SoftwareX 17 (2022) 100861

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

PowerDynamics.jl—An experimentally validated open-source package for the dynamical analysis of power grids

Anton Plietzsch [a,b,*], Raphael Kogler [a,b], Sabine Auer [c], Julia Merino [d], Asier Gil-de-Muro [d], Jan Lißе [c], Christina Vogel [c], Frank Hellmann [a]

[a] Potsdam Institute for Climate Impact Research, Germany
[b] Humboldt-Universität zu Berlin, Germany
[c] elena international GmbH, Germany
[d] TECNALIA, Basque Technology and Research Alliance (BRTA), Spain

ARTICLE INFO

Article history:
Received 15 December 2020
Received in revised form 26 August 2021
Accepted 19 October 2021

Keywords:
Power systems
Dynamic simulation
Microgrids
Inverters
Julia

ABSTRACT

PowerDynamics.jl is a Julia package for time-domain modeling of power grids that is specifically designed for the stability analysis of systems with high shares of renewable energies. It makes use of Julia's state-of-the-art differential equation solvers and is highly performant even for systems with a large number of components. Further, it is compatible with Julia's machine learning libraries and allows for the utilization of these methods for dynamical optimization and parameter fitting. The package comes with a number of predefined models for synchronous machines, transmission lines and inverter systems. However, the strict open-source approach and a macro-based user-interface also allows for an easy implementation of custom-built models which makes it especially interesting for the design and testing of new control strategies for distributed generation units. This paper presents how the modeling concept, implemented component models and fault scenarios have been experimentally tested against measurements in the microgrid lab of TECNALIA.

Code metadata

Current code version	v2.5.1
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-20-00105
Legal Code License	GPL-3.0 License
Code versioning system used	git
Software code languages, tools, and services used	Julia
Compilation requirements, operating environments & dependencies	see Project.toml
Link to developer documentation	https://juliaenergy.github.io/PowerDynamics.jl/stable
Questions and support	https://github.com/JuliaEnergy/PowerDynamics.jl/issues

1. Motivation and significance

The massive integration of renewable energy sources comes with challenges for the dynamic stability and control of power grids. In conventional grids the dynamical stability is ensured by controlling a relatively small number of large conventional generators that are synchronized over the high voltage transmission grid. The inertia of the rotating generator masses stabilizes the operating state against short term fluctuations. In contrast, solar and wind power plants produce much less power and are therefore typically installed in the distribution grid, interfaced by inverters with control schemes that lock onto the grid frequency. A transition towards a larger share of renewable energy sources therefore not only implies a decentralization and an increase in the number of generating units but also significantly decreases the total amount of stabilizing inertia. To tackle this issue, the development of more decentralized control schemes and so-called grid-forming inverter controls has recently become a very active research field [1].

* Corresponding author.
E-mail address: plietzsch@pik-potsdam.de (Anton Plietzsch).

https://doi.org/10.1016/j.softx.2021.100861

However, models of grid-connected inverters represent black-box models, do not allow a transparent verification of the model results and therefore do not allow a fast transfer of the latest research results. As the International Council on Large Electric Systems (CIGRE) also emphasizes, different inverter types and manufacturer models are often available in different software and in different degrees of complexity. Therefore, harmonization in a common simulation model is impractical [2].

At the same time, the energy transition requires rapid development and continuous improvement of reliable simulation tools for dynamic network analysis [3]. For this reason, Open-Source (OS) solutions are an important contribution to the implementation of a grid-stable energy transition. While in the area of static network analysis (for load flow calculations and optimal power flow problems) there are already numerous OS tools, there is little open source available in the area of dynamic modeling of frequency and voltage stability as well as the synchronization behavior of the network in the seconds and sub-seconds range. One reason for the lack of current OS solutions is the lack of a high-performance software environment.

PowerDynamics.jl [4] is an open source software for the dynamic simulation of energy systems that aims to fill this gap [5]. It is based on the programming language Julia [6] and developed in a collaboration between the Potsdam Institute for Climate Impact Research (PIK) and the company elena international GmbH.[1] *PowerDynamics.jl* has been shown to easily beat the simulation times of common commercial simulation environments, such as DIgSILENT *PowerFactory* or MATLAB *Simulink* [7]. This opens up the possibility to run very large test cases or apply Monte Carlo sampling based methods [8,9].

PowerDynamics.jl comes with a library of predefined component types for generators, loads, inverters, transmission lines and transformers, as well as a number of fault scenarios. Users can easily build their own test cases or read in existing test case files. We encourage users to also implement their own component types and eventually share it with others by making a pull request on GitHub.[2]

The construction of the differential equation system is based on *NetworkDynamics.jl* [10], a Julia package for simulating large dynamical systems on complex network structures that is also developed at PIK. The numerical integration is based on Julia's *DifferentialEquations.jl* package [11]. This makes *PowerDynamics.jl* also compatible with Julia's scientific machine learning ecosystem[3] and thereby opens up the possibility to apply machine learning methods to transient stability analysis.

