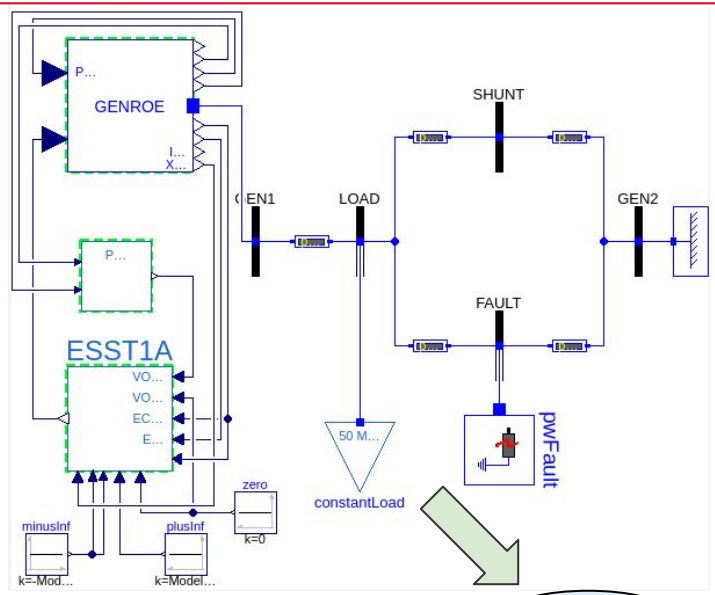
- Testings have been conducted with FMI export for different software tools:
  - Simulink
  - Python Libraries (FMPy PyFMU)
- FMUs for individual components
  - Dynamic elements vs. Algebraic elements
- FMUs for entire system
  - Size is challenge



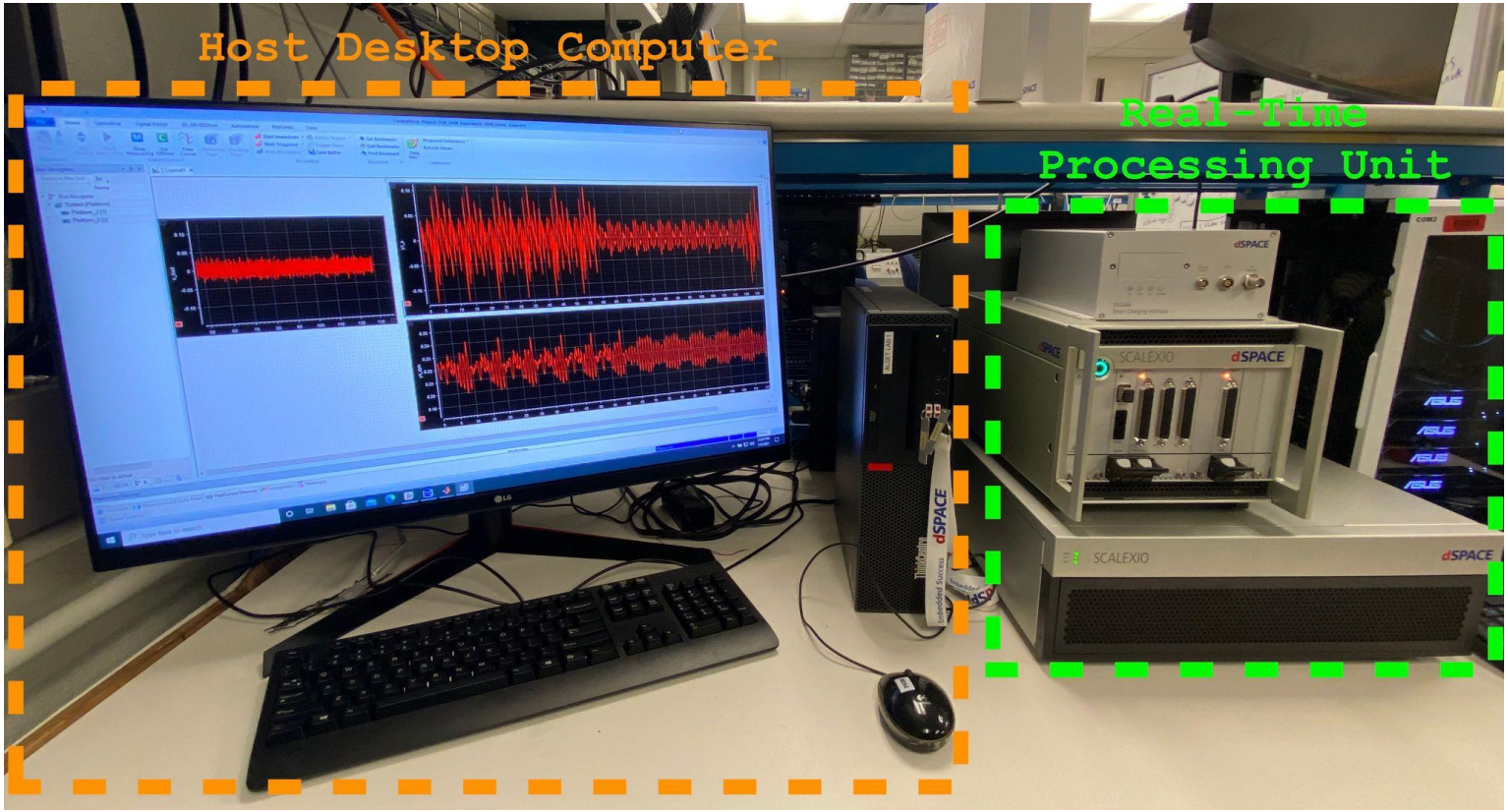
OpenIPSL and FMUs for Real Time Simulation

FMI-Export for Real Time Simulation

- Entire systems were used rather than different FMUs for each basic component
- Co-simulation FMU, with solver tolerance and simulation time already pre-defined
- Exported with source code via Dymola
- Loaded into dSPACE SCALEXIO:
  - Natively loads FMUs
  - No extra step is required



Real Time Simulation Setup

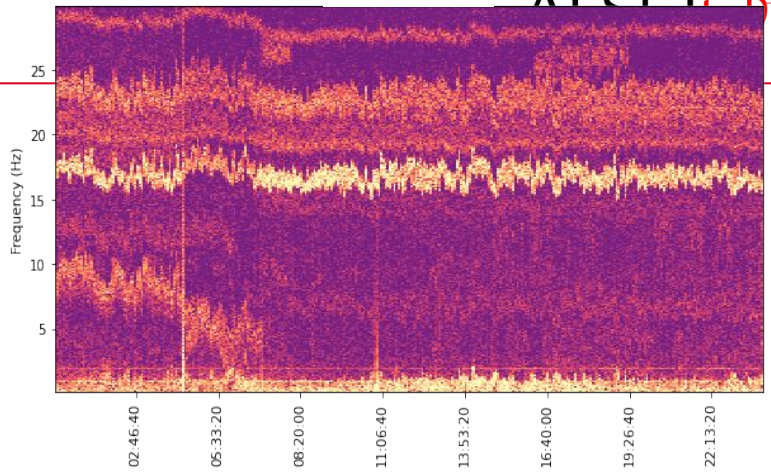


Real Time Simulation Result

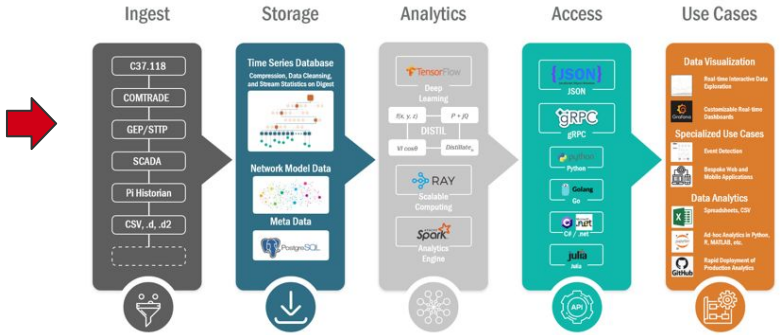
Periodic Task 1/Overrun Count	Periodic Task 1/Task Call Counter	Periodic Task 1/Task Turnaround Time
13	30013	8.95870489417323E-05

Adaptive for New Use Cases:
Dominion Energy's Needs for Model Calibration

- **Dominion's models for planning are used for operations analysis, forensics and control settings.**
- **Modeling challenges**
  - Conventional model validation require events happening but system mostly in ambient conditions.
  - Operation conditions change throughout the day due to changing nature of load, line switching, V setpoint change, etc.
  - **Existing model needs to be updated due to unmodeled dynamics.**
  - **Difficult to do when models and data are segregated.**
- ***Vision:* Cloud-based Data-assisted modeling with Modelica-based technologies**
  - Quickly accessible synchrophasor data using *PredictiveGrid*™
  - Portable model modules for various generator stations with enhanced functionalities to match to data (linearization).
  - Quickly do model validation and calibration "on-demand" to support planning and operation tasks.



