Open-Source Steady-State Models for Integration of Wave Energy Converter into Microgrids

Alexander Barajas-Ritchie, Derek Jackson, Eduardo Cotilla-Sanchez, Yue Cao
School of Electrical Engineering and Computer Science
Oregon State University
Corvallis, Oregon 97331, USA
Email: barajale@oregonstate.edu

***Abstract*—This paper proposes a software framework, WEC-Grid, for integrating wave energy converters (WECs) into power flow software, such as Siemens PSS®E, to aid the integration of alternative energy sources into Microgrids. While integrating alternative sources such as WECs presents specific challenges such as cost, power quality, and power variability, wave energy is a promising renewable energy resource. Evaluating the integration of WECs into the power grid is a complex and nuanced problem that requires seamless communication between a WEC model and power flow software. The presented WEC-Grid software framework bridges and extends the functionality of WEC-Sim, an open-source WEC modeling package for MATLAB, through a wave-to-wire (W2W) electro-mechanical power conversion and processing model. WEC-Grid acts as a software wrapper, handler, and communication layer between the W2W modeler and power flow software. The software is designed to represent each grid system as a class object, allowing power system operators to perform power system duties such as contingency planning and dispatch operations. The integration of WECs with PSS®E's power flow calculations workflow is demonstrated with an IEEE RTS case study.**

***Index Terms*—Power systems, microgrid, wave energy, renewable energy integration**

I. Introduction

Integrating variable renewable energy sources, such as wind, solar, and wave energy, is a key objective in building sustainable and eco-friendly power grids. The integration of these alternative sources presents specific challenges such as cost, power quality, and power variability. Wave energy has the potential to be a reliable and abundant renewable resource. However, its technology is relatively immature, with all the challenges associated with developing technologies. Wave energy converters (WECs), such as point absorber types, can capture wave energy and turn it into electro-mechanical energy, thus producing clean, renewable energy for communities.

The design and operation of integrating WECs into the power grid is a complex and nuanced problem that requires seamless communication between a WEC model and power flow software. Power flow solvers are routinely used for power system engineering applications such as contingency planning and dispatch operations. However, there is currently no open-source solution for integrating WECs into power flow software, which adds an additional hurdle to wave energy integration.

This paper proposes a software framework that enables power system modelers to integrate WECs into either an open-source power system solver or a commercial solver such as Matpower and Siemens PSS®E. We introduce a wave-to-wire (W2W) model consisting of the power electronics required to transfer wave generated energy to the grid that extends the functionality of WEC-Sim [1], an existing open-source WEC modeling package for MATLAB, and demonstrate its integration with PSS®E's power flow calculations workflow. The rest of this paper is organized as follows: Section II provides background information and motivation for wave energy and power systems. Section III outlines the methodology for the proposed WEC-Grid software. Section IV reviews the W2W and power flow models used in our demonstration and results. Section V explores our program's results and the microgrid's steady-state status. Finally, Section VI concludes the paper and discusses future work.

II. Background and Motivation

Power system planning and operations computational tools such as Siemens PSS®E, MiGRID, and HOMER model power system configurations with variable amounts of buses, loads, and generators. These electrical networks are dynamic and range from one to thousands of buses and generators depending on the service size of the grid. Using our knowledge of generator and load types, one can model how a type of generator will perform on a given grid configuration using these tools. In the case of WECs, energy system modeling and simulation have largely focused on wave energy mechanics rather than the connected electrical system. The open-source WEC-Sim library, which has been applied to many WEC design studies (examples in [2]–[4]), lacks a robust electrical system modeling interface. While a sub-library within WEC-Sim called PTO-Sim is capable of calculating the electrical power generated from a WEC-connected generator, the electrical load is modeled as a constant resistance and is not compatible with realistic control schemes for linear or nonlinear loads [5].

The WEC electrical conversion system typically includes an AC/DC and a DC/AC conversion stage (or a DC/DC conversion stage for DC transmission), both key components of the W2W system [6]. The electrical conversion and controls have previously been modeled in Saber [7] and in Simulink

[8]. However, both models are not open-source. Without these models, it is hard to reproduce the results and integrate the WEC systems with power system computational tools. Once the WEC power is converted into electrical power, it can be distributed within a microgrid (or a regional grid model). However, WECs have problems with power quality [9] due to irregular wave periods at the tens of seconds timescale that result in high peak-to-average power ratios. This problem can be mitigated with energy storage, back-to-back converters, and/or a combination with other WECs (wave farms) [10], [11]. Additional benefits of energy storage in a microgrid include reinforced grid infrastructure, reduced operational costs, and more reliable renewable energy integration [12].

