

# Machine Learning-Based Power Flow Solver

## CEC Progress Report for HiPAS Project

Lily Buechler, Adithya Antonysamy, and David Chassin  
*SLAC National Accelerator Laboratory*

## 1 Introduction

Power system simulation engines are essential tools for power grid operation and planning by utility companies and academic researchers. Power system modeling is inherently challenging due to the wide range of timescales of grid operation. For example, voltage regulation happens on the timescale of seconds to minutes, while seasonal temperature variation occurs on the timescale of several months. Therefore, simulations are often run over a long horizon with a small simulation timestep, which may require millions of solver iterations. This can be extremely computationally expensive, especially for scenario-based analysis, and may require researchers to utilize high performance computing resources.

Many types of grid analysis use quasi-static timeseries (QSTS) simulation tools such as GridLAB-D [1] or OpenDSS. In quasi-static power system simulation, the inputs to the powerflow solver vary from one timestep to another, but the powerflow at each timestep is assumed to be static. In other words, the bus voltage at a specific timestep is only dependent on the power injections at that timestep and not any previous timestep. Powerflow solutions must be performed chronologically because of the time-dependence of various resources in the network (thermostatically controlled loads, voltage regulators, and capacitor banks), which prevents sequential solutions from being parallelized.

QSTS powerflow solvers utilize either Newton Raphson or fixed point iterative methods for solving the powerflow equations. These methods have been highly optimized and are already very efficient. Therefore, recent work on accelerating powerflow simulations have focused primarily on either reducing the number of powerflow solutions, reducing the problem complexity, or replacing the powerflow solver altogether. Approaches include reducing feeder size [2], parallelizing solutions through geographic or temporal decomposition [3], reducing the total number of powerflow solutions through variable timestepping [4], linearizing the powerflow equations, vector quantization [5], and approximating the powerflow mappings using machine learning-based methods [6, 7]. Temporal decomposition and variable timestepping can lead to errors in modeling the dynamic response of controllable devices to changes in voltage. Methods such as vector quantization, geographic decomposition, feeder simplification, linearization of the powerflow equations, and machine learning-based approximation can potentially lead to errors in the estimated voltage profile.

Past studies have shown that machine learning-based methods in particular can result in highly accurate approximation of the powerflow equations. These studies have focused primarily on linear regression [6], SVR [7], and neural networks with relatively simple architectures [8, 9]. These data-driven methods would be particularly useful in applications where real-time simulation is required. These models can also be used in cases where no network model is available but there is significant historical network data.

The objective of this project is to develop a machine learning-based powerflow solver that can replace or be used in tandem with a traditional powerflow solver to accelerate simulation speed. This requires further development of data-driven methods for powerflow approximation, validation of model accuracy, scalability, robustness, and computational complexity on a variety of network scenarios. The developed framework will also be implemented and validated in the GridLAB-D source code. This report gives an overview of current progress on model development and validation.

## 2 Powerflow Equations

The purpose of a powerflow solver is to perform the inverse powerflow mapping. That is, given the real and reactive power injections at each bus in the network, solve for the magnitude and phase of the voltage at each bus in the network. The GridLAB-D powerflow solver solves the powerflow equations using the current injection method [10] for three-phase unbalanced powerflow using Newton Raphson iteration. The powerflow equations can be expressed in the following form

$$\Delta I_{rk}^s = \frac{P_k^s V_{rk}^s + Q_k^s V_{mk}^s}{(V_{rk}^s)^2 + (V_{mk}^s)^2} - \sum_{i=1}^n \sum_t (G_{ki}^{st} V_{ri}^t - B_{ki}^{st} V_{mi}^t) \quad (1)$$

$$\Delta I_{mk}^s = \frac{P_k^s V_{mk}^s + Q_k^s V_{rk}^s}{(V_{rk}^s)^2 + (V_{mk}^s)^2} - \sum_{i=1}^n \sum_t (G_{ki}^{st} V_{mi}^t - B_{ki}^{st} V_{ri}^t) \quad (2)$$