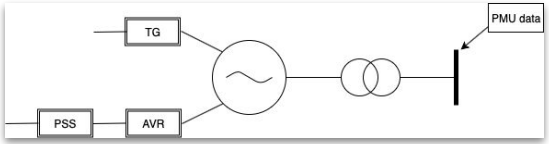
Voltage Magnitude Spectrogram at Unmodeled Generating Unit



Predictive Grid Synchrophasor Platform

Cloud-Based *Proof-of-Concept* Prototype

- ***Challenge:*** Typical generator plant models are isolated in simulation tool (PSS/E):
  - Limited to in-built capabilities of the tool
  - Not possible to deploy existing PSS/E model in PredictiveGrid platform.



- ***Solution:*** use Modelica and FMI to create a portable model! *However, the models needed were not available in OpenIPSL.*
- ***Approach:***
  - Implement the model in Modelica and verify against PSS/E.
  - If results are the same, export Modelica model as an FMU
  - Deploy model in platform and build toolchain for model calibration:
    - Use Python functionalities to deploy the model in platform.
    - Use Python and Jupyter notebooks to build calibration "notebook" in cloud platform.

SW-to-SW verification of the plant model (PSS@E vs. Modelica)

↓

Export Modelica model as FMU ***with source code***

fmi Functional Mock-Up Interface

↓

Predictive Grid Integration:

- Query measurements data
- Implement signal processing of PMU data
- Couple the model (FMU) with PMU data
- Integrate tools for model calibration, i.e. optimization-based parameter estimation.

↓

Manually Update PSS/E Model Data (Could also be automated)

Import a specific user defined library for connection to the platform and retrieve data

```
from Chetan_lib02 import *
conn = btrdb.connect("internal.api.dominion.predictivegrid
```

Import standard Python modules for mathematical calculations, data processing and ModestPy tool after its installation

```
import time
import os
import pandas as pd
import numpy as np
from modestpy import Estimation
from modestpy.utilities.sysarch import get_sys_arch
from modestpy.fmi.model import Model
import matplotlib.pyplot as plt
import seaborn as sns
```

Instantiation of the FMU

```
# Instantiate FMU
fmu_file = 'WC_ST01.fmu'
model = Model(fmu_file)
```

Defining inputs/outputs after signal processing

```
# Inputs
inp = pd.DataFrame()
t = tnew
inp['time'] = t
inp['Vreal'] = Vre
inp['Vim'] = Vim
inp = inp.set_index('time')

# Load measurements (ideal results)
ideal = pd.DataFrame()
ideal['time'] = t
ideal['Pout'] = P
ideal['Qout'] = Q
ideal = ideal.set_index('time')
```

Defining parameters to be estimated

```
# Load definition of estimated parameters (name, initial value, bounds)
est = {'eSST1A1.K_A':(1.26,1.,1.5),
       'eSST1A1.T_A':(1.4e-06,1.0e-6,0.0001),
       'P0.k':(183600000.,183000000.,184000000),
       'Q0.k':(-28800000.,-30000000.,-28000000)}
```

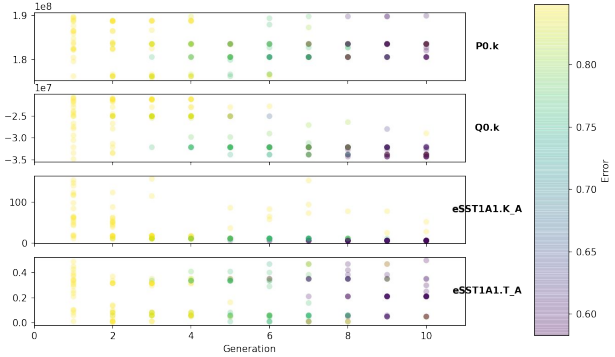
Defining estimation algorithms and settings

```
# Session
session = Estimation(workdir, fmu_file, inp, known, est, ideal,
                     lp_n=1, lp_len=None, lp_frame=None,
                     vp=None,
                     methods=('GA','SCIPY'),
                     ga_opts={'maxiter': 10, 'tol': 1e-6, 'lhs': True},
                     ps_opts={'maxiter': 100, 'tol': 1e-5},
                     scipy_opts={'solver': 'Nelder-Mead',
                                 'options': {'eps': 1e-6}},
                     ftype='RMSE', seed=1,
                     default_log=True, logfile='WC.log')
```

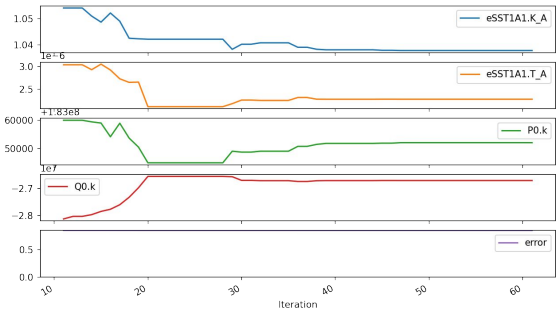
Parameter Estimation Under Ambient Conditions

- After a linear analysis of the plant, it has been noticed that the exciter could contribute to the anomalous behavior.
- Therefore, an estimation of the voltage regulator gain **Ka** and time constant **Ta** and the steady state active (**P0**) and reactive power (**Q0**), has been performed for ambient conditions.

GA Algorithm

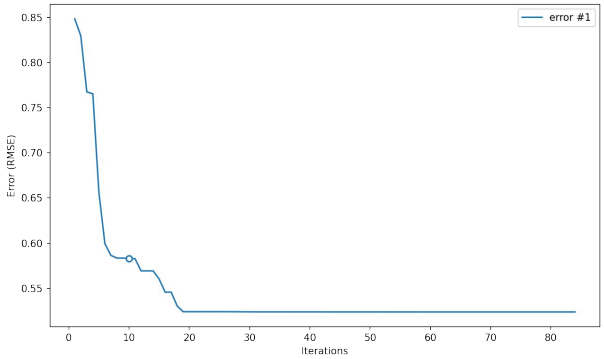


Nelder-Mead



estimation elapsed time ≈ 1431s

Error



GA Nelder-Mead

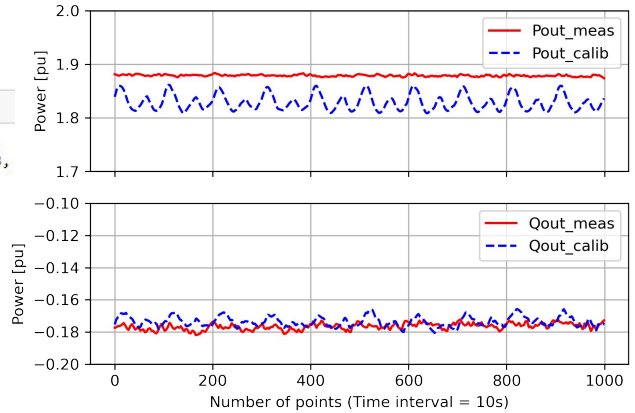
Sequence of algorithms used for the estimation