2. Software description

2.1. Software architecture

The general workflow for a dynamic simulation with *PowerDynamics.jl* is shown in Fig. 1. A power grid is defined by specifying the grid structure and the dynamic equations for the node and line components. Once a power grid is defined, the data can be stored in a .json-file by using `write_powergrid` and loaded again by using `read_powergrid`. The function `find_operationpoint` is used to find fixed points of the dynamical system. Different fault scenarios can be applied to the system in operation state.

2.1.1. Nodes

Synchronous machines, inverters and loads are considered as node components. *PowerDynamics.jl* contains a library of already implemented node components. Additionally, users can define their own node components by using the node macro `@DynamicNode`.

2.1.2. Lines

Transformers and electrical lines are considered as line components. As for the node components, *PowerDynamics.jl* includes a standard library of different line and transformer types, including a static admittance line, `StaticLine`, a dynamic admittance line, `RLLine`, the (static) `PiModelLine` and a simple transformer model based on the Pi-model.

2.1.3. Grid structure

A power grid in *PowerDynamic.jl* is a composite type that contains the line and node components, as well as the graph of the grid structure. It can either be constructed from arrays or ordered dictionaries of the line and node components using the `PowerGrid` constructor or loaded from a .json file using `read_powergrid`. For the dynamical simulations the right-hand-side `ODEFunction` of the differential equation is build using *NetworkDynamics.jl*. This way the model definitions and the simulation engine are decoupled.

2.1.4. Operationpoint

In *PowerDynamics.jl* the global reference frame is the co-rotating frame of the nominal grid frequency (50 or 60 Hz, respectively). Every stable operating state therefore represents a stable steady state solution of the dynamic power system. These states can be found using the `find_operationpoint` function for which different solver methods are implemented. For the operation point search we primarily build on solver methods from the *SteadyStateDiffEq.jl* package.[4] The default setting is `:rootfind` that uses the `SSRootfind` method to perform a NLsolve.jl-based root finding algorithm. Alternatively, `:dynamic` uses the `DynamicSS` method to integrate the dynamical equations until a steady state is reached.

For very large systems with a high number of dynamical states it might be useful to first find a solution for the static power flow in the network before finding the steady state solution for the internal (control) variables of complex machine and inverter models. This can be done by setting `solve_powerflow = true`. The power flow solution is found using the AC power flow solver from the *PowerModels.jl*[5] package [12].

2.1.5. Fault scenarios

In *PowerDynamics.jl* there are different simulation scenarios implemented.

On the one hand simulation scenarios are faults, that are derived from `AbstractPerturbation` and persist for a certain time span `tspan_fault`. The `AbstractPerturbation` scenario is implemented using Callbacks,[6] i.e. the system is numerically integrated until the fault begins, then the system parametrization or topology is changed and the integration is continued until the fault ends. At the moment there are four faults implemented: For the `LineFailure` fault a line is removed from the topology of the grid. For the `NodeShortCircuit` fault the shunt admittance at a node can be changed to a high value. With the `NodeParameterChange` fault, any parameter of a dynamic node model can be

1 https://www.elena-international.com/
2 https://github.com/JuliaEnergy/PowerDynamics.jl
3 https://sciml.ai/
4 https://github.com/SciML/SteadyStateDiffEq.jl
5 https://github.com/lanl-ansi/PowerModels.jl
6 https://github.com/SciML/DiffEqCallbacks.jl

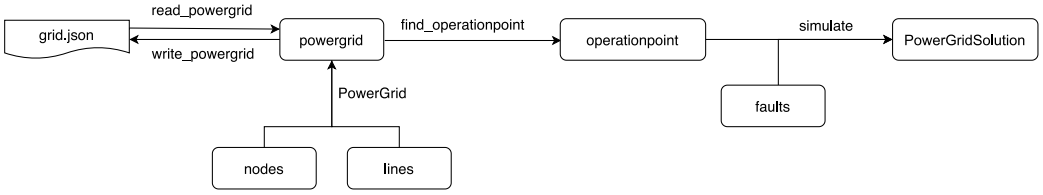

Fig. 1. Simple data and workflow between components of *PowerDynamics.jl*.

changed to a different value. The `PowerPerturbation` is a special case, where the power infeed parameter at a node is changed. This can for example be used to model a load drop at a load node. A detailed example for simulating faults in *PowerDynamics.jl* can be found in Section 3.

On the other hand, there is the theoretical `ChangeInitialConditions` scenario that allows the user to set dynamic variables at single nodes to certain dynamic states out of the equilibrium. This is especially useful for applying probabilistic stability analysis [8,9] that rely on sampling the phase space.

2.1.6. Numerical solution & plotting

The simulation of a fault scenario is done with the function `simulate`. It builds a right-hand-side `ODEFunction` using *NetworkDynamics.jl*, numerically integrates it using the *DifferentialEquations.jl* package and returns a `PowerGridSolution`. This solution can then be easily plotted with *Plots.jl*[7] using a predefined plotting recipe for the solution type.

