Modeling Challenges and Opportunities in Transient Simulations for Power Systems with Large Penetration of Converter-Interfaced Generation

Duncan Callaway

Jose Daniel Lara

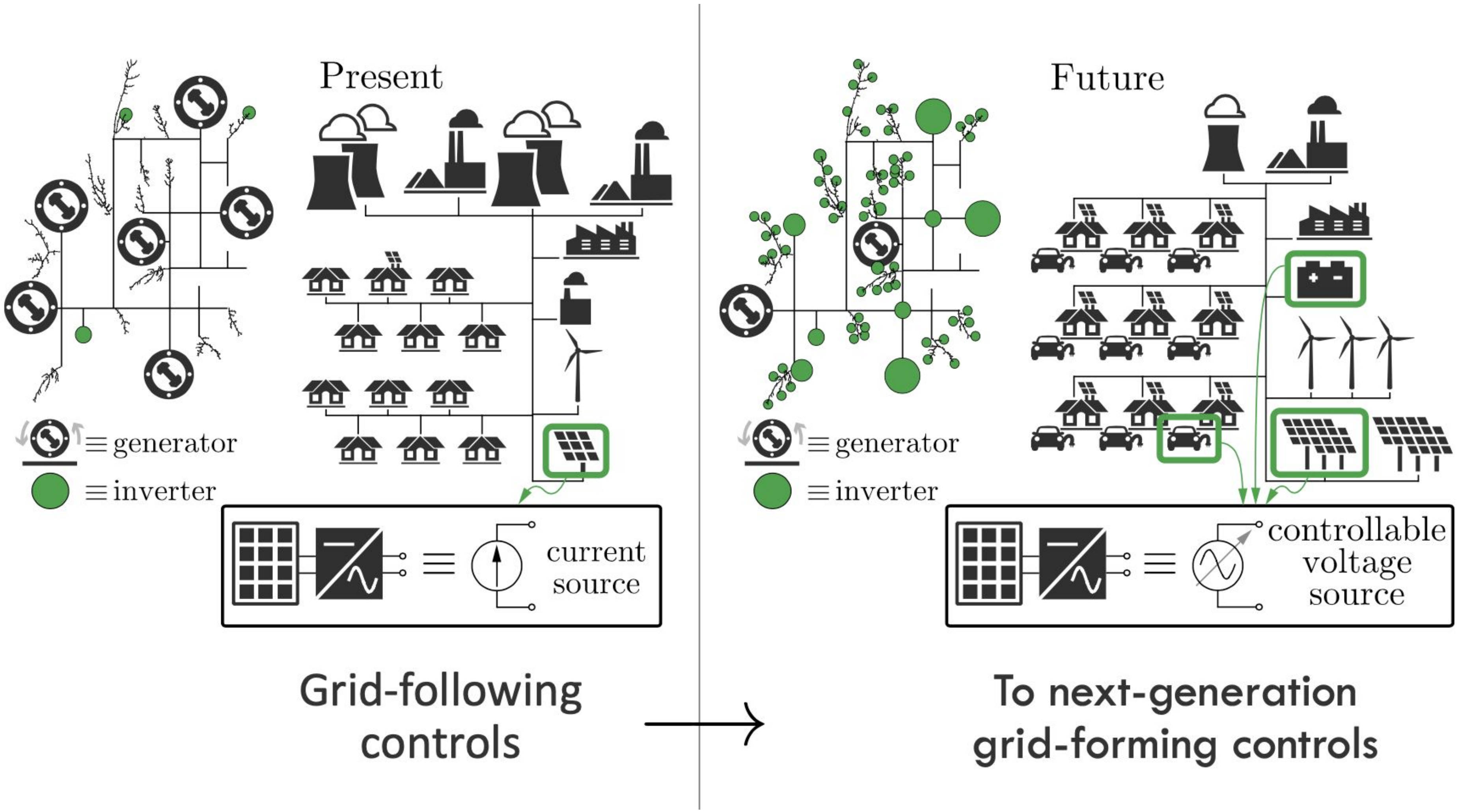
Rodrigo Henriquez-Auba

University of California, Berkeley

{dcal, jdlara, rhenriquez}@berkeley.edu

PSERC Webinar
November 24, 2020

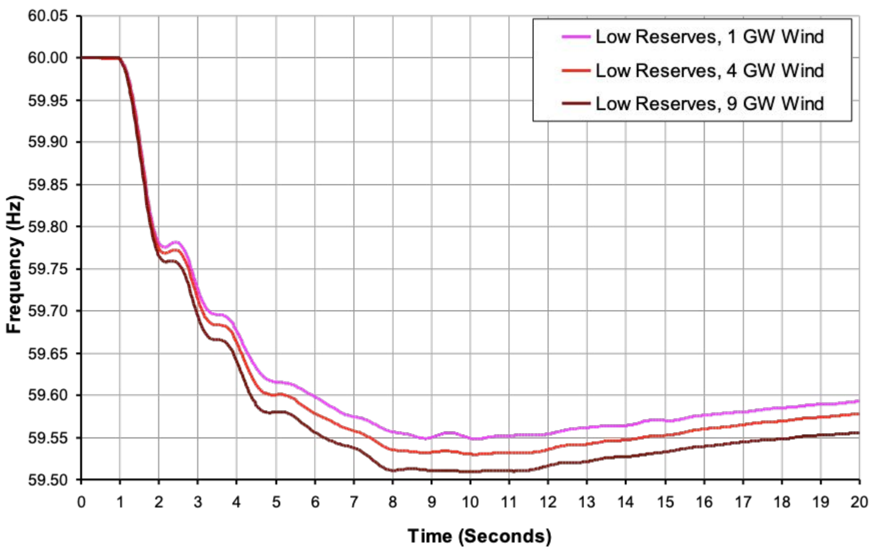
Next generation grids



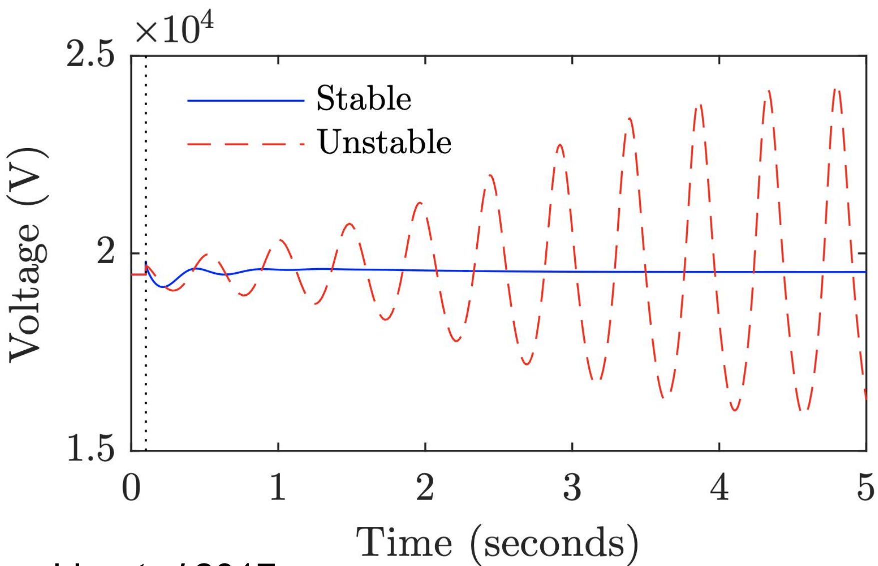
Credit: Yashen Lin, NREL

Will they operate reliably?

