OpenIPSL and RaPId

Open Source Software Tools for Smart Grid Modeling, Simulation and Model Validation

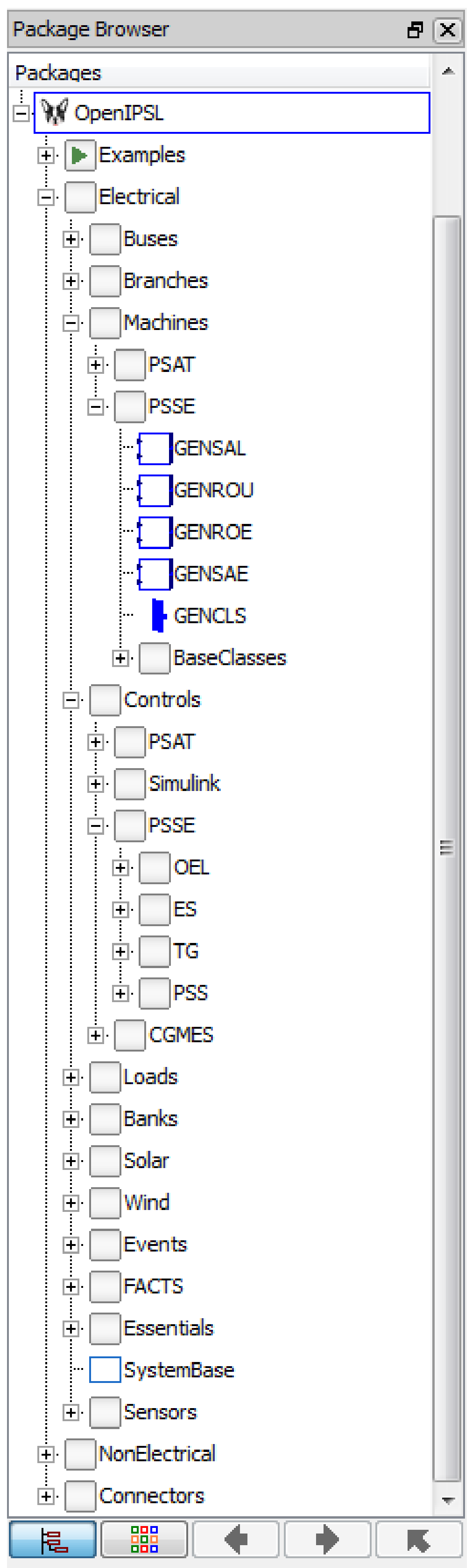
Luigi Vanfretti, Maxime Baudette, and Francisco José Gómez
KTH Royal Institute of Technology
luigiv@kth.se, baudette@kth.se, fragom@kth.se