Results: Parameter Estimation Results for 4 parameters

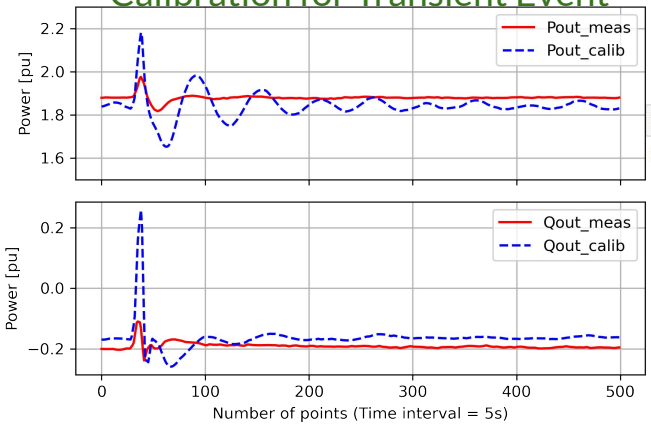
- From the results, the exciter gain **Ka** (uncalibrated value 160) keeps a value of the same order of magnitude in both scenarios whereas the time constant **Ta** (uncalibrated value 0.029s) has a difference of several orders of magnitude.
- Current parameters being used do not represent dynamics accurately (damped response (measurements) vs. undamped response of model):
  - More parameters for different parts of the model need to be included (e.g. turbine, PSS, etc).
  - Component models may need to be revisited (e.g. many parameters not used, modelers don't know why).
- More scenarios and different combinations of parameters will be tested since the preliminary results could also be affected by correlation between parameters:
  - Uncertainty quantification and sensitivity analysis methods need to be available in the platform.

Calibration for Ambient Conditions

```
estimates
{'eSST1A1.K_A': 2.6426506177827225,
 'eSST1A1.T_A': 0.24553453016003393,
 'P0.k': 182750836.3976619,
 'Q0.k': -34307843.08675239}
```



Calibration for Transient Event




```
estimates
{'eSST1A1.K_A': 1.0379577400856557,
 'eSST1A1.T_A': 2.2805494426998525e-06,
 'P0.k': 183052047.85619536,
 'Q0.k': -26700844.37074689}
```

Model Verification and Model Transformation

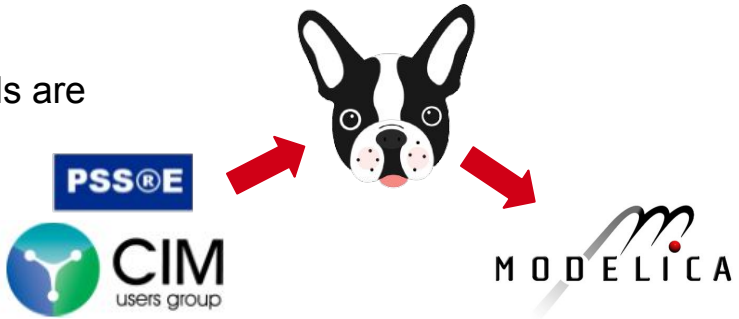
Implementation in Modelica and Model Verification

Main Challenges

- For a translation to work, it is necessary to be sure that models are consistent between tools
  - Mapping needs to be precise
  - Models need to be
    - Implemented in Modelica
    - Automatically tested
    - Verified against PSSE
  - Testing routine should be automatic:
    - Continuous Integration Framework

Overcoming Barriers

- Basic components (Machines, Exciters ...) are tested under different conditions in tiny models assembled using Modelica and OpenIPSL
- Regression testing to be implemented
- Continuous integration for library consistency

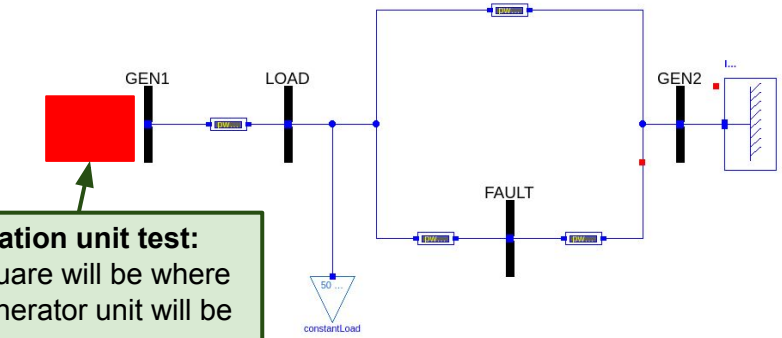


OpenIPSL is the means to represent PSSE and CIM models in Modelica

OpenIPSL must have the component being mapped and its behavior must be consistent with source

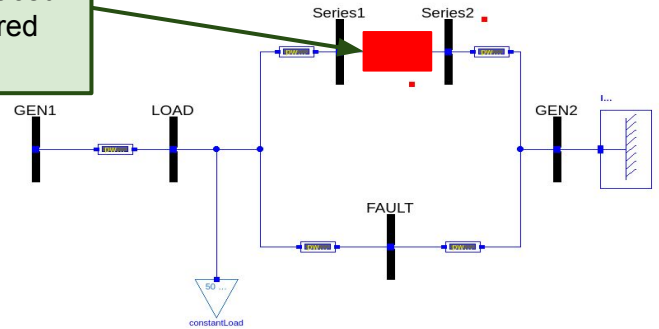
Templates: System Models for Verification of Basic Components

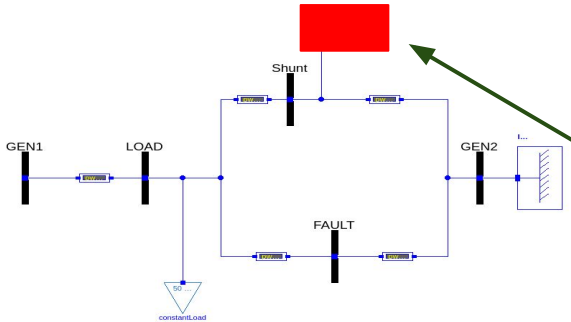
- A set of different small-scale systems was created for the testing and verification of basic components
  - Machines, Exciters, Stabilizers, Turbine Governors, etc.
  - Three different types of systems
  - Three different tests can be performed:
    - Fault (F)
    - Load Variation (LV)
    - Step in Reference value (SR)



Generation unit test: red square will be where the generator unit will be placed.

Series component test: Series component placed between two buses (red square).

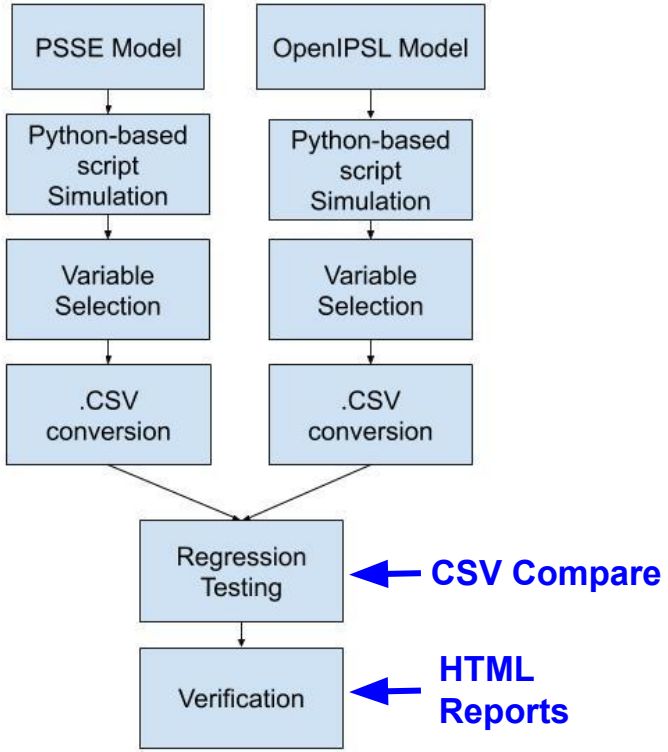




Shunt component test: Shunt component connected to a shunt bus (red square).

