PGLEARN – AN OPEN-SOURCE LEARNING TOOLKIT FOR OPTIMAL POWER FLOW

Anonymous authors
Paper under double-blind review

ABSTRACT

Machine Learning (ML) techniques for Optimal Power Flow (OPF) problems have recently garnered significant attention, reflecting a broader trend of leveraging ML to approximate and/or accelerate the resolution of complex optimization problems. These developments are necessitated by the increased volatility and scale in energy production for modern and future grids. However, progress in ML for OPF is hindered by the lack of standardized datasets and evaluation metrics, from generating and solving OPF instances, to training and benchmarking machine learning models. To address this challenge, this paper introduces PGLearn, a comprehensive suite of standardized datasets and evaluation tools for ML and OPF. PGLearn provides datasets that are representative of real-life operating conditions, by explicitly capturing both global and local variability in the data generation, and by, for the first time, including time series data for several large-scale systems. In addition, it supports multiple OPF formulations, including AC, DC, and second-order cone formulations. Standardized datasets are made publicly available to democratize access to this field, reduce the burden of data generation, and enable the fair comparison of various methodologies. PGLearn also includes a robust toolkit for training, evaluating, and benchmarking machine learning models for OPF, with the goal of standardizing performance evaluation across the field. By promoting open, standardized datasets and evaluation metrics, PGLearn aims at democratizing and accelerating research and innovation in machine learning applications for optimal power flow problems.

1 INTRODUCTION

The rapid evolution of energy systems, driven by mass integration of renewable and distributed energy resources, is creating new challenges in the maintenance, expansion, and operation of power grids. The increased volatility and scale of power generation in modern and future grids calls for innovative solutions to manage uncertainty and ensure reliability (Zhang et al., 2021). Machine learning (ML) has emerged as a powerful tool in this context, offering the potential to address key problems such as optimizing grid operations and predicting power demand, enabling the use of previously intractable applications such as real-time risk analysis (Chen et al., 2024). However, the success of ML models depends on the availability of high-quality data, which is essential for training accurate and reliable models (Khaloie et al., 2024; Lovett et al., 2024).

Optimal Power Flow (OPF) is a fundamental problem in power systems operations, focusing on how to efficiently operate a power transmission system while satisfying physics, engineering, and operations constraints. Most market-clearing algorithms for real-time electricity markets are based on OPF, which makes it paramount to real-time operations. In addition, OPF forms the building block of security-constrained unit commitment (SCUC) formulations used in day-ahead markets (Chen et al., 2023), as well as transmission expansion planning problems. In practice, many instances of OPF need to be solved at once in order to account for uncertainties in renewable power generation and/or demand, making it a computationally intensive task – especially given the non-convex physics of AC power flow. Besides its real-world applications, OPF has also garnered significant attention as a test-bed for research methods integrating mathematical programming and machine learning.

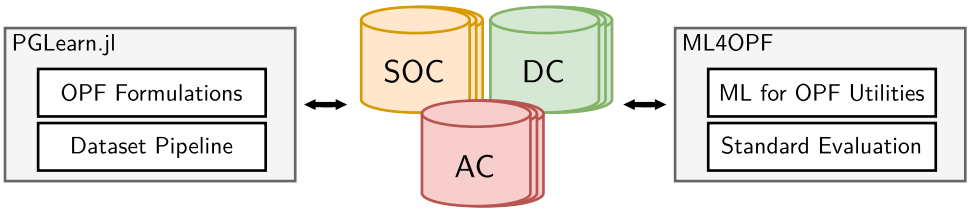

Figure 1: The PGLearn Toolkit: publicly available AC, DC, and SOC optimal power flow datasets, `PGLearn.jl` for data generation, and the `ML4OPF` ML toolkit.

The key motivation for using ML to address parametric OPF stems from the increased volatility and scale in power generation in modern and future grids. Indeed, the growing use of intermittent renewable energy sources such as wind and solar generation is driving significant growth in operational uncertainty. This motivates a shift from deterministic to, e.g., stochastic optimization formulations that explicitly consider uncertainty. Such a change results in OPF problems that are, or will be, orders of magnitude larger than today's instances. In addition, the risk of energy shortage, congestion, and voltage issues has become substantially larger and requires novel methods to manage in real-time. Machine learning offers some hope in addressing these challenges by moving much of the computational burden offline and delivering orders of magnitude speedups for real-time operations.

This research avenue is further justified by the fact that practitioners often solve OPF instances on very similar – or even identical – transmission systems, with only renewable generation and/or power demand varying across instances. Readers are referred to Hasan et al. (2020); Khaloie et al. (2024) for a detailed review of prior works in machine learning for OPF. Note that many of the works therein do not consider the non-convex AC-OPF formulation directly, but rather focus on more tractable, i.e., convex, OPF formulations, such as the DC-OPF linear approximation or second-order cone relaxation (Molzahn and Hiskens, 2019). Although these relaxed formulations do not capture the exact physics, they more closely match the problems that real power market operators and participants solve every day (Ma et al., 2009). For example, Chen (2023) uses a linear formulation inspired by problem solved by the Midcontinent Independent System Operator (MISO) to clear its real-time market.

1.1 Motivation: Data Scarsity in ML for OPF Research

Despite strong interest from industry, real, industry-scale data is scarce, mainly due to regulatory barriers that restrict the sharing of sensitive information on power grids. Thus, most previous ML for OPF works generate their own artificial datasets, often based on the Power Grid Lib Optimal Power Flow (PGLib-OPF) (Babaeinejadsarookolaee et al., 2019) benchmark library – a collection of grid snapshots originally designed for benchmarking AC-OPF optimization algorithms.

While machine learning methods usually require many thousands of data points to train accurate models, PGLib-OPF only provides a single snapshot per grid. Hence, a data augmentation strategy is needed to "sample around" the provided snapshots in a realistic fashion. There is however no consensus in prior literature for how to perform this sampling, which has led to a highly fragmented ecosystem where it is impossible to directly compare results from different works due to the use of very different data distributions (Khaloie et al., 2024). Indeed, the characteristics of the learning problem may vary substantially when considering different augmentation schemes, for example correlated versus uncorrelated noise.