2.2. Software functionalities

PowerDynamics.jl allows for modeling symmetric 3-phase power grids in dq-coordinates. Voltage and current are both represented by complex phasors, where the real part represents the d-coordinate and the imaginary part the q-coordinate. The advantage of this complex representation is that the complex power can be calculated by $s = p + jq = u \cdot i^*$.

The package is designed for transient stability analysis. Dynamics on shorter timescales such as harmonics and inverter switching cannot be modeled using the phasor approach (see Fig. 2).

2.3. Sample code snippets

The following code snippet is an explanatory example of how to use the `DynamicNode`-Macro to define new inverters, generators or loads in *PowerDynamics.jl*. It starts with the definition of parameters as arguments and the definition of the mass matrix. The boolean is true for differential equations and false for algebraic constraints. The assertion statements on the parameters is followed by a list of dynamics variables and finally the system of ordinary differential equations (ODEs) itself. The ODE system is expected to contain the right-hand-side for all dynamic variables stated before. The complex voltage `u` is always required for all dynamic nodes, however it can be replaced by an algebraic constraint by setting `m_u=false`. The given example of a grid-informing inverter, `VSIVoltagePT1`, corresponds to the one visualized in the block diagram in Fig. 3.

```
@DynamicNode VSIVoltagePT1(τ_v,τ_P,τ_Q,K_P,K_Q,V_r,P,Q)  begin
    MassMatrix(m_u = true, m_int = [true,true])
end
begin
    @assert τ_v > 0 "time constant voltage droop delay should be >0"
    @assert τ_P > 0 "time constant active power measurement should be >0"
    @assert τ_Q > 0 "time constant reactive power measurement should be >0"
    @assert K_Q > 0 "reactive power droop constant should be >0"
    @assert K_P > 0 "active power droop constant reactive power measurement should be >0"
end [[ω, dω],[q_m,dq_m]] begin
    p = real(u * conj(i))
    q = imag(u * conj(i))
    dφ = ω
    v = abs(u)
    dv = 1/τ_v*(-v+V_r - K_Q*(q_m-Q))
    dq_m = 1/τ_Q*(q-q_m)
    du = u * 1im * dφ + dv*(u/v)
    dω = 1/τ_P*(-ω-K_P*(p-P))
end
```

Fig. 2. Source code for a droop controlled voltage source inverter as described by Schiffer et al. [13].

2.4. Future developments & improvements

The *PowerDynamics.jl* community has just started to work heavily on the modularization of *PowerDynamics.jl* such that power generators, inverters and other equipment can be represented in the well-known block diagram structure. We plan to undertake this modularization with the help of *ModelingToolkit.jl*,[8] that uses symbolic equations for modeling differential equations systems [14]. The basic idea of this package is to optimize the performance of the simulation by so called transformations that include symbolic manipulations of the equations but also automatic parallelization of the simulation. Hereby, it aims to take away the responsibility from the modeler to write performant code. It thus takes a similar approach to the modeling language *Modelica*, but at the same time has the feature of great composability with other packages in Julia's differential equations ecosystem [15]. In Fig. 3 a block diagram is shown for the example of a grid-forming inverter. It consists of active- and reactive power droop control (as an outer control loop), a frequency integrator and active as well as a reactive power filters. The causal modeling representation from the block diagram is planned for the new major *PowerDynamics.jl* release. A prototype for an MTK based block structure is already available via the *BlockSystems.jl* package.[9]

[7] https://github.com/JuliaPlots/Plots.jl

[8] https://github.com/SciML/ModelingToolkit.jl

[9] https://github.com/hexaeder/BlockSystems.jl.

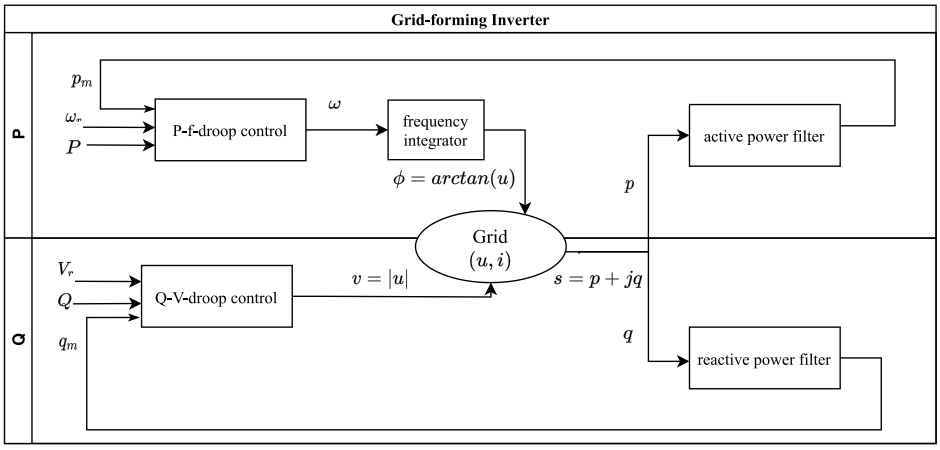

Fig. 3. Example block diagram for a grid-forming inverter.