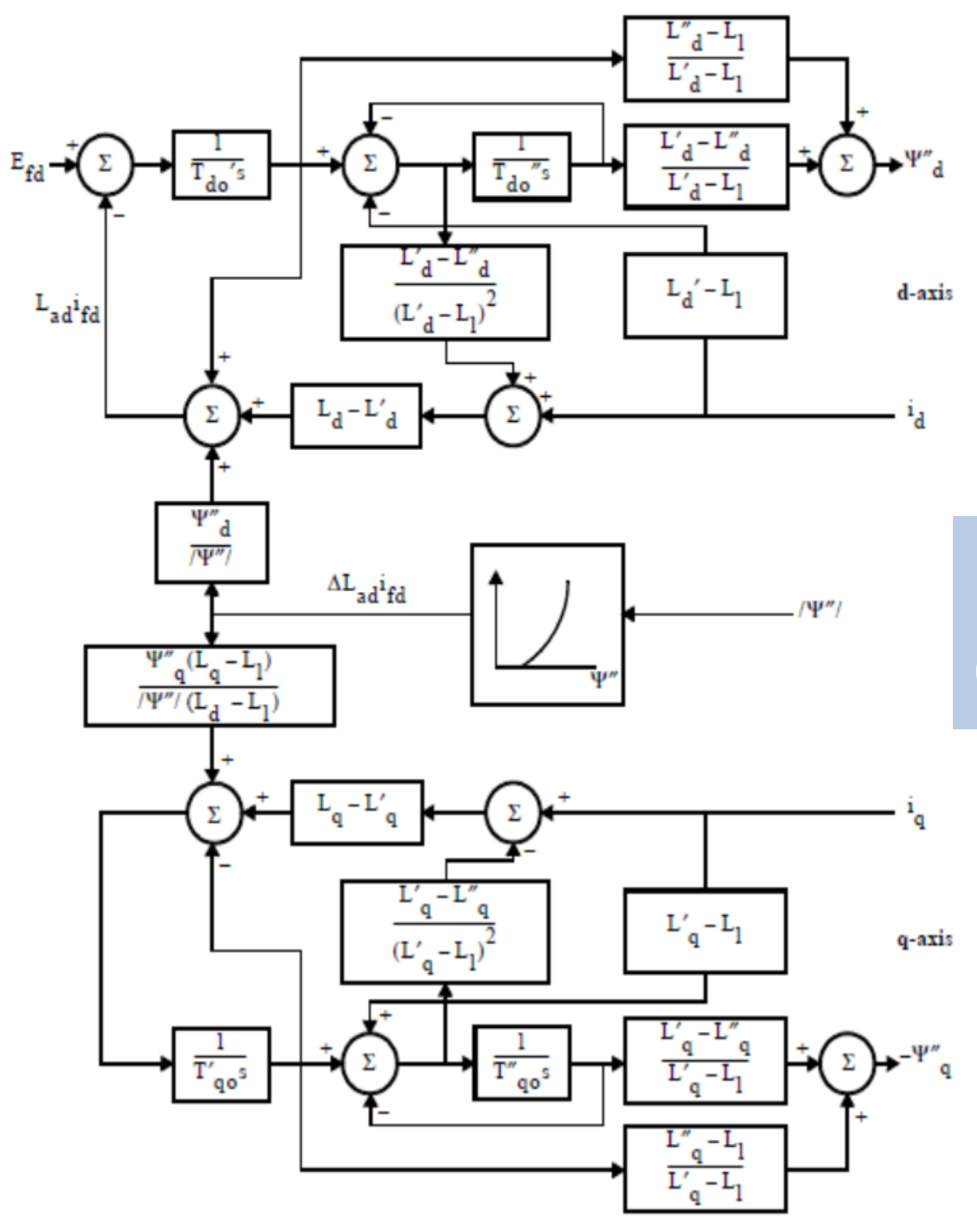
OpenIPSL

The OpenIPSL – A Modelica Library

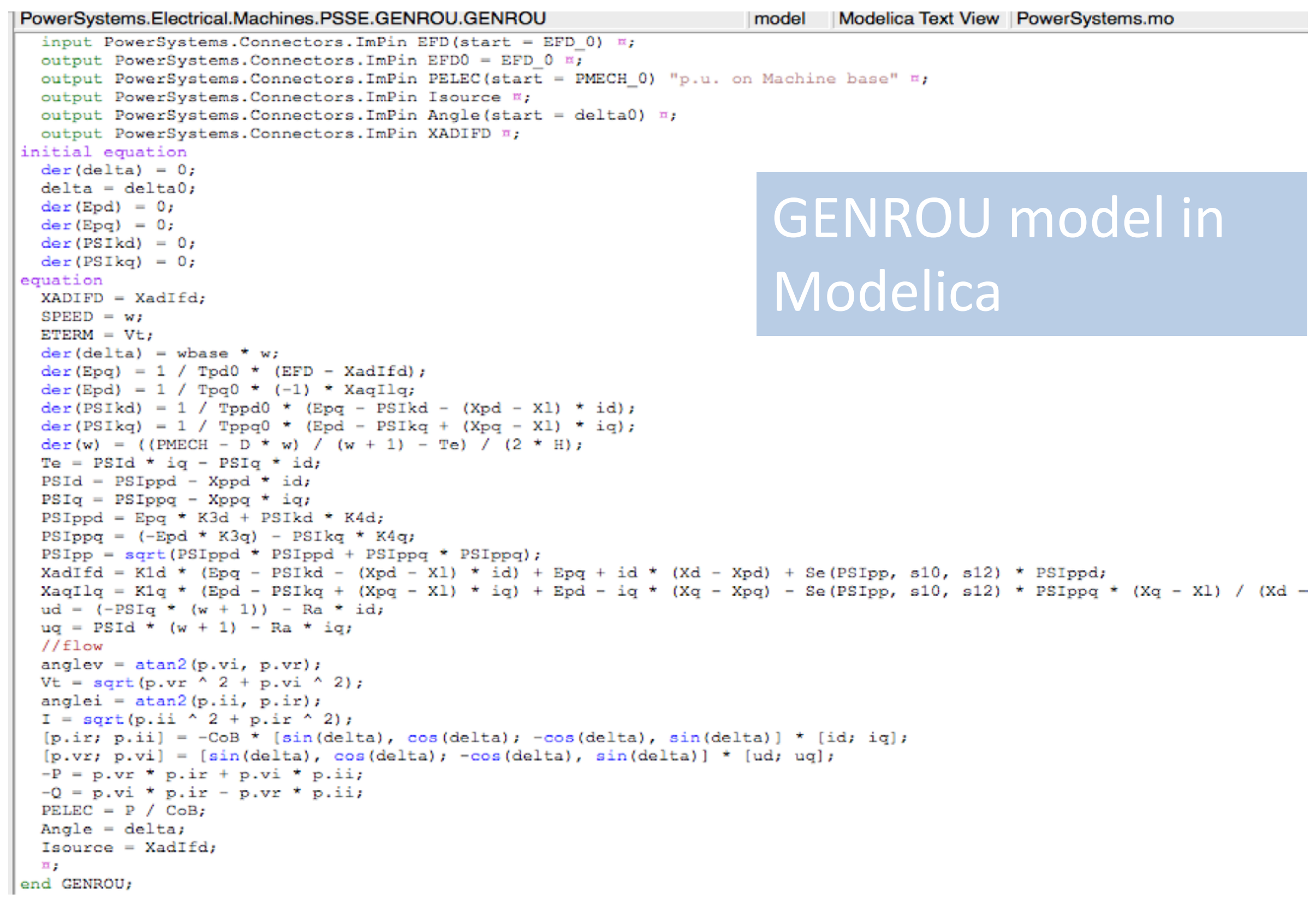
- Modeling and simulation should not be ambiguous: it should be consistent across different simulation platforms.
- For unambiguous modeling, model sharing and simulation, Modelica and Modelica Tools can be used due to their ***standardized equation-based modeling language***.
- The Power Systems library developed using as reference domain specific software tools (e.g. PSS/E, Eurostag, PSAT and others)
- The library is being tested in several Modelica supporting software: OpenModelica, Dymola, SystemModeler
- Components and systems are validated against proprietary tools and one OSS tool used in power systems (domain specific)