where  $t$  and  $s$  index the phase such that  $s, t \in \{a, b, c\}$ ,  $n$  is the number of nodes in the network,  $V_{rk}^s$  and  $V_{mk}^s$  are the real and imaginary components of the voltage, and  $G_{ki}^{st}$  and  $B_{ki}^{st}$  are the real and imaginary components of the nodal admittance matrix. The real and reactive power injections at bus  $k$  for phase  $s$  is equal to the power generated  $P_{gk}^s$  minus the load  $P_{lk}^s$ :

$$P_k^s = P_{gk}^s - P_{lk}^s \quad (3)$$

$$Q_k^s = Q_{gk}^s - Q_{lk}^s \quad (4)$$

The dependency of the load on the voltage is modeled using a ZIP model

$$P_{lk}^s = \bar{P}_{lk}^s [a_{pk}^s + b_{pk}^s V_k^s + c_{pk}^s (V_k^s)^2] \quad (5)$$

$$Q_{lk}^s = \bar{Q}_{lk}^s [a_{qk}^s + b_{qk}^s V_k^s + c_{qk}^s (V_k^s)^2] \quad (6)$$

where  $a_{pk}^s + b_{pk}^s + c_{pk}^s = 1$  and  $a_{qk}^s + b_{qk}^s + c_{qk}^s = 1$ . The coefficients  $a_{pk}^s$ ,  $b_{pk}^s$ , and  $c_{pk}^s$  can be interpreted as the constant power, constant current, and constant impedance portions of the load. The voltage magnitude is given by  $V_k^s = |V_{rk}^s + jV_{mk}^s|$  and the voltage phase is given by  $a_k^s = \angle(V_{rk}^s + jV_{mk}^s)$ . It is assumed that the real and reactive power injections are specified at all buses except for the slack bus (e.g. there are  $n - 1$  PQ buses).

In the existing Newton Raphson-based GridLAB-D implementation, the powerflow solution is initialized with the solution from the previous timestep. Therefore, the number of

iterations it takes for the solver to converge to a solution is dependent on the magnitude of the change in the power injections between timesteps.

### 3 Data-driven Powerflow Estimation

Our objective is to learn a data-driven approximation of the physics-based powerflow mapping given in (1 - 6). We define separate mappings from a feature vector  $x$  to the voltage magnitude  $V_k^s$  and phase  $a_k^s$ :

$$V_k^s = f(x) \quad (7)$$

$$a_k^s = f(x) \quad (8)$$

The feature vector  $x$  is composed of the real and reactive power injections at nominal voltage at each PQ bus in the network:

$$x = [\bar{P}_2^a \ \bar{P}_2^b \ \bar{P}_2^c \ \dots \ \bar{P}_n^a \ \bar{P}_n^b \ \bar{P}_n^c \ \bar{Q}_2^a \ \bar{Q}_2^b \ \bar{Q}_2^c \ \dots \ \bar{Q}_n^a \ \bar{Q}_n^b \ \bar{Q}_n^c]^T \in \mathbb{R}^{6(n-1)} \quad (9)$$

Here we assume that node 1 is the slack bus where the voltage is fixed. If the power factor of each load and generator is constant over time, the real and reactive power injections are proportional. Therefore, the feature vector can be simplified to:

$$x = [\bar{P}_2^a \ \bar{P}_2^b \ \bar{P}_2^c \ \dots \ \bar{P}_n^a \ \bar{P}_n^b \ \bar{P}_n^c]^T \in \mathbb{R}^{3(n-1)} \quad (10)$$

The training set used to train the data-driven model is generated by running the standard powerflow solver for a length of time. It is important to design the inputs to this simulation such that the outputs are in the typical operating regime of the network and the data-driven model can be learned most efficiently.