To ensure that ML for OPF research is useful in practice, it is important to carefully consider the choices involved in designing a data augmentation scheme. For example, most works surveyed in Khaloie et al. (2024) sample instances by perturbing individual loads independently of each other. Figure 2 illustrates the limitations the resulting data distribution, for a system with 1354 buses. The figure shows that, i) the resulting distribution displays a very narrow range of total demand, and ii) the resulting OPF solutions exhibit simplistic patterns that do not capture global dynamics over a

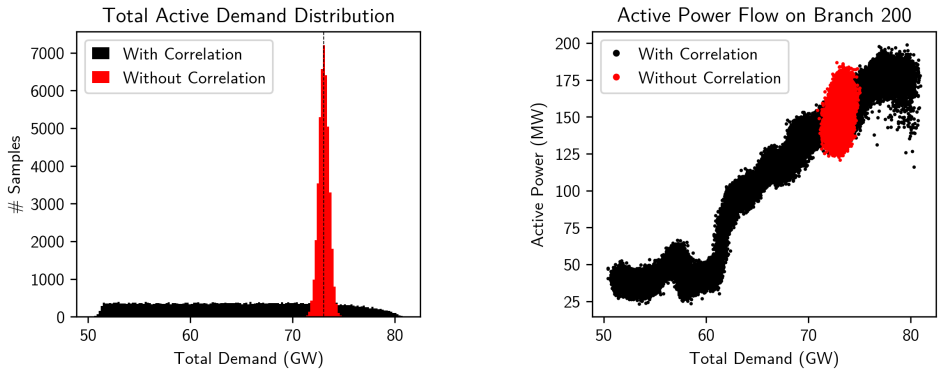

Figure 2: Limitations of sampling strategies that do not consider correlations across individual loads. Left: histogram of total demand: the absence of correlations yields a narrow range of total demand. Right: active power flow on branch 200; the absence of correlations in input data leads to datasets with low variance and diversity.

wider range of total demand. As a consequence, ML models trained on data that does not capture correlations can only be expected to perform well for a small total demand range, severely limiting their usability in practice.

Finally, the lack of publicly available standardized datasets requires individual teams to expend considerable computational resources to generate new datasets. This comes at a high financial and environmental cost, and results in most academic studies considering small, synthetic power grids which incur lower data generation costs, but are irrelevant to industry practitioners. More importantly, it represents a significant barrier to entry to teams without substantial computational resources.

1.2 Related Work

Due to strong interest from academia and industry, several OPF datasets and data generation packages have been released in recent years. However, none simultaneously meet the requirements of being actively maintained, considering large power networks, and using realistic sampling schemes.

Some prior works (Donon et al., 2020; Chatzos et al., 2022; Klamkin et al., 2024) report results on industry-based datasets, i.e., using data obtained from transmission system operators. Although these works report more closely match real-world systems, the corresponding datasets are not publicly available due to regulations around privacy and security.

Despite a growing literature on ML for OPF, few datasets have been made publicly available. The datasets initially released alongside the OPFSampler (Robson et al., 2019) codebase are no longer available, and the code is unmaintained. The OPFLearn library (Joswig-Jones et al., 2022) provides data augmentation tools built on top of PowerModels Coffrin et al. (2018), as well as a collection of 10,000 samples of AC-OPF instances and their solution for five systems with up to 118 buses. This code is no longer maintained, only considers uncorrelated demand perturbations, and reports incomplete primal/dual solutions. More recently, and closest to this paper, OPFData (Lovett et al., 2024) is a collection of AC-OPF datasets which considers systems with up to 13,659 buses. The collection also includes instances with perturbed topology, obtained by randomly removing individual lines or generators in the system. The main limitation of OPFData, however, is that it only considers uncorrelated demand perturbation, leading to simplistic data distributions as illustrated in Figure 2. Additionally, OPFData only reports AC-OPF solutions, does not include dual solutions nor metadata such as solve times, and does not include the source code used to generate and solve samples.

1.3 Contributions and Outline

To address the above challenges, this paper proposes PGLearn, a collection of datasets and tools for ML and OPF. The contributions of PGLearn are as follows:

- PGLearn provides a collection of standardized datasets for large-scale OPF instances, generated using a realistic and reproducible data augmentation methodology. The entire collection totals over 10,000,000 OPF samples, split between training and testing data to allow for direct comparison of results. Crucially, PGLearn is also the first OPF dataset to incorporate realistic time-series data for large-scale power systems.
- Each PGLearn dataset comprises complete primal and dual solutions for several OPF formulations, a unique feature compared to existing literature.
- PGLearn provides several evaluation metrics for objective and fair comparison of various ML methodologies for OPF problems, together with guidelines on performance benchmarking.
- The `PGLearn.jl` Julia (Bezanson et al., 2017) library containing the source code to generate the PGLearn datasets. The modular design of `PGLearn.jl` simplifies the implementation and execution of new data augmentation schemes and OPF formulations.
- The `ML4OPF` PyTorch (Ansel et al., 2024) library containing data parsers, optimized GPU-friendly implementations of the supported OPF formulations (objective, constraints, etc.), and other utilities for developing new ML methods for OPF.

The code used to generate PGLearn is fully open-source and relies only on open-source solvers, allowing to interrogate, reproduce, and extend each part of the dataset generation process. By openly distributing these tools and datasets, PGLearn seeks to lower the barrier of entry for researchers in the field, promoting innovation and accelerating the development of ML techniques for OPF and optimization more broadly.

The rest of this paper is structured as follows. Section 2 describes each OPF formulation included in PGLearn. Section 3 introduces the data augmentation procedure, and Section 4 presents relevant features of the `PGLearn.jl` and `ML4OPF` libraries. Section 5 provides recommendations for evaluation metrics and their reporting. Section 6 reviews the limitations of PGLearn, and Section 7 concludes the paper.

2 OPF Formulations in PGLearn

A unique feature of PGLearn is that it provides, for each OPF instance, solutions to several OPF formulations. This allows to compare, for the same input data, the performance of ML models trained using different formulations. Namely, PGLearn currently supports the nonlinear, non-convex AC-OPF, the second-order cone relaxation SOC-OPF, and the linear approximation DC-OPF. A brief summary of each formulation is provided below. Due to space considerations, full OPF formulations are stated in Appendix A.

2.1 OPF Formulations

AC-OPF The AC-OPF is considered the "full" steady-state optimal power flow formulation. PGLearn uses the rectangular-power polar-voltage form, matching the `ACPPowerModel` formulation implemented in PowerModels (Coffrin et al., 2018). This formulation includes non-convex AC power flow physics to accurately model the power system. The full non-linear programming formulation is included in Model 1.

SOC-OPF The SOC-OPF is a second-order-cone relaxation of the AC-OPF proposed by Jabr (2006a). The SOC-OPF better approximates the full-physics AC-OPF compared to the linear DC-OPF, but is more complicated to solve. A description of how to derive the SOC-OPF, and its full conic programming formulation, is included with Model 2.

DC-OPF The DC-OPF is a sparse linear approximation to the AC-OPF (Christie et al., 2000). It is commonly used in industry to approximate AC-OPF in cases where solving AC-OPF within time constraints is intractable. Among other simplifications, it considers only active power and fixes all voltage magnitudes to 1. A list of all assumptions required to derive the DC-OPF, and its full linear programming formulation, is included with Model 4.