Another goal for the next major release of *PowerDynamics.jl* is to make it compatible with the power system data management package *PowerSystems.jl*[10]

We further plan to implement more fault scenarios, such as noise perturbations [17,18] and dynamically induced cascading failures [19]. These enhancements will make use of the state-of-the-art stochastic differential equations solvers and callback functions of *DifferentialEquations.jl* [11].

We also want to extend the library of component models for lines and nodes. Here, we also expect user contributions for various inverter controls that can be shared via pull requests on GitHub.

More future contributions are expected from the MARiE-project (Dynamic modeling for analysis and control of intelligent energy networks) undertaken by BTU Cottbus together with elena international GmbH and funded by the German Federal Ministry for Economic Affairs and Energy (BMWi). The key innovation of the project is the development of canonical base models of grid-connected power converters by BTU Cottbus. Together with elena international, the models are then implemented in Julia and integrated in *PowerDynamics.jl*. To ensure practical relevance of the models they are finally experimentally validated with a Power-in-the-loop test bed by BTU Cottbus.

3. Illustrative example

Our illustrative example is the simulation of a tripping line in the IEEE 14-bus system. This system contains 5 generators, 11 loads, 17 lines and 3 transformers. We model the loads as constant power loads and the generators by a 4th order generator model [20]. The parameters are taken from [21].

The implementation of this introductory example can be found in the package PowerDynamicsExamples.[11] It is available both as a Julia script and a Jupyter Notebook. The latter can also be directly launched in a browser by using BinderHub (see Fig. 4).

The simulation of this test case is straightforward and requires only a few lines of code. *PowerDynamics.jl* is a registered package. This means it can be directly installed from the Julia REPL (read-eval-print loop) with: For creating a new test case we would have to define an `Array` or `OrderedDict` of buses and lines and execute the function `PowerGrid(buses,lines)`. Here, we assume that the IEEE-14bus testcase is already saved as a .json

```
using Pkg
Pkg.add("PowerDynamics")
```

Fig. 4. Installing *PowerDynamics.jl*.

```
using PowerDynamics
powergrid = read_powergrid("ieee14-4th-order.json", Json)
operationpoint = find_operationpoint(powergrid)
timespan= (0.0,5.)
fault = LineFailure(line_name="branch2", tspan_fault=(1.,5.))
solution = simulate(fault, powergrid, operationpoint, timespan)
include("plotting.jl")
plt = create_plot(solution)
display(plt)
```

Fig. 5. Source code for simulating the IEEE14 grid with *PowerDynamics.jl*. The function `create_plot` is defined in the script *plotting.jl* that can be found in the *PowerDynamicsExamples* repository. The plotting script has additional dependencies on Julia packages, that need to be installed similar to Fig. 5.

file that we can read in with `read_powergrid`. As the initial state for the simulation we determine the operation point with `find_operationpoint`. The fault scenario is the predefined function `LineFailure`, which removes a line from the grid. The numerical integration is done with the `simulate` function and the results are shown in Fig. 6.

4. Software validation

4.1. Testlab description

An experimental validation of *PowerDynamics.jl* was undertaken in January 2020 at the Smart Grid Technologies Laboratory at TECNALIA[12] within the context of the ERIGrid[13]-funded Valeria project. The primary goal was to implement detailed dynamical models for different inverter controls, loads and perturbation scenarios in *PowerDynamics.jl* and to validate the numerical simulations against experimental measurements of different dynamical scenarios in a small test grid setup.

The laboratory at TECNALIA comes with a grid-forming and a grid-following inverter that both have self-built control schemes [22,23]. Therefore, the dynamic equations and parameters are known and can be translated into *PowerDynamics.jl* code. The

10 https://github.com/NREL-SIIP/PowerSystems.jl [16], in order to use its ability to parse larger testcases in the industry standard data formats MATPOWER and PSS/e.

11 https://github.com/JuliaEnergy/PowerDynamicsExamples

12 https://www.tecnalia.com

13 *European Research Infrastructure supporting Smart Grid Systems Technology Development, Validation and Roll Out* (https://erigrid.eu/).