Regression Testing

- Manual verification of models is tiresome and can be subjective
  - How close the two curves should be so they are considered to be capturing the same behavior?
- Automatic procedures is needed for consistency:
  - Base results from PSSE are generated from Python scripts and saved into CSV
    - For each basic component
    - For each disturbance scenario
  - Results from OpenIPSL are obtained automatically via Python script and converted into CSV
  - CSV Compare tool is used
    - 1% Tolerance Tube



CSV Compare Report Example (1/2)

Meta report - CSV file comparison

Timestamp:	12/16/2019 3:25:40 PM [UTC]
Mode:	CsvTreeCompare
Base directory:	/home/marcelo/Dev/Gitted_Reps/NYPAModelTransformation/OpenIPSLVerification /VerficationRoutines/Dymola/Results/Generators/
Compare directory:	/home/marcelo/Dev/Gitted_Reps/NYPAModelTransformation/OpenIPSLVerification /VerficationRoutines/PSSE/Results/Generators/
Verbosity:	4
Tolerance:	1e-2
Compared files:	The compare directory contained 4 files. 4 files were tested. 1 file failed. Success rate is 75.0%.
Results	
	FAILED - At least one result failed its check with the base file. UNTESTED - No base file has been found or an exception occurred. SUCCEEDED - All results have been checked and are valid.
SUCCEEDED	GENROE.csv
SUCCEEDED	GENSAL.csv
SUCCEEDED	GENROU.csv
FAILED Ø0.24	GENSAE.csv

/home/marcelo/Dev/Gitted_Reps/Result/Generators/GENSAE_report.html

Meta report:	/home/marcelo/Dev/Gitted_Reps/Result/Generators/
Base file:	/home/marcelo/Dev/Gitted_Reps/NYPAModelTransformation/OpenIPSLVerification /VerficationRoutines/Dymola/Results/Generators/GENSAE.csv
Compare file:	/home/marcelo/Dev/Gitted_Reps/NYPAModelTransformation/OpenIPSLVerification /VerficationRoutines/PSSE/Results/Generators/GENSAE.csv
Tolerance:	0.01
Timestamp:	12/16/2019 3:25:40 PM [UTC]
Compared results:	The compare file contained 7 results. 7 results were tested. 1 result failed. Success rate is 85.7%.
Average relative error:	0.24
Failed tests:	GENSAE.gENSAE.delta

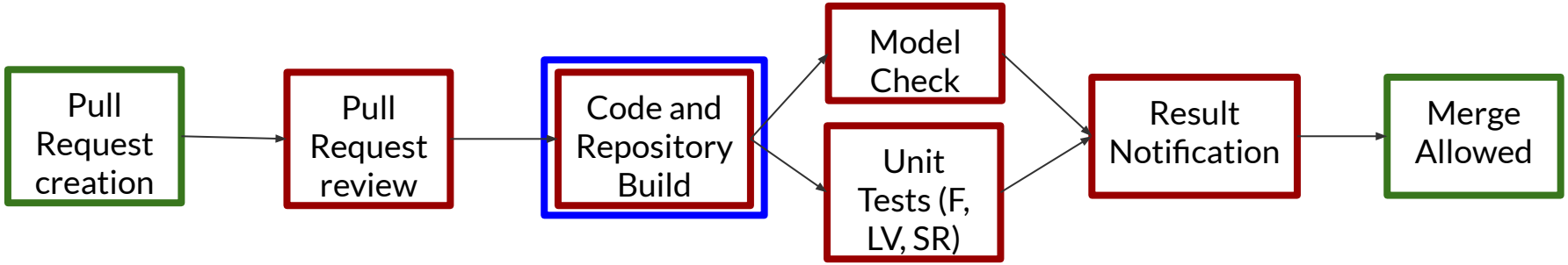
CSV Compare Report Example (2/2)

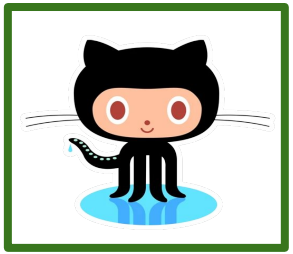
Offset between the reference and resulting signal!

The result signal is inside the ‘tolerance tube’ at all times.

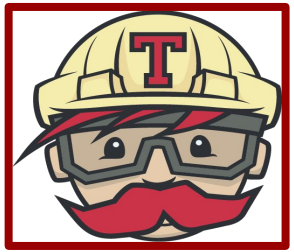
Continuous Integration

- OpenIPSL models are stored on a Github repository:
  - Need for an automatized procedure for code checking and model behavior verification
    - Continuous Integration - Software Engineering Solution

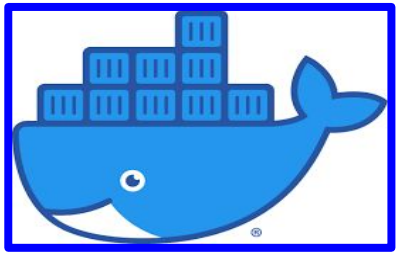




Github: repository where the code is located. Version control solution.



Travis CI: application to trigger and automate functions for the CI



Docker: container for our testing machine. Solution to the "it works on my machine" issue.

Automated Model Transformation

Main Idea

- Survey and analyze existing tools.
  - Existing Tools: Ditto, BIM2Modelica, CIM2Modelica
- Create the appropriate mappings from
  - PSSE to Modelica
  - CIM to Modelica
- Build Model Translation Tools
- Test and debug the tool with different systems
  - Single Machine Infinite Bus Systems
  - Components can be tested almost individually
    - Machines
    - Exciters
    - Governors
    - Stabilizers
    - Wind Machines
    - Compensators

OpenIPSL models can be exported and used in many tools

OpenIPSL models are shown to be consistent with PSSE

Need to design the basics of the translation tool with Modelica models as targets

Existing Tools - What's out there?

DiTTo (NREL)

- General idea: *many-to-one-to-many parser framework.*
  - Readers (inputs) and writers (outputs)
- Made for translating distribution systems and models from one data format to another.

BIM2Modelica

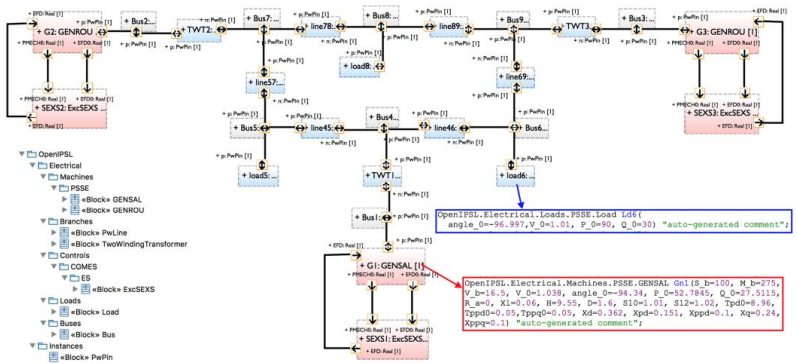
- Open source framework for generating and simulating thermal models by using data from BIM models.