Table 1: Summary statistics of the PGLearn datasets.

Case name	$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{L}\rvert$	$\lvert\mathcal{G}\rvert$	$\lvert\mathcal{E}\rvert$	Total PD	Total PG	Global range
`14_ieee`	14	11	5	20	0.3 GW	0.4 GW	70% – 110%
`30_ieee`	30	21	6	41	0.3 GW	0.4 GW	60% – 100%
`57_ieee`	57	42	7	80	1.3 GW	2 GW	60% – 100%
`89_pegase`	89	35	12	210	6 GW	10 GW	60% – 100%
`118_ieee`	118	99	54	186	4 GW	7 GW	80% – 120%
`300_ieee`	300	201	69	411	24 GW	36 GW	60% – 100%
`1354_pegase`	1354	673	260	1991	73 GW	129 GW	70% – 110%
`NewYork2030`	1576	1446	323	2427	33 GW	42 GW	70% – 110%
`1888_rte`	1888	1000	290	2531	59 GW	89 GW	70% – 110%
`2869_pegase`	2869	1491	510	4582	132 GW	231 GW	60% – 100%
`6470_rte`	6470	3670	761	9005	97 GW	118 GW	60% – 100%
`Texas7k`	6717	4541	637	9140	75 GW	97 GW	80% – 120%
`9241_pegase`	9241	4895	1445	16049	312 GW	530 GW	60% – 100%
`13659_pegase`	13659	5544	4092	20467	381 GW	981 GW	60% – 100%
`Midwest24k`	23643	11727	5646	33739	104 GW	318 GW	90% – 130%

2.2 Dual OPF Formulations and Solutions

Another unique feature of PGLearn is that it provides, for each OPF formulation, complete primal and dual solutions. This novel capability is motivated by the recent interest in leveraging dual information in ML contexts. For instance, Qiu et al. (2024) and Tanneau and Van Hentenryck (2024) both consider learning dual optimization proxies, wherein an ML model outputs dual-feasible solutions to conic optimization problems. In a similar fashion, Kotary and Fioretto (2024) leverage insights from Augmented Lagrangian methods to learn Lagrange multipliers for nonlinear problems. Finally, several recent works attempt to predict dual solutions in the context of mixed-integer optimization, with the aim of obtaining high-quality dual bounds through Lagrangian duality Parjadis et al. (2023); Demelas et al. (2024).

PGLearn leverages Lagrangian duality for nonlinear, non-convex problems such as AC-OPF, and conic duality for convex relaxations and approximations such as SOC-OPF and DC-OPF. Dual formulations for SOC-OPF and DC-OPF are stated in Appendix A. Note that a key advantage of conic (convex) relaxations is that dual-feasible solutions provide valid certificates of optimality, which can be used to validate the quality of primal predictions.

Dual solutions are also at the core of price formation in electricity markets. Hence, by systematically providing dual solutions, PGLearn will support future research at the intersection of optimization, machine learning, and the economics of electricity markets.

3 The PGLearn Collection of Datasets for Learning OPF

Like most prior work in the field, PGLearn uses a sampling scheme to convert static snapshots to datasets of OPF instances. PGLearn considers a total of 14 snapshots, split into four categories based on the number of buses:

Small (<1k): `14_ieee`, `30_ieee`, `57_ieee`, `89_pegase`, `118_ieee`, `300_ieee`
Medium (<5k): `1354_pegase`, `NewYork2030`, `1888_rte`, `2869_pegase`
Large (<10k): `6470_rte`, `Texas7k`, `9241_pegase`
Extra-Large (>10k): `13659_pegase`, `Midwest24k`

The `89_pegase`, `1354_pegase`, `2869_pegase`, `9241_pegase`, and `13659_pegase` cases from Fliscounakis et al. (2013) are based on the European power grid, the `1888_rte` and `6470_rte` cases from Josz et al. (2016) are based on the French power grid, the `nyiso_2030_v11` case from University of Wisconsin-Madison (2024) is based on the planned 2030 New York power grid, and the `Texas7k` and `Midwest24k` cases from Kunkolienkar et al. (2024) are based on the

Texas power grid and the US Midwest power grid, respectively. The smaller `14_ieee`, `30_ieee`, `118_ieee`, and `300_ieee` cases from University of Washington, Dept. of Electrical Engineering (1999) are synthetic. These cases are chosen to span a range of scales and to match cases used in prior ML for OPF literature.

Table 1 reports statistics of each snapshot used in the PGLearn collection. Namely, for each reference snapshot, the table reports: the number of buses $|\mathcal{N}|$, the number of loads $|\mathcal{L}|$, the number of generators $|\mathcal{G}|$, the number of branches $|\mathcal{E}|$, which includes power lines and transformers, the total active power demand in the reference case (Total PD), the total maximum active power generation (Total PG), and the range of the global scaling factor used in the data augmentation scheme described next (Global range).

Demand Sampling The PGLearn datasets sample each load's active and reactive demand by combining a global (per-sample) correlation term with local (per-load per-sample) noise. The power factor of each load is varied by sampling the local noise independently for the active and reactive components. This sampling procedure mimics real power system behavior since in practice, machine learning training takes time, so models are trained day-ahead on demand ranges given by forecasting systems for the next day (Chen et al., 2022). The local noise is applied to generate diverse samples at all values of total load, ensuring the usability of the machine learning system under various demand settings. The global correlation is important to ensure that the model captures a wide total demand *range* rather than being specialized to a particular total demand level. This captures more operating regimes of the power system, as shown in Figure 2. The demand sampling process is described in Algorithm 1. The Global Range column in Table 1 contains the values for b_l and b_u for each case. ϵ is set to 20% for all cases. The width of the global range is fixed to 40% with b_u determined by incrementally scaling the reference load values in 10% steps until an infeasible case is hit.

Algorithm 1 Demand Sampling

Input: Reference demand $(\mathbf{p}^{\mathrm{d}}, \mathbf{q}^{\mathrm{d}})$, global range (b_l, b_u), noise level ϵ
Output: Sampled demand $(\tilde{\mathbf{p}}^{\mathrm{d}}, \tilde{\mathbf{q}}^{\mathrm{d}})$

$b \sim \mathrm{Uniform}(b_l, b_u)$
for $i = 1 \dots |\mathcal{L}|$ **do**
  $\epsilon_i^{\mathrm{p}} \sim \mathrm{Uniform}(1-\epsilon, 1+\epsilon)$
  $\epsilon_i^{\mathrm{q}} \sim \mathrm{Uniform}(1-\epsilon, 1+\epsilon)$
  $\tilde{\mathbf{p}}_i^{\mathrm{d}} \leftarrow b \cdot \epsilon_i^{\mathrm{p}} \cdot \mathbf{p}_i^{\mathrm{d}}$
  $\tilde{\mathbf{q}}_i^{\mathrm{d}} \leftarrow b \cdot \epsilon_i^{\mathrm{q}} \cdot \mathbf{q}_i^{\mathrm{d}}$
end for
return $(\tilde{\mathbf{p}}^{\mathrm{d}}, \tilde{\mathbf{q}}^{\mathrm{d}})$

Status Sampling To generate the $N-1$[1] datasets, disabled branches/generators are sampled following the procedure used in OPFData (Lovett et al., 2024). Either one generator or one (non-bridge, to preserve connectedness of the network) branch is disabled per instance.