- Low inertia: Early grid-scale work focused on swing dynamics
  - RoCoF
  - Nadir
  - Steady state



Undrill *et al* 2010



Lin *et al* 2017

- However higher order dynamics are important, too
  - Small signal stability
  - Transient stability

Today's Objective

- Introduce a new simulation tool that supports scientific computing for large-scale dynamics simulations
  - design philosophy,
  - validation,
  - short tutorial

PowerSimulationsDynamics.jl

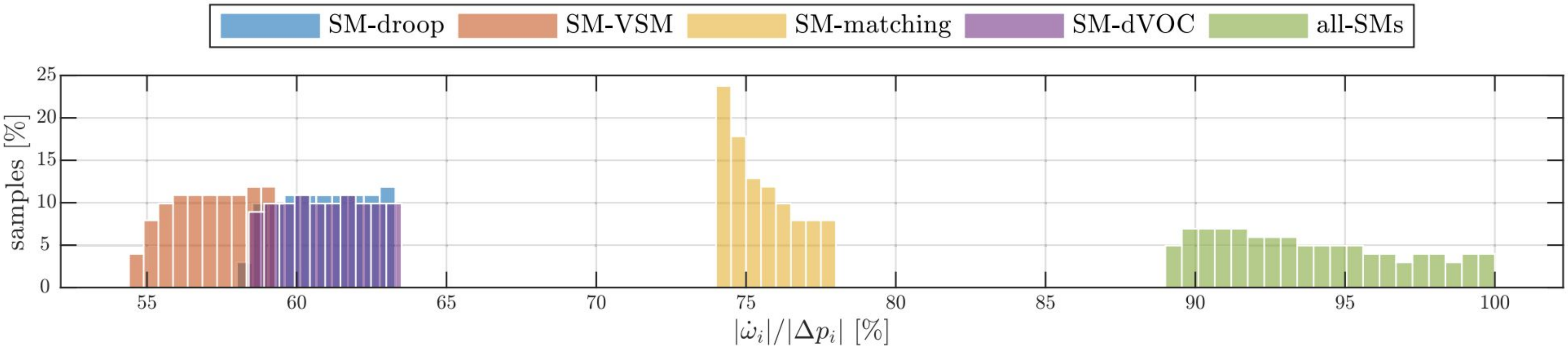
```julia
inverter = DynamicInverter(
    name = get_name(storage),
    ω_ref = 1.0, # ω_ref,
    converter = AverageConverter(rated_voltage = 138.0, rated_current = 100.0),
    outer_control = OuterControl(
        VirtualInertia(Ta = 2.0, kd = 400.0, kω = 20.0),
        ReactivePowerDroop(kq = 0.2, ωf = 1000.0),
    ),
    inner_control = CurrentControl(
        kpv = 0.59,     #Voltage controller proportional gain
        kiv = 736.0,    #Voltage controller integral gain
        kffv = 0.0,     #Binary variable enabling the voltage feed-forward in output of current controllers
        rv = 0.0,       #Virtual resistance in pu
        lv = 0.2,       #Virtual inductance in pu
        kpc = 1.27,     #Current controller proportional gain
        kic = 14.3,     #Current controller integral gain
        kffi = 0.0,     #Binary variable enabling the current feed-forward in output of current controllers
        ωad = 50.0,     #Active damping low pass filter cut-off frequency
        kad = 0.2,
    ),
    dc_source = FixedDCSource(voltage = 600.0),
    freq_estimator = KauraPLL(
        ω_lp = 500.0, #Cut-off frequency for LowPass filter of PLL filter.
        kp_pll = 0.084,  #PLL proportional gain
        ki_pll = 4.69,   #PLL integral gain
    ),
    filter = LCLFilter(lf = 0.08, rf = 0.003, cf = 0.074, lg = 0.2, rg = 0.01),
)
```

NREL Transforming ENERGY

Berkeley UNIVERSITY OF CALIFORNIA

ENERGY MODELING, ANALYSIS, & CONTROL
ENERGY & RESOURCES GROUP

Example: Low Inertia Frequency Dynamics

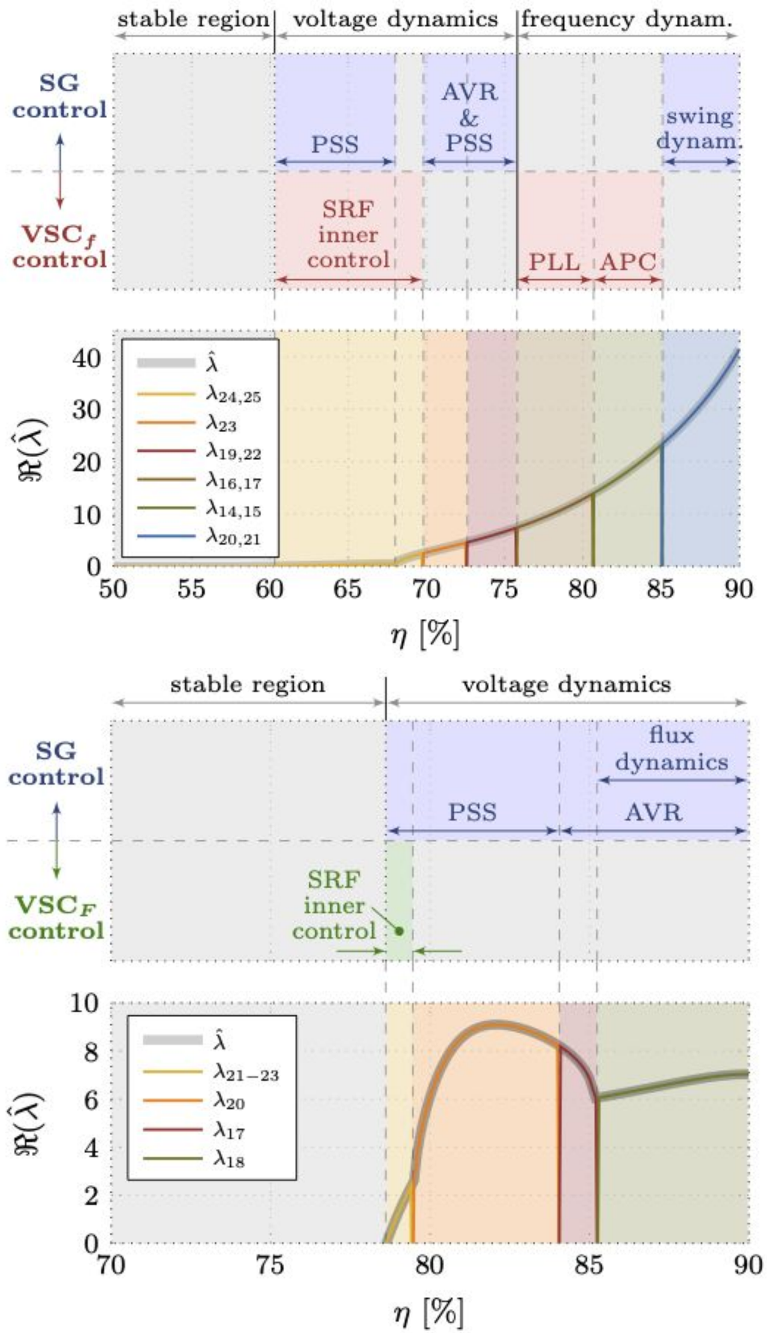


Tayyebi *et al.*

- Explore the performance of different grid-forming inverter controls on frequency response metrics
- Customized 9-bus test system model built in Simulink

Tayyebi *et al,* "Frequency Stability of Synchronous Machines and Grid-Forming Power Converters" *IEEE J-ESTPE* 2020

Example: Low Inertia Small Signal Stability



Markovic *et al*:

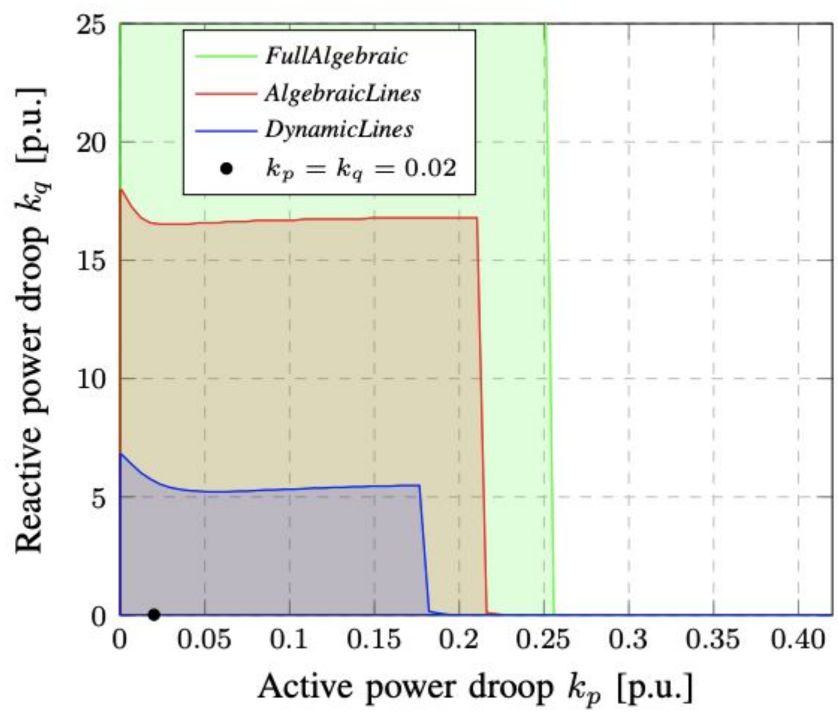
- Explore small signal stability for large-scale systems with high penetrations of power electronic converters
- Constructed a customized small-signal model
- Validated in Matlab Simulink

Markovic *et al,* “Understanding Stability of Low-Inertia Systems,” *in rev.*

Example: What is the role of line dynamics?

