`PowerSimulationsDynamics.jl` - An Open Source Modeling Package for Modern Power Systems with Inverter-Based Resources

Jose Daniel Lara, *Senior Member, IEEE,* Rodrigo Henriquez-Auba, *Member, IEEE,* Matthew Bossart, *Student Member, IEEE,* Duncan S. Callaway, *Member IEEE* and Clayton Barrows, *Senior Member, IEEE,*

***Abstract*—The inclusion of inverter-based resources from renewable energy creates new challenges for the stability and transient behavior of power systems which are best understood by studying their dynamic responses through simulation. In this paper, we develop an open source simulation toolbox, `PowerSimulationDynamics.jl`, to study the dynamic response of a balanced system with high penetration of inverter-based resources. `PowerSimulationDynamics.jl` is implemented in the Julia language and features a rich library of synchronous generator components and inverter models. In addition, it allows the study of both quasi-static phasors that employ an admittance matrix representation for the network and electromagnetic `dq` models that use a dynamic representation of the network. Case studies and validation exercises show that`PowerSimulationDynamics.jl` results closely match Quasi-Static Phasor (QSP) tools like Siemens PSS®E, ANDES, and wave-form Electro-magnetic Transient (EMT) simulations like PSCAD.**

***Index Terms*—Inverter-Based Resources, Low-Inertia Power Systems, Software Implementation, Positive Sequence, Electro-magnetic Simulations.**

I. Introduction

COMPUTATIONAL simulations have been the primary tool for the study of power systems dynamics and stability for many decades. The scale and complexity of interconnected power systems limits the application of analytical techniques for their analysis. Although there is a wealth of knowledge about models, simulation techniques, and software implementations [1], continuous advances in computer hardware, programming languages, and numerical integration techniques make the field of power system dynamic simulations a perpetually evolving area. The changing nature of the grid also catalyzes innovation in simulation models and techniques. Dynamic analyses of power systems relies on two methods to assess the system's dynamic behavior: (1) Time-domain simulations and (2) Small-signal Stability analysis.

More recently, with the addition of Inverter-based Resources (IBRs) into the electric system, new dynamics in the controls of IBRs are changing modeling requirements for system-wide stability studies that rely on time-domain simulations [2]–[4]. Further, Interactions at higher frequency, driven by the increasing reliance on IBRs, underscore the need to revisit simulation methods and models for such systems [3] and the implementation and application of simulation methods and models needs to be revisited for systems with significant penetration of IBRs.

The new control and analysis challenges that come with the integration of IBRs also bring modeling and solution difficulties for incumbent approaches, highlighting the need for fundamental changes in the domain of power system simulation. Practitioners in the power system community have reported limitations of existing commercial QSP tools for systems that have sizable IBR penetration and/or weak interconnections. For instance, an ERCOT report [5] highlights existing non-convergence and numerical instabilities in weak grid situations. QSP commercial tools rely on solution techniques and heuristics that have limitations when handling stiff Ordinary Differential Equation (ODE) systems which limits the level of detail in IBR models; they also restrict the possibility of accounting for fast network dynamics [6]. In cases where QSP simulations do not provide valid results, researchers and practitioners need to rely on point-on-wave EMT models that are computationally prohibitive for system-wide studies. Functionally, nearly all power system dynamic simulation packages integrate the modeling layer (where the behavior of system elements is specified) and the simulation layer (where the algorithmic solution approach is defined). In almost all cases, the details of these layers are not accessible to the developer and limit model development to the pre-defined structures of the solution algorithms available. The close integration between the algorithm and the models, combined with the development hurdles, licensing, and software limitations described above, restricts the scientific study of power systems with high IBR penetration.

On the other hand, the development of new algorithms or models requires bespoke dynamic simulation applications with a high setup cost and often will not support large-scale,

J. D. Lara, R. Henriquez-Auba and C. Barrows are with the Grid Planning and Analysis Center at the National Renewable Energy Laboratory.

M. Bossart is with the Department of Electrical, Computer, and Energy Engineering at the University of Colorado Boulder.

D. S. Callaway is with the Energy and Resources Group at the University of California Berkeley.

This work was supported by the National Renewable Energy Laboratory, operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308; and by the U.S. Department of Energy by the Advanced Grid Modeling program within the Office of Electricity, under contract DE-AC02-05CH1123; and by the Solar Energy Technologies Office through award 38637 (UNIFI Consortium). The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes.

reproducible experiments that are easily extensible with new models or methods. We note that there are important open-source efforts that have been used successfully by the research community. Notably, the Matlab-based library PSAT has been a pioneer in extensible dynamic modeling for power systems [7]. The Python-based tool ANDES [8] offers a modeling approach where a symbolic layer describes the components and a numeric layer performs vector-based numerical computations. Dynaωo is a hybrid C++/Modelica open-source suite of a simulation tool for power systems [9]. In the Julia ecosystem, `PowerDynamics.jl` [10] also uses symbolic representation of the dynamic models to provide a two-stage process of symbolic and compiled representation of a dynamic network similar to the approach used in ANDES. However, the level of customizability and modularity of open source tools has some limitations. In most cases implementing new models requires direct modification of the underlying simulation logic source code. Although there is support for component-level flexibility for generator models, there is a limited focus on inverters. Critically, existing libraries are deeply integrated with the solution methods and it very difficult to use them to explore integration algorithms.

A. Scope and Contributions

In this paper we describe a new modeling platform, `PowerSimulationsDynamics.jl` (`PSID.jl`), a Julia-based open-source package designed to provide the computational tools to develop models and algorithms to simulate systems with large penetration of IBRs. The tool has the flexibility to formulate balanced EMT models in dq and QSP modeling of the network and devices. To our knowledge, `PSID.jl` is the only open-source dynamics tool developed explicitly to enable extensibility at multiple points of the modeling stack from the component to the integration algorithm.

`PSID.jl` exploits Julia's *multiple dispatch* to support its modeling flexibility. The approach developed in this paper uses a data model for inverters and generators for component-level customization, and flexible ODE and Algebraic representation of the network circuit dynamics. The system model is also flexible enabling the usage of different numerical integration routines via `DifferentialEquations.jl` [11] common API. `PSID.jl` uses novel automatic differentiation techniques to calculate the Jacobians of existing and custom models without user input.

This work exploits the features of `PSID.jl` to implement new modeling and integration techniques to tackle the challenges in modeling systems with large penetrations of IBRs at large scale. The key contributions to the state of the art systems with large penetration of IBR modeling are:

- An implementation of IBR and machine models based on generic data models that supports developing encapsulated sub-component models. The modular design enables code and model reuse; this reduces development requirements and enables fast and simple prototyping of controls and models.
- `PSID.jl` is the first open-source tool to support model and integration algorithm decoupling that provides residual and mass matrix formulations of for the system model enabling the exploration of solution techniques.
- The network circuit model using dq enables the formulation of QSP and balanced EMT models for the study of system stability including fast dynamics and the exploration of stiff model solution integration algorithms in large models.

The rest of the paper is structured as follows: Section II discusses the simulation models implemented in `PSID.jl` including the residuals and mass-matrix formulations. Section III discusses the details of the software design and structure as well as the modeling details for generators, inverters, loads and network. The simulation case studies and verification results are shown in Section IV where we compare the results from `PSID.jl` with commercial tools to verify the validity of the results. Finally the conclusions are presented in Secion V.

II. Power System Models

To simulate an interconnected transmission level system comprised of a collection of balanced 3-phased interconnected components through RLC circuits, an initial-value-problem can be formulated as follows:

$$\frac{d\vec{x}}{dt} = F(\vec{x}, \vec{y}, \vec{\eta}), \quad \vec{x}(t_0) = \vec{x}^0, \tag{1a}$$

$$\frac{d\vec{y}}{dt} = G(\vec{x}, \vec{y}, \vec{\psi}), \quad \vec{y}(t_0) = \vec{y}^0, \tag{1b}$$

where $\vec{x}$ and $F(\cdot)$ represent the devices (e.g., IBRs, machines, loads) states and equations with $\vec{\eta}$ parameters. The circuit states and dynamics of the network are represented as the sub-system $\vec{y}$ and $G(\cdot)$ with network parameters $\vec{\psi}$. The general `PSID.jl` model is a standard current injection model; therefore, $\vec{y}$ represents the network voltages and currents. The numerical advantages of current injection models outweigh the complexities of implementing constant power loads for longer-term transient stability analysis and support the modeling of fast network dynamics [1]. The network is defined in a common Synchronous Reference Frame (SRF) rotating at frequency ω_{sys}, while each device is modeled in its own dq SRF.