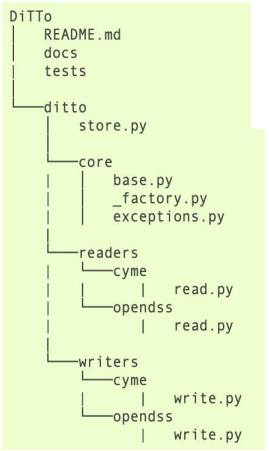
CIM2Modelica

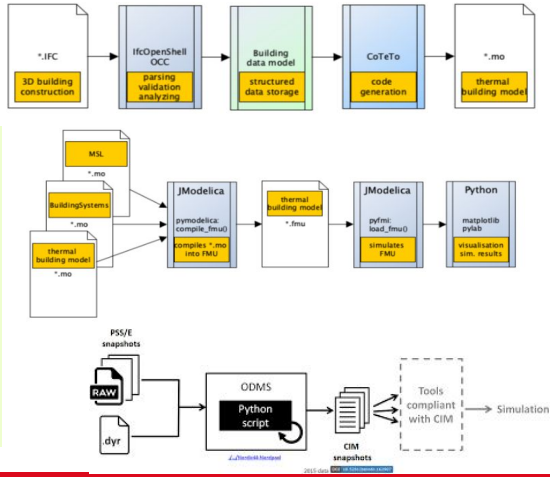
- Model-to-model transformation tool made for power systems
- XML schemas are used together with model-driven engineering concepts and paradigms

PSS@ODMS

- Proprietary tool with PSSE to CIM translation



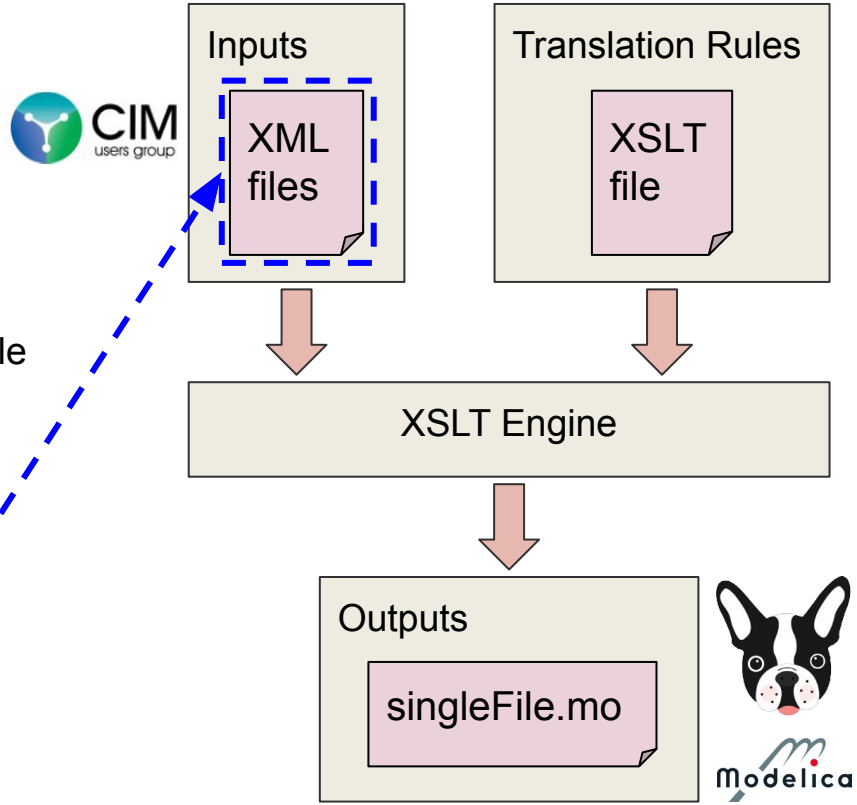




CIM to Modelica

Overview

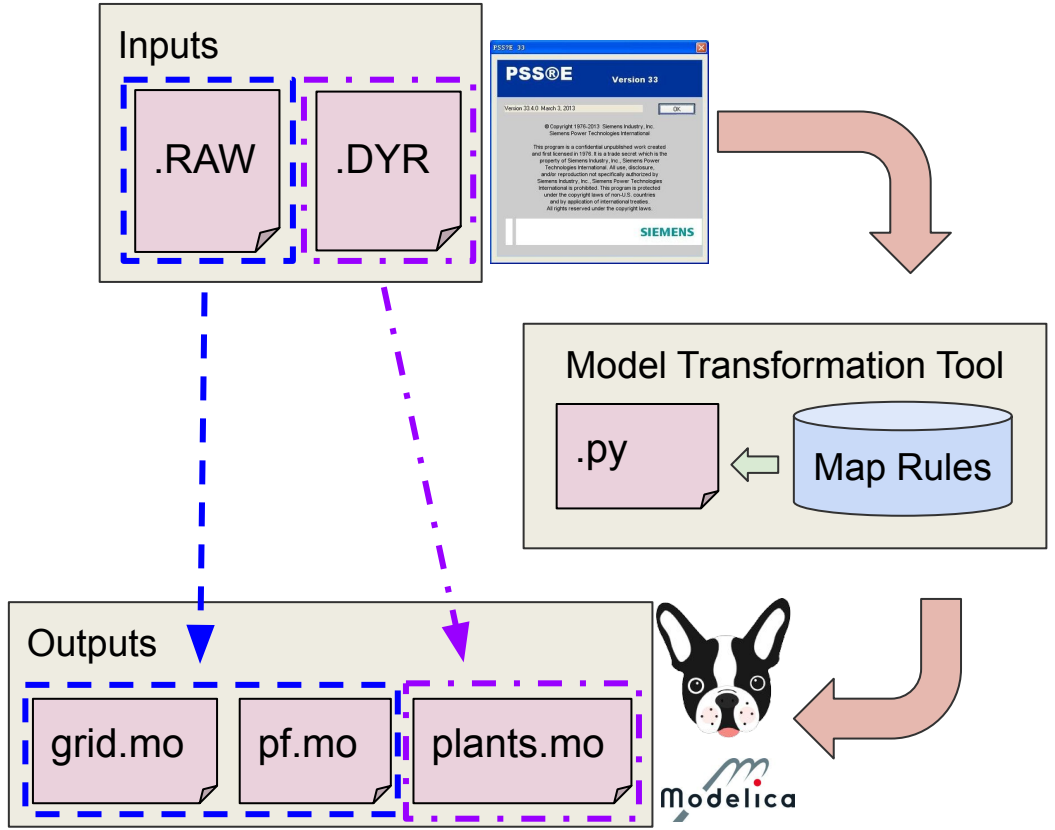
- Uses XSLT to translate CIM to Modelica
  - Proprietary tool - Windows Engine
- Common Information Model:
  - Readable (eXtensible Markup Language - XML).
  - Normalized - common data used in multiple types, linked using keys
  - Multiple CIM files including:
    - State Variables (SV)
    - Dynamic (DY)
    - Equipment (EQ)
    - Topology (TP)
    - Steady-state Hypothesis (SSH)



PSSE to Modelica

Overview

- Tool's general structure
  - Parsers, Readers and Writers
  - Based on templates
  - Written in Python
  - PSSE to Modelica
    - Raw files are translated into modelica network:
      - Lines, buses, shunts, loads
    - Dyr files are translated into dynamic models:
      - Machines, exciters, turbine-governors