OpenIPSL





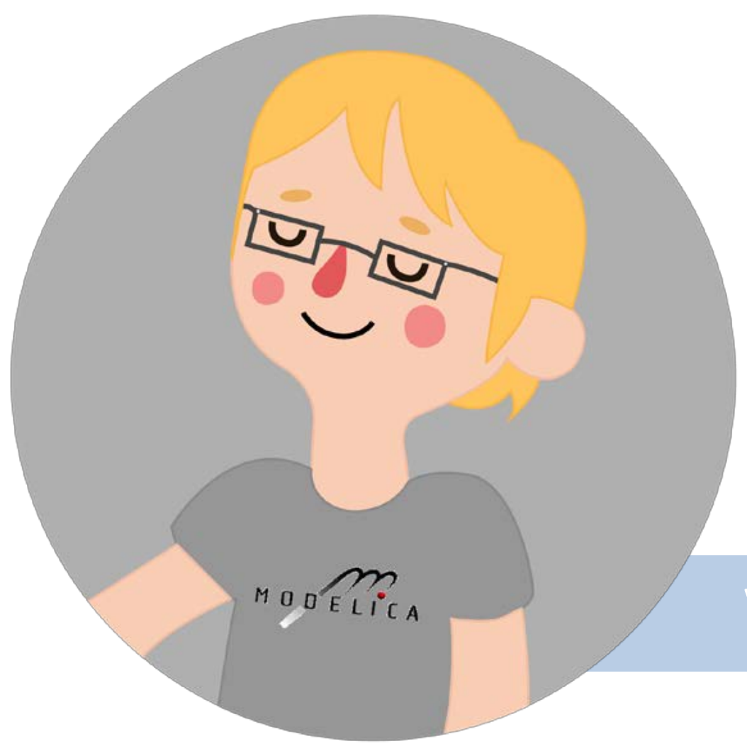
PSS/E model: GENROU



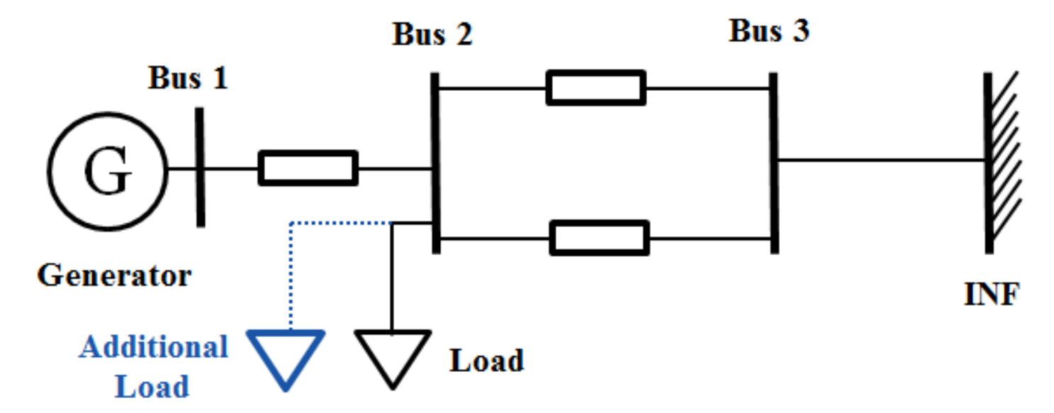
PowerSystems.Electrical.Machines.PSSE.GENROU.GENROU | model | Modelica Text View | PowerSystems.mo