Given the system model (1a)-(1b), a simulation can be defined as follows: given an initial condition for the device and network states $\vec{x}(t_0), \vec{y}(t_0)$ advance the solution in time t from one point to the next considering a discrete timeline $\{t_0, t_1, \ldots, t_n, \ldots, T\}$. A simulation requires a stepping algorithm that finds the solution at time t_{n+1} provided the values of the variables at $\{t_k | t_0 \leq t_k < t_{n+1}\}$.

A. Solution methods for dynamic simulations

Fully dynamic models of a power system tend to be large-scale and stiff due to the multi-rate nature of power systems dynamics. As a result, model (1a)-(1b) needs to be reformulated to employ appropriate integration algorithms that can find a solution to the system dynamics. Most commercial tools use Partitioned-Explicit solution methods and fixed time-stepping. In [12], B. Stott presents a detailed account of the techniques found in present-day commercial software that apply to the solution of the dynamical response of power systems. The use of partitioned methods has many merits from the computational performance perspective. However, it also has mathematical drawbacks that can lead to numerical instabilities [1]. Partition methods used by conventional transient stability simulators are inherently tied to the quasi-stationary assumption of the network variables, which imposes limitations on the modeling detail that can be included in a simulation.

Because the challenges of integrating IBRs stem from both interactions across time scales and interactions with line dynamics [13], `PSID.jl` implements a formulation suitable for the exploration of simultaneous solution methods. This design choice is geared towards the exploration of implicit solution methods that can handle unknown stiffness a priori, since explicit methods are not A-stable [14]. Further, the potential challenge of stiffness requires different approaches depending on the properties of the system being simulated – for example, stiffness arising from eigenvalues that exhibit large negative parts versus cases with large imaginary eigenvalues. As a result, a software package that relies on a single approach to stiffness in its solution method can limit the models and situations that can be simulated. The need to provide flexible formulations for power systems simulation models is described in detail in [15], where the author notes the flexibility and algorithmic improvements afforded by a semi-implicit formulation of the dynamic model. `PSID.jl` is able to formulate Differential Algebraic Equation (DAE) models of dynamical equations in two different ways which adapt to the requirements of implicit numerical methods using error control and variable step properties.

• **Residual Model:**[1]. This model is implemented for methods that find the solution to H$(t, z_t, z_t') = 0$ at each time step t. This formulation distinguishes between a subset of differential states z_d and algebraic states z_a, where the differential states are described by at least one derivative, resulting in the following system formulation:

$$\vec{r}_{x_d} = \frac{d\vec{x}_d}{dt} - F_d(\vec{x}_d, \vec{x}_a, \vec{y}_d, \vec{y}_a, \vec{\eta}) \tag{2a}$$

$$\vec{r}_{x_a} = F_a(\vec{x}_d, \vec{x}_a, \vec{y}_d, \vec{y}_a, \vec{\eta}) \tag{2b}$$

$$\vec{r}_{y_d} = \frac{d\vec{y}_d}{dt} - G_d(\vec{x}_d, \vec{x}_a, \vec{y}_d, \vec{y}_a, \vec{\psi}) \tag{2c}$$

$$\vec{r}_{y_a} = G_a(\vec{x}_d, \vec{x}_a, \vec{y}_d, \vec{y}_a, \vec{\psi}) \tag{2d}$$

where $\vec{x}_d$ and $\vec{x}_a$ are respectively the differential and algebraic states of the device and $\vec{y}_d$ and $\vec{y}_a$ are network differential and network algebraic counterparts. Functions F_d, F_a, G_d and G_a are the subsystems of equations that define the system of non-linear equations solved at each time step t. The terms $\vec{r}_{x_d}$, $\vec{r}_{x_a}$, $\vec{r}_{y_d}$, $\vec{r}_{y_a}$ correspond to the residuals of the non-linear system of equations.

It is important to note that models like (2a)-(2d) arise in power systems from the application of Singular Perturbation Theory (SPT) to (1a)-(1b) to reduce overall model stiffness. By selectively zero-out some of the differential terms and transform the model into a system of index-1 DAE, the "slow" states maintain their differential representation and the "fast" states are simplified into the algebraic terms. However, in `PSID.jl` there is no requirement that the separation between differential and algebraic states matches the slow and fast simplifications.

• **Mass Matrix Model:** This model is implemented for methods derived from the solution of mechanics problems where the differential terms z' are multiplied by constants. It represents system dynamics with the equation $Mz' = h(z)$. Although mass matrix models may have arbitrary structure, in `PSID.jl` the focus is on models where M is a diagonal matrix[2]. The resulting model is as follows:

$$M_x \frac{d\vec{x}}{dt} = F(\vec{x}, \vec{y}, \vec{\eta}) \tag{3a}$$

$$M_y \frac{d\vec{y}}{dt} = G(\vec{x}, \vec{y}, \vec{\psi}) \tag{3b}$$

where M_x corresponds to the mass matrix for the device states and M_y corresponds to the matrix for the network states. The diagonal elements of M_x are determined by the time constants of the device models and can be 0 if the dynamics of the state are not included, which is a common occurrence in large data sets where filtering dynamics are ignored. On other hand, since `PSID.jl` employs a current injection model, M_y has 0 when a line is modeled using an algebraic relationship or when the constant multiplying the node voltage derivative terms. Similar to the residual model, SPT can be used by setting the entries M diagonal to $\epsilon \to 0$ in (3a)-(3b) and analyzing the conditions under which those entries reduce model stiffness.

The flexibility in the model formulation enables the use of solvers that employ Backwards Differentiation Formula (BDF) and Backward Euler approaches as well as collocation algorithms derived from the Radau, Rosenbrock, and Rodas methods. This wide variety of solvers is enabled by `PSID.jl`'s integration with the `DifferentialEquations.jl` API [16], which supports a wide variety of solution methods for both residual and mass-matrix formulations of the simulation model. Integrating the simulation model through a generic solver API allows the `PSID.jl` framework to be used to formulate large-scale stiff power systems simulation models. This in turn supports the development of novel integration algorithms and improvements in existing methods.

III. Software design and structure

`PSID.jl` needs to support two major use cases: 1) package users that want to develop scripts to perform analysis and 2) users that want to develop new device or branch models. The usability objectives require that `PSID.jl` handles the routines for initialization, Jacobian calculation, interface with `DifferentialEquations.jl` execution and post-processing.

Figure 1 shows the relationships with the different packages used in `PSID.jl`. The system model is used to get the Jacobian using `ForwardDiff.jl` [17] and also used to find the initial conditions. Finally the Jacobian and the initial conditions are interfaced into the integrator through

[1]In the differential equations literature, this model is also named *semi-explicit* system

[2]These classes of mass matrix models are also named *Lumped mass matrix* models

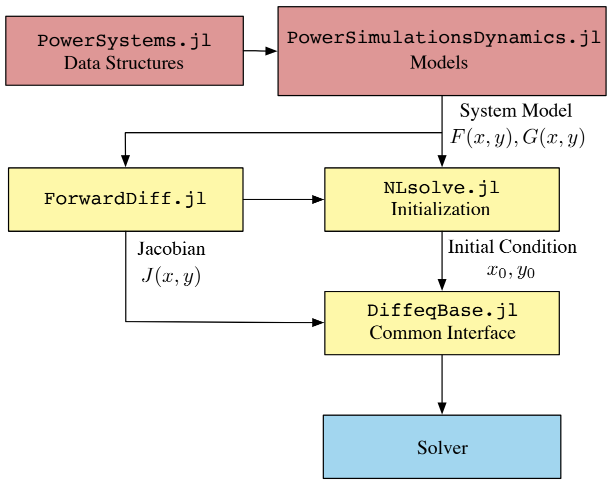

Fig. 1. Software dependencies in `PSID.jl`.

the common interface. The forthcoming sections describe how the flow in Fig. 1 maintains the objectives of flexibility and scalability.

Our choice of Julia as the software development environment is central to enabling many of the desired features of of a power systems simulation software as described in [1]. Julia is a scripting language like Python and Matlab, but offers the performance one would associate with low-level compiled languages [6]. Julia's support of multiple dispatch and composition allows us to design a software and model library that is computationally efficient yet easy to use and extend through method overloading [18]. Julia's multiple dispatch is particularly useful for mathematical modeling since methods can be defined based on abstract data structures to enable code reuse and easy interfacing with existing models [19]. `PSID.jl` evaluates different models by selecting the appropriate method version based on the signature of arguments passed into the function. This approach is different from traditional scripting languages, where dispatch is based on a special argument syntax and is sometimes implied rather than explicitly written.