Henriquez *et al.*, explore the role of:

- line model assumptions
- grid forming control assumptions on region of s.s. stability.
- Leveraged recent Julia libraries to enable fast Jacobian calculations



Henriquez *et al.* “Grid Forming Inverter Small Signal Stability: Examining Role of Line and Voltage Dynamics” IECON 2020

Example: Teaching

"21st Century Power System Dynamics" (EECS 290O) at UC Berkeley

- Course objective: Equip students with the theoretical knowledge to model power system dynamics with voltage source converters *and* synchronous machines
- Semester project: Implement one grid-forming inverter model from the literature
- Tremendous effort involved in just getting models to run!

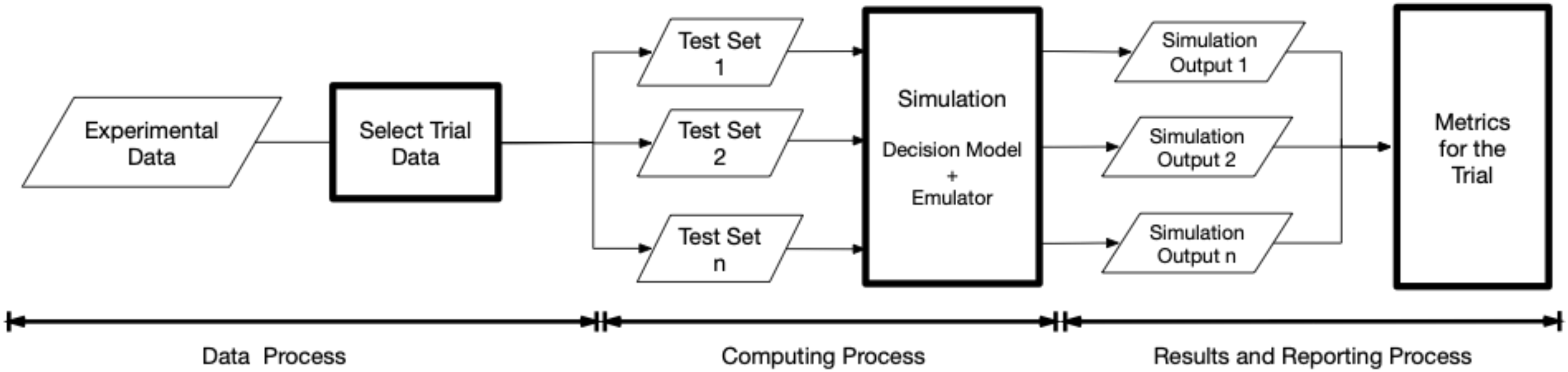
There is a gap in available computing tools

In each example, researchers and students had to build their own simulation models to study emerging questions on dynamics in low-inertia systems

- This takes time (a significant part of someone's PhD…)
- Hinders reproducibility
- Slows down the review process

We argue that principles of scientific computing are ripe for application in power systems research, and will address the issues above.

Scientific computing principles for power systems research



1. **Data Process**: experiment parameters, test system for the experiment, number of sample sets of confounding variables.
2. **Computing Process**: This enables investigating a range of discrete simulation scenarios
3. **Results and Reporting Process**: Report distribution of results across distribution of confounding variables.

Lara *et al* “Computational Experiment Design for Operations Model Simulation” *Electric Power Systems Research, 2020*

Q: What's *needed* to facilitate scientific computing in power systems research?

...Especially if we wish to focus on large-scale systems with high penetration of converter-interfaced generation

A: Open-source tools for power system analysis

- Built in a fast, interactive, technical language
- Separation between modeling and algorithms
- Modular component descriptions, EMT capabilities
- Seamless capacity for scripting and automating scenario generation and execution
- Parse industry standard data files, validation against industry standard tools.

julia

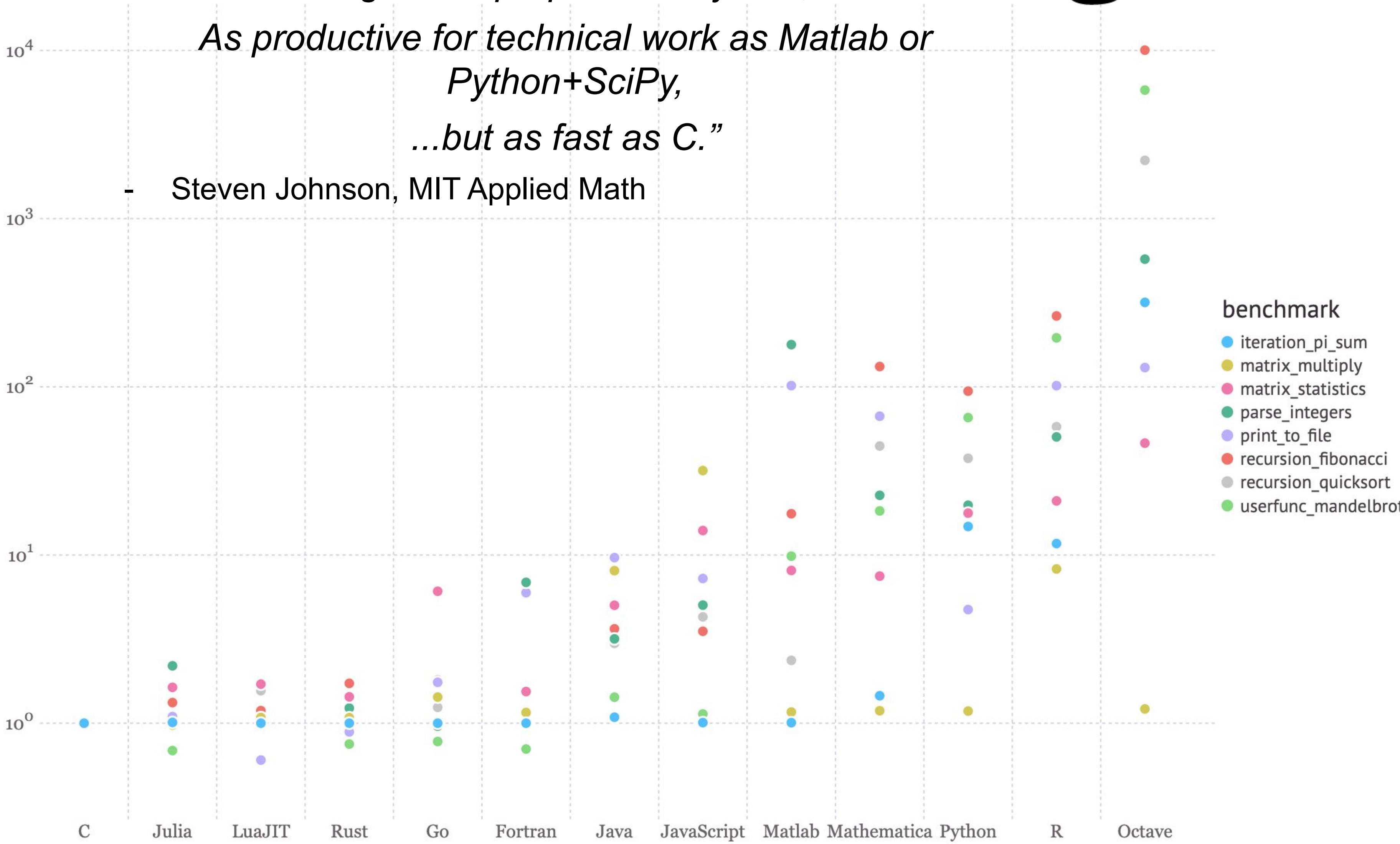
"As high-level and interactive as Matlab or Python+IPython,

As general-purpose as Python,

As productive for technical work as Matlab or Python+SciPy,

...but as fast as C."