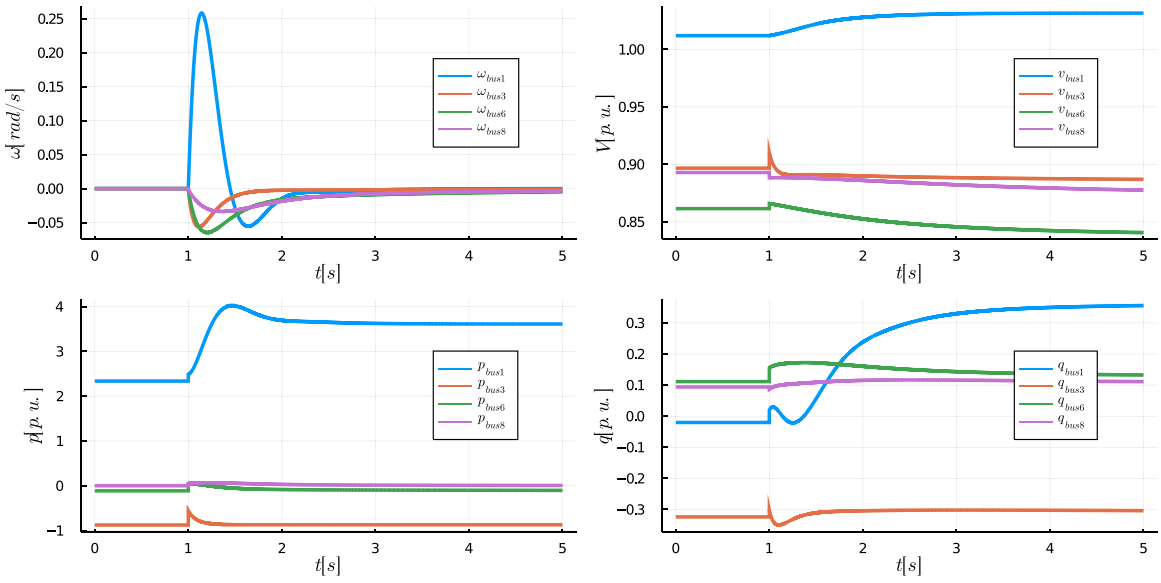

Fig. 6. Simulation of a line tripping in the IEEE 14-bus system.

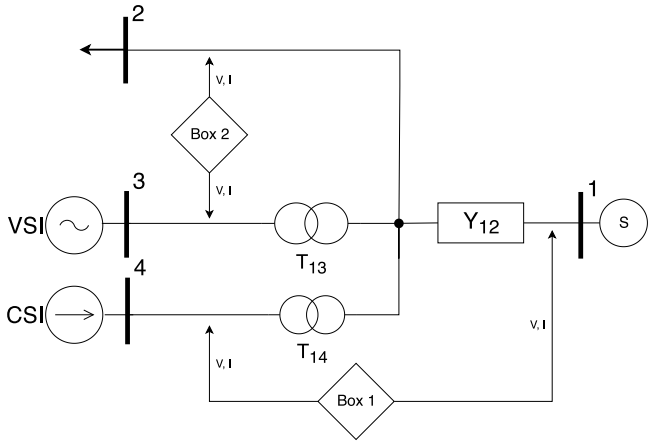

Fig. 7. Test grid setup in the laboratory. The setup includes a grid-forming voltage source inverter (bus 3) and a grid-following current source inverter (bus 4) that are connected to the grid by transformers, a load (bus 2) and a line with admittance Y_{12}. The test grid has an interconnection to the Spanish grid (bus 1) which we model as a slack node. Measurement are taken out with two Boxes that can measure voltage and current signals for two phases. We assume the phases to be balanced and thereby calculate the third phase.

grid-following control consists of a low-pass filter for the voltage signal, a phase-locked loop (PLL) and a droop control for active and reactive power. The grid-forming control consists of several filters for the voltage and current signals, a droop control for frequency and voltage amplitude and a fictitious impedance.

4.2. Test case description

For demonstrative purposes in this paper we chose a test case setup including a grid-forming, a grid-following and a load in grid connected mode (Fig. 7). The comparison between measurements and simulations for the scenario of a power set point dispatch at the voltage source inverter is shown in Fig. 8. A more detailed description and a larger variety of test cases can be found in the technical report of the Valeria project [24].

5. Impact

5.1. Scientific impact

Programming languages, such as Python or the commercial language MATLAB, are not up to the special requirements of modeling renewable power grids. This includes the integration of stochastic differential equations for the representation of fluctuating renewable energy sources [3] as well as the simulation of delayed differential equations to represent the influence of measurement and delay times of inverter controls [25]. Due to Julia's just-in-time compilation, *PowerDynamics.jl* has also strong performance advantages compared to simulations in MATLAB or Python. It has also been shown to be much more performant than its commercial competitors MATLAB *Simulink* and DIgSILENT *PowerFactory* [7]. This does not only enable the dynamic simulation of very large test cases with a large number of nodes but also Monte Carlo sampling based methods. This includes for example efficient calculations of stability and performance measures such basin stability [8] and survivability [9] that complement standard linear stability analysis and capture also the nonlinear behavior of the system. Additionally, *PowerDynamics.jl* supports automatic differentiation and is therefore compatible with Julia's scientific machine learning packages such as *DiffEqFlux.jl*, a package for combining differential equations with neural networks [26]. This will eventually lay the foundation for a new branch of research, a machine learning based energy system stability analysis [27].