A. System model implementation

Equations (2a)-(2d) and (3a)-(3b) are implemented in the software as a Julia structure with pre-allocated vectors to perform in-place updates of $\frac{d\vec{x}}{dt}$, $\frac{d\vec{y}}{dt}$, $\vec{x}$, $\vec{y}$. In addition, a vector $\vec{a}$ is used for intermediate inner variables when it is necessary to share information between the components of a model. This design avoids the unnecessary addition of algebraic equations to the state vector, which is a critical consideration since the computational complexity of an implicit method is dominated by the linear solution method. The linear solution method for its part is a function of the required precision and the number of variables with an upper bound $O(n^3)$ [20]. The system model also keeps track of global variables to which all models have visibility; one example of this is the system frequency ω_{sys} variable.

These vectors are stored in a system container to be used as caches. During the simulation build, these caches collect each device together with its states, inner variables, and bus location

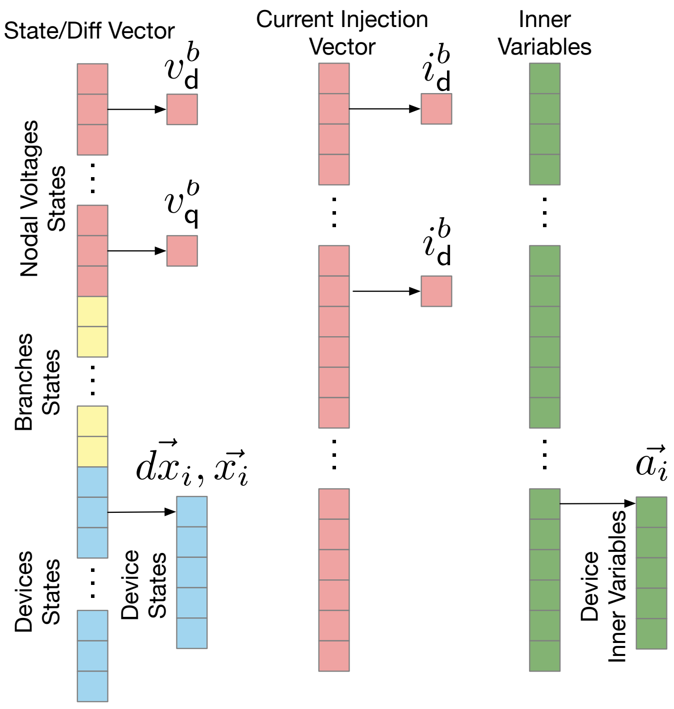

Fig. 2. Implementation of the state space indexing.

to generate an index of the cached objects. The index is a tree structure, with the device on the first level and the component on the second level. Figure 2 shows the organization of the internal cached vectors in the simulation struct.

B. Auto-Differentiation (AD) of Jacobians

Performant Jacobian computations are critical requirement when employing implicit solution methods for stiff DAE systems since it is used in the non-linear solve and to implement adaptive timestepping techniques. Further, there is a wealth of sparsity techniques to reduce the computational cost of Jacobian evaluations that present opportunities to tackle the challenges of simulating systems with IBRs. Computing Jacobians numerically is a large area of numerical methods research that has exploded over the last two decades with the demands in applications like machine learning and optimization [21]. Computing the Jacobian function requires deriving the matrix function:

$$J(\vec{x},\vec{y},\vec{\eta},\vec{\psi}) = \begin{bmatrix} \frac{\partial}{\partial \vec{x}}F(\vec{x},\vec{y},\vec{\eta}) & \frac{\partial}{\partial \vec{y}}F(\vec{x},\vec{y},\vec{\eta}) \\ \frac{\partial}{\partial \vec{x}}G(\vec{x},\vec{y},\vec{\psi}) & \frac{\partial}{\partial \vec{y}}G(\vec{x},\vec{y},\vec{\psi}) \end{bmatrix} \tag{4}$$

The flexibility afforded by `PSID.jl` makes it challenging to know a priori the potential structure of (4). Tools that use symbolic layers such as ANDES can exploit symbolic operation libraries to calculate the Jacobian expressions. However, symbolic differentiation isn't always effective since the length of the expressions grows exponentially.

In `PSID.jl`, the approach taken is to employ modern AD techniques using `ForwardDiff.jl` [17] as the backend. `ForwardDiff.jl` employs a dual-number approach the computation of vector Jacobians. Since the Jacobians of dynamic simulation models are square, forward-differentiation is also the most efficient method for AD since forward AD computes an Jacobian column-wise, whereas reverse AD schemes use row-wise computations [22].

Providing the users with performant Jacobian Matrix function evaluations using AD techniques that rely on dual number implementation is that the caching design specified in Section III-A needs to be implemented for `Float` and `Dual` number types. Further, since dynamic power systems problems result in sparse Jacobian matrices, it it possible to generate and cache the Jacobian matrix and provide an in-place calculation of the system model Jacobian for the integration process. Since the Jacobian is obtained from the indexed system model described in Section III it is possible to map the entries of the AD-derived Jacobian directly to the states to perform analysis the reductions necessary for small signal stability.

1) Small Signal stability analysis: Take the residual formulation (2a)-(2d) from dynamic components $\vec{x} := (\vec{x}_d^\top, \vec{y}_d^\top)$ and algebraic variables $\vec{y} := (\vec{x}_a^\top, \vec{y}_a^\top)$. These are used to define $S(\cdot) := (F_a(\cdot)^\top, G_a(\cdot)^\top)$ as the vector of differential equations and $R(\cdot) := (F_d(\cdot)^\top, G_d(\cdot)^\top)$ as the vector of algebraic equations of the entire system. Setting the residuals to zero, the resulting non-linear differential algebraic system of equations can be written as:

$$\begin{bmatrix} \frac{d}{dt}\vec{x} \\ 0 \end{bmatrix} = \begin{bmatrix} S(\vec{x},\vec{y}) \\ R(\vec{x},\vec{y}) \end{bmatrix} \tag{5}$$

We are interested in the stability around an equilibrium point $\vec{x}_{eq}, \vec{y}_{eq}$ that satisfies $d\vec{x}/dt = 0$, or equivalently $S(\vec{x}_{eq}, \vec{y}_{eq}) = 0$, while satisfying $R(\vec{x}_{eq}, \vec{y}_{eq}) = 0$. To do that we use a first order approximation:

$$\begin{bmatrix} \frac{d}{dt}\Delta\vec{x} \\ 0 \end{bmatrix} = \underbrace{\begin{bmatrix} S(\vec{x}_{eq},\vec{y}_{eq}) \\ R(\vec{x}_{eq},\vec{y}_{eq}) \end{bmatrix}}_{=0} + J[\vec{x}_{eq},\vec{y}_{eq}] \begin{bmatrix} \Delta\vec{x} \\ \Delta\vec{y} \end{bmatrix} \tag{6}$$

The Jacobian matrix $J[\vec{x}_{eq}, \vec{y}_{eq}]$ can be split in four blocks depending on the specific variables and functions modeled:

$$J[\vec{x}_{eq},\vec{y}_{eq}] = \begin{bmatrix} S_x & S_y \\ R_x & R_y \end{bmatrix}. \tag{7}$$

Assuming that R_y is not singular we can eliminate the algebraic variables to obtain the reduced Jacobian:

$$J_{\text{red}} = S_x - S_y R_y^{-1} R_x \;\rightarrow\; \frac{d}{dt}\Delta\vec{x} = J_{\text{red}}\Delta\vec{x} \tag{8}$$

that defines our reduced system for the differential variables on which we can compute its eigenvalues to analyze local stability.