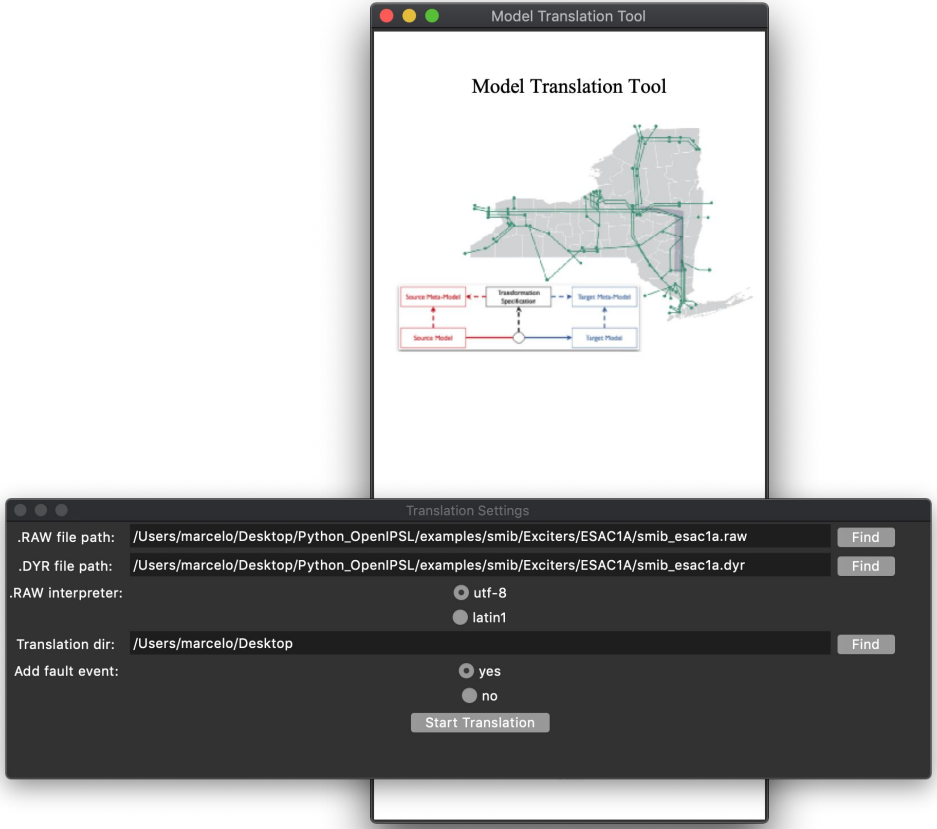
Assessment of Tool's Performance

Main Idea

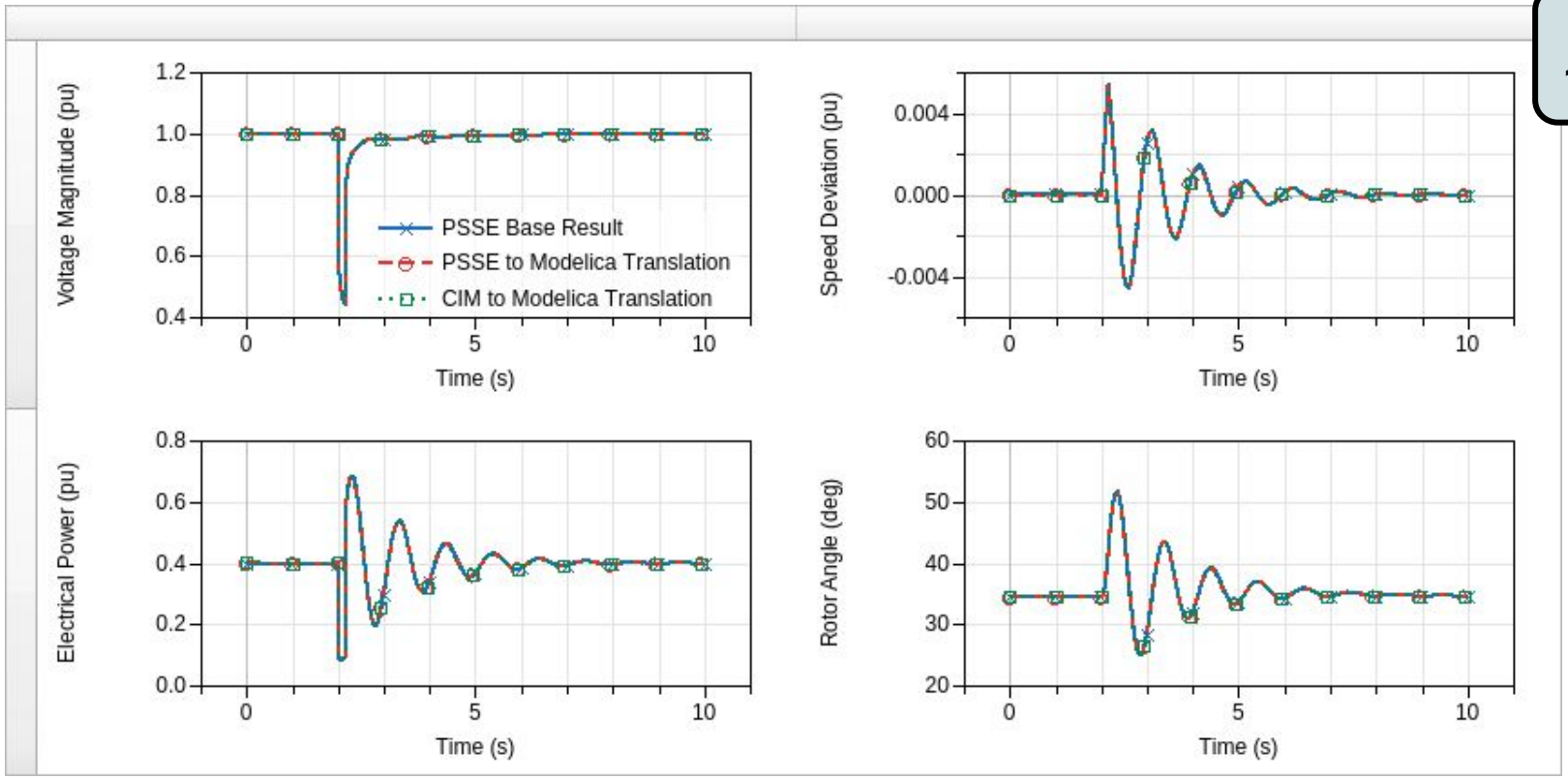
- Test translations performed by tool
- Test tool's scalability with different systems:
  - Brazilian 7-bus System, IEEE14, IEEE23, Nordic 44, Icelandic System, REN System, NYPA 500

Methods

- Test system for many different testing:
  - Assessing results
    - Check if translated systems simulate and if their results are the same of translated versions
  - Timing translation for metrics (PSSE to Modelica)
    - Test tool in different machines, translating different systems.



Simulation of Translated Models Using MT Tools



PSSE to Modelica Performance Test: Settings

Tests with different machines:

- **Goal is not to compare** the translation between different operating systems/machines
- **Goal is to assess** tool's **scalability** in different operating systems/machines

	Linux-based	**Mac**	**Windows**
Operating Software	Ubuntu 18.04.5 LTS	MacOS Mojave	Windows 10 20H2
Processor	i7 2.9 GHz x8	i7 2.8 GHz	AMD Ryzen 7 PRO 2700 3.20 GHz x8
Memory	16GB 1.6GHz DDR3	16GB 1.6GHz DDR3	64GB 1.6GHz DDR3