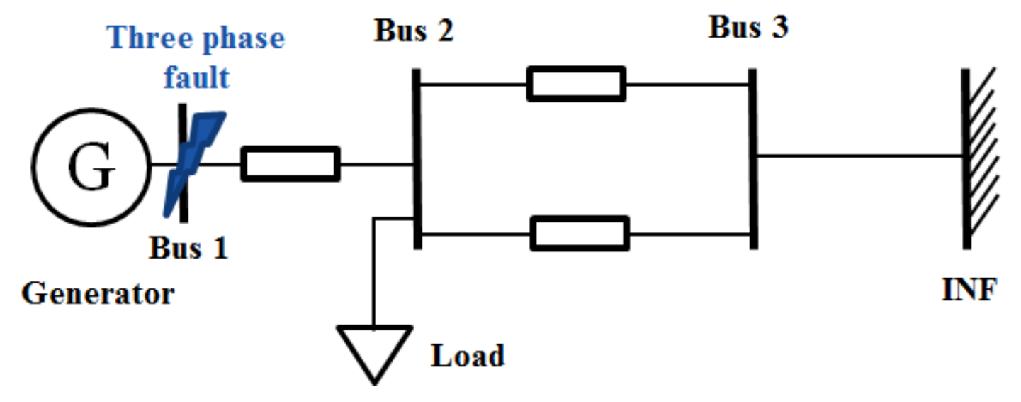
```
  input PowerSystems.Connectors.ImPin EFD(start = EFD_0) ¤;
  output PowerSystems.Connectors.ImPin EFD0 = EFD_0 ¤;
  output PowerSystems.Connectors.ImPin PELEC(start = PMECH_0) "p.u. on Machine base" ¤;
  output PowerSystems.Connectors.ImPin Isource ¤;
  output PowerSystems.Connectors.ImPin Angle(start = delta0) ¤;
  output PowerSystems.Connectors.ImPin XADIFD ¤;
initial equation
  der(delta) = 0;
  delta = delta0;
  der(Epd) = 0;
  der(Epq) = 0;
  der(PSIkd) = 0;
  der(PSIkq) = 0;
equation
  XADIFD = XadIfd;
  SPEED = w;
  ETERM = Vt;
  der(delta) = wbase * w;
  der(Epq) = 1 / Tpd0 * (EFD - XadIfd);
  der(Epd) = 1 / Tpq0 * (-1) * XaqIlq;
  der(PSIkd) = 1 / Tppd0 * (Epq - PSIkd - (Xpd - Xl) * id);
  der(PSIkq) = 1 / Tppq0 * (Epd - PSIkq + (Xpq - Xl) * iq);
  der(w) = ((PMECH - D * w) / (w + 1) - Te) / (2 * H);
  Te = PSId * iq - PSIq * id;
  PSId = PSIppd - Xppd * id;
  PSIq = PSIppq - Xppq * iq;
  PSIppd = Epq * K3d + PSIkd * K4d;
  PSIppq = (-Epd * K3q) - PSIkq * K4q;
  PSIpp = sqrt(PSIppd * PSIppd + PSIppq * PSIppq);
  XadIfd = K1d * (Epq - PSIkd - (Xpd - Xl) * id) + Epq + id * (Xd - Xpd) + Se(PSIpp, s10, s12) * PSIppd;
  XaqIlq = K1q * (Epd - PSIkq + (Xpq - Xl) * iq) + Epd - iq * (Xq - Xpq) - Se(PSIpp, s10, s12) * PSIppq * (Xq - Xl) / (Xd -
  ud = (-PSIq * (w + 1)) - Ra * id;
  uq = PSId * (w + 1) - Ra * iq;
  //flow
  anglev = atan2(p.vi, p.vr);
  Vt = sqrt(p.vr ^ 2 + p.vi ^ 2);
  anglei = atan2(p.ii, p.ir);
  I = sqrt(p.ii ^ 2 + p.ir ^ 2);
  [p.ir; p.ii] = -CoB * [sin(delta), cos(delta); -cos(delta), sin(delta)] * [id; iq];
  [p.vr; p.vi] = [sin(delta), cos(delta); -cos(delta), sin(delta)] * [ud; uq];
  -P = p.vr * p.ir + p.vi * p.ii;
  -Q = p.vi * p.ir - p.vr * p.ii;
  PELEC = P / CoB;
  Angle = delta;
  Isource = XadIfd;
  ¤;
end GENROU;
```