C. Injection device modeling

`PSID.jl` employs the data structures and type hierarchy from `PowerSystems.jl` [23] to dispatch into the model functions. Generators and inverters are defined as a composition of components (see Fig. 3). This design enables the interoperability of components within a generic device container. As a result, it is possible to implement custom component models and interface them with other existing components and models by just overloading the `device!`[3] method. Whenever

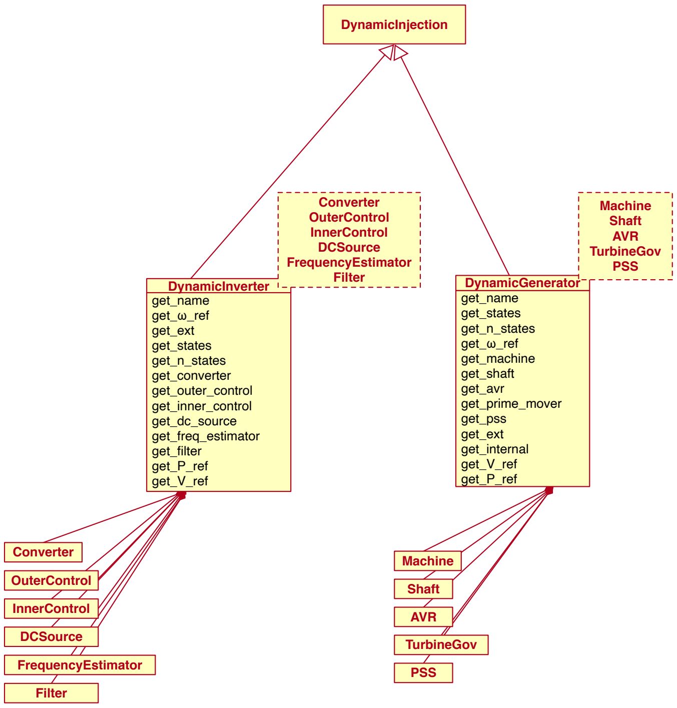

Fig. 3. `DynamicInjection` data structures from `PowerSystems.jl`.

```
struct MyCustomDevice <: DynamicInjection
    field1
    field2
end

function device!(
    states::AbstractArray{T},           # x_i
    diff::AbstractArray{T},             # dx_i
    v_r::T,
    v_i::T,
    i_r::AbstractArray{T},
    i_i::AbstractArray{T},
    global_vars::AbstractArray{T},
    inner_vars::AbstractArray{T},       # a_i
    device::MyCustomDevice,
    t,
) where {T <: ACCEPTED_REAL_TYPES}
    # Update diff vector
    diff[1] = f_1(states, v_r, v_i,...)
    diff[2] = f_2(states, v_r, v_i,...)

    # Update inner vars
    inner_vars[1] = inner_var_1(states, v_r, v_i,...)
    inner_vars[2] = inner_var_2(states, v_r, v_i,...)

    # add to the bus currents
    i_r += current_r(states, v_r, v_i,...)
    i_i += current_i(states, v_r, v_i,...)
    return
end
```

Listing 1: Injection device method prototype

the function iterates over the vector containing the entire state space, only the relevant portions are accessed by `device!`. Listing 1 shows a prototype of method overload required to develop a model for a `MyCustomDevice`. The method `device!()` is dispatched based on the type of the `device` field which is defined the user. At the device modeling layer, the sub-components from the meta-models share local

[3]In Julia's diction, functions that mutate arguments have a !

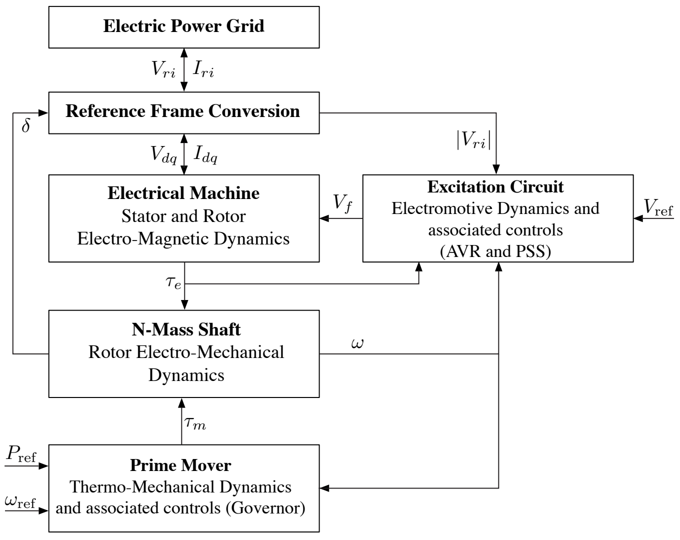

Fig. 4. Generator metamodel.

information internal to each device. For instance, in an IBR model, the current reference variable i_{olc}^{ref} from the outer loop needs to shared with the inner loop control and in a generator model the mechanical torque variable τ_m is required by the shaft model. This design further modularizes the implementation of the sub-components within an injection device. Intra-device information passing is standardized via port variables that enable efficient information passing between component modeling functions. The arrows in Figs. 5 and 4 showcase the port variables and their source-destination relationships for the inverter and generator respectively.

1) Generator Modeling: Generators models follow common practices well established in the literature [1], [24]. The machine model in `PSID.jl` benefits from the existing standardization in generator models already implemented in most software solutions based on the underlying physics of the generator. Figure 4 depicts component levels used in `PSID.jl` to describe synchronous generators where each generator is defined by a machine model, an excitation circuit and associated controls (Automatic Voltage Regulator (AVR) and Power System Stabilizer (PSS)), a shaft model, and a prime mover. The metamodel in Fig. 4 also allows the implementation of the initialization routine for generators described in [1].

2) Inverter Modeling: IBR simulation models are structured according to the cascaded control architectures, and as a result there is less standardization on the simulation models. Figure 5 depicts the relationships for `DynamicInverter` model as implemented in `PSID.jl`. The sub-component separation and variable sharing in the inverter is able to support both grid-following industrial models commonly used in QSP[4] and grid-following models as well as more advanced control architectures like Virtual Synchronous Machine (VSM) or droop control. Each inverter is defined by a filter, converter

[4]The adaptation of industrial models to the proposed meta model is discussed in detail in https://github.com/NREL-Sienna/PowerSystems.jl/discussions/767

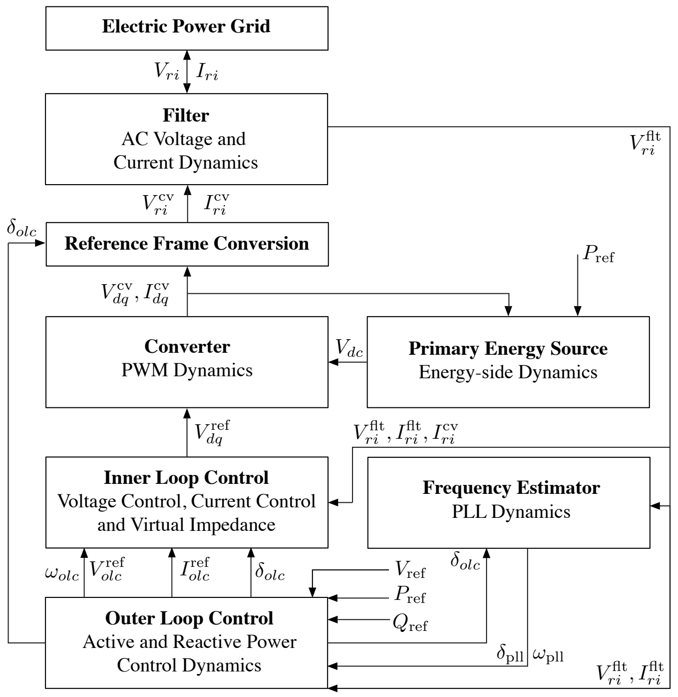
Electric Power Grid
V_{ri} I_{ri}
Filter
AC Voltage and Current Dynamics
V_{ri}^{flt}
V_{ri}^{cv} I_{ri}^{cv}
δ_{olc}
Reference Frame Conversion
P_{ref}
$V_{dq}^{\text{cv}}, I_{dq}^{\text{cv}}$
Converter
PWM Dynamics
V_{dc}
Primary Energy Source
Energy-side Dynamics
V_{dq}^{ref}
$V_{ri}^{\text{flt}}, I_{ri}^{\text{flt}}, I_{ri}^{\text{cv}}$
Inner Loop Control
Voltage Control, Current Control and Virtual Impedance
Frequency Estimator
PLL Dynamics
ω_{olc} V_{olc}^{ref}
I_{olc}^{ref}
δ_{olc}
V_{ref}
P_{ref}
Q_{ref}
δ_{olc}
Outer Loop Control
Active and Reactive Power Control Dynamics
δ_{pll} ω_{pll}
$V_{ri}^{\text{flt}}, I_{ri}^{\text{flt}}$

Fig. 5. Inverter metamodel.