While in the context of static power flow analysis there are already numerous OS tools (which are mostly Python-based), e.g. *PyPSA* [28] and *pandapower* [29], there is little open source available in the area of dynamic modeling of frequency and voltage stability as well as the synchronization behavior of the network in the seconds and subseconds range. Only the *Power System Analysis Toolbox* (PSAT) library which has not been maintained for 10 years and therefore does not contain any current inverter models, and the new library *PowerSimulationDynamics.jl*[14] (another Julia package, continuation of *LITS.jl* [30]) are known. The latter is developed by the National Renewable Energy Laboratory (NREL) and uses the parsing capabilities of *PowerSystems.jl* to build their industry standard power system data structures. The

[14] https://github.com/NREL-SIIP/PowerSimulationsDynamics.jl

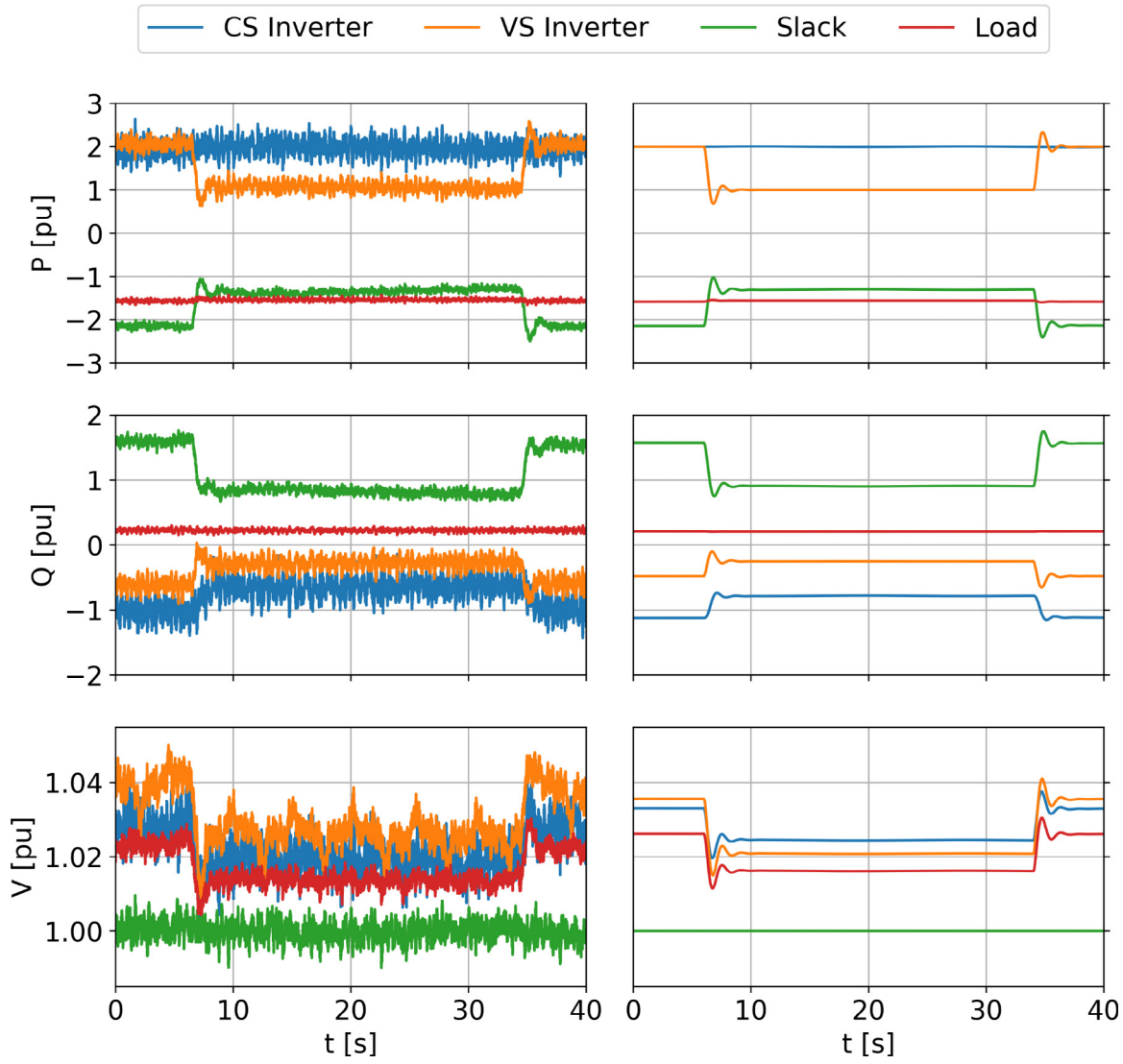

Fig. 8. Comparison of measurements and simulations. The scenario is a set point dispatch for the active power at the grid-forming inverter in the test grid setup shown in Fig. 7. Voltage and power signals are calculated from the measured current and voltage signals. The dynamic models for grid-forming and grid-following inverter have been implemented and simulated in *PowerDynamics.jl*.

package is using implicit and explicit differential algebraic equations (DAEs) representations in combination with the Sundials solvers [31].