- Steven Johnson, MIT Applied Math



Examples of existing open-source tools

- Power flow and OPF:
  - MATPOWER (Matlab Based)
  - PyPower (Python Based)
  - PowerModels.jl (Julia Based)
- Dynamics:
  - PSAT (Power flow, dynamics and stability, harmonics) (Matlab Based)
  - ANDES (Python Based)
  - PST (Matlab Based)
  - iTesla Power Systems Library (iPSL) (OpenModelica Based)
  - GridDyn (C++ Based)

`PowerSimulationsDynamics.jl` (PSID)

PSID is a Julia-based open-source power system modeling and simulation toolbox designed to study system stability in **large-scale, low-inertia** power systems.

- Data models & modularity facilitate fast model development.
- Julia: provides cutting edge solvers for large, stiff systems

Our vision: enable *scientific computing* approaches to EMT simulations to study emerging power system dynamics topics. For example,

- Rigorous study of model complexity vs fidelity
- Benchmarking emerging converter control strategies

Software Architecture

Data model and inputs

PowerSystems.jl

- PSS/e raw and dyr files
- Matpower power flow cases
- PSAT matlab files (upcoming)

Model Formulation

PowerSimulationsDynamics.jl

- Implicit Model
- Explicit Model
- Small Signal Stability

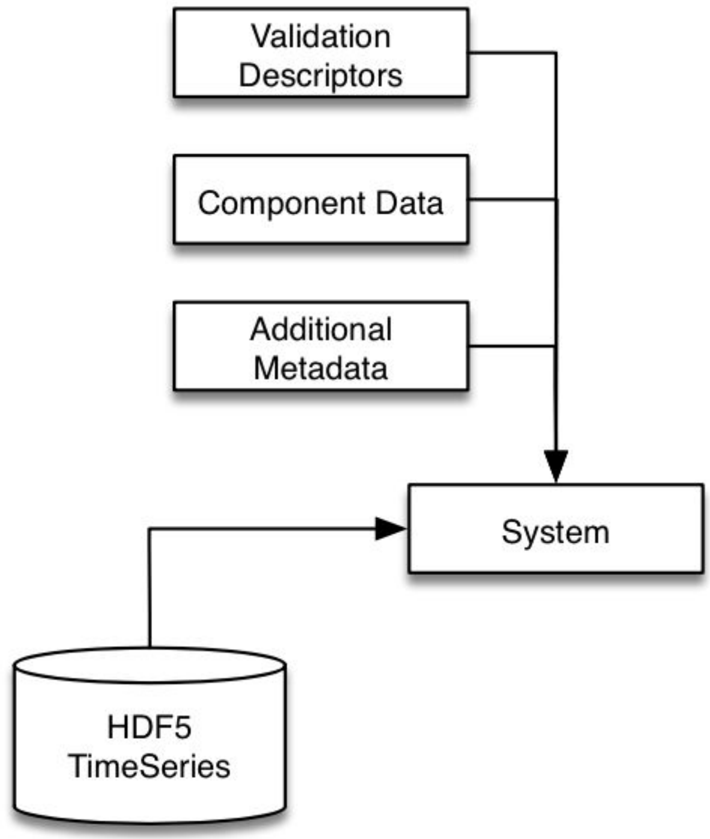
Integration Algorithm

DifferentialEquations.jl

- Implicit:
  - BDF (through Sundials.jl)
  - BDF method pure Julia
- Mass Matrix:
  - Rosenbrock Methods
  - Rosenbrock-W Methods
  - FIRK Methods
  - SDIRK Methods

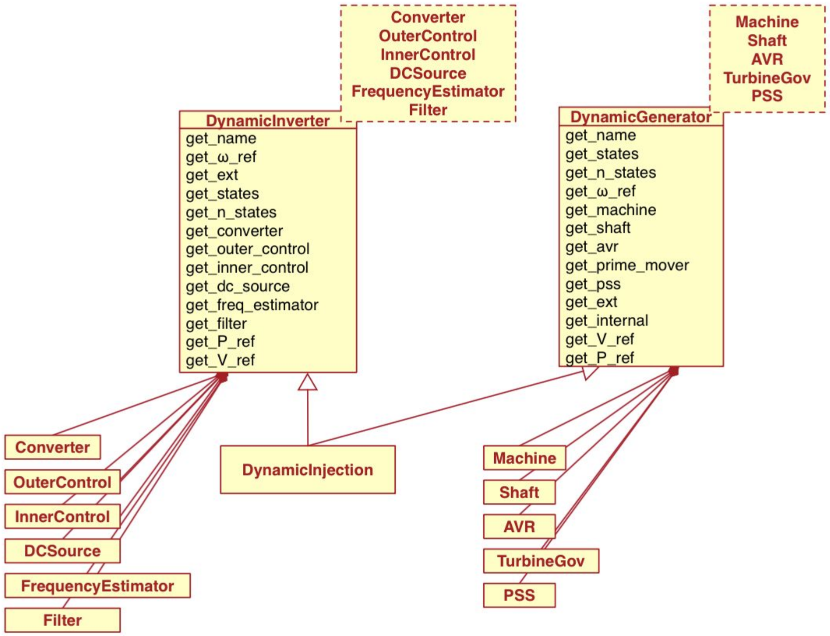
PowerSystems.jl

- PowerSystems.jl is a package to organize and manipulate data with diverse modeling requirements.
- Provides a generic data model for the simulations
- Implements the metamodels for machines and inverters.



PowerSystems.jl

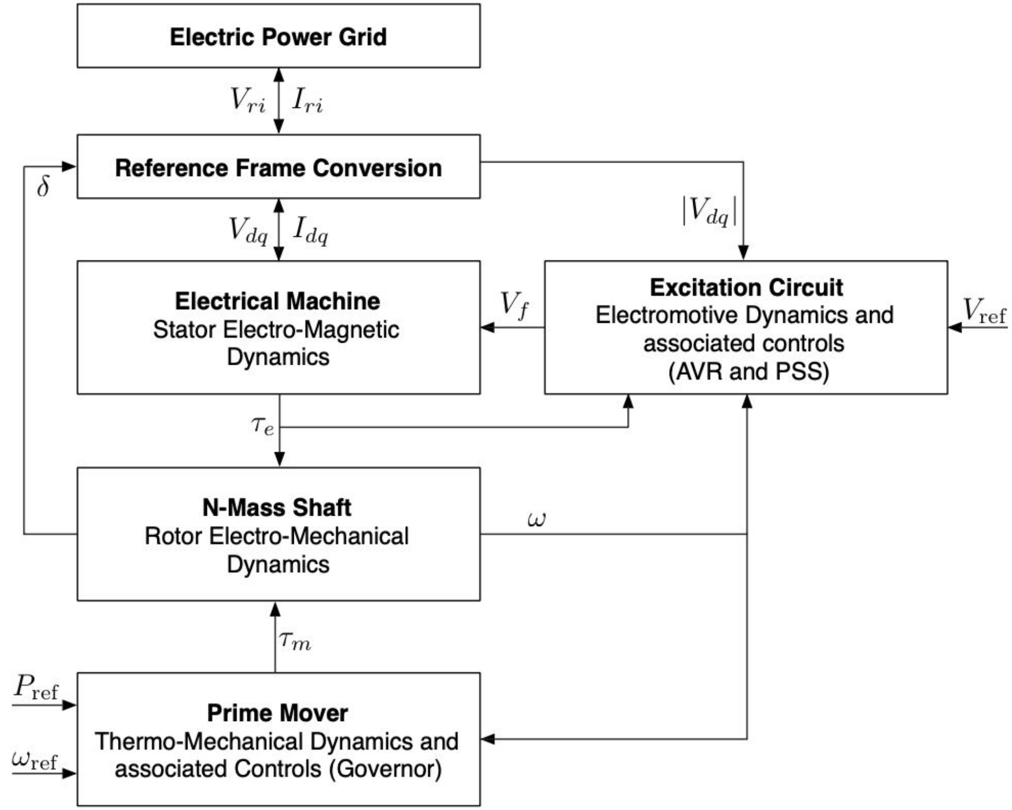
- PowerSystems.jl currently can parse pss/e dynamic data files.
- Once the system is read, it can be modified and serialized to disk with the modifications.
- Any data additions of modification can be recorded in reproducible scripts.