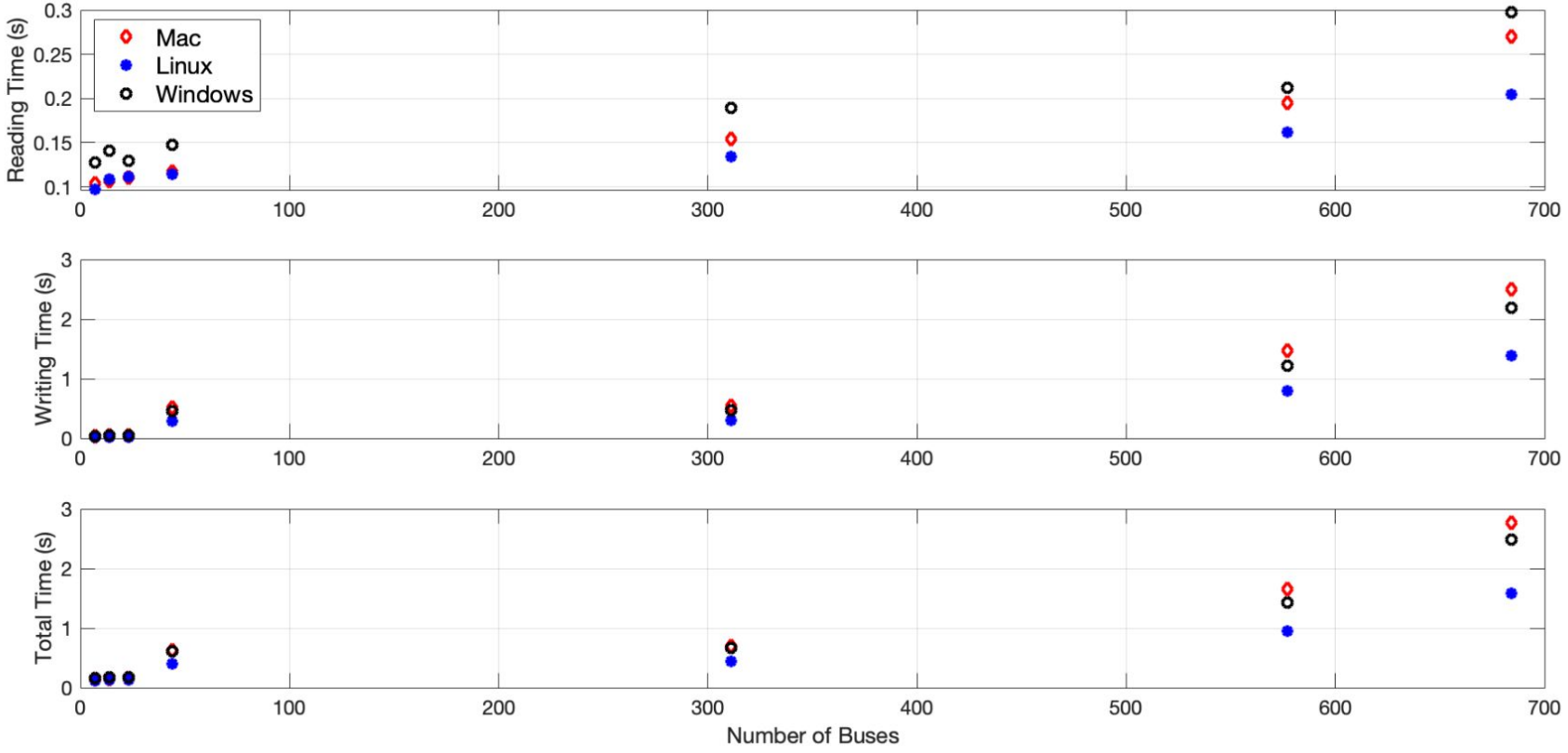
PSSE to Modelica Performance Test: Test Systems

ALSETlab

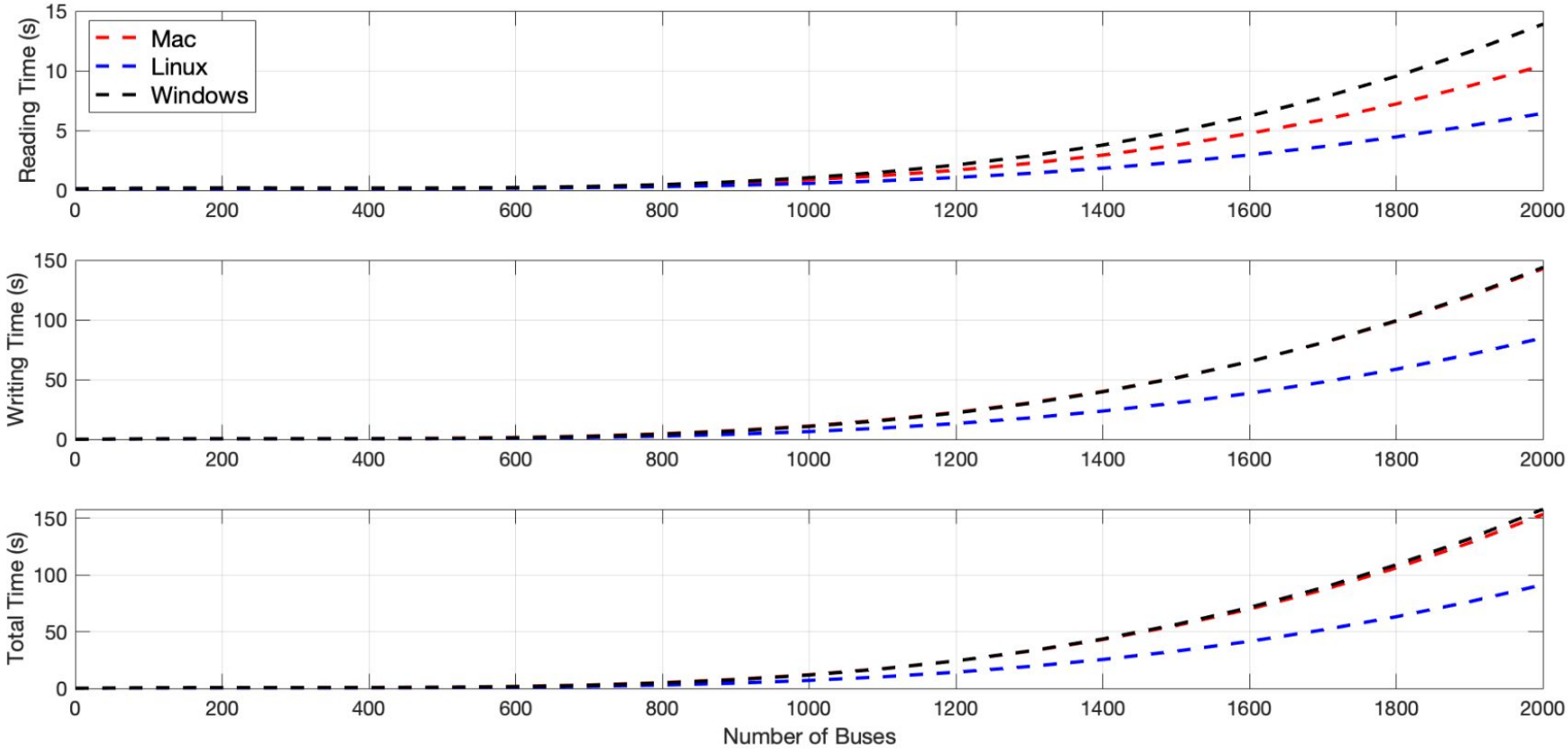
Different test systems

	Number of Buses	Number of Machines	Total Number of DAE
Brazilian 7-Bus	7	5	1032
IEEE 14	14	5	1346
IEEE 23	23	6	1619
Nordic 44	44	80	15976
Icelandic System	311	61	20342
REN System	577	138	34878
NY System	684	234	63094

PSSE to Modelica - Performance Results

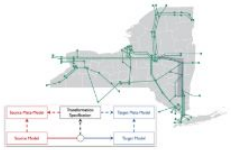


PSSE to Modelica - Performance Projection



Public Website

Model Transformation Tool - BabelGrid



Model Transformation for High Performing Grid Applications.

View the Project on GitHub
marcelofcastro/Python_OpenIPSL

This is the website for the 'Model Transformation for High Performing Smart Grid Applications' project. The project has been on going since Fall 2018. Joint project between New York Power Authority and Rensselaer Polytechnic Institute.