One of the simplest models to consider is a mapping where the output variables  $\hat{V}_k^s$  and  $\hat{a}_k^s$  are a linear function of the features  $x$ :

$$\hat{V}_k^s = (\theta_{vk}^s)^T x + b_{vk}^s \quad (11)$$

$$\hat{a}_k^s = (\theta_{ak}^s)^T x + b_{ak}^s \quad (12)$$

The parameters  $\theta_{vk}^s$ ,  $\theta_{ak}^s$ ,  $b_{vk}^s$ , and  $b_{ak}^s$  are learned using L2 regularized linear regression, with the following loss functions

$$J_k^s(\theta_{vk}^s, b_{vk}^s) = \sum_{i=1}^m \left( V_k^{s(i)} - (\theta_{vk}^s)^T x^{(i)} - b_{vk}^s \right)^2 + \alpha_v \|\theta_{vk}^s\|_2^2 \quad (13)$$

$$J_k^s(\theta_{ak}^s, b_{ak}^s) = \sum_{i=1}^m \left( a_k^{s(i)} - (\theta_{ak}^s)^T x^{(i)} - b_{ak}^s \right)^2 + \alpha_a \|\theta_{ak}^s\|_2^2 \quad (14)$$

where  $x^{(i)}$  is the feature vector corresponding with the  $i^{th}$  training example.  $V_k^{s(i)}$  and  $a_k^{s(i)}$  are the magnitude and phase of the voltage of the  $k^{th}$  bus for phase  $s$  for the  $i^{th}$  training example. Regularization coefficients  $\alpha_v$  and  $\alpha_a$  can be selected using cross-validation.

Model performance was compared to that of a baseline model, which assumes that the

Table 1: Networks simulated in GridLAB-D and the corresponding size of the input and output variables of the ML algorithm for each case.

Network	# of Nodes	Voltage dimension	Power dimension
IEEE 4 bus	4	12	3
IEEE 13 bus	13	48	20
IEEE 123 bus	123	402	95
PNNL GC-12.47-1	27	108	9
PNNL R1-12.47-3	52	297	20
PNNL R2-12.47-2	250	2553	22

predicted voltage magnitude and phase is equal to the mean values in the training set:

$$\hat{V}_k^{s(j)} = \frac{1}{m} \sum_{i=1}^m V_k^{s(i)} \quad (15)$$

$$\hat{a}_k^{s(j)} = \frac{1}{m} \sum_{i=1}^m a_k^{s(i)} \quad (16)$$

## 4 Power Network Models

Six different networks were evaluated: three different IEEE test feeders [11] (4 bus, 13 bus and 123 bus) and three different PNNL taxonomy feeders [12]. The sizes of the networks and the dimension of inputs and outputs for the corresponding powerflow problem are shown in Table 1. These models were selected in order to include a variety of networks of different sizes and characteristics. Three of the networks (IEEE 123, GC-12.47-1, and R2-12.46-2) have both voltage regulators and capacitor banks, two networks have voltage regulators (IEEE 13 and R1-12.47-3), and one network has neither (IEEE 4).

The static spot loads in the original feeder models were replaced by time-varying load profiles. These profiles were generated from 1-minute Pecan Street residential load data [13] from July 2018. Each profile is defined as the sum of a specified number of randomly selected individual home profiles, which is then scaled such that the mean is equal to the original spot load and the standard deviation is equal to a specified fraction of the original spot load. The power factor of the load and the coefficients of the ZIP model were held constant for each individual simulation, but varied for different scenarios.

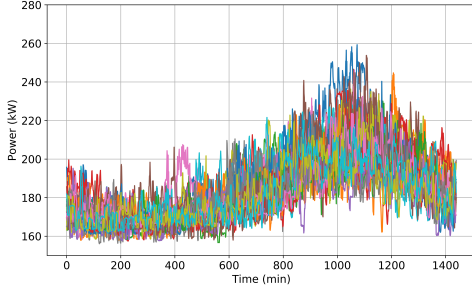


Figure 1: Example nominal power profiles used to populate the IEEE 13 bus network.

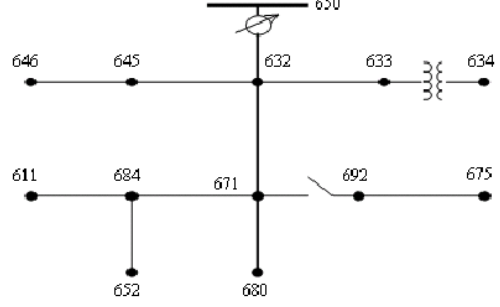


Figure 2: Network diagram of IEEE 13 bus network [11].