In contrast *PowerDynamics.jl* is based on the *DifferentionalEquations.jl* package which enables to simulate stochastic differential equations (SDEs), delayed differential equations (DDEs) and explicit DAEs in mass matrix representation. This allows the modeling of inverter measurement delays and renewable power fluctuations. Additionally, since *PowerDynamics.jl* uses the *NetworkDynamics.jl* package for simulation model building which decouples the representation of nodes and lines, major performance optimizations with parallelization and GPU calculations are possible. This gives the possibility to model large-scale networks. Currently, we are in contact with the developers of *PowerSimulationDynamics.jl* and collaborate on the integration of *ModelingToolkit.jl* and *PowerSystems.jl*.

PowerDynamics.jl is still very young and under heavy development but the user base is steadily growing. At the moment the software is developed by and used for research at PIK, at TU Berlin, at BTU Cottbus, at TU Delft and at FZ Jülich. Further, TU Delft plans to use *PowerDynamics.jl* for teaching electrical engineering courses.

5.2. Commercial impact

PowerDynamics.jl will serve as a basis or backend for web applications and other commercial products of elena international which is a spin-off of the Potsdam-Institute of Climate Impact Research. In particular, *PowerDynamics.jl* together with existing Open-Source tools for static network analyses will be the backend of an innovative web application which will be offered as an analysis tool for the renewable conversion of power systems first in islanded and then also in interconnected operation. Here elena int. is already working on a prototype for the planning of micro power grids. This software can then be used for the planning of micro power grids (in the global south) and for the new concept of cellular energy systems in Europe.

As an Open-Source Software *PowerDynamics.jl* was also already used for creating an online simulation environment for an online course about "inertia requirements for renewable power systems".[15] The goal of the course is to guide participants through

[15] https://www.renac.de/trainings-services/trainings/ready-made-trainings/product/inertia-requirements-for-renewable-power-systems

the topic of inertia issues and possible solutions for highly renewable power grids.[16]

6. Conclusions

PowerDynamics.jl is an OS Julia package for transient stability analysis which is both suitable for highly renewable power grids, especially relevant for microgrids, and large-scale networks with a large number of buses due to its performance advantages. It allows for the integration of delays and fluctuation dynamics of renewable power sources. Due to its decoupled representation of node and line components, it can be parallelized and allows for future performance optimization with GPU (graphics processing unit) calculations. *PowerDynamics.jl* is still a young library, however, several research projects have started that relate to *PowerDynamics.jl* and push its development. Currently, it is work in progress to integrate a causal modeling approach in *PowerDynamics.jl* that allows the representation of inverters as block diagrams as it is coming in control theory to bridge the gap to this community and engage a wider user base into the development and spreading of *PowerDynamics.jl*.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research has been performed using the ERIGrid Research Infrastructure and is part of a project that has received funding from the European Union's Horizon 2020 Research and Innovation Programme under the Grant Agreement No. 654113. The support of the European Research Infrastructure ERIGrid and its partner TECNALIA is very much appreciated. We further acknowledge the support by BMBF (CoNDyNet2 FK. 03EK3055A), the DFG (ExSyCo-Grid, 410409736), the Leibniz competition (T42/2018) and the Federal Ministry of Economics (MAriE, FK. 03Ei4012B).