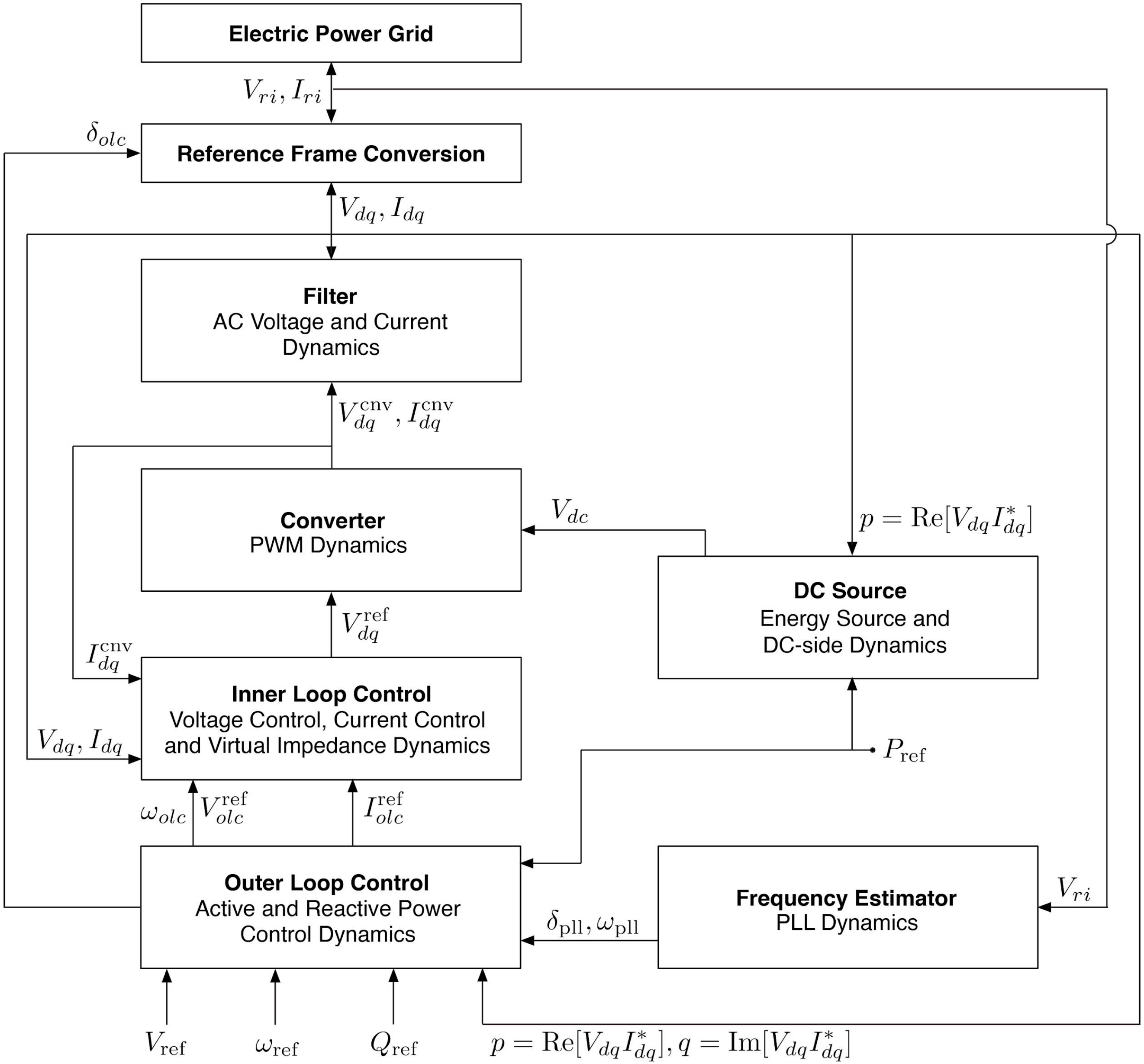
For additional details, check: https://github.com/NREL-SIIP/PowerSystems.jl

Modeling Strategy

- A key aspect is that `PSID` is driven by the data model in `PowerSystems.jl`.
- PSID automatically constructs DAEs from data model
- Can use metamodels for dynamic devices → allows `PSID` to construct mathematical models with different levels of stiffness



Modular Inverter "Meta" Model



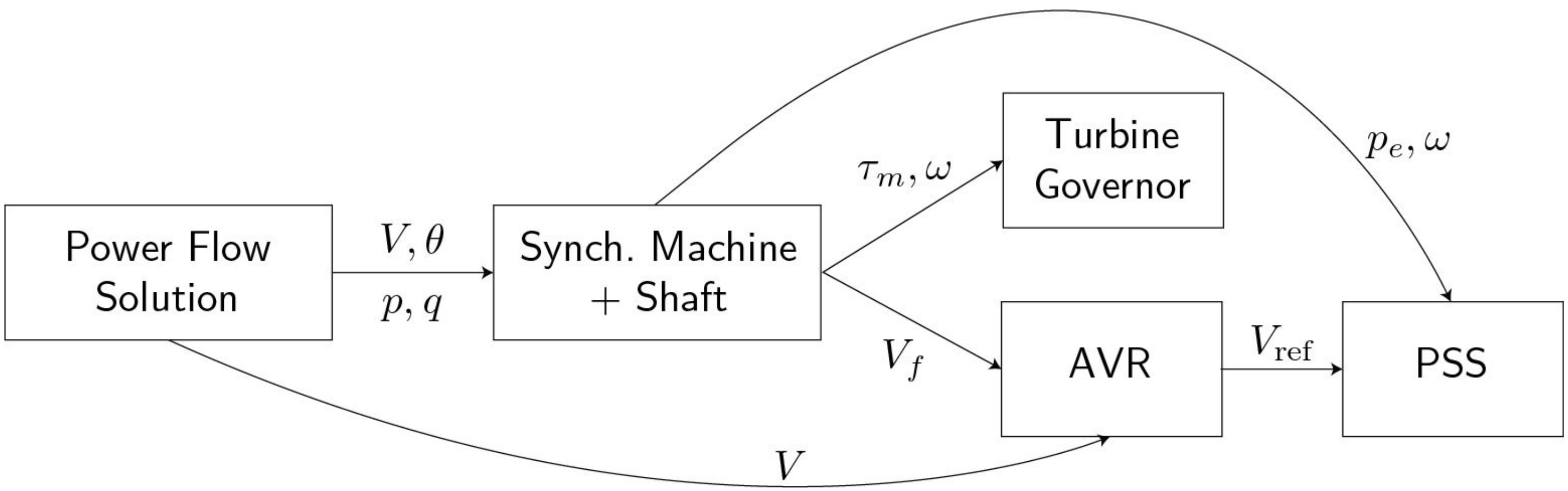
Running a simulation

- The simulation specification is based on methods defined to model each dynamic component.
- Reads directly the information from the data in the system.
- Define the perturbations into the system:
  - `NetworkSwitch`: Used for large network reconfigurations
  - `BranchTrip`: Trip a Line or Transformer in the system
  - `ControlReferenceChange`: Change the reference points on a device
- Define the timespan