GENROU model in Modelica



Validation of GENROU model



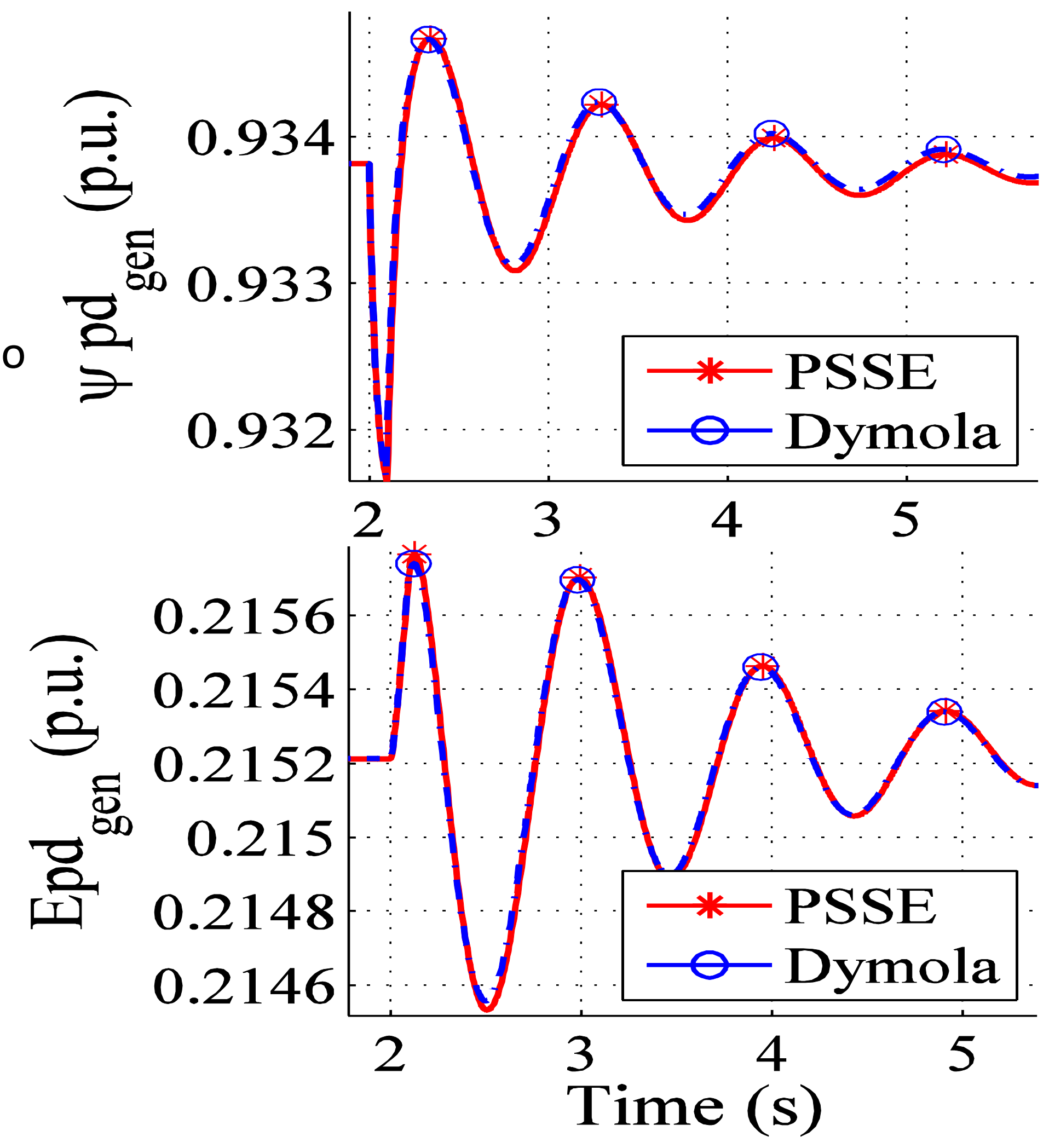


Set-up a model in each tool with the simulation scenario configured

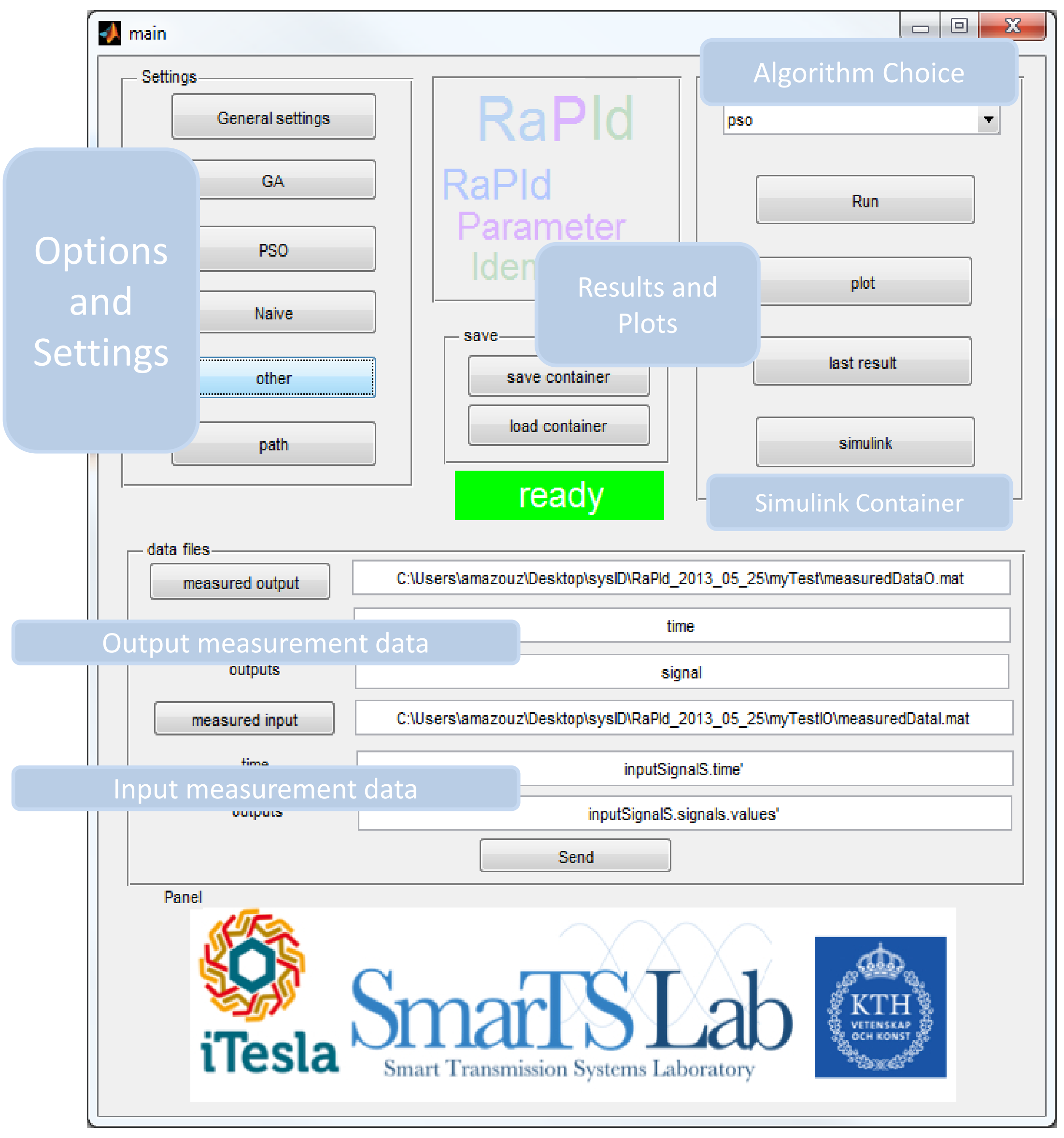
In the case of Modelica, the simulation configuration can be done within the model

In the case of PSS/E, a Python script is created to perform the same test.

Sample Test:
1. Running under steady state for 2s.
2. Vary the system load with constant P/Q ratio.
3. After 0.1s later, the load was restored to its original value .
4. Run simulation to 10s.
5. Apply three phase to ground fault.
6. 0.15s later clear fault by tripping the line.
7. Run simulation until 20s.