III. Methodology

The electrical power grid is one of the largest machines created, and managing this machine is a complex and challenging problem. We have many tools for managing and understanding our grid system, and our primary tool for managing our power grid is a power flow solver. A power flow solver is a specialized software designed to take a grid model with all the details of electrical lines, generators, and loads and find a balanced solution that provides enough power to each load without compromising any generators or lines. Power system engineers use these power flow solvers to understand demand and adjust loads and generators to achieve grid stability. As detailed in this section, we designed WEC-Grid with the intention that our primary user be a power systems operator. We develop WEC-Grid to be modular to fit whichever workflow a power system operator is using and make the process of WEC integration accessible.

A. Structure

WEC-Grid acts as a software wrapper, handler, and communication layer between a W2W modeler and power flow software. WEC-Grid is designed to represent each grid system as a class object to take advantage of object-oriented programming. The operator interacts directly with WEC-Grid by importing the package into a main script and creating a WEC-Grid object to perform their power systems duties such as dispatch and system analysis. The WEC-Grid object allows our user to perform the W2W modeling, power flow injection, and system analysis. WEC-Grid currently utilizes two application program interfaces (APIs), MATLAB and PSS®E, for the WEC and grid modeling [13], [14]. These APIs are user-facing Python functions that allow for the use of program-specific source code functions. An SQL database which can be seen in orange in Fig. 1, has been implemented to organize and store simulation output from both W2W models and PSS®E. The database is automatically maintained by WEC-Grid with minimal user interaction. The W2W modeler and power flow software only has write-access privileges, which only allows these programs to write their output to the SQL database. All data is served to these modules via the WEC-Grid handler. Each WEC is represented as its whole SQLite table with a corresponding output and time stamp (Fig. 4). The timestamp is critical for maintaining consistency across simulations and models and ensures simulation output from different models can be compared and analyzed accurately.

To maintain and manage the development environment, we employ Anaconda [15]. Anaconda is software that allows users to create an isolated environment that has specific package versions and programming languages. Anaconda lets users switch between, manage, and share environments that allow specialized and intentionally crafted package configurations to ensure software compatibility. An Anaconda environment is strongly encouraged while using WEC-Grid due to the use of specific package version requirements. Python version 3.7 is required for compatibility with PSS®E 35.3 and MATLAB 2021b.

Currently, WEC-Grid has been developed to be interfaced via a Jupyter notebook [16]. Jupyter Notebook is an open-source web-based integrated development environment (IDE) that allows users to interact with code in a series of live cells. This cell format is very useful for data science research due to the ability to include text explanations, equations, and visualizations alongside code cells. This gives the user the ability to understand and interpret and share their results. Jupyter Notebooks were chosen for these reasons in addition to community support. This was a specific goal in our software development to use a platform with an established community that offers tutorials and guides. An example Jupyter Notebook script can be seen in Fig. 2. In Cell 1, one can see WEC-Grid being imported and establishing the PSS®E API connection. We were able to abstract away a significant portion of the setup, and all the user needs to do is to import WEC-Grid, and the PSS®E connection is established. We can see the initialization of a WEC-Grid object in Cell 2, along with the PSS®E output showing a converging steady state. Cell 4 shows a Pandas data frame of the Power Grid. This reformatting from PSS®E to Pandas allows for ease of data exploration and manipulation. Having the grid system represented as a Pandas data frame allows the user to use the Pandas data manipulation functions to perform independent system analysis with a support framework. In the Results and Demonstration section, we demonstrate built-in plotting functions which render directly into Jupyter Notebook output cells.