Project Overview

With the upcoming changes in the energy landscape of New York State, new modeling capabilities and simulation tools will be required in order for renewable and distributed energy technologies to be properly studied and incorporated into the electricity grid. In today's environment, grid modeling and operation in New York State are limited to few specific and proprietary software tools traditionally used for modeling and simulation over many decades. The ability of introducing new tools has been impractical, due to the way models have been developed and maintained. By using interoperable standards, and technologies, this project will create a new basis for developing and transforming grid models that could facilitate the utilization of multiple tools to advantage of the best features each tool has to offer in order to support the engineering and decision making process driven by the NYS Energy Plan and NY Reforming the Energy Vision (REV).

Main Resources

In this website you can find the following:

Pages	Links
Overview	About the Project
Model Validation	CSV Compare Results
Tool Description	About the Tool
Tool Guide	Using the Tool
About us	Contact Info

Website:
https://marcelofcastro.github.io/Python_OpenIPSL/index

- General information about the project is available there.

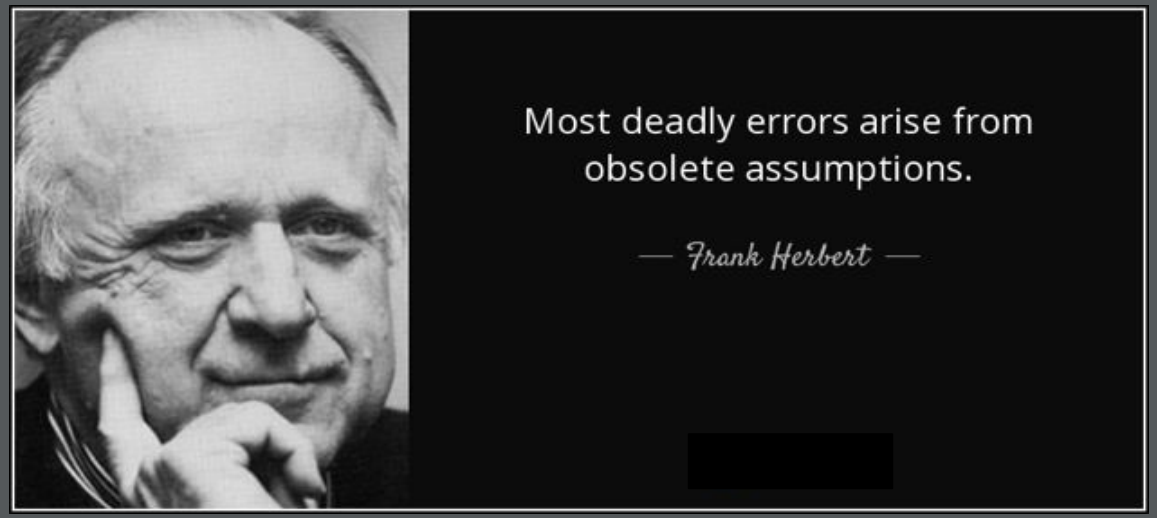
Dymola Reports

General results for Dymola tests can be found in the table below.

Model Type	General Reports
1. Machines	(a)Fault Report (b)Load Variation Report
2. Exciters	(a)Fault Report (b)Load Variation Report (c)Reference Step Report
3. Turbine Governors	(a)Fault Report (b)Load Variation Report
4. Power System Stabilizers	(a)Fault Report (b)Load Variation Report
5. Wind Machines	(a)Fault Report

All validation reports are available here

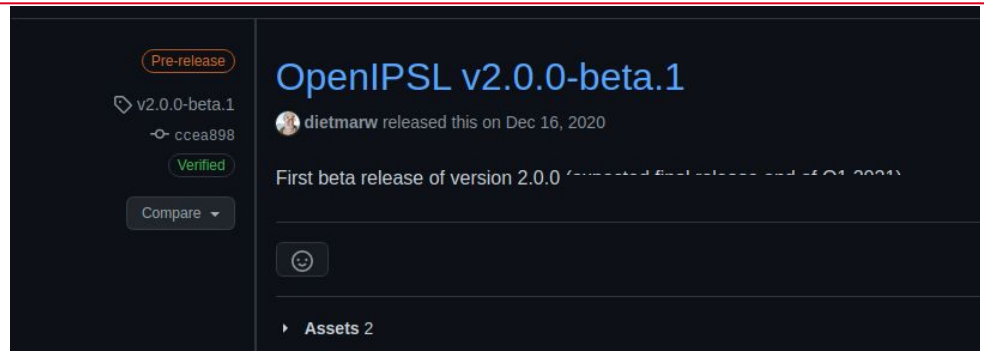
Final Remarks

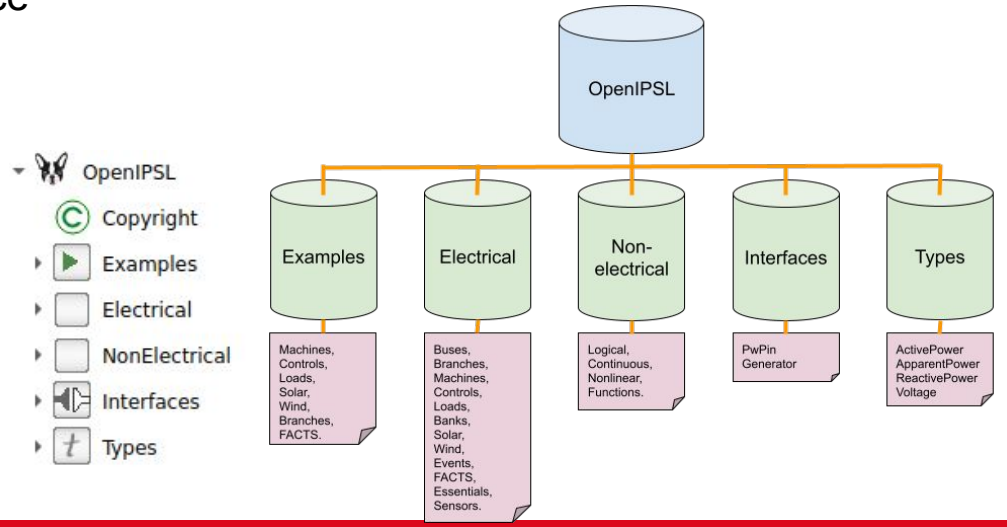


OpenIPSL - New Library Release

OpenIPSL v2.0 - Beta

- New library “beta” is available
- Models have their behavior verified against PSSE models
  - Mapping is now certified to be valid because models are consistent
- Models can be automatically checked once the library is updated via pull request
  - Soon to be implemented in Github Actions
- Library has a more stable base
  - Modelica solutions implemented
    - Types
  - Consistent with Modelica notations
- Final v2 release expected by end of 2021
- Version 2.1 will include the RES models, planned for Q2 2022.





Conclusion and Final Remarks

- ***Modelica and OpenIPSL can help address today's challenges of the de facto simulation tools*** used in power grid analysis and design, however, "social aspects" related to resistance to change have to be addressed
  - Need to convince practitioners that it is as good or better than domain specific approaches/tools, we use SW-to-SW validation and other studies.
  - Regulators should require open access standards and not favor a specific software/vendor.
- *Modelica can be used to develop open source models of renewable energy sources, without having to reimplement the models in multiple platforms:* ***cost reduction and productivity gains!***
  - The open access standards allow models from one language to be used in many tools without loss of information and consistency of simulation results.
  - The library of Modelica models developed in this project is compatible with many different Modelica-compliant tools and models can be exported via FMI standard
    - Modelica model is a good way to describe model (open access specification supported by multiple tools natively)
    - Target Model can be exported to various tools using FMI (+100!) for different applications
- ***Ongoing and Future Work***
  - Model transformation tool to "transform" models from typical "parameter files", i.e. PSS/E, CIM, into Modelica representations to automate the building of large system models → ease adoption by practitioners.
  - Release the RES models presented in v2.1 of OpenIPSL.
  - Continue expanding capabilities of the library.
  - Online: http://openipsl.org

Rensselaer

why not change the world?®

Thank you for your attention!