RApid Parameter IDentification toolbox



pso

C:\Users\amazouz\Desktop\sysID\RaPId_2013_05_25\myTest\measuredDataO.mat

time

signal

C:\Users\amazouz\Desktop\sysID\RaPId_2013_05_25\myTestIO\measuredDataI.mat

inputSignalS.time'

inputSignalS.signals.values'

Options and Settings

Algorithm Choice

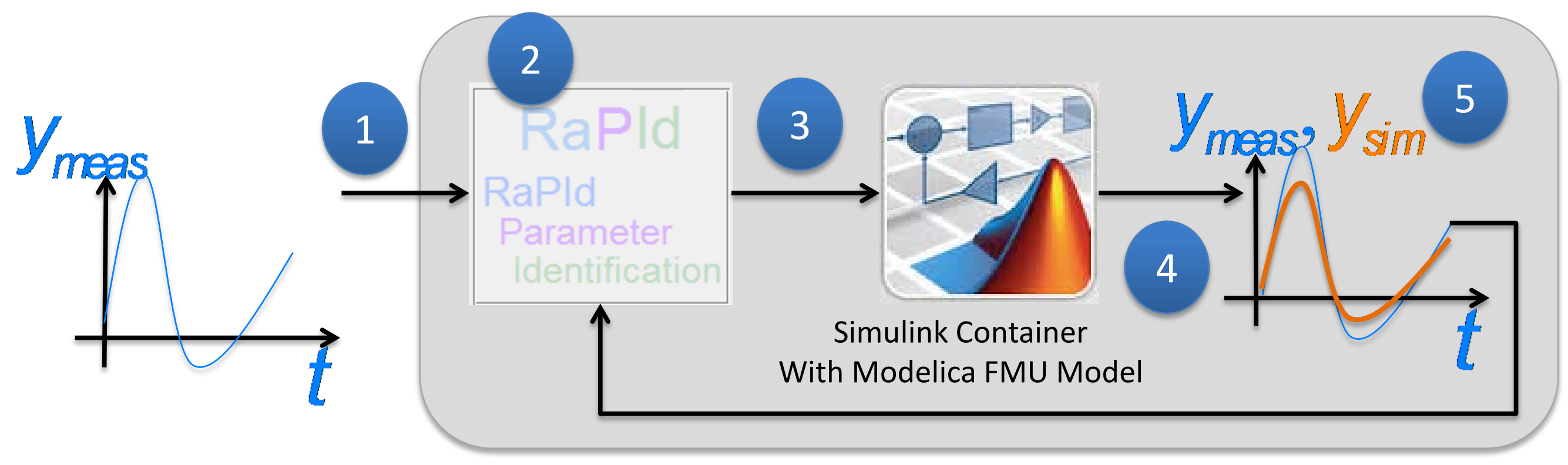
Results and Plots

Simulink Container

Output measurement data

Input measurement data

- **RaPId was developed in MATLAB.**
- The MATLAB code acts as ***wrapper*** to provide interaction with several other programs (which may not need to be coded in MATLAB).
- Advanced users can simply use MATLAB scripts instead of the graphical interface.
- **Plug-in Architecture:**
- Completely extensible and open architecture allows advanced users to add:
- Identification methods
- Optimization methods
- Specific objective functions
- Solvers (numerical integration routines)



1 Output (and optionally input) measurements are provided to RaPId by the user.

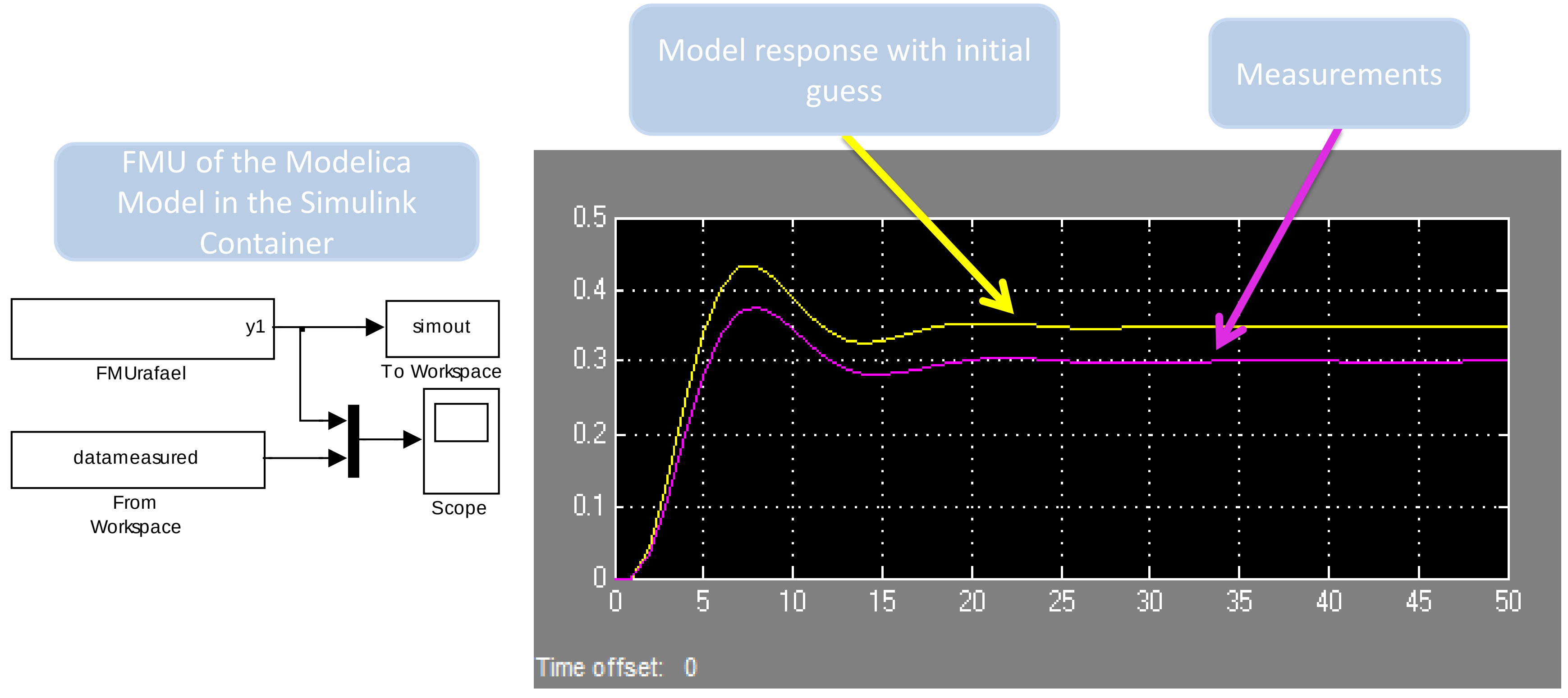
2 At initialization, a set of parameters is pre-configured (or generated randomly by RaPId)