B. Processing Pipeline

We can follow the flow diagram (Fig. 1) to understand the entire processing pipeline. The process begins with the initialization of a WEC-Grid object represented as a yellow object block. The system is initialized, the proper inputs are passed to the W2W, and power flow models are chosen by the user. The dotted sections of the flow chart indicating the blocks that are modular and can be substituted with supported W2W models and power flow solvers. Only the W2W Simulink model (using WEC-Sim) and PSS®E are currently supported, with plans to support other modelers in later revisions. Once a solved state and W2W simulations are completed, the W2W output will be saved to the SQL database. The grid state is currently saved in local memory. The main loop is then

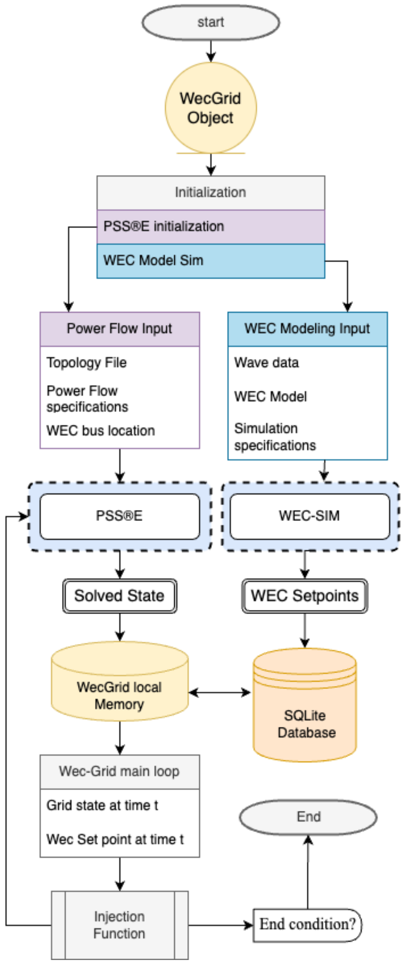
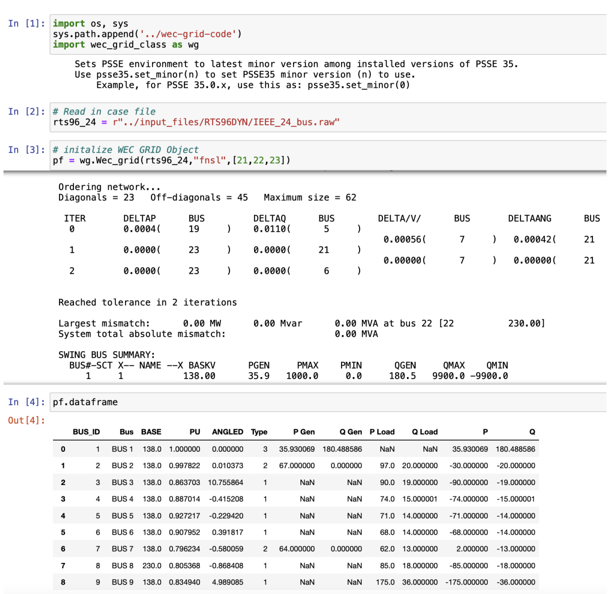


Fig. 1: The flow chart pipeline illustrates the end-to-end processing of the WEC model integration of steady stead state power flows. One of the software's key features is its modularity and use of open-source power flow solvers and WEC modelers. The dotted boxes indicate that the blocks (PSS®E and WEC-SIM) can be substituted with support modeling software.



```
In [1]: import os, sys
        sys.path.append('../wec-grid-code')
        import wec_grid_class as wg

    Sets PSSE environment to latest minor version among installed versions of PSSE 35.
    Use psse35.set_minor(n) to set PSSE35 minor version (n) to use.
        Example, for PSSE 35.0.x, use this as: psse35.set_minor(0)

In [2]: # Read in case file
        rts96_24 = r"../input_files/RTS96DYN/IEEE_24_bus.raw"

In [3]: # initalize WEC GRID Object
        pf = wg.Wec_grid(rts96_24,"fnsl",[21,22,23])

 Ordering network...
 Diagonals = 23   Off-diagonals = 45   Maximum size = 62

  ITER       DELTAP      BUS          DELTAQ      BUS          DELTA/V/      BUS        DELTAANG      BUS
   0         0.0004(     19     )     0.0110(      5     )
                                                                0.00056(      7     )    0.00042(     21
   1         0.0000(     23     )     0.0000(     21     )
                                                                0.00000(      7     )    0.00000(     21
   2         0.0000(     23     )     0.0000(      6     )

 Reached tolerance in 2 iterations

 Largest mismatch:      0.00 MW      0.00 Mvar      0.00 MVA at bus 22 [22         230.00]
 System total absolute mismatch:                    0.00 MVA

 SWING BUS SUMMARY:
   BUS#-SCT X-- NAME --X BASKV      PGEN     PMAX     PMIN      QGEN     QMAX     QMIN
      1     1            138.00      35.9   1000.0      0.0     180.5   9900.0 -9900.0

In [4]: pf.dataframe
Out[4]:
```