Each simulation was run for 15 days (7/1/18-7/15/18) using a simulation timestep of 1 minute. The first day of simulation was discarded, days 2-7 (6 days) were used for model training, and days 8-15 (8 days) were used for model testing.

## 5 Performance Metrics

The performance of the prediction algorithm can vary by bus as the voltage variation depends highly on the location in the network. The prediction accuracy also can vary significantly with respect to time, as the loading of the network changes. Therefore, we use performance metrics that are based on the least accurate prediction in the network, and analyze the distribution of that error over time.

Let  $V^{(i)}$  and  $\hat{V}^{(i)}$  be the vectors of the actual and predicted voltage magnitudes for the  $i$ th example in the test set. Then the normalized voltage magnitude error for example  $i$  is defined as

$$\epsilon_v^{(i)} = \left\| \frac{\hat{V}^{(i)} - V^{(i)}}{V^{(i)}} \right\|_{\infty} \quad (17)$$

Similarly, let  $a^{(i)}$  and  $\hat{a}^{(i)}$  be the vectors of the actual and predicted voltage phases for the  $i$ th example in the test set. Then the absolute voltage phase error is defined as

$$\epsilon_a^{(i)} = \|\hat{a}^{(i)} - a^{(i)}\|_{\infty} \quad (18)$$

Therefore  $\epsilon_v^{(i)}$  and  $\epsilon_a^{(i)}$  are equal to the voltage magnitude and phase errors for the bus in the network with the least accurate predictions for sample  $i$ . The distribution of  $\epsilon_v^{(i)}$  and  $\epsilon_a^{(i)}$  over all samples in the test set gives information about the accuracy and robustness of the method.

## 6 Results

The mean voltage magnitude and phase prediction errors for different networks and loading scenarios are shown in Tables 2 and 3 and Fig. 3. Five days of training data were used to fit each model. The simulations were run using the default ZIP model parameters in the test feeder models and a load power factor of 0.9. As shown, the prediction error for both the voltage magnitude and phase increases with the level of loading. The power-voltage behavior is fairly linear for small power injections, such that linear regression gives a very accurate approximation of the behavior. At higher loading levels the behavior becomes more nonlinear. For all networks, linear regression is significantly more accurate than the baseline averaging method, often by one or two orders of magnitude. However, the error magnitude can vary by network. This is because the specification of the spot load values are not consistent from one network to another, which results in differing degrees of linearity of the powerflow equations at the spot load value. For example, the spot loads of for the PNNL GC-12.47-1 network represent a very lightly loaded feeder relative to the other networks. A mean value of the normalized voltage magnitude prediction error of less than 0.001 is likely sufficient for many simulation applications.

Table 2: Mean voltage magnitude prediction error ( $\epsilon_v$ ) for the test set as a function of the standard deviation of the load for different networks. Five days of training data were used to fit each model.

Network	Load standard deviation (as % of spot load)				
	5%	10%	20%	30%	40%
IEEE 4 bus	0.00045	0.00188	-	-	-
IEEE 13 bus	-	0.00009	0.00032	0.00071	0.00125
IEEE 123 bus	-	0.00011	0.00128	0.00739	-
PNNL GC-12.47-1	-	$1.74 \cdot 10^{-6}$	$2.84 \cdot 10^{-6}$	$5.97 \cdot 10^{-6}$	$7.10 \cdot 10^{-6}$
PNNL R1-12.47-3	-	0.00565	0.01175	0.02047	0.02541
PNNL R2-12.47-2	-	0.00248	0.00395	0.00423	0.00455

Results indicate that there are not significant benefits from training linear regression with large amounts of data. Fig. 4 shows the mean normalized voltage magnitude error ( $\epsilon_v$ ) and the mean absolute voltage phase error ( $\epsilon_a$ ) as a function of the training set size for all six networks. For these simulations, the power factor of the load was set to 0.90, the standard deviation of the load was set to 10% of the spot load value, and the load ZIP parameters were set equal to their default values. Results for all networks show that there are not significant benefits from using more than 0.4 days (approximately 600 training examples) of data to train the model. However, it is important that these datapoints sufficiently cover the portion of the operating regime of the network that is fairly linear.