Time-Series Sampling There are several public resources, i.e. ENTSO-E Hirth et al. (2018) and OEDI[2], which provide time-series power demand information. Some system operators, e.g. RTE, also publish load information in real-time.[3] Although these data sources are useful for e.g. power demand forecasting studies, they are not detailed enough to formulate an OPF; namely a full description of the power system and load-level demand information is required. Motivated by this mismatch, the `Texas7k` and `Midwest24k` cases include one year of synthetic time-series data at an hourly granularity. Readers are referred to Li et al. (2020a) for details on how the coarse time-series data is created. PGLearn uses cubic spline interpolation to augment the coarse time-series in order to provide AC, DC, and SOC-OPF solutions at a five-minute granularity for `Texas7k` and ten-minute granularity for `Midwest24k`.

[1] $N-1$ refers to the common security requirement that the system remain stable under any single failure; here, N refers to the number of components which are susceptible to failure (branches and generators).

[2] `https://data.openei.org/s3_viewer?bucket=arpa-e-perform`

[3] `https://www.rte-france.com/en/eco2mix/market-data`

Train-Test Split The training and testing sets contain only feasible input samples, i.e. those inputs for which a solution was found for all formulations. The feasible samples are then shuffled using a seeded random number generator (`MersenneTwister(42)` from Julia 1.11.5). Then, the first 80% of the shuffled feasible samples are selected as training data and the remaining 20% as testing data. Users should further split the training data to create validation and/or calibration sets as needed; the testing data should be kept unchanged to allow for direct comparison of reported results. This creates three datasets – `train`, `test`, and `infeasible`, where samples in `infeasible` are those for which a (locally) optimal solution could not be found for at least one of the formulations considered.

4 Open-Source Implementations

PGLearn leverages two MIT-licensed open-source repositories: `PGLearn.jl` to generate datasets and `ML4OPF` to build machine learning models.

4.1 PGLearn.jl: OPF Data Generation

The `PGLearn.jl` repository contains the JuMP (Lubin et al., 2023) implementations of the OPF formulations as well as utilities for sampling and solving datasets of instances. For each case, it first uses the `make_basic_network` function from PowerModels (Coffrin et al., 2018) to parse and pre-process the corresponding raw Matpower (Zimmerman et al., 2010) file. The `MathOptSymbolicAD` (LANL-ANSI, 2022) automatic differentiation backend is used to accelerate derivative calculations. `PGLearn.jl` leverages the GNUparallel utility (Tange, 2022) for parallelization across CPU cores and the SLURM workload manager (Jette and Wickberg, 2023) for parallelization across nodes.

4.2 ML4OPF: Model Training, Evaluation and Benchmarking

The `ML4OPF` library – specifically the `parsers` submodule – is the standard way to work with the PGLearn datasets. `ML4OPF` also includes several other submodules that allow researchers to quickly combine and modify existing methods, implement new methods, and easily compare results to prior works. The `layers` submodule contains implementations of several useful differentiable layers implemented in PyTorch (Ansel et al., 2024) for producing predictions which satisfy constraints. The `functional` submodule contains PyTorch JIT implementations of each formulation's constraints, objective, and incidence matrices. `formulations` contains a higher-level API which makes some common assumptions (e.g. only $\mathbf{p}^{\mathrm{d}}$ and $\mathbf{q}^{\mathrm{d}}$ vary per sample) to simplify common workflows. `models` contains ready-to-train implementations of various optimization proxy model architectures including the Lagrangian Dual Framework (Fioretto et al., 2021), a penalty method, and the E2ELR network from Chen (2023).

5 Benchmarking Machine Learning Models for OPF

This section describes evaluation metrics for optimization proxies, catering to the specific context of learning the solution maps of optimization problems. It also provides guidelines for comparing and benchmarking models. It is important to recognize that there is no universal metric, and that researchers should report a combination of metrics to accurately capture the behavior and performance of their models, keeping in mind the downstream use-cases of their contribution.

5.1 Accuracy metrics

The following lists several important metrics in the evaluation of optimization proxy models, i.e. models that predict the output of a parametric optimization problem given the parameters. They are stated below on a per-instance basis; aggregations should be performed by taking the mean/standard deviation and maximum (e.g. over the test set samples).

Optimality gap This metric reports how close the predicted objective value (i.e. the objective value of the predicted solution) is to the true optimal value. It is important to note that predicted solutions are often infeasible, hence it is not fair to assess solutions based on optimality gap alone.

Constraint violations This metric reports the magnitude of constraint violation, aggregated per group of constraints. Here, "group" refers to constraint of the same type, e.g. the $|\mathcal{N}|$ Kirchhoff's current law constraints in DC-OPF. In addition to average/maximum violation magnitude, researchers should also report (for each group of constraints) the proportion of constraints violated and the total violation (the sum of violations within each group).

Distance to feasible set This metric reports how far the predicted solution is from the closest feasible point. This requires solving the corresponding projection problem for each instance (replacing the objective with $\|x - x^\star\|$ where $x^\star$ is the predicted solution and x is subject to the original constraints).

Distance to optimal solution This metric reports how far the predicted solution is from the optimal solution. Note that a solution can exhibit small residuals but large distance to feasibility. In real-life, this can mean that a solution with small residual may need large changes to become feasible, with potentially a large increase in cost.

5.2 Computational performance metrics

These metrics evaluate how fast the proxy models are, compared to optimization solvers. It's important to recognize that ML proxies are only heuristics, whereas optimization solvers have stronger guarantees. Two types of metrics should be reported: computing time for applications where a single instance is solved at a time and throughput for applications that need to solve large batches of instances (e.g. large-scale simulations). Timing results should be reported using CPU/GPU.hour (i.e. 2 CPU cores for 1 hour corresponds to 2 CPU.hr). Similarly to the accuracy metrics, aggregated timing results should report both the mean/standard deviation and the maximum across samples. Finally, in line with general guidelines for reporting ML results, researchers should always report the device (CPU and/or GPU) used for running experiments.