	BUS_ID	Bus	BASE	PU	ANGLED	Type	P Gen	Q Gen	P Load	Q Load	P	Q
0	1	BUS 1	138.0	1.000000	0.000000	3	35.930069	180.488586	NaN	NaN	35.930069	180.488586
1	2	BUS 2	138.0	0.997822	0.010373	2	67.000000	0.000000	97.0	20.000000	-30.000000	-20.000000
2	3	BUS 3	138.0	0.863703	10.755864	1	NaN	NaN	90.0	19.000000	-90.000000	-19.000000
3	4	BUS 4	138.0	0.887014	-0.415208	1	NaN	NaN	74.0	15.000001	-74.000000	-15.000001
4	5	BUS 5	138.0	0.927217	-0.229420	1	NaN	NaN	71.0	14.000000	-71.000000	-14.000000
5	6	BUS 6	138.0	0.907952	0.391817	1	NaN	NaN	68.0	14.000000	-68.000000	-14.000000
6	7	BUS 7	138.0	0.796234	-0.580059	2	64.000000	0.000000	62.0	13.000000	2.000000	-13.000000
7	8	BUS 8	230.0	0.805368	-0.868408	1	NaN	NaN	85.0	18.000000	-85.000000	-18.000000
8	9	BUS 9	138.0	0.834940	4.989085	1	NaN	NaN	175.0	36.000000	-175.000000	-36.000000
9	10	BUS 10	138.0	0.881702	3.723669	1	NaN	NaN	100.0	23.000000	-100.000000	-23.000000

Fig. 2: A Jupyter Notebook script that imports and creates a WEC-Grid Object.

triggered, and the program injects the WEC set points at each time step until an end condition occurs.

```
1 def main():
2   wecGridObj = Wec_grid(case.raw,solver,[wec_buses])
3   for i in range(len([wec_buses])):
4     wecGridObj.run_WEC_Sim(wec_id,sim_length,sample,
     waveHeight,wavePeriod,waveSeed)
5   wec_data = read_sql_query("SELECT * from WEC_output")
6   for i in range(len([wec_data])):
7     wecGridObj.inject([Pgen], [V_mag], solver, time)
8 main()
```

Listing 1: The code snippet above is a representation of the WecGrid main loop for Steady-state WEC integration. The main script initializes a WecGrid object and runs a WEC-Sim simulation for each WEC pulls the SQL data and loops over the entire SQL data injecting each set of setpoints.

We can look at code listing I to see a representation of what the main loop looks like for WEC steady-state integration using AC injection. We begin by initializing a WEC-Grid object, WecGridObj. Next, we need to pass the case file in either raw or sav format for PSS®E, a power flow-solving algorithm, and buses to which we would like to add our WECs. Typical solvers are Newton-Raphson and Gauss-Seidel. The W2W model is called sequentially based on the desired amounts of WEC on the grid. The WEC-Sim call needs an identifier, simulation length, sample resolution, wave height, wave period, and wave seed. Currently, this part of the program is the most computationally heavy; however, it is significantly faster relative to running it via the MATLAB GUI. Data is pulled from the SQLite3 shared database. The datasets can then be reformatted and iterated over to perform the AC injection (Line 5). The function takes two arrays for active power and voltage of length WEC buses. We can also specify the time step value for different levels of quasi-steady state simulations.

IV. System Models

A. Wave-to-Wire System Modeling

The W2W system consists of the WEC and accompanying electronics to generate power and deliver it to the grid. The W2W model used in this work includes the WEC-Sim RM3 two-body point absorber open-source model [1] [17], PTO controller, direct-drive permanent magnet linear generator PTO connected to an active rectifier, generic onboard energy storage, and the grid power controller. A system diagram is shown in Fig. 3. The model is developed in MATLAB/Simulink and exports simulation results into a SQL database to be used by WEC-Grid asynchronously. To maintain a generic representation of grid-connected WECs, the only losses included in the W2W model are from the hydrodynamics within the WEC-Sim module and generator resistance.