Table 3: Mean voltage phase prediction error ( $\epsilon_a$ ) for the test set as a function of the standard deviation of the load for different networks. Five days of training data were used to fit each model.

Network	Load standard deviation (as % of spot load)				
	5%	10%	20%	30%	40%
IEEE 4 bus	0.02353	0.09381	-	-	-
IEEE 13 bus	-	0.00435	0.01770	0.03660	0.06969
IEEE 123 bus	-	0.00799	0.04365	0.20409	-
PNNL GC-12.47-1	-	0.00006	0.00009	0.00016	0.00022
PNNL R1-12.47-3	-	0.34517	0.68771	1.17677	1.44059
PNNL R2-12.47-2	-	0.00557	0.04136	0.05580	0.09325

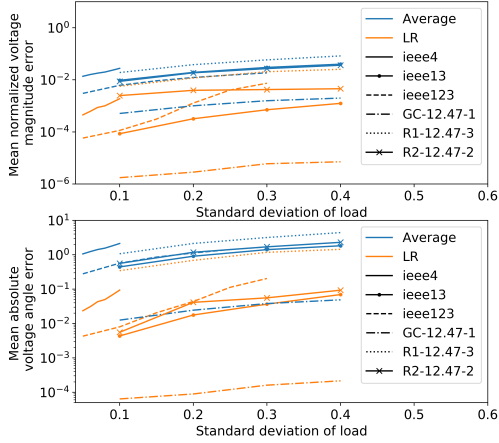


Figure 3: Mean prediction error as a function of the standard deviation of the load for regularized linear regression and the averaging baseline for different networks.

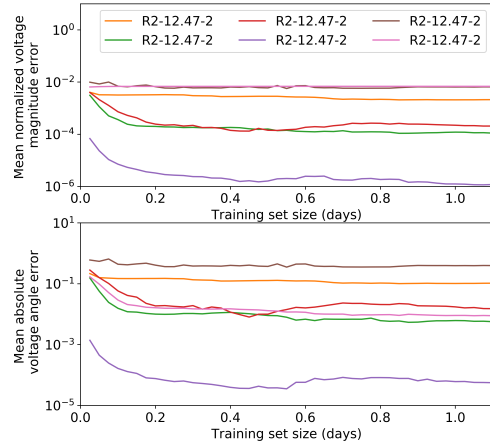


Figure 4: Mean prediction error as a function of training set size for regularized linear regression for different networks.

The voltage magnitude in distribution feeders is lower further away from the slack bus. Results show that linear regression is less accurate for larger voltage deviations, and therefore is most accurate at the head of the feeder. This can be observed in Fig. 5, which shows the voltage magnitude prediction error for each phase at each node in the network. The marker color represents the phase of the voltage, and the marker size represents the magnitude of the prediction error. As shown the prediction error is correlated with the location in the network, and varies slightly between the three different phases.

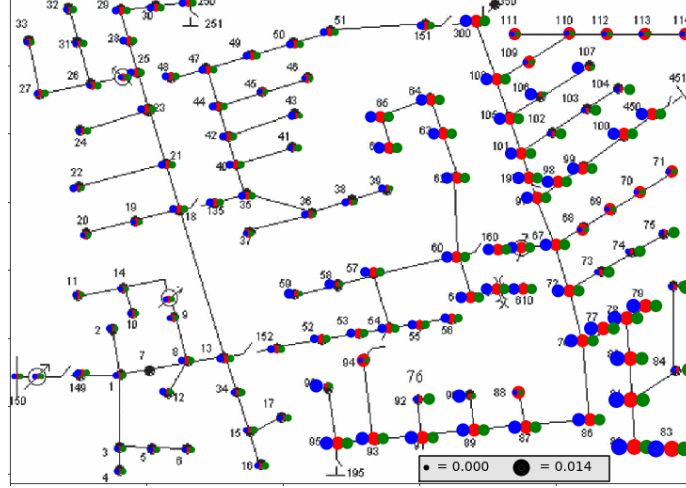


Figure 5: Mean voltage magnitude error in IEEE 123 network. The color indicates the phase (blue=a, red=b, green=c) and marker size indicates the magnitude. The slack bus is node 150, located in the lower left corner.