Initialization Procedure

Initialization for each device

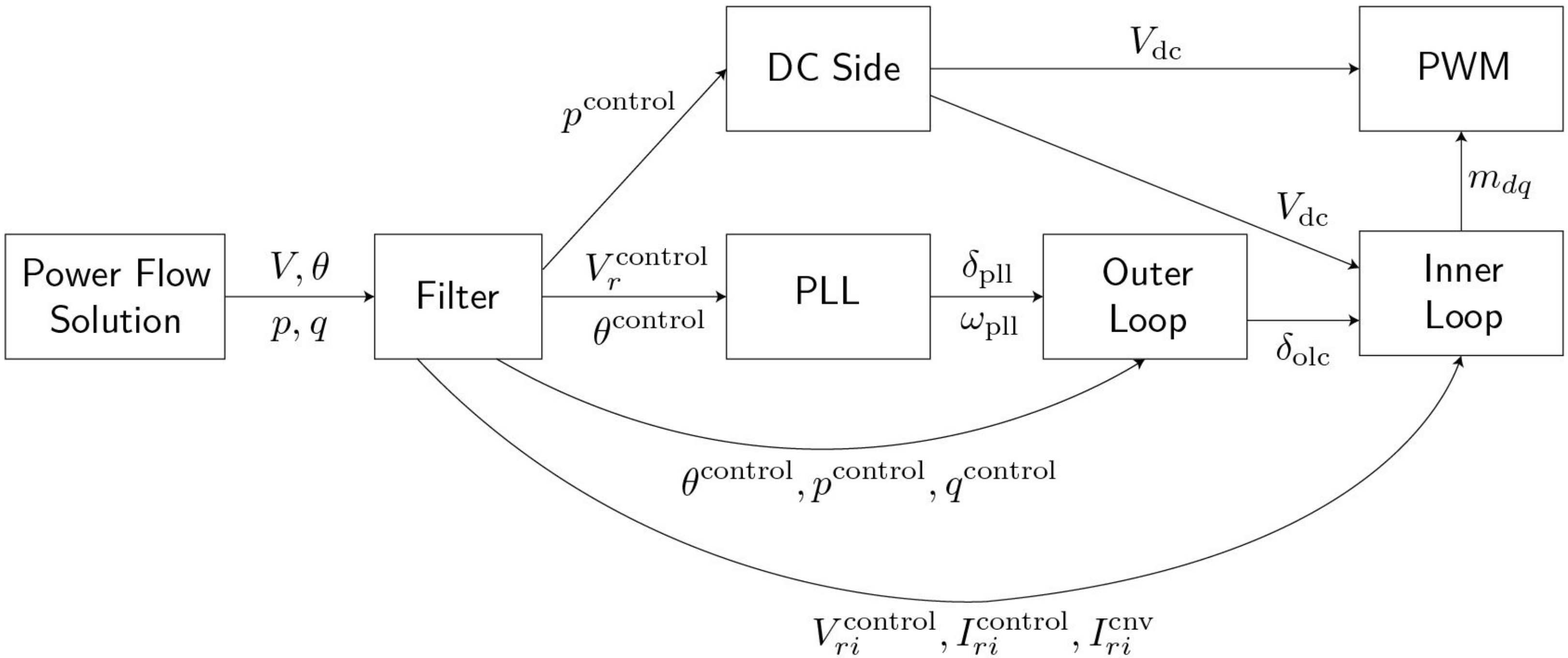
Generator:



Initialization Procedure

Initialization for each device

Inverter:



Software comparison

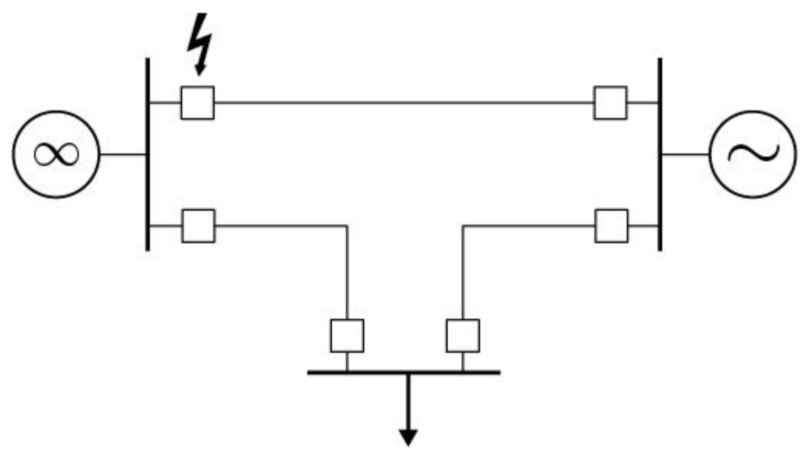
- We rely on computational tools to accurately represent power systems.
- Any new software must be validated against industry accepted software tools.
- Multiple options: PSS/E, PSLF, PSCAD, DIgSILENT, PowerWorld, EUROSTAG.
- Parser for PSS/E already available in PowerSystems.jl

T. Overbye, M. Venkatsubramanian "Validation and Accreditation of Transient Stability Results"
PSERC Publication 11-08, September 2011.

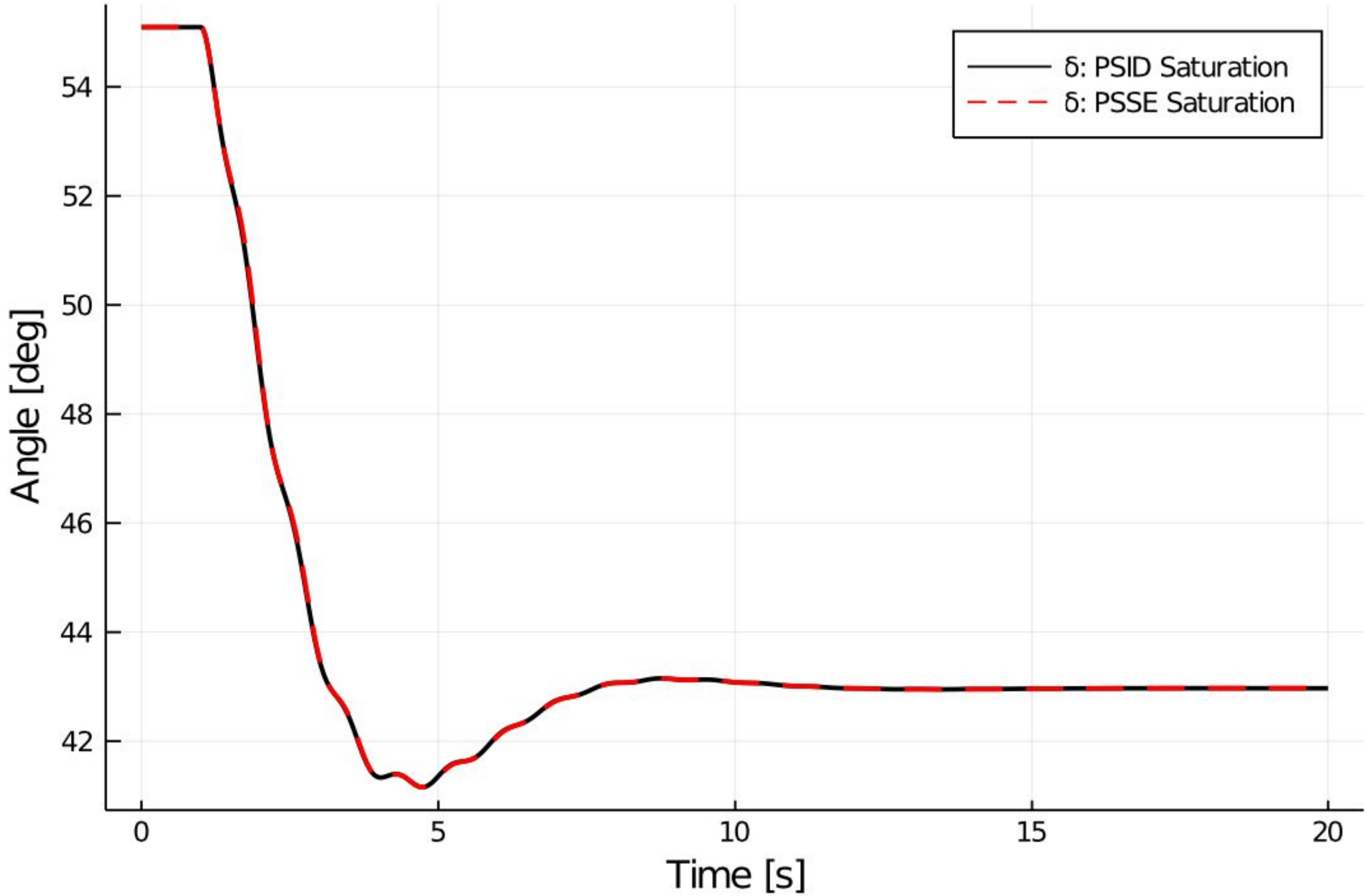
Validation PSID vs PSS/E

- PSID implementation based on DAE.
- PSS/E models implementation: block diagrams into differential equations.
- Different options for implementing saturation functions and anti-windup.
- Solver algorithms and tolerances will affect results
- Validation of each model via:
  - Operating point (steady-state initialization)
  - Transient simulation under a disturbance

- 3-bus test case:

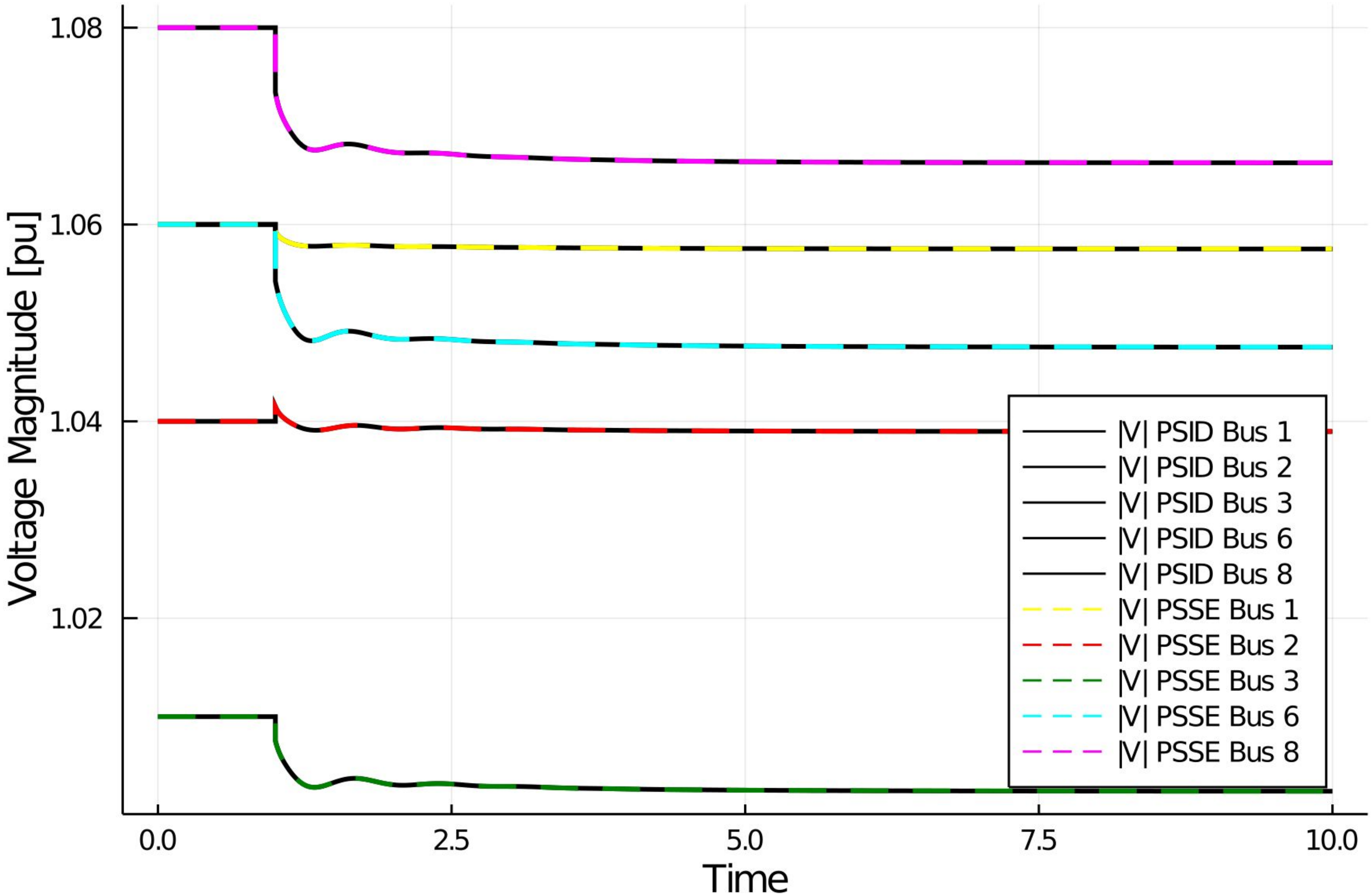


Validation PSID vs PSS/E



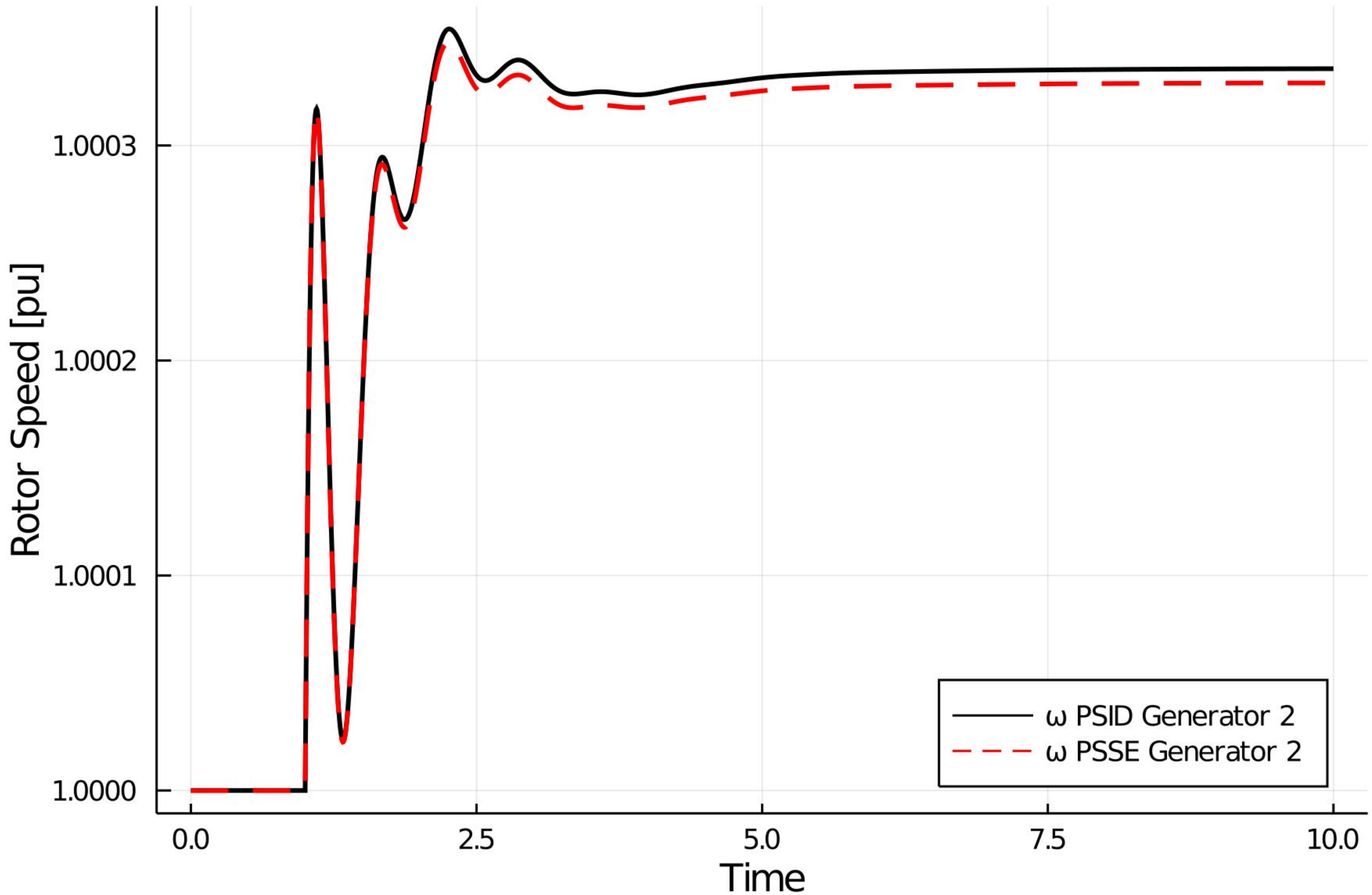
Validation GENROU + AC1A + TGOV1: 3 buses with IB

Validation PSID vs PSS/E



Validation: Voltages IEEE 14 Bus - GENROU - GAST

Validation PSID vs PSS/E



Validation: Speed IEEE 14 Bus - GENROU + GAST

Current Generator Modeling Capabilities

Machine	Shaft	Turbine Governor	AVR	PSS
Classic (GENCLS)	Single-Shaft	TGType I (PSAT)	Type II (PSAT)	SimplePSS (PSAT)
GENROU/E	Five-Mass-Shaft (PSAT)	TGType II (PSAT)	Type II (PSAT)	
GENSAL/E		TGOV1 (PSS/E)	ESAC1A (PSS/E)	
One d- One q-machine (PSAT)		GAST (PSS/E)		
Anderson-Fouad Simplified (PSAT)				
Anderson-Fouad (PSAT)				
Marconato Simplified (PSAT)				
Marconato (PSAT)				