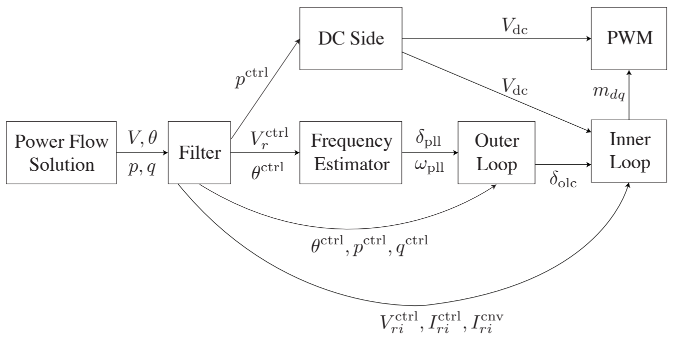

Fig. 6. Inverter initialization model.

model, inner loop control, outer loop control, frequency estimator (typically models employ a Phase-locked Loop (PLL) although is not required in all applications), and a primary energy source model. The proposed component decomposition deviates from the one implemented for QSP modeling, which distinguishes between three main modeling blocks: Generator, Electrical, and Plant-level controls. However, the default QSP structure cannot integrate frequency measurement modeling and integration with additional control models like VSM. Further, the representation of the filter in QSP is limited to an algebraic representation due the nature of the simulation tools which restricts the implementation of the model in EMT settings. Since there is no well established initialization methods for generic inverter models, `PSID.jl` employs the sequence described in Fig. 6 to initialize the inverter states. The initialization routine generalizes to any inverter implemented in a compatible fashion with the metamodel.

D. Load Models

`PSID.jl` supports the two major static load models: ZIP and Exponential. Load models are implemented by aggregating the individual loads located at a bus b and using the network SRF. Each aggregated load lumps all the currents from the static loads using a rectangular model. All the load models follow the same methods as the injection devices by updating the total current at each bus b. For example, the total load currents in a bus with ZIP loads is implemented as follows:

$$i^b_{\mathsf{d}} = \frac{1}{|v^b||v^b_0|^2} \sum_{l\in\mathcal{L}^b_i} |v^b_0| \left(p_l v^b_{\mathsf{d}} + q_l v^b_{\mathsf{q}}\right) + \sum_{l\in\mathcal{L}^b_p} |v^b_0|^2 \left(p_l v^b_{\mathsf{d}} + q_l v^b_{\mathsf{q}}\right) + \sum_{l\in\mathcal{L}^b_z} |v^b| \left(p_l v^b_{\mathsf{d}} + q_l v^b_{\mathsf{q}}\right) \tag{9a}$$

$$i^b_{\mathsf{q}} = \frac{1}{|v^b||v^b_0|^2} \sum_{l\in\mathcal{L}^b_i} |v^b_0| \left(p_l v^b_{\mathsf{q}} - q_l v^b_{\mathsf{d}}\right) + \sum_{l\in\mathcal{L}^b_p} |v^b_0|^2 \left(p_l v^b_{\mathsf{q}} - q_l v^b_{\mathsf{d}}\right) + \sum_{l\in\mathcal{L}^b_z} |v^b| \left(p_l v^b_{\mathsf{q}} - q_l v^b_{\mathsf{d}}\right) \tag{9b}$$

where $\mathcal{L}^b_p$, $\mathcal{L}^b_i$, $\mathcal{L}^b_z$ are the sets of constant power, constant current and constant impedance loads at each bus b respectively. v^b_0, p_l, q_l are the bus voltage and load's power from the powerflow used to estimate the total load currents i^b_{d} and i^b_{q} at the bus.

In addition to the static load models, two models of dynamic loads are available: 5- and 3-state induction machine models.

E. Network Modeling

`PSID.jl` implements a lumped parameter π-circuit of a transmission line in a dq reference frame. This general representation of the network can capture the circuit network dynamics depending on requirements for the study. Further, it enables the modeler to selectively choose which specific branches should be modeled including circuit dynamics. The model is implemented by splitting the real and imaginary portions of the network quantities using a *rectangular* representation as follows:

$$\frac{1}{\Omega_b}\begin{bmatrix} L & 0 \\ 0 & L \end{bmatrix} \frac{d}{dt}\begin{bmatrix} \vec{i}^{\ell}_{\mathsf{d}} \\ \vec{i}^{\ell}_{\mathsf{q}} \end{bmatrix} = E_\ell \begin{bmatrix} \vec{v}_{\mathsf{d}} \\ \vec{v}_{\mathsf{q}} \end{bmatrix} - \begin{bmatrix} R & -L \\ L & R \end{bmatrix} \begin{bmatrix} \vec{i}^{\ell}_{\mathsf{d}} \\ \vec{i}^{\ell}_{\mathsf{q}} \end{bmatrix} \tag{10}$$

$$\frac{1}{\Omega_b}\begin{bmatrix} \frac{1}{B} & 0 \\ 0 & \frac{1}{B} \end{bmatrix} \frac{d}{dt}\begin{bmatrix} \vec{v}^{b}_{\mathsf{d}} \\ \vec{v}^{b}_{\mathsf{q}} \end{bmatrix} = \begin{bmatrix} \vec{i}^{b}_{\mathsf{d}} \\ \vec{i}^{b}_{\mathsf{q}} \end{bmatrix} - \begin{bmatrix} \Re(Y_a) & \Im(Y_a) \\ -\Im(Y_a) & \Re(Y_a) \end{bmatrix} \begin{bmatrix} \vec{v}^{b}_{\mathsf{d}} \\ \vec{v}^{b}_{\mathsf{q}} \end{bmatrix} \tag{11}$$

where (10) represents the Kirchoff Voltage Laws across the series element of a branch, and is used to evolve the electric current of branches modeled dynamically. E_ℓ is the rectangular incidence matrix that returns the difference between the sending and receiving buses, and R and L are the the series resistance and inductance matrices of each dynamic branch.

Equation (11) represent Kirchoff Current Laws across the network (for each bus), where $\vec{i}^b_{\mathsf{d}}$ and $\vec{i}^b_{\mathsf{q}}$ are the total real (d-axis) and imaginary current (q-axis) injections from devices and dynamic branches at the bus b. The admittance matrix Y_a corresponds to a modified admittance matrix without the series elements of the branches modeled dynamically. The blocks $1/B$ on the left-hand side corresponds to the total lump capacitance at the nodes. The classical network model in QSP models employ SPT to eliminate the differential terms since the terms $\ell/\Omega_b, c/\Omega_b \approx 10^{-3}$ have small values in most systems operating at 50 or 60 Hz. If the model doesn't include dynamics lines, the left-hand-side of (11) is zero yielding the equivalent QSP network representation $0 = V - \boldsymbol{Y}I$.

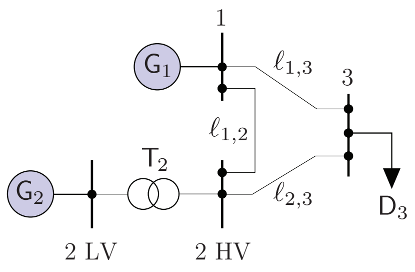

Fig. 7. Three-bus high-voltage (HV) network system for case studies.

Since the square matrices on the left and right hand side are sparse and have the same pattern in each block, `PSID.jl` implements a sparse matrix vector multiplication using the `nzrange` to reduce computational effort. Note that effectively, the matrices on the left hand side of the equations is M_y, the mass matrix in equations (3b).

F. Perturbations

Solver-agnostic perturbations are a major challenge in the development of simulation software because the re-initialization procedure can be different depending on both the solver and whether the perturbation introduces a change in the algebraic versus the dynamic portions of the model. `PSID.jl` simplifies the implementation of perturbations (e.g., line faults and step changes) through the use of callbacks to the integrator.

Callbacks enable controlling the solution flow and intermediate initializations without requiring simulation re-starts and other heuristics typically used in power systems dynamic modeling. `PSID.jl` includes built-in features to manipulate perturbation objects that internally define the solver callbacks. Modelers can extend the perturbation library by defining custom callback structures which allows further flexibility in the type of modeling.

IV. Simulation Validation and Case Studies

This section presents comparisons of simulation using `PSID.jl`, the QSP simulator PSS®E, and the waveform simulation EMT platform PSCAD. We also show the verification results of a large QSP case with large penetration of IBRs and small-signal capabilities using the results as published in [8]. All the simulation code and results is available at https://github.com/NREL-Sienna/PSIDValidation. Listing 2 shows the `PSID.jl` code used to run the simulations with Residual with the Sundials solver `IDA()`.

A. Four-bus network case

For validation purposes, the 3-bus HV network system (4-bus) depicted in Figure 7 will be used to analyze the dynamic

```julia
using PowerSystems
using PowerSimulationsDynamics
using Sundials

sys = System("system_data.json")

sim = Simulation(
    ResidualModel, # model type
    sys, # system
    pwd(), # directory
    (0.0, 20.0), # time span
    BranchTrip(1.0, Line, "BUS 1-BUS 2 HV-i_1"), # fault
)

execute!(sim, IDA(), abstol = 1e-10)
results = read_results(sim)
```

Listing 2: Example of setting up a simulation

TABLE I
MODELS USED FOR 4-BUS QSP VALIDATION

Generator	Machine	Excitation	Governor	PSS
1 - G1	GENROU	None	None	
2 - EG/SG/RG	GENROU	SEXS	GAST	
2 - ND	GENROU	SEXS	TGOV1	
2 - WG	GENROU	SEXS	GAST	IEEEST
2 - SH	GENROU	SEXS	HYGOV	
Inverter	**Outer**	**Inner**	**Converter**	**Filter Freq. Est.**
2 - S/SW	REPCA1	REECB1	REGCA1	None

response of PSID.jl against other simulation tools. The definitions of G_1 and G_2 will change according to the study under consideration.