Results suggest that variation in the load composition only slightly affects the voltage magnitude or phase error. Fig. 6 shows a histogram of the voltage magnitude test error for the IEEE 123 network for three different load compositions: constant impedance ( $c_{pk}^s = 1$ ,  $c_{qk}^s = 1$ ), constant current ( $b_{pk}^s = 1, b_{qk}^s = 1$ ), and constant power ( $a_{pk}^s = 1$ ,  $a_{qk}^s = 1$ ) loads. While the mean values of the three distributions are nearly identical, there is a small difference in the length of the tail of the distribution. Variation in the power factor has a similarly small affect on prediction error.

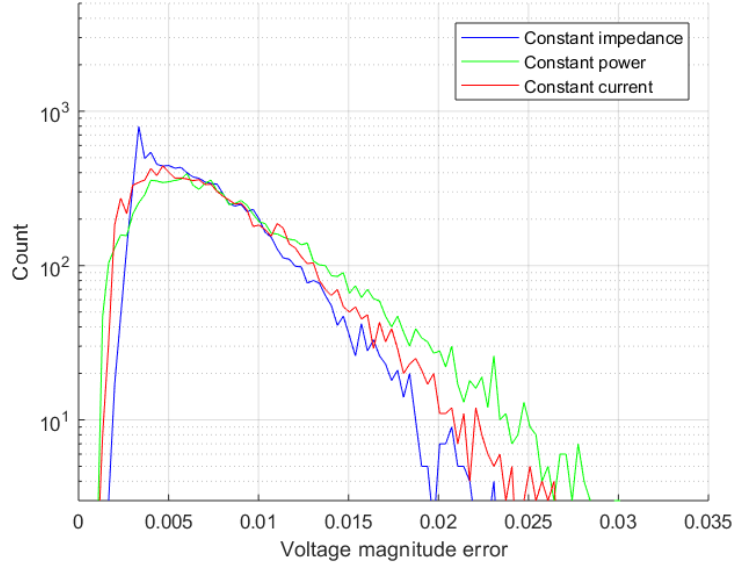


Figure 6: Histogram of the voltage magnitude prediction test error for linear regression for different load compositions for the IEEE 123 bus network.



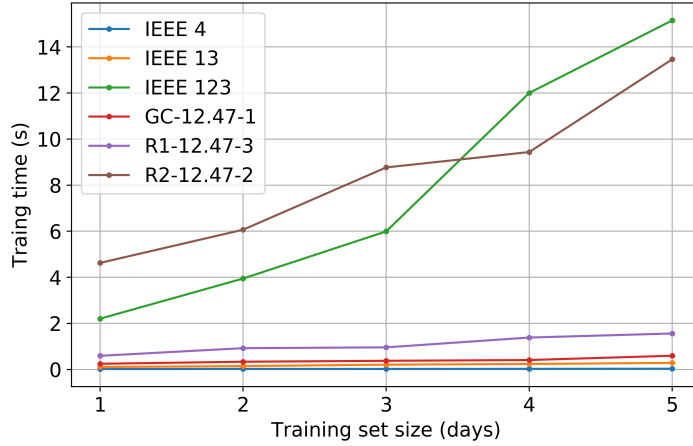


Figure 7: Model training time for linear regression as a function of the training set size (1440 samples per day) for different networks.

The primary advantage from using a simple model such as linear regression for powerflow approximation is the fast computation time for parameter estimation. Fig. 7 shows the training time required for different sizes of training sets for the six different networks. Computation time scales approximately linearly with the size of the training set. However as previously discussed, less than one day of data is generally required to learn an accurate model. The model training time for this amount of training data ( $< 10$  s) is negligible compared to the amount of time generally required for powerflow simulation. Computation time also increases with the size of the input and output dimension of the powerflow problem.