References

[1] Schiffer J, Zonetti D, Ortega R, Stanković AM, Sezi T, Raisch J. A survey on modeling of microgrids—From fundamental physics to phasors and voltage sources. Automatica 2016;74:135–50.
[2] 4/C6.35/CIRED JWGC. Modelling of inverter-based generation for power system dynamic studies. 2018, European Commission.
[3] Auer S. The stability and control of power grids with high renewable energy share. (Ph.D. thesis), Humboldt-Universität zu Berlin; 2018.
[4] Kittel T, Auer S, Horn C. Sneak preview: PowerDynamics.jl – An open-source library for analyzing dynamic stability in power grids with high shares of renewable energy. Wind Integr Workshop 2018 2018. URL https://arxiv.org/abs/2012.05175.
[5] Auer S, Kittel T. Modeling the dynamics and control of power systems with high share of renewable energies. Wind Integr Workshop 2018 2018. URL https://arxiv.org/abs/2012.05164.
[6] Bezanson J, Edelman A, Karpinski S, Shah VB. Julia: A fresh approach to numerical computing. SIAM Rev 2017;59(1):65–98.
[7] Liemann S, Strenge L, Schultz P, Hinners H, Porst J, Sarstedt M, et al. Probabilistic stability assessment for active distribution grids. In: 2021 IEEE madrid powertech. 2021, p. 1–6. http://dx.doi.org/10.1109/PowerTech46648.2021.9494855.
[8] Menck PJ, Heitzig J, Marwan N, Kurths J. How basin stability complements the linear-stability paradigm. Nat Phys 2013;9(2):89–92.
[9] Hellmann F, Schultz P, Grabow C, Heitzig J, Kurths J. Survivability of deterministic dynamical systems. Sci Rep 2016;6:29654.
[10] Lindner M, Lincoln L, Drauschke F, Koulen JM, Würfel H, Plietzsch A, et al. Networkdynamics. jl—Composing and simulating complex networks in julia. Chaos 2021;31(6):063133.
[11] Rackauckas C, Nie Q. Differentialequations. jl–A performant and feature-rich ecosystem for solving differential equations in julia. J Open Res Softw 2017;5(1).
[12] Coffrin C, Bent R, Sundar K, Ng Y, Lubin M. Powermodels. jl: An open-source framework for exploring power flow formulations. In: 2018 Power systems computation conference. IEEE; 2018, p. 1–8.
[13] Schiffer J, Ortega R, Astolfi A, Raisch J, Sezi T. Conditions for stability of droop-controlled inverter-based microgrids. Automatica 2014;50(10):2457–69.
[14] Ma Y, Gowda S, Anantharaman R, Laughman C, Shah V, Rackauckas C. Modelingtoolkit: A composable graph transformation system for equation-based modeling. 2021, arXiv:2103.05244.
[15] Rackauckas C. Modelingtoolkit, modelica, and modia: The composable modeling future in julia. The Winnower 2021. http://dx.doi.org/10.15200/winn.162133.39054.
[16] Lara JD, Barrows C, Thom D, Krishnamurthy D, Callaway D. Powersystems. jl—A power system data management package for large scale modeling. SoftwareX 2021;15:100747.
[17] Auer S, Hellmann F, Krause M, Kurths J. Stability of synchrony against local intermittent fluctuations in tree-like power grids. Chaos 2017;27(12):127003.
[18] Plietzsch A, Auer S, Kurths J, Hellmann F. A generalized linear response theory of complex networks with an application to renewable fluctuations in microgrids. 2019, arXiv preprint arXiv:1903.09585.
[19] Schäfer B, Witthaut D, Timme M, Latora V. Dynamically induced cascading failures in power grids. Nature Commun 2018;9(1):1–13.
[20] Sauer PW, Pai MA. Power system dynamics and stability. vol. 101, Wiley Online Library; 1998.
[21] Kodsi SKM, Canizares CA. Modeling and simulation of IEEE 14-bus system with FACTS controllers. Tech. Rep., Canada: University of Waterloo; 2003.
[22] Planas E, Gil-de-Muro A, Andreu J, Kortabarria I, de Alegría IM. Stability analysis and design of droop control method in dq frame for connection in parallel of distributed energy resources. In: IECON 2012-38th annual conference on IEEE industrial electronics society. IEEE; 2012, p. 5683–8.
[23] Planas E, Gil-de-Muro A, Andreu J, Kortabarria I, de Alegría InM. Design and implementation of a droop control in d–q frame for islanded microgrids. IET Renew Power Gener 2013;7(5):458–74.
[24] Vogel C, Auer S, Deß T, Plietzsch A, Kogler R. Validation of low-voltage energy and renewables integration analysis. Tech. rep., European Research Infrastructure supporting Smart Grid (ERIGrid); 2020, https://erigrid.eu/wp-content/uploads/2020/06/ERIGrid_TA_VALERIA_Technical-Report_v01.pdf.
[25] Schiffer J, Fridman E, Ortega R, Raisch J. Stability of a class of delayed port-Hamiltonian systems with application to microgrids with distributed rotational and electronic generation. Automatica 2016;74:71–9.
[26] Rackauckas C, Innes M, Ma Y, Bettencourt J, White L, Dixit V. Diffeqflux. jl-a julia library for neural differential equations. 2019, arXiv preprint arXiv:1902.02376.
[27] Nauck C, Lindner M, Schürholt K, Zhang H, Schultz P, Kurths J, et al. Predicting dynamic stability of power grids using graph neural networks. 2021, arXiv preprint arXiv:2108.08230.
[28] Brown T, Hörsch J, Schlachtberger D. PyPSA: Python for power system analysis. J Open Res Softw 2018;6(4). http://dx.doi.org/10.5334/jors.188, arXiv:1707.09913.
[29] Thurner L, Scheidler A, Schäfer F, Menke J-H, Dollichon J, Meier F, et al. Pandapower—An open-source python tool for convenient modeling, analysis, and optimization of electric power systems. IEEE Trans Power Syst 2018;33(6):6510–21.
[30] Henriquez-Auba R, Lara JD, Roberts C, Pallo N, Callaway DS. LITS. jl–An open-source julia based simulation toolbox for low-inertia power systems. 2020, arXiv preprint arXiv:2003.02957.
[31] Hindmarsh AC, Brown PN, Grant KE, Lee SL, Serban R, Shumaker DE, et al. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. ACM Trans Math Softw 2005;31(3):363–96.

16 https://github.com/SabineAuer/OnlineCourse-Inertia