1) QSP simulation: In this case, G_1 represents a large grid connection using a GENROU machine model without additional controllers and G_2 represents a collection of 8 generation sources, including industrial grid following IBRs. The models used are summarized in Table I. This simulation is performed using a residual formulation with the solver IDA and abstol $= 10^{-10}$.

To perform the validation, we run two perturbations 1) a generation trip of inverter Bus2-S at $t = 1$s to assess the active power balancing and generator speed modeling and 2) a trip of the line between buses 1 and 3 to assess the voltage control and generator electromechanical modeling. Figure 9 presents the rotor speed for the generation units and Fig. 8 depicts the active power output of the generation units which are able to recover the lost power and output enough additional generation to prevent a frequency collapse. Figures 10 and 11 showcases the voltage and field current of the generators after the line trip which follow precisely the trace of the benchmark solution. Note that the field current is not a state in the GENROU model but rather a variable derived from several internal states, highlighting the accuracy of PSID.jl.

B. Balanced EMT simulation

To validate the dq -EMT simulation capabilities, we conduct a waveform simulation of the system in Fig. 7 using a π-line and substituting G_1 with an inverter featuring VSM control [25] and G_2 with a droop-control [26]. The control blocks of the inverters are implemented as custom PSCAD components

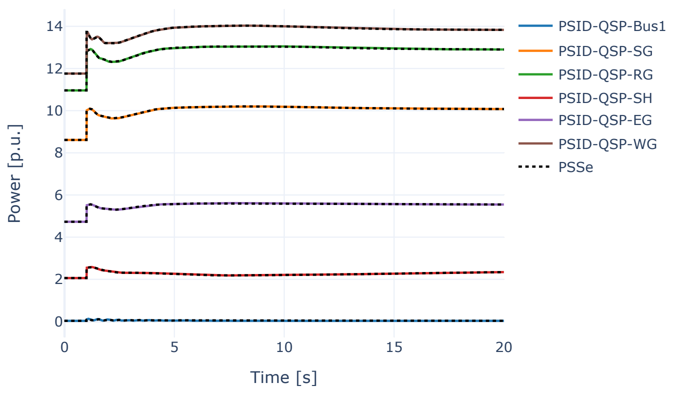

Fig. 8. Power output after Gen Bus2-S trip case 4-buses QSP

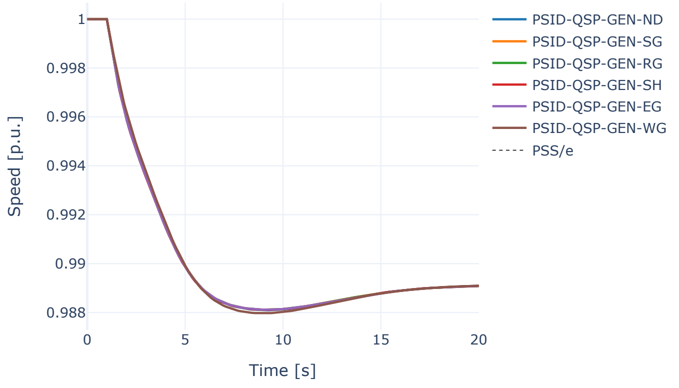

Fig. 9. Generator speed after Gen Bus2-S trip case 4-buses QSP

to match the models available in PSID.jl. In the PSCAD model, the output of the inner loop control ($V_{\text{dq}}^{\text{ref}}$) and the outer loop control angle (δ_{olc}) define an ideal three-phase voltage source at the converter side of the filter.

In this system we conducted a line trip between buses 1 and 2 using a mass-matrix model with abstol $= 10^{-14}$ in PSID.jl, and setting 5μs time step in PSCAD. Additionally, we conducted a simulation specifying the lines to be modeled algebraically as usually done in QSP simulations. Figure 12 showcases the close match between PSID.jl and PSCAD for the bus voltages and the capability to capture high frequency dynamics which the algebraic lines can't capture. Figure 13 shows a comparison of the internal states of the PLL in the VSM inverter. These results display the effects of the line models in the internal PLL states which in this case show that the additional dynamics from the lines do not affect the PLL angle estimation $\delta\theta_{olc}$ but the EMT does reflect an oscillatory dynamic in $v_{\text{q},pll}$ that the algebraic model can not.

C. 240-bus WECC case

To assess its modeling capabilities at large scales, we tested PSID.jl on the 240-bus WECC case for the study of large-scale penetration of IBR [27]. The system is comprised of the devices described in Table II and 329 Lines. The resulting dynamic model has a total of 2420 dynamic states and 506 algebraic states.

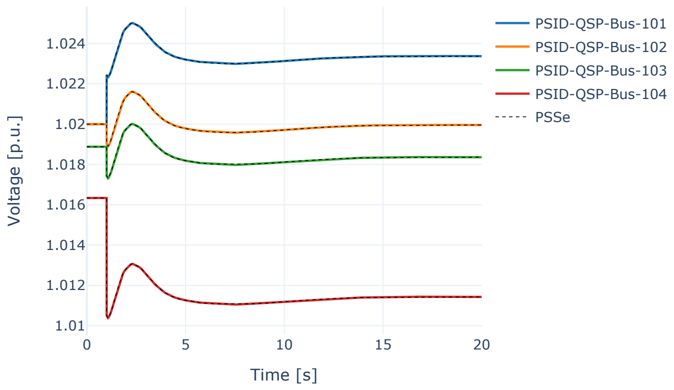

Fig. 10. Bus Voltages after Line 1-3 trip case 4-buses QSP

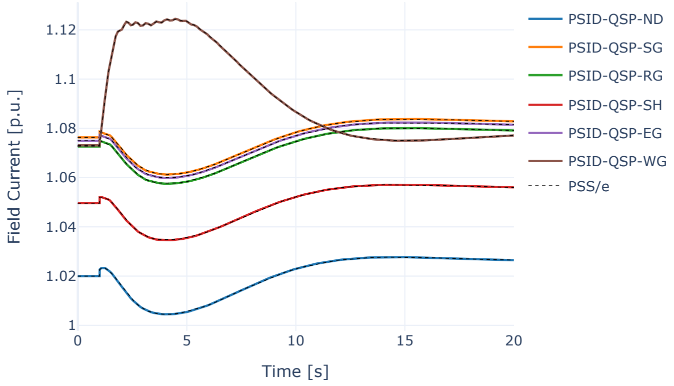

Fig. 11. Generator field current after Line 1-3 trip case 4-buses QSP

We executed 329 line trips and 195 generator trips in the system using PSS®E and `PSID.jl` and for each contingency we calculated and compared the Root Mean Square Error (RMSE) for every bus voltage, angle, and generator speed, resulting in a total of 595 trace comparisons between `PSID.jl` and PSS®E. Overall, we calculated the RMSE of 311,780 individual traces to verify the validity of `PSID.jl` simulation results with respect to a commercial tool for QSP modeling. We employed the `Rodas5P()` solver with adaptive time-stepping and a 10^{-9} absolute tolerance and PSS®E using a $\Delta t = 0.005$. Table III shows the results of this exercise; the average error is calculated as the average of the max errors over the set of perturbations.