Using measurements of the floating body velocity and position, the PTO subsystem implements a reactive controller covered in [18]. The force applied by the linear generator F_{PTO} is equal to the linear combination of velocity v and

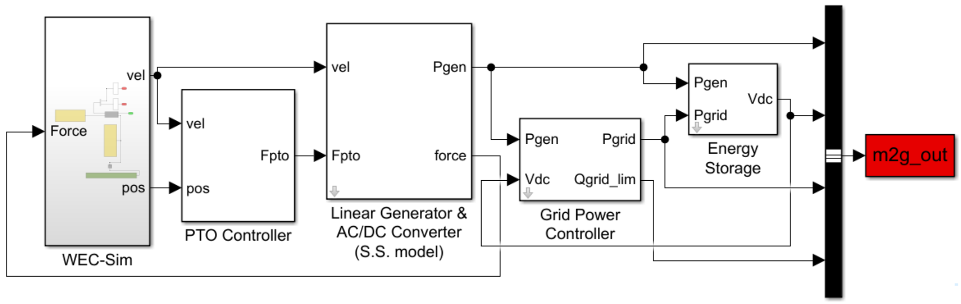


Fig. 3: Wave-to-Wire system diagram in Simulink.

position x scaled by damping coefficients B and K. The PTO force equation is given as

$$F_{PTO} = Bv + Kx \tag{1}$$

The direct-drive linear generator and active rectifier is modeled as a steady-state lumped system. The generator is modeled in the dq-reference frame following the methods in [19]. In steady-state conditions, the dq-frame currents i_d and i_q are constant and the generator equations become time-invariant. The force applied by a linear generator is calculated using (2) where τ_{pm} is the generator pole pitch, L is the dq-frame inductance, and λ_m is the permanant magnet flux linkage [5]. By equating (1) and (2), and setting generator reactive power to 0, closed-form solutions of the dq-frame currents and voltages are found. The active power output by the generator is found using the power-invariant dq-transformation relation $P_{gen} = v_d i_d + v_q i_q$.

$$F_{PTO} = \frac{\pi}{\tau_{pm}} \left((L i_d + \lambda_m) i_q - (L i_q) i_d \right) \tag{2}$$

The large peak-average power ratio produced by a WEC must be reduced to prevent stability issues caused by large power variations on the grid. The dc-link capacitor or energy storage device serves as a buffer to mitigate this variaton. The generic energy storage system is modeled as an ideal capacitor with capacitance C_{dc} and voltage V_{dc}, but can represent a variety of energy storage devices. The cycle-averaged voltage change equation (3) is a function of the power generated and delivered to the grid P_{grid} [20].

$$\frac{dV_{dc}}{dt} = \frac{P_{gen} - P_{grid}}{C_{dc} V_{dc}} \tag{3}$$

The W2W system is configured as a PV-bus in PSS®E. The W2W model must then supply the active power setpoint P_{grid}, voltage setpoint, and the reactive power limit Q_{lim} the W2W system is capable of supplying. This is computed in the grid power controller. P_{grid} is modulated to keep V_{dc} between user-configured upper and lower limits while also smoothing out P_{gen}. This is accomplished through an averaging unit with period T_{avg} and input P_{gen}, and a PI-controller to regulate V_{dc}. The equation form is omitted from the paper to save space. Reactive power limits are configured similar to the type-IV wind turbine model included in PSS®E [13], which is dependent on P_{grid}, the grid converter's current rating I_{lim}, and grid-voltage magnitude $|V|$. The positve reactive power limit Q_{lim} is calculated using (4), and the negative reactive power limit is simply the negative equivalent of (4).

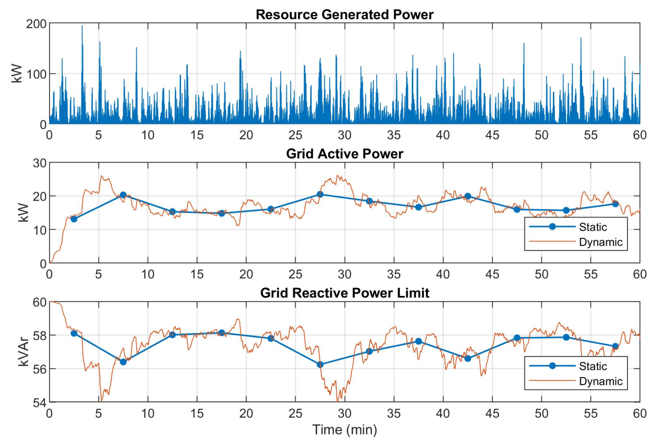


Fig. 4: Example output of the W2W Simulink model.

$$Q_{lim} = \sqrt{(|V| I_{lim})^2 - P_{grid}^2} \tag{4}$$

An example of the active power setpoints P_{grid} and reactive power limits Q_{lim} for the instantaneous WEC-generated power P_{gen} is shown in Fig. 4. The W2W Simulink model runs with a simulation time step of 0.1-seconds and the simulated waveforms are labeled as "dynamic" in Fig. 4. The waveforms are then downsampled to 5-minute intervals by averaging P_{gen} and Q_{lim} within the interval, labeled as "static" in the plots. The "static" timeseries are saved to a SQL database in per unit form, which is then accessed by WEC-Grid.