Current CIG Modeling Capabilities

Filter	LCL (6-states)
Converter (PWM)	AverageDynamics
Inner Loop Control	Voltage/Current PI Controller (4-states)
Outer Loop Control	Virtual Inertia + QV droop (3-states)
Frequency Estimator	Kaura PLL (4-states)
DC Source	Fixed-DC Source

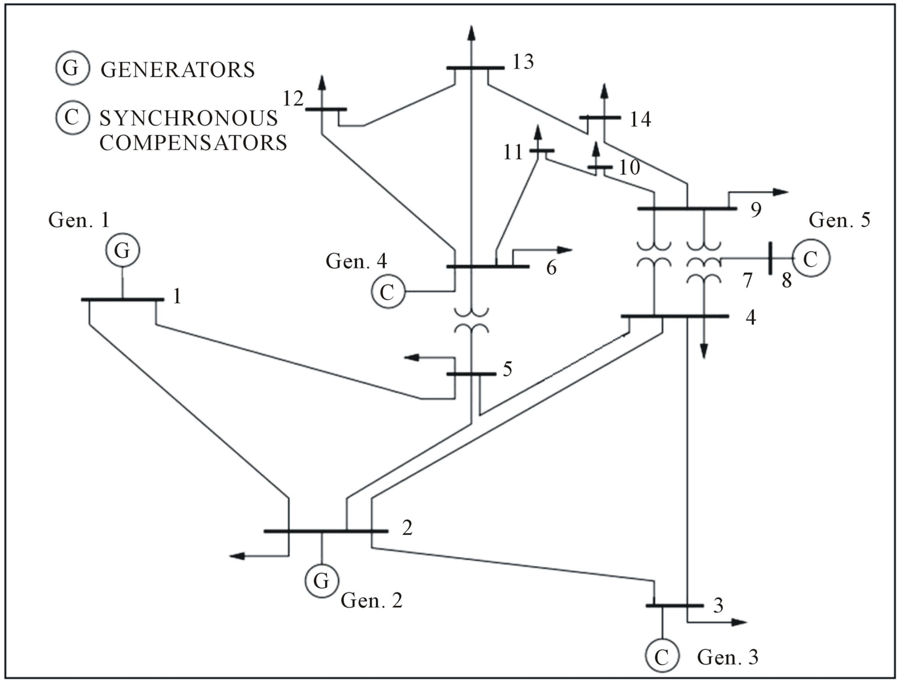
Inverter validation:

S. D'Arco, J. A. Suul and O. B. Fosso: "A Virtual Synchronous Machine implementation for distributed control of power converters in SmartGrids", *Electric Power Systems Research*, vo. 122, pp. 180-197, 2015.

Demonstration with 14-Bus System

- Gen1:
  - GENROU,
  - GAST,
  - ESAC1A

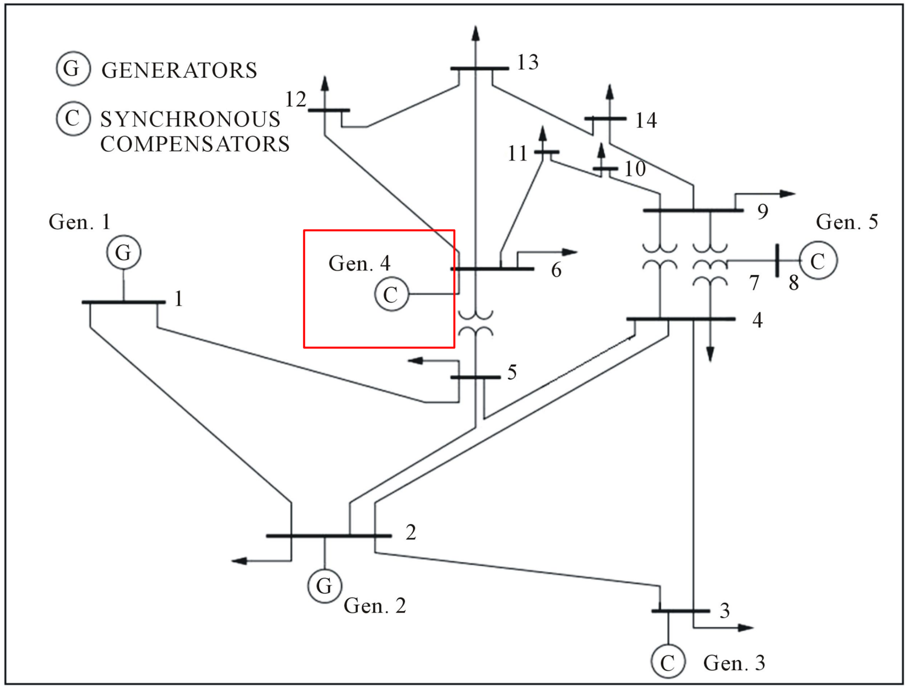
- Gens 2 - 5:
  - GENROU,
  - ESAC1A
  - Fixed Turbine Governor output



Follow along in https://bit.ly/3mckKUI

Demonstration with 14-Bus System

Remove Generator 4 and substitute with storage using a Virtual Synchronous Machine in the same bus.



To an example of code use ….

How to get involved

- PSID is functional, but still in development!
- We encourage you to:
  - use the tool and flag areas for improvement
  - take part in open source development
  - Join our slack channel (NREL-SIIP)

PowerSimulationsDynamics.jl

Search docs

Welcome Page

- Overview
- Installation
- Structure

Quick Start Guide

Tutorials

Models

Initialization

Small Signal

Reference Frames

Generator Component Library

- Machine
- Shaft
- AVR
- PSS
- Turbine and Governor

CIG Component Library

Welcome Page

Edit on GitHub

PowerSimulationsDynamics.jl

Overview

`PowerSimulationsDynamics.jl` is a `Julia` package for doing Power Systems Dynamic Modeling with Low Inertia Energy Sources.

The synchronous machine components supported here are based on commercial models and the academic components are derived from Power System Modelling and Scripting.

Inverter models support the model in "A Virtual Synchronous Machine implementation for distributed control of power converters in SmartGrids"

Installation

The latest stable release of PowerSimulationsDynamics can be installed using the Julia package manager with

```
] add PowerSimulationsDynamics
```

For the current development version, "checkout" this package with

```
] add PowerSimulationsDynamics#master
```

Structure

The following figure shows the interactions between `PowerSimulationsDynamics.jl`, `PowerSystems.jl`, `DifferentialEquations.jl` and the integrators. The architecture of `PowerSimulationsDynamics.jl` is such that the power system models are all self-contained and return the model function evaluations. The Jacobian is calculated through `DifferentialEquations.jl`'s common-interface enabling the use of any solver available in Julia. Considering that the resulting models are differential-algebraic equations (DAE), the implementation focuses on the use of implicit solvers, in particular SUNDIALS since it has exceptional features applicable to large models — for

README.md

PowerSimulationsDynamics.jl

Master - CI passing | codecov 94% | Documentation passing | slack @SIIP/PSID | DOI 10.5281/zenodo.4287804

`PowerSimulationsDynamics.jl` is a Julia package for power system modeling and simulation of Power Systems dynamics. The objectives of the package are:

- Provide a flexible modeling framework that can accommodate different device models according to modeling needs.
- Streamline the construction of large scale differential equations problems to avoid repetition of work when adding/modifying model details.
- Exploit Julia's capabilities to improve computational performance of large scale power system dynamic simulations.
- Provide State-of-Art modeling to assess Low-Inertia Power Systems.

Contributors 2

jd-lara Jose Daniel Lara

rodrigomha Rodrigo Henríquez-Au...

Languages

Julia 97.7% MATLAB 2.3%

What's next?

- Home-turf research:
  - DOE-sponsored collaboration on computing tools for accelerating simulation and learning regions of transient stability
  - Exploring standards for dynamic simulation with CIG
- Continued development of component libraries
- Expanded research community involvement

NREL Transforming ENERGY

Berkeley Lab

Berkeley UNIVERSITY OF CALIFORNIA

Acknowledgements to:

NREL SIIP LDRD, Led by Clayton Barrows

https://github.com/NREL-SIIP/PowerSimulationsDynamics.jl