D. Small Signal analysis

`PSID.jl` AD routine for Jacobian evaluation provides the capability to evalate function (4) at a desired operating point to analyze the small signal stability of a system. We employ a known small-signal unstable system, the 11-bus Kundur system (see Table IV) to verify the eigenvalues calculation in `PSID.jl`. We compare the eigenvalues and the damping ζ with outputs from the Python package ANDES [8] which provides a similar capability to `PSID.jl`. ANDES eigenvalue routine analyses has already been validated against the commercial tool DSATools SSAT. The RMSE of the eigenvalues between ANDES and `PSID.jl` is $\frac{1}{N}||\lambda_{\texttt{PSID}} - \lambda_{\text{ANDES}}||_2 \approx 0.2872$. The results confirm that these small signal analyses

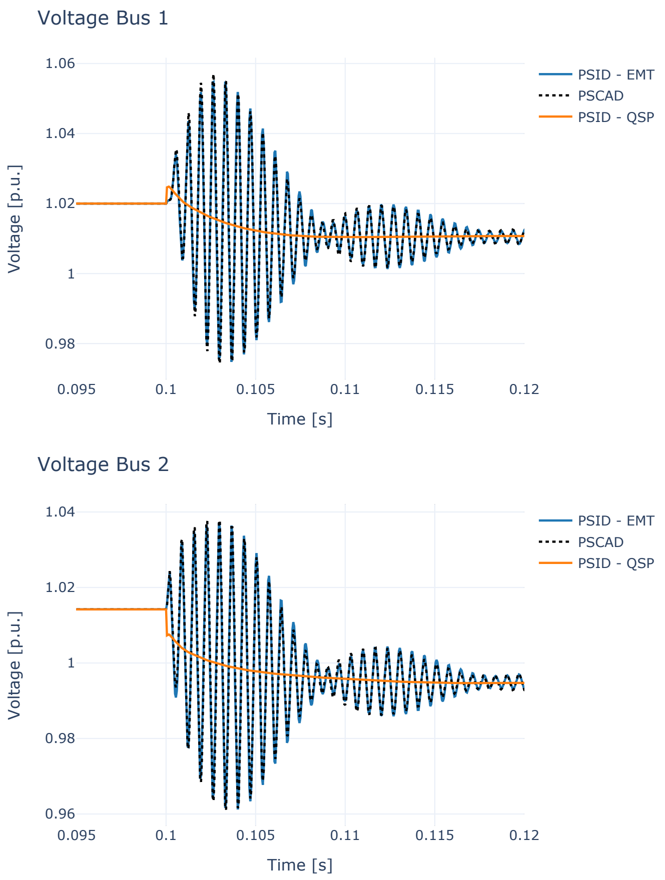

Fig. 12. Transient response PSCAD vs PSID - EMT.

TABLE II
MODELS USED IN WECC 240-BUS QSP PHASOR VALIDATION

Count	Machine	Excitation	Governor	PSS	
6	GENROU	SEXS	GAST	IEEEST	
41	GENROU	SEXS	GAST		
3	GENROU	SEXS	HYGOV	IEEEST	
22	GENROU	SEXS	HYGOV		
1	GENROU	SEXS	TGOV	IEEEST	
36	GENROU	SEXS	TGOV		
Inverters	Outer	Inner	Converter	Filter	Freq. Est.
121	REPCA1	REECB1	REGCA1	None	None

closely match other for QSP models with packages like ANDES. The small differences observed are explained by power-torque approximations performed in shafts and turbine governor models.

The network modeling flexibility between QSP and EMT also extends to more small signal analyses allowing a comparison of the eigenvalues between the two network models. Figure 14 presents a comparison of the eigenvalues for both formulations in `PSID.jl` for the system in Section IV-B. The EMT model has an additional 12 states for the network, with a large imaginary part that explains the fast oscillations observed in Figure 12.

V. CONCLUSION

This paper introduced the open-source simulation toolbox `PSID.jl`, which focuses on positive sequence and electro-

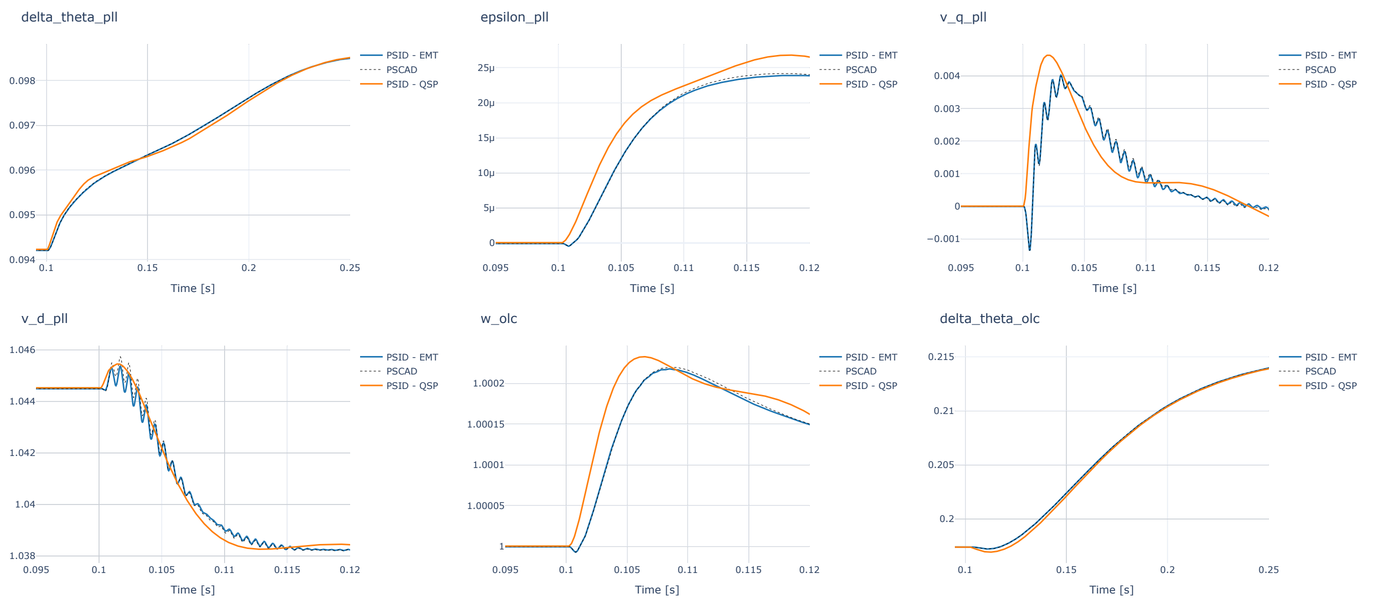

Fig. 13. Transient response PSCAD vs `PSID.jl` - EMT.

TABLE III
ERROR ANALYSIS OF LARGE QSP SIMULATION

	Generator Trip	Line Trip
Maximum Angle RMSE [deg]	4.10×10^{-1}	4.50×10^{-1}
Maximum Voltage RMSE [pu]	2.59×10^{-5}	2.62×10^{-5}
Maximum Speed RMSE [pu]	6.57×10^{-6}	1.22×10^{-5}
Average Angle RMSE [deg]	3.17×10^{-2}	1.57×10^{-4}
Average Voltage RMSE [pu]	1.08×10^{-5}	7.83×10^{-7}
Average Speed RMSE [pu]	1.60×10^{-6}	5.23×10^{-7}

TABLE IV
EIGENVALUE RESULT COMPARISON FOR 11-BUS SYSTEM

	`PSID.jl`		ANDES	
	Eigenvalue	$\zeta[\%]$	Eigenvalue	$\zeta[\%]$
#1	$-37.2633 + 0j$	100	$-37.2633 + 0j$	100
#2	$-37.1698 + 0j$	100	$-37.1699 + 0j$	100
#3	$-36.2028 + 0j$	100	$-36.2031 + 0j$	100
⋮	⋮	⋮	⋮	⋮
#39	$0.5381 - 2.7415j$	–19.26	$0.5537 - 2.7459j$	–19.77
#40	$0.5581 + 2.7415j$	–19.26	$0.5537 + 2.7459j$	–19.77

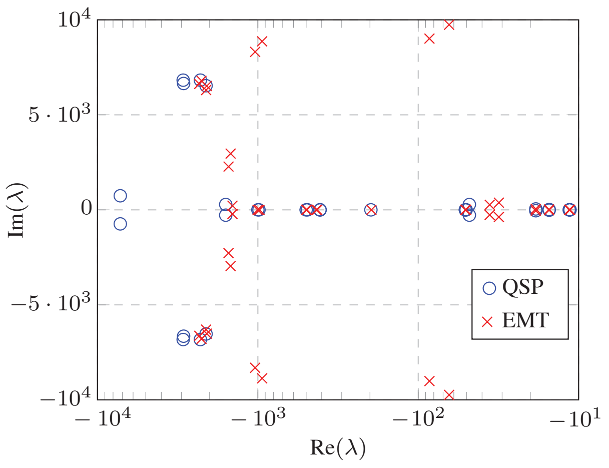

Fig. 14. Eigenvalue comparison for balanced EMT simulation.

magnetic simulations of low-inertia power systems. `PSID.jl` is built on Julia scripting language and incorporates a myriad of power system devices and component models while enabling the analysis of transient responses of dynamical systems using different model complexity. The development of device metamodels for generators and inverters standardize ports and states interaction among devices and their internal components. The proposed software design enables researchers to define a new component's model within the proposed metamodel and quickly explore novel architectures and controls.