B. PSS®E Microgrids

The IEEE RTS 96 Area 1 test case was used for the demonstration of WEC integration. The test case grid comprises a 24-bus system with 11 generators and 16 loads. Fig. 5 is annotated to show the system swing bus and modified buses where WECs have been integrated. This is a standardized test case from IEEE for benchmarking purposes. This case is well understood and studied, which makes it a great candidate for WEC integration. For our current software iteration, we can process grids represented in raw and sav files.

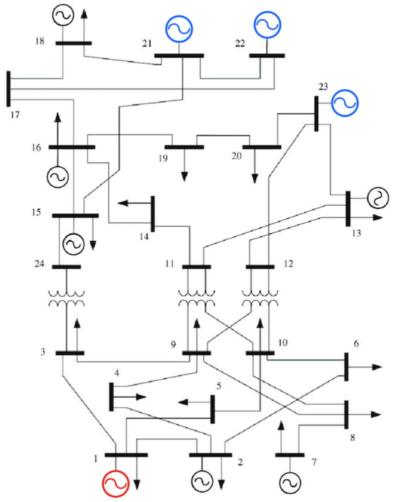


Fig. 5: One-line diagram of the IEEE RTS 96 bus test case (Area 1). The red generator indicates the swing bus. The blue generators at Buses 21, 22, and 23 represent the WECs that replace the traditional generators at those buses.

V. Results and Demonstration

Our target user is a power systems operator overseeing a microgrid wanting to integrate a WEC farm onto their power grid. We assume a goal of a 5-minute resolution to ensure the reliability of their system. This grid operator is able to use WEC-Grid to solve their grid over a quasi-steady state sliding window to understand the grid and WEC interaction. WEC-Grid can be a valuable tool in small microgrids where relatively small injections can produce large swings in the buses.

The WEC power generation simulation process is streamlined using the WEC-Grid class that configures and executes the W2W model within the Python environment. WEC-Sim includes many configuration options to customize WEC performance and the simulated scenario that are necessary for power system analysis. Thus, many of those details can be abstracted and simplified down to wave conditions, which have the largest impact on power generation quantity, quality, and stability. Using WEC-Grid, the power system operator simply selects the number of WECs connected to the grid and each of their wave conditions. This straightforward interface also allows more advanced features to be easily implemented, such as Monte Carlo analysis, where WEC simulations are run multiple times with randomly generated states.

Once the WEC data is pulled from the SQL database, the user runs the main injection loop, which can be seen in Code Listing 1, Line 5. Active power and voltage magnitude are injected from each WEC-bus time t, a new steady-state is solved, and the time step is moved forward. One can keep track of the steady-state values to understand how the system is behaving. The first step is initializing the system and solving for a steady state.

In Table I, we can see the initialized grid state of the IEEE RTS 96 24 bus case. The data frame is organized by calling the PSS®E API and pulling all the information a power systems engineer would typically need. Table I is a subset only containing half of the buses on the system. This is the initial solved state when the system is initialized. This can be considered to be a baseline of the system. In Table II, we can see a subset of WEC 1's output pulled from the SQL database and formatted.

In Fig. 6 and 7 we show example setpoints for active and reactive power of their respective buses after a steady-state solution was found for every time step. Each time step successfully converged using a Full Newton Raphson solver while injecting WEC electrical power at the selected WEC

TABLE I: Subset of the solved state output from WEC-Grid

Bus	PU	P	Q
Bus 1	1.00	35.93	180.49
Bus 2	1.00	-30.00	-20.00
Bus 3	0.86	-90.00	-19.00
Bus 4	0.89	-74.00	-15.00
Bus 5	0.93	-71.00	-14.00
Bus 6	0.91	-68.00	-14.00
Bus 7	0.80	2.00	-13.00
Bus 8	0.81	-85.00	-18.00
Bus 9	0.83	-175.00	-36.00
Bus 10	0.88	-100.00	-23.00

TABLE II: A subset of the WEC-Sim electrical output for WEC 1. The data was pulled from the SQL database and stored locally in a pandas data frame.