3 The model is simulated with the parameter values given by RaPId.

4 The outputs of the model are recorded and compared to the user-provided measurements.

5 A fitness function is computed to judge how close the measured data and simulated data are to each other

2' **Simulations continue until a min. fitness or max no. of iterations (simulation runs) are reached.**

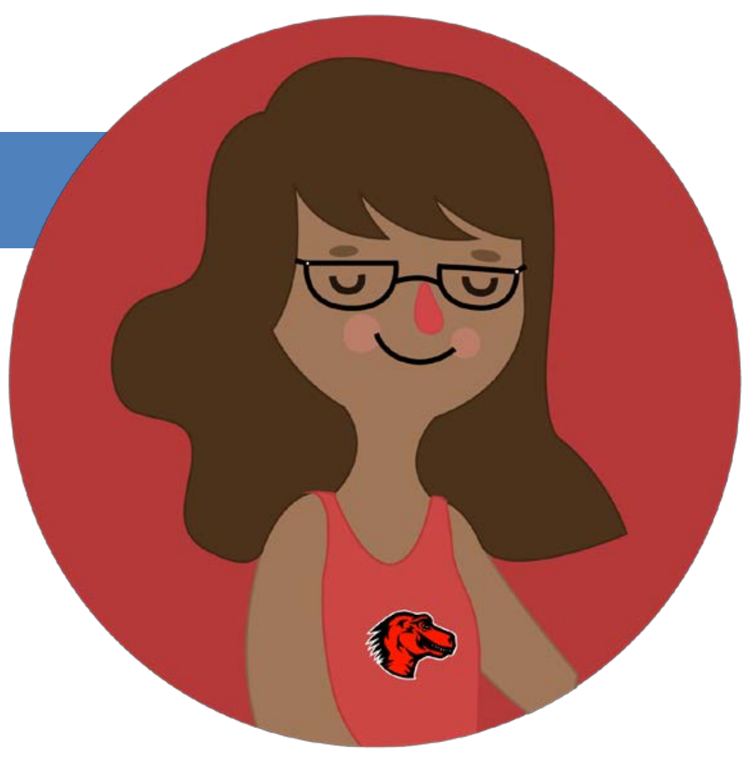


Model response with initial guess

Measurements

FMU of the Modelica Model in the Simulink Container

Useful Resources – Our Open Source Software



OpenIPSL is available on Github:
- Get involved at https://github.com/SmarTS-Lab/OpenIPSL
- Documentation: http://openipsl.readthedocs.io/en/latest/
- See video-demos on-line: https://www.youtube.com/watch?v=nKOCulNiy3o

RaPId toolbox is available on GitHub:
- Get involved at https://github.com/SmarTS-Lab/iTesla_RaPId
- See video-demos on-line about RaPId:

GUI example: https://www.youtube.com/watch?v=e7OkVEtcz6A
CLI example: https://www.youtube.com/watch?v=4qrPASIWdiY