We presented several numerical experiments and benchmarks to highlight the capabilities of `PSID.jl` and the validity of the models included in the library. Case studies show that `PSID.jl` enables easier assessment of the trade-off between model complexity and computational requirements by exchanging device models and comparing simulation results under different assumptions.

The ongoing development of `PSID.jl` focuses on extending the available models and analytical capabilities. This includes reduced-order inverter models, FACTS device models, and HVDC and DC-side dynamics of inverters. Finally, using data-driven techniques for developing surrogate models to represent aggregation of networks is a next research direction.

REFERENCES

[1] F. Milano, *Power system modelling and scripting*. Springer Science & Business Media, 2010.

[2] M. Paolone, T. Gaunt, X. Guillaud, M. Liserre, S. Meliopoulos, A. Monti, T. Van Cutsem, V. Vittal, and C. Vournas, "Fundamentals of power systems modelling in the presence of converter-interfaced generation," *Electric Power Systems Research*, vol. 189, p. 106811, 2020.

[3] N. Hatziargyriou, J. Milanovic, C. Rahmann, V. Ajjarapu, C. Canizares, I. Erlich, D. Hill, I. Hiskens, I. Kamwa, B. Pal *et al.*, "Definition and classification of power system stability revisited & extended," *IEEE Transactions on Power Systems*, 2020.

[4] F. Milano, F. Dörfler, G. Hug, D. J. Hill, and G. Verbič, "Foundations and challenges of low-inertia systems," in *2018 Power Systems Computation Conference (PSCC)*. IEEE, 2018, pp. 1–25.

[5] E. Rehman, M. Miller, J. Schmall, and S. H. F. Huang, "Dynamic stability assessment of high penetration of renewable generation in the ercot grid," ERCOT, Tech. Rep., 04 2018, version 1.0.

[6] R. Henriquez-Auba, J. D. Lara, C. Roberts, and D. S. Callaway, "Grid forming inverter small signal stability: Examining role of line and voltage dynamics," in *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2020, pp. 4063–4068.

[7] F. Milano, "An open source power system analysis toolbox," *IEEE Transactions on Power systems*, vol. 20, no. 3, pp. 1199–1206, 2005.

[8] H. Cui, F. Li, and K. Tomsovic, "Hybrid symbolic-numeric framework for power system modeling and analysis," *IEEE Transactions on Power Systems*, vol. 36, no. 2, pp. 1373–1384, 2020.

[9] A. Guironnet, M. Saugier, S. Petitrenaud, F. Xavier, and P. Panciatici, "Towards an open-source solution using modelica for time-domain simulation of power systems," in *2018 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*. IEEE, oct 2018.

[10] A. Plietzsch, R. Kogler, S. Auer, J. Merino, A. Gil-de Muro, J. Liße, C. Vogel, and F. Hellmann, "PowerDynamics.jl—an experimentally validated open-source package for the dynamical analysis of power grids," *SoftwareX*, vol. 17, p. 100861, 2022.

[11] C. Rackauckas and Q. Nie, "Differentialequations.jl – a performant and feature-rich ecosystem for solving differential equations in Julia," *Journal of Open Research Software*, vol. 5, no. 1, 2017.

[12] B. Stott, "Power system dynamic response calculations," *Proceedings of the IEEE*, vol. 67, no. 2, pp. 219–241, 1979.

[13] U. Markovic, O. Stanojev, P. Aristidou, E. Vrettos, D. Callaway, and G. Hug, "Understanding small-signal stability of low-inertia systems," *IEEE Transactions on Power Systems*, vol. 36, no. 5, pp. 3997–4017, 2021.

[14] U. M. Ascher and L. R. Petzold, *Computer methods for ordinary differential equations and differential-algebraic equations*. Siam, 1998, vol. 61.

[15] F. Milano, "Semi-implicit formulation of differential-algebraic equations for transient stability analysis," *IEEE Transactions on Power Systems*, vol. 31, no. 6, pp. 4534–4543, 2016.

[16] C. Rackauckas and Q. Nie, "Confederated modular differential equation apis for accelerated algorithm development and benchmarking," *Advances in Engineering Software*, vol. 132, pp. 1–6, 2019.

[17] J. Revels, M. Lubin, and T. Papamarkou, "Forward-mode automatic differentiation in julia," *arXiv preprint arXiv:1607.07892*, 2016.

[18] "Methods: The Julia Language," https://docs.julialang.org/en/v1/manual/-methods/, accessed: 2019-09-25.

[19] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM review*, vol. 59, no. 1, pp. 65–98, 2017.

[20] A. Bojańczyk, "Complexity of solving linear systems in different models of computation," *SIAM Journal on Numerical Analysis*, vol. 21, no. 3, pp. 591–603, 1984.

[21] A. Griewank and A. Walther, *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

[22] C. C. Margossian, "A review of automatic differentiation and its efficient implementation," *Wiley interdisciplinary reviews: data mining and knowledge discovery*, vol. 9, no. 4, p. e1305, 2019.

[23] J. D. Lara, C. Barrows, D. Thom, D. Krishnamurthy, and D. Callaway, "Powersystems. jl—a power system data management package for large scale modeling," *SoftwareX*, vol. 15, p. 100747, 2021.

[24] P. Kundur, *Power system stability and control*. McGraw-Hill New York, 1994.

[25] 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*, vol. 122, pp. 180 – 197, 2015.

[26] U. Markovic, J. Vorwerk, P. Aristidou, and G. Hug, "Stability analysis of converter control modes in low-inertia power systems," in *2018 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*. IEEE, 2018, pp. 1–6.

[27] H. Yuan, R. S. Biswas, J. Tan, and Y. Zhang, "Developing a reduced 240-bus wecc dynamic model for frequency response study of high renewable integration," in *2020 IEEE/PES Transmission and Distribution Conference and Exposition (T&D)*. IEEE, 2020, pp. 1–5.

José Daniel Lara (Senior Member, IEEE) received the B.Sc. and Licentiate degrees in electrical engineering from the University of Costa Rica, San Pedro, Costa Rica, in 2009 and 2012, respectively, the M.Sc. degree in Electrical and Computer Engineering from the University of Waterloo, Waterloo, ON, Canada, in 2014, M.S. in Energy and Resources with focus on Engineering and Business for Sustainability in 2017, and Ph.D. in Energy and Resources with Designated Emphasis in Computational and Data Science and Engineering from the University of California, Berkeley, CA, USA in 2022. He currently works at the National Renewable Energy Laboratory (NREL) with the Grid Planning and Analysis Center (GPAC). His research interests include the analysis and simulation of power systems and their applications to power systems operation and quantitative energy policy.

Rodrigo Henriquez-Auba (Member, IEEE) received the B.Sc. and M.Sc. in electrical engineering from the Pontifical Catholic University of Chile, Santiago, Chile, in 2014 and 2016, respectively, and the Ph.D. degree in Electrical Engineering and Computer Sciences from the University of California, Berkeley, CA, USA, in 2022. He currently works at the National Renewable Energy Laboratory (NREL) with the Grid Planning and Analysis Center (GPAC), and his research interests include operation and planning of electric systems with large shares of renewable sources, and control, modeling and simulation of power systems and power electronics.

Matthew Bossart (Member, IEEE) is a fourth year PhD student in Electrical, Computer and Energy Engineering at CU Boulder, and an NSF Graduate Research Fellow. Matt's research goals are to aid in the transition to a renewable energy electric grid by examining the impact of power electronics on the system at large. Prior to coming to CU, Matt worked at a start-up company designing power electronics and test methods for diagnosing faults in Lithium-ion batteries.

Duncan S. Callaway (Member, IEEE) is an Associate Professor of Energy and Resources at the University of California, Berkeley. He is also a faculty affiliate in Electrical Engineering and Computer Science, and a faculty scientist at Lawrence Berkeley Laboratory. He received his PhD from Cornell University. He has held engineering positions at Davis Energy Group and PowerLight Corporation, and academic positions at UC Davis, the University of Michigan and UC Berkeley. Duncan teaches courses on electric power systems and at the intersection of statistical learning and energy. His research focuses on grid integration of renewable electricity, and models and control strategies for power system dynamics, demand response, electric vehicles and electricity storage.

Clayton Barrows (Senior Member, IEEE) received the B.S. degree in electrical engineering from the University of Wyoming, Laramie, WY, USA, in 2006, and the Ph.D. degree in energy and mineral engineering from The Pennsylvania State University, State College, PA, USA, in 2013. He is a Senior Engineer with the Grid Planning and Analysis Center (GPAC), National Renewable Energy Laboratory (NREL), Golden, CO, USA. At NREL, he focuses on understanding the impacts of evolving infrastructure systems by improving the speed and accuracy of models through advanced computation and analysis.