Time	WEC	Pg	PU
0.00	1	0.00	1.00
450.00	1	22.21	1.10
750.00	1	18.31	1.10
1050.00	1	17.62	1.10
1350.00	1	21.58	1.10
1650.00	1	19.59	1.10
1950.00	1	19.09	1.10
2250.00	1	14.12	1.10
2550.00	1	13.90	1.10
2850.00	1	17.17	1.10

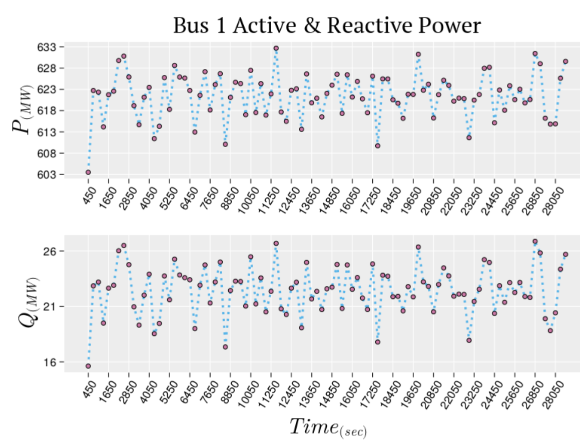


Fig. 6: Swing bus's active (P) and reactive (Q) power levels after steady state convergence over simulation time.

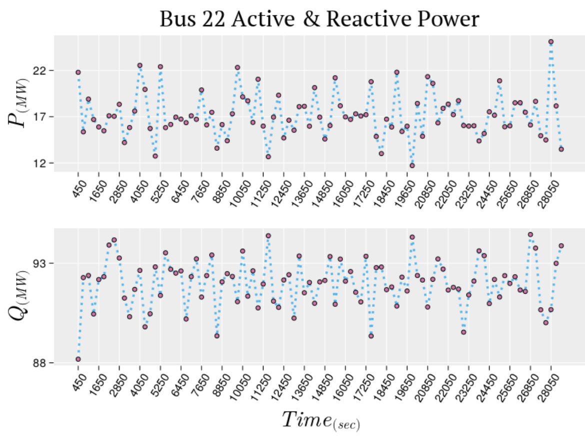


Fig. 7: Bus 22's active (P) and reactive (Q) power levels after steady state convergence over simulation time. This bus has the injection of WEC 2.

buses. For Bus 1 or the system swing bus, we can see a stable behavior of a swing bus over the 6-hour period. The largest ramp-up of the system was approximately 20 MW with the system still converging while leveraging the wave energy sources. The Wec buses all show similar behavior and active power output between the ranges of 11 - 23 MW. We used the modified RM3 model for each WEC simulation with the electrical input scaled by a factor of 10^4 to resemble a relatively sized WEC farm electrical output. PSS®E was able to successfully converge at every time step which shows the successful integration of the WECs onto the RTS 96 24-bus grid using WEC-Grid.

VI. Conclusion

In this work, we have demonstrated a software framework that integrates wave energy converters into PSS®E's power flow solver through a power conversion stage consisting of an electrical generator, short-term energy storage, and power electronics. This enables power system operators with limited knowledge of WEC modeling, power electronics, MATLAB and Python scripting, and SQL data storage to add renewable wave energy onto a microgrid via our open source software framework. This represents a significant step towards making wave energy integration more accessible and practical. Our work lays the foundation for future research into dynamic modeling simulations for wave energy integration. Specifically, we plan to develop a dynamically aware feature of WEC-Grid that can solve the dynamics of the power system as a sliding window. This would allow for the benefits of dynamic simulation while maintaining a lower computational time compared to full dynamic simulation. Such a feature could help power system operators and researchers better understand and optimize the performance of WECs on the power grid. Overall, our study highlights the continued potential of WECs as a promising and reliable renewable energy source and provides a framework for their successful integration into the power grid.

Acknowledgment

This work was supported in part by the US Department of Energy, Office of Energy Efficiency and Renewable Energy, Water Power Technology Office under the grant number #DE-EE0009445

References

[1] K. Ruehl, D. Ogden, Y.-H. Yu *et al.*, "Wec-sim v5.0.1," 2022. [Online]. Available: https://zenodo.org/badge/latestdoi/20451353

[2] R. So, B. Bosma, K. Ruehl *et al.*, "Modeling of a Wave Energy Oscillating Water Column as a Point Absorber Using WEC-Sim," *IEEE Transactions on Sustainable Energy*, vol. 11, no. 2, pp. 851–858, 2020.