## 7 Conclusions and Future Work

This analysis demonstrates the performance of linear regression-based models for learning the inverse powerflow mapping under a variety of network and loading scenarios. Because the powerflow mapping for most networks is generally quite linear under low loading conditions, linear regression can provide a very accurate approximation of the powerflow equations under certain conditions. The small computational expense of fitting linear regression models offers significant benefits in terms of speeding up traditional powerflow solvers. These findings will help inform the development of more complex models, such as support vector regression (SVR) and neural networks, in the future.

In addition to model development, future work will also focus on integrating methods directly into the GridLAB-D source code. Online implementation will require developing methods generating appropriate training sets for each network, as well as algorithms to determine the region of the operating space where the learned model is accurate. This will enable adaptively switching between using the traditional solver and the machine learning-based approximation depending on the inputs to the powerflow problem.

One limitation of the proposed solver is that the accuracy of the method cannot be strictly guaranteed given a simulation problem. One method that addresses this is using the

machine learning-based approximation in order to seed the Newton Raphson solver. The standard approach is to initialize the Newton Raphson solver with the solution from the previous iteration. However, the number of NR iterations could be significantly reduced by initializing the solver more accurately using the machine learning-based estimate.

## References

- [1] D. P. Chassin, K. Schneider, and C. Gerkenmeyer, “Gridlab-d: An open-source power systems modeling and simulation environment,” in *2008 IEEE/PES Transmission and Distribution Conference and Exposition*. IEEE, 2008, pp. 1–5.
- [2] Z. K. Pecanak, V. R. Disfani, M. J. Reno, and J. Kleissl, “Multiphase distribution feeder reduction,” *IEEE Transactions on Power Systems*, vol. 33, no. 2, pp. 1320–1328, 2017.
- [3] D. Montenegro, R. C. Dugan, and M. J. Reno, “Open source tools for high performance quasi-static-time-series simulation using parallel processing,” in *2017 IEEE 44th Photovoltaic Specialist Conference (PVSC)*. IEEE, 2017, pp. 3055–3060.
- [4] M. J. Reno and R. J. Broderick, “Predetermined time-step solver for rapid quasi-static time series (qsts) of distribution systems,” in *2017 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*. IEEE, 2017, pp. 1–5.
- [5] J. Deboever, S. Grijalva, M. J. Reno, and R. J. Broderick, “Fast quasi-static time-series (qsts) for yearlong pv impact studies using vector quantization,” *Solar Energy*, vol. 159, pp. 538–547, 2018.
- [6] S. Powell, A. Ivanova, and D. Chassin, “Fast solutions in power system simulation through coupling with data-driven power flow models for voltage estimation,” *arXiv preprint arXiv:2001.01714*, 2020.
- [7] J. Yu, Y. Weng, and R. Rajagopal, “Robust mapping rule estimation for power flow analysis in distribution grids,” in *2017 North American Power Symposium (NAPS)*. IEEE, 2017, pp. 1–6.
- [8] A. Karami and M. Mohammadi, “Radial basis function neural network for power system load-flow,” *International Journal of Electrical Power & Energy Systems*, vol. 30, no. 1, pp. 60–66, 2008.
- [9] H. H. Muller, M. J. Rider, C. A. Castro, and V. L. Paucar, “Power flow model based on artificial neural networks,” in *2005 IEEE Russia Power Tech*. IEEE, 2005, pp. 1–6.
- [10] P. A. Garcia, J. L. R. Pereira, S. Carneiro, V. M. da Costa, and N. Martins, “Three-phase power flow calculations using the current injection method,” *IEEE Transactions on Power Systems*, vol. 15, no. 2, pp. 508–514, 2000.
- [11] W. H. Kersting, “Radial distribution test feeders,” *IEEE Transactions on Power Systems*, vol. 6, no. 3, pp. 975–985, 1991.
- [12] K. P. Schneider, Y. Chen, D. P. Chassin, R. G. Pratt, D. W. Engel, and S. E. Thompson, “Modern grid initiative distribution taxonomy final report,” Pacific Northwest National Lab.(PNNL), Richland, WA (United States), Tech. Rep., 2008.
- [13] Pecan Street Inc. (2017) Dataport from pecan street. [Online]. Available: <https://dataport.cloud/>