Data-generation time This metric reports the total time spent obtaining the ground-truth primal/dual solutions required for the training (and validation) set of the proxy model – the time spent generating the test set can be excluded.

Training time This metric reports the time spent training the optimization proxy model. In addition, researchers should comment on how often the model would need to be re-trained in a practical application. For instance, it is not realistic to train a model every hour if the training time is 6 hours.

Inference time This metric reports the time spent producing a solution to a single instance. This is useful if the downstream application involves solving instances sequentially, for instance, when solving a market-clearing problem every 5 minutes. Researchers should report the maximum time across samples, especially for architectures that involve an iterative scheme such as gradient correction (Donti et al., 2021), implicit layers (Agrawal et al., 2019), or optimization solvers, because of performance variability.

Instance throughput This metric reports how many instances can be processed per unit of time with a fixed computational resource budget. This metric is relevant for settings where a large number of instances need to be evaluated, e.g., when running large-scale simulations that require solving multiple OPF instances. In addition, it better captures the batched processing advantages of GPU devices.

The `solve_time` metadata included with the PGLearn datasets can be used to obtain data-generation time and instance throughput for optimization solvers. However, it is important to note that PGLearn datasets are generated using a single thread per instance, and utilize multiple processes per machine. Such settings are known to have an adverse impact on the performance of optimization solvers. Hence, the solving times reported in the metadata are likely over-estimates compared to solving one instance at a time in a "clean" environment or with multi-threaded solvers.

6 LIMITATIONS

6.1 DATA LIMITATIONS

In the absence of publicly-available, granular datasets released directly by system operators, PGLearn is limited like prior works to using synthetic time-series and data augmentation schemes to generate samples. Nevertheless, it is important to note that the reference snapshots selected for PGLearn are based on real power grids in France, Europe, Texas, and the American midwest (Josz et al., 2016; Fliscounakis et al., 2013; Kunkolienkar et al., 2024; University of Wisconsin-Madison, 2024), and the time-series used are based on real-world characteristics Li et al. (2020b).

Another limitation of PGLearn is the limited variety of topologies, and the nature of topology changes. Namely, PGLearn considers topology variations by removing individual lines or generators. In contrast, real-life operations include multiple categories of topology changes, such as switching multiple lines and reconfiguring buses within a substation. Additional research is needed to better capture this lesser-studied facet of power grid operations.

6.2 FUTURE COLLECTIONS

PGLearn aims to provide curated datasets that are updated over time, to integrate new data-generation procedures and OPF formulations. To that end, future versions of PGLearn will comprise OPF formulations that include elements present in market-clearing formulations used by system operators. This includes, for instance, the integration of reserve products and support for piece-wise linear production curves Ma et al. (2009).

7 CONCLUSION

This paper has introduced PGLearn, an open-source learning toolkit for optimal power flow. PGLearn addresses the lack of standardized datasets for ML and OPF by releasing several datasets of large-scale OPF instances. It is the first collection that comprises complete primal and dual solutions for multiple OPF formulations. In addition to releasing public datasets, PGLearn provides open-source tools for data generation, and for the training and evaluation of ML models. These open-source tools enable reproducible and fair evaluation of methods for ML and OPF, thereby democratizing access to the field.

The PGLearn collection contains, in its initial release, over 10,000,000 OPF samples. It is released alongside extensive documentation and code, allowing users to generate additional datasets as needed, and to benchmark the performance of ML models. The paper has also provided several performance metrics and guidelines on how to report them. These guidelines aim at capturing specific aspects of ML for OPF that fall outside the scope of traditional ML applications. This includes, for instance, the fundamental importance of measuring and reporting constraint violations, as well as accurate reporting of data-generation, training and inference times when evaluating computational performance.

Finally, PGLearn aims to democratize access to research on ML and OPF by removing the barrier to entry caused by the computational requirements of large-scale data generation. It also aims to align academic research more closely to the scale and complexity of real-world power systems. This will, in turn, unlock the potential for modern AI techniques to assist in making future energy systems more efficient, reliable, and sustainable.

REPRODUCIBILITY STATEMENT

Since the entire pipeline for generating the PGLearn datasets is open-source, the dataset is completely reproducible. The Julia random number generator `MersenneTwister` is used to ensure random number generation is consistent across machines. The `ML4OPF` repository similarly makes use of seeded random number generators, e.g. when instantiating neural network weights and shuffling training data. Besides allowing to fully recreate the PGLearn datasets, the focus on reproducibility also allows practitioners to easily extend or modify the dataset, for example generating new datasets based on custom formulations.

REFERENCES

Junshan Zhang, Yonghong Chen, Mohammad Faqiry, Bernard Knueven, Manuel Joseph Garcia, and Roger Treinen. Future of grid operations and markets: Uncertainty management., 2021. URL https://www.osti.gov/biblio/1861473.

Wenbo Chen, Mathieu Tanneau, and Pascal Van Hentenryck. Real-time risk analysis with optimization proxies. *Electric Power Systems Research*, 235:110822, 2024.

Hooman Khaloie, Mihaly Dolanyi, Jean-Francois Toubeau, and François Vallée. Review of machine learning techniques for optimal power flow. *Available at SSRN 4681955*, May 2024. doi: 10.2139/ssrn.4681955.

Sean Lovett, Miha Zgubic, Sofia Liguori, Sephora Madjiheurem, Hamish Tomlinson, Sophie Elster, Chris Apps, Sims Witherspoon, and Luis Piloto. OPFData: Large-scale datasets for AC optimal power flow with topological perturbations. *arXiv preprint arXiv:2406.07234*, 2024.

Yonghong Chen, Feng Pan, Feng Qiu, Alinson S Xavier, Tongxin Zheng, Muhammad Marwali, Bernard Knueven, Yongpei Guan, Peter B. Luh, Lei Wu, Bing Yan, Mikhail A. Bragin, Haiwang Zhong, Anthony Giacomoni, Ross Baldick, Boris Gisin, Qun Gu, Russ Philbrick, and Fangxing Li. Security-constrained unit commitment for electricity market: Modeling, solution methods, and future challenges. *IEEE Transactions on Power Systems*, 38(5):4668–4681, 2023. doi: 10.1109/TPWRS.2022.3213001.

Fouad Hasan, Amin Kargarian, and Ali Mohammadi. A survey on applications of machine learning for optimal power flow. In *2020 IEEE Texas Power and Energy Conference (TPEC)*, pages 1–6. IEEE, 2020.

Daniel K. Molzahn and Ian A. Hiskens. A survey of relaxations and approximations of the power flow equations. *Foundations and Trends® in Electric Energy Systems*, 4(1-2):1–221, 2019. ISSN 2332-6557. doi: 10.1561/3100000012. URL http://dx.doi.org/10.1561/3100000012.