[3] N. Aji, E. Ekawati, and I. N. Haq, "The Simulation of Wave Energy Conversion by Floating Point Absorber Buoy in Indonesian Sea Waves," in *2021 International Conference on Instrumentation, Control, and Automation (ICA)*, 2021, pp. 112–117.

[4] J. S. Artal-Sevil, J. A. Domínguez, H. El-Shalakany *et al.*, "Modeling and simulation of a wave energy converter system. Case study: Point absorber," in *2018 Thirteenth International Conference on Ecological Vehicles and Renewable Energies (EVER)*, 2018, pp. 1–7.

[5] R. So, A. Simmons, T. Brekken *et al.*, "Development of PTO-Sim: A Power Performance Module for the Open-Source Wave Energy Converter Code WEC-Sim," in *Volume 9: Ocean Renewable Energy*. St. John's, Newfoundland, Canada: American Society of Mechanical Engineers, 2015, pp. 1–10.

[6] I. López, J. Andreu, S. Ceballos *et al.*, "Review of wave energy technologies and the necessary power-equipment," *Renewable and Sustainable Energy Reviews*, vol. 27, pp. 413–434, 2013.

[7] M. Jasinski, M. Malinowski, M. P. Kazmierkowski *et al.*, "Control of AC/DC/AC Converter for Multi MW Wave Dragon Offshore Energy Conversion System," in *2007 IEEE International Symposium on Industrial Electronics*, 2007, pp. 2685–2690.

[8] N. Müller, S. Kouro, J. Glaría *et al.*, "Medium-voltage power converter interface for Wave Dragon wave energy conversion system," in *2013 IEEE Energy Conversion Congress and Exposition*, 2013, pp. 352–358.

[9] H. Polinder and M. Scuotto, "Wave energy converters and their impact on power systems," in *2005 International Conference on Future Power Systems*, 2005, pp. 1–9.

[10] F. Wu, X.-P. Zhang, P. Ju *et al.*, "Modeling and Control of AWS-Based Wave Energy Conversion System Integrated Into Power Grid," *IEEE Transactions on Power Systems*, vol. 23, no. 3, pp. 1196–1204, 2008.

[11] T. Ahmed and A. Zobaa, "Offshore power conditioning system connecting arrays of wave energy converters to the electric power grid," in *8th International Conference on Advances in Power System Control, Operation and Management (APSCOM 2009)*, 2009, pp. 1–6.

[12] M. A. H. Rafi and J. Bauman, "A Comprehensive Review of DC Fast-Charging Stations With Energy Storage: Architectures, Power Converters, and Analysis," *IEEE Transactions on Transportation Electrification*, vol. 7, no. 2, pp. 345–368, 2021.

[13] S. PSS®E, *PSS®E Use Manual 35*, Siemens PSS®E, 2021. [Online]. Available: https://www.siemens.com/global/en.html

[14] I. MathWorks, "Matlab python api," 2021. [Online]. Available: https://www.mathworks.com/help/matlab/matlab-python-api.html

[15] I. Anaconda, "Anaconda distribution," 2023, accessed February 13, 2023. [Online]. Available: https://www.anaconda.com/products/distribution

[16] P. Jupyter, "Jupyter notebook," 2015. [Online]. Available: https://jupyter.org/

[17] V. S. Neary, M. Previsic, R. A. Jepsen *et al.*, "Methodology for Design and Economic Analysis of Marine Energy Conversion (MEC) Technologies," Sandia National Laboratories, Tech. Rep. SAND2014-9040, 2014.

[18] A. Maria-Arenas, A. J. Garrido, E. Rusu *et al.*, "Control Strategies Applied to Wave Energy Converters: State of the Art," *Energies*, vol. 12, no. 16, pp. 1–19, 2019.

[19] N. Mohan, *Advanced electric drives: analysis, control, and modeling using MATLAB/Simulink*. Hoboken, New Jersey: Wiley, 2014.

[20] E. Twining and D. G. Holmes, "Modelling grid-connected voltage source inverter operation," *Proc. Australasian Universities Power Engineering Conference (AUPEC)*, pp. 1–6, 2001.