Xingwang Ma, Haili Song, Mingguo Hong, Jie Wan, Yonghong Chen, and Eugene Zak. The security-constrained commitment and dispatch for midwest iso day-ahead co-optimized energy and ancillary service market. In *2009 IEEE Power & Energy Society General Meeting*, pages 1–8. IEEE, 2009.

Wenbo Chen. End-to-end feasible optimization proxies for large-scale economic dispatch. INFORMS Annual meeting, 2023.

Sogol Babaeinejadsarookolaee, Adam Birchfield, Richard D Christie, Carleton Coffrin, Christopher DeMarco, Ruisheng Diao, Michael Ferris, Stephane Fliscounakis, Scott Greene, Renke Huang, et al. The Power Grid Library for benchmarking AC optimal power flow algorithms. *arXiv preprint arXiv:1908.02788*, 2019.

Balthazar Donon, Rémy Clément, Benjamin Donnot, Antoine Marot, Isabelle Guyon, and Marc Schoenauer. Neural networks for power flow: Graph neural solver. *Electric Power Systems Research*, 189:106547, 2020.

Minas Chatzos, Mathieu Tanneau, and Pascal Van Hentenryck. Data-driven time series reconstruction for modern power systems research. *Electric Power Systems Research*, 212:108589, 2022.

Michael Klamkin, Mathieu Tanneau, Terrence WK Mak, and Pascal Van Hentenryck. Bucketized active sampling for learning ACOPF. *Electric Power Systems Research*, 235:110697, 2024.

Alex Robson, Mahdi Jamei, Cozmin Ududec, and Letif Mones. Learning an optimally reduced formulation of OPF through meta-optimization. *arXiv preprint arXiv:1911.06784*, 2019.

Trager Joswig-Jones, Kyri Baker, and Ahmed S Zamzam. OPF-Learn: An open-source framework for creating representative AC optimal power flow datasets. In *2022 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pages 1–5. IEEE, 2022.

Carleton Coffrin, Russell Bent, Kaarthik Sundar, Yeesian Ng, and Miles Lubin. Powermodels.jl: An open-source framework for exploring power flow formulations. In *2018 Power Systems Computation Conference (PSCC)*, pages 1–8, June 2018. doi: 10.23919/PSCC.2018.8442948.

Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017. doi: 10.1137/141000671. URL https://epubs.siam.org/doi/10.1137/141000671.

Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, CK Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xiaodong Wang, William Wen, Shunting Zhang, Xu Zhao, Keren Zhou, Richard Zou, Ajit Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster machine learning through dynamic Python bytecode transformation and graph compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, April 2024. doi: 10.1145/3620665.3640366. URL https://pytorch.org/assets/pytorch2-2.pdf.

Rabih A Jabr. Radial distribution load flow using conic programming. *IEEE transactions on power systems*, 21(3):1458–1459, 2006a.

Richard D Christie, Bruce F Wollenberg, and Ivar Wangensteen. Transmission management in the deregulated environment. *Proceedings of the IEEE*, 88(2):170–195, 2000.

Guancheng Qiu, Mathieu Tanneau, and Pascal Van Hentenryck. Dual conic proxies for ac optimal power flow. *Electric Power Systems Research*, 236:110661, 2024.

Mathieu Tanneau and Pascal Van Hentenryck. Dual lagrangian learning for conic optimization. *arXiv preprint arXiv:2402.03086*, 2024.

James Kotary and Ferdinando Fioretto. Learning constrained optimization with deep augmented lagrangian methods. *arXiv preprint arXiv:2403.03454*, 2024.

Augustin Parjadis, Quentin Cappart, Bistra Dilkina, Aaron Ferber, and Louis-Martin Rousseau. Learning lagrangian multipliers for the travelling salesman problem, 2023. URL https://arxiv.org/abs/2312.14836.

Francesco Demelas, Joseph Le Roux, Mathieu Lacroix, and Axel Parmentier. Predicting lagrangian multipliers for mixed integer linear programs. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 10368–10384. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/demelas24a.html.

Stéphane Fliscounakis, Patrick Panciatici, Florin Capitanescu, and Louis Wehenkel. Contingency ranking with respect to overloads in very large power systems taking into account uncertainty, preventive, and corrective actions. *IEEE Transactions on Power Systems*, 28(4):4909–4917, 2013.

Cédric Josz, Stéphane Fliscounakis, Jean Maeght, and Patrick Panciatici. AC power flow data in MATPOWER and QCQP format: iTesla, RTE snapshots, and PEGASE. *arXiv preprint arXiv:1603.01533*, 2016.

University of Wisconsin-Madison. University of Wisconsin-Madison ARPA-E PERFORM Electric Grid Test Cases, 2024. URL https://electricgrids.engr.tamu.edu/university-of-wisconsin-madison-perform-cases/.

Sanjana Kunkolienkar, Farnaz Safdarian, Jonathan Snodgrass, Adam Birchfield, and Thomas Overbye. A description of the Texas A&M University Electric Grid Test Case Repository for power system studies. In *2024 IEEE Texas Power and Energy Conference (TPEC)*, pages 1–6. IEEE, 2024.

University of Washington, Dept. of Electrical Engineering. Power systems test case archive, 1999. URL http://www.ee.washington.edu/research/pstca/.

Wenbo Chen, Seonho Park, Mathieu Tanneau, and Pascal Van Hentenryck. Learning optimization proxies for large-scale security-constrained economic dispatch. *Electric Power Systems Research*, 213:108566, 2022.

Lion Hirth, Jonathan Mühlenpfordt, and Marisa Bulkeley. The entso-e transparency platform – a review of europe's most ambitious electricity data platform. *Applied Energy*, 225:1054–1067, 2018. ISSN 0306-2619. doi: https://doi.org/10.1016/j.apenergy.2018.04.048. URL https://www.sciencedirect.com/science/article/pii/S0306261918306068.

Hanyue Li, Ju Hee Yeo, Ashly L Bornsheuer, and Thomas J Overbye. The creation and validation of load time series for synthetic electric power systems. *IEEE Transactions on Power Systems*, 36 (2):961–969, 2020a.

Miles Lubin, Oscar Dowson, Joaquim Dias Garcia, Joey Huchette, Benoît Legat, and Juan Pablo Vielma. JuMP 1.0: Recent improvements to a modeling language for mathematical optimization. *Mathematical Programming Computation*, 2023. doi: 10.1007/s12532-023-00239-3.

Ray Daniel Zimmerman, Carlos Edmundo Murillo-Sánchez, and Robert John Thomas. MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education. *IEEE Transactions on power systems*, 26(1):12–19, 2010.

LANL-ANSI. MathOptSymbolicAD.jl, 2022. URL https://github.com/lanl-ansi/MathOptSymbolicAD.jl.

Ole Tange. GNU Parallel 20220522 ('NATO'), May 2022. URL https://doi.org/10.5281/zenodo.6570228.

Morris A Jette and Tim Wickberg. Architecture of the Slurm workload manager. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 3–23. Springer, 2023.

Ferdinando Fioretto, Pascal Van Hentenryck, Terrence WK Mak, Cuong Tran, Federico Baldo, and Michele Lombardi. Lagrangian duality for constrained deep learning. In *Machine Learning and Knowledge Discovery in Databases. Applied Data Science and Demo Track: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part V*, pages 118–135. Springer, 2021.

Priya Donti, David Rolnick, and J Zico Kolter. DC3: A learning method for optimization with hard constraints. In *International Conference on Learning Representations*, 2021.

Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. *Advances in neural information processing systems*, 32, 2019.

Hanyue Li, Jessica L Wert, Adam Barlow Birchfield, Thomas J Overbye, Tomas Gomez San Roman, Carlos Mateo Domingo, Fernando Emilio Postigo Marcos, Pablo Duenas Martinez, Tarek Elgindy, and Bryan Palmintier. Building highly detailed synthetic electric grid data sets for combined transmission and distribution systems. *IEEE Open Access Journal of Power and Energy*, 7: 478–488, 2020b.

Aharon Ben-Tal and Arkadi Nemirovski. *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. SIAM, 2001.

Ray D. Zimmerman and Carlos E. Murillo-Sánchez. Matpower user's manual, May 2024. URL https://doi.org/10.5281/zenodo.11212313.

Lorenz T Biegler and Victor M Zavala. Large-scale nonlinear programming using Ipopt: An integrating framework for enterprise-wide dynamic optimization. *Computers & Chemical Engineering*, 33(3):575–582, 2009.

Iain S Duff and John K Reid. MA27 - a set of Fortran subroutines for solving sparse symmetric sets of linear equations. *UKAEA Atomic Energy Research Establishment*, 1982.

J Fowkes, A Lister, A Montoison, and D Orban. LibHSL: the ultimate collection for large-scale scien-tific computation. *Les Cahiers du GERAD ISSN*, 711:2440, 2024.

S. Gopinath, H.L. Hijazi, T. Weisser, H. Nagarajan, M. Yetkin, K. Sundar, and R.W. Bent. Proving global optimality of acopf solutions. *Electric Power Systems Research*, 189:106688, 2020. ISSN 0378-7796. doi: https://doi.org/10.1016/j.epsr.2020.106688. URL https://www.sciencedirect.com/science/article/pii/S0378779620304910.

R. A. Jabr. Radial distribution load flow using conic programming. *IEEE Transactions on Power Systems*, 21(3):1458–1459, Aug 2006b. ISSN 0885-8950. doi: 10.1109/TPWRS.2006.879234.

Rabih A Jabr. A conic quadratic format for the load flow equations of meshed networks. *IEEE Transactions on Power Systems*, 22(4):2285–2286, 2007.

Paul J. Goulart and Yuwen Chen. Clarabel: An interior-point solver for conic programs with quadratic objectives. *arXiv prepring arXiv:2405.12762*, 2024.

Q. Huangfu and J. A. J. Hall. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1):119–142, 2018. doi: 10.1007/s12532-017-0130-5.

The HDF Group. Hierarchical Data Format, version 5, 2024. URL https://github.com/HDFGroup/hdf5.

A FORMULATIONS

A.1 BACKGROUND MATERIAL

This section provides a brief overview of Lagrangian and conic duality. PGLearn uses the former for nonlinear non-convex problems such as AC-OPF, and the latter for convex formulations such as SOC-OPF and DC-OPF.

A.1.1 NONLINEAR OPTIMIZATION

Consider a nonlinear, non-convex optimization problem of the form

$$\min_{x} \quad f(x) \tag{1a}$$
$$\text{s.t.} \quad g(x) \geq 0 \tag{1b}$$
$$h(x) = 0 \tag{1c}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$, $g : \mathbb{R}^n \mapsto \mathbb{R}^m$ and $h : \mathbb{R}^n \mapsto \mathbb{R}^p$ are continuous functions, assumed to be differentiable over their respective domains.

Denote by $\mu \in \mathbb{R}^m$ and $\lambda \in \mathbb{R}^p$ the Lagrange multipliers associated to constraints (1b) and (1c), respectively. The first order Karush-Kuhn-Tucker optimality conditions read

$$J_h(x)^\top \lambda + J_g(x)^\top \mu = \nabla_x f(x) \tag{2a}$$
$$g(x) \geq 0 \tag{2b}$$
$$h(x) = 0 \tag{2c}$$
$$\mu \geq 0 \tag{2d}$$
$$\mu^\top g(x) = 0 \tag{2e}$$

where $J_h(x) = \nabla_x h(x)$ and $J_g(x) = \nabla_x g(x)$ denote the Jacobian matrices of h and g, respectively.

Given Lagrange multipliers λ, μ, the following Lagrangian bound is a valid lower bound on the optimal value of problem (1):

$$\mathcal{L}(\lambda, \mu) = \min_{x} f(x) - \lambda^\top h(x) - \mu^\top g(x). \tag{3}$$

Note that computing this Lagrangian bound requires solving a nonlinear, non-convex problem, which is NP-hard in general. Hence, it is generally intractable to compute valid dual bounds from Lagrangian duality in the context of non-convex problems.

A.1.2 CONIC OPTIMIZATION

Consider a conic optimization problem of the form

$$\min_{x} \quad c^\top x \tag{4a}$$
$$\text{s.t.} \quad Ax \succeq_{\mathcal{K}} b \tag{4b}$$

where $A \in \mathbb{R}^{m \times n}$ and $\mathcal{K}$ is a proper cone, i.e., a closed, pointed, convex cone with non-empty interior. The corresponding conic dual problem reads

$$\max_{y} \quad b^\top y \tag{5a}$$
$$\text{s.t.} \quad A^\top y = c \tag{5b}$$
$$y \in \mathcal{K}^* \tag{5c}$$

where $\mathcal{K}^*$ is the dual cone of $\mathcal{K}$. The reader is referred to Ben-Tal and Nemirovski (2001) for a more complete overview of conic optimization and duality.

As shown by Tanneau and Van Hentenryck (2024), in many real-life applications, it is straightforward to obtain dual-feasible solutions. This is the case, for instance, when all primal variables have finite lower and upper bounds, as is the case for all formulations considered in this work. By weak conic duality, such dual-feasible solutions then yield valid dual bounds.

A.2 OPF FORMULATIONS

This section presents the optimization models for each OPF formulation in PGLearn. Readers are referred to the Matpower manual (Zimmerman and Murillo-Sánchez, 2024) for a general introduction to power systems, as well as the underlying concepts and relevant notations. Readers are also referred to the `build_opf` functions in the `PGLearn.jl` source code for implementations of each using the JuMP (Lubin et al., 2023) modeling language, and to the `extract_primal` and `extract_dual` functions for how primal and dual solutions are extracted and stored.

To formulate OPF problems, the following sets are introduced. The set of buses is denoted by $\mathcal{N}$. The sets of generators and loads attached to bus $i \in \mathcal{N}$ are denoted by $\mathcal{G}_i$ and $\mathcal{L}_i$, respectively. The set of branches, i.e., power lines and transformers, is denoted by $\mathcal{E}$. Each edge $e \in \mathcal{E}$ is associated with a pair of buses (i, j) corresponding to the edge's origin and destination. Note that power grids often include parallel branches, i.e., two branches may have identical endpoints. For ease of reading, using a slight abuse of notation, edges are identified with their endpoints using the notation $e = (i, j) \in \mathcal{E}$; this indicates that branch e has endpoints i, j. The set of edges leaving (resp. entering) bus $i \in \mathcal{N}$ is denoted by $\mathcal{E}_i$ (resp. $\mathcal{E}_i^R$). Finally, each branch e is characterized by its complex admittance matrix

$$Y_e = \begin{pmatrix} Y_e^{\mathrm{ff}} & Y_e^{\mathrm{ft}} \\ Y_e^{\mathrm{tf}} & Y_e^{\mathrm{tt}} \end{pmatrix} = \begin{pmatrix} g_e^{\mathrm{ff}} + \mathbf{j} b_e^{\mathrm{ff}} & g_e^{\mathrm{ft}} + \mathbf{j} b_e^{\mathrm{ft}} \\ g_e^{\mathrm{tf}} + \mathbf{j} b_e^{\mathrm{tf}} & g_e^{\mathrm{tt}} + \mathbf{j} b_e^{\mathrm{tt}} \end{pmatrix} \in \mathbb{C}^{2\times 2} \tag{6}$$

where $\mathbf{j}$ is the imaginary unit, i.e., $\mathbf{j}^2 = -1$.

A.2.1 AC OPTIMAL POWER FLOW

Model 1 states the nonlinear programming formulation of AC-OPF used in PGLearn.

Model 1 AC Optimal Power Flow (AC-OPF)

$$\min_{\mathbf{p}^{\mathrm{g}},\mathbf{q}^{\mathrm{g}},\mathbf{p}^{\mathrm{f}},\mathbf{q}^{\mathrm{f}},\mathbf{p}^{\mathrm{t}},\mathbf{q}^{\mathrm{t}},\mathbf{v},\boldsymbol{\theta}} \quad \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{G}_i} c_j \mathbf{p}_j^{\mathrm{g}} \tag{7a}$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{G}_i} \mathbf{p}_j^{\mathrm{g}} - \sum_{j\in\mathcal{L}_i} \mathbf{p}_j^{\mathrm{d}} - g_i^{\mathrm{s}} \mathbf{v}_i^2 = \sum_{e\in\mathcal{E}_i} \mathbf{p}_e^{\mathrm{f}} + \sum_{e\in\mathcal{E}_i^R} \mathbf{p}_e^{\mathrm{t}} \qquad \forall i \in \mathcal{N} \tag{7b}$$

$$\sum_{j\in\mathcal{G}_i} \mathbf{q}_j^{\mathrm{g}} - \sum_{j\in\mathcal{L}_i} \mathbf{q}_j^{\mathrm{d}} + b_i^{\mathrm{s}} \mathbf{v}_i^2 = \sum_{e\in\mathcal{E}_i} \mathbf{q}_e^{\mathrm{f}} + \sum_{e\in\mathcal{E}_i^R} \mathbf{q}_e^{\mathrm{t}} \qquad \forall i \in \mathcal{N} \tag{7c}$$

$$\mathbf{p}_e^{\mathrm{f}} = g_e^{\mathrm{ff}} \mathbf{v}_i^2 + g_e^{\mathrm{ft}} \mathbf{v}_i \mathbf{v}_j \cos(\boldsymbol{\theta}_i - \boldsymbol{\theta}_j) + b_e^{\mathrm{ft}} \mathbf{v}_i \mathbf{v}_j \sin(\boldsymbol{\theta}_i - \boldsymbol{\theta}_j) \qquad \forall e = (i,j) \in \mathcal{E} \tag{7d}$$

$$\mathbf{q}_e^{\mathrm{f}} = -b_e^{\mathrm{ff}} \mathbf{v}_i^2 - b_e^{\mathrm{ft}} \mathbf{v}_i \mathbf{v}_j \cos(\boldsymbol{\theta}_i - \boldsymbol{\theta}_j) + g_e^{\mathrm{ft}} \mathbf{v}_i \mathbf{v}_j \sin(\boldsymbol{\theta}_i - \boldsymbol{\theta}_j) \qquad \forall e = (i,j) \in \mathcal{E} \tag{7e}$$

$$\mathbf{p}_e^{\mathrm{t}} = g_e^{\mathrm{tt}} \mathbf{v}_j^2 + g_e^{\mathrm{tf}} \mathbf{v}_i \mathbf{v}_j \cos(\boldsymbol{\theta}_i - \boldsymbol{\theta}_j) - b_e^{\mathrm{tf}} \mathbf{v}_i \mathbf{v}_j \sin(\boldsymbol{\theta}_i - \boldsymbol{\theta}_j) \qquad \forall e = (i,j) \in \mathcal{E} \tag{7f}$$

$$\mathbf{q}_e^{\mathrm{t}} = -b_e^{\mathrm{tt}} \mathbf{v}_j^2 - b_e^{\mathrm{tf}} \mathbf{v}_i \mathbf{v}_j \cos(\boldsymbol{\theta}_i - \boldsymbol{\theta}_j) - g_e^{\mathrm{tf}} \mathbf{v}_i \mathbf{v}_j \sin(\boldsymbol{\theta}_i - \boldsymbol{\theta}_j) \qquad \forall e = (i,j) \in \mathcal{E} \tag{7g}$$

$$(\mathbf{p}_e^{\mathrm{f}})^2 + (\mathbf{q}_e^{\mathrm{f}})^2 \le \overline{S_e}^2 \qquad \forall e \in \mathcal{E} \tag{7h}$$

$$(\mathbf{p}_e^{\mathrm{t}})^2 + (\mathbf{q}_e^{\mathrm{t}})^2 \le \overline{S_e}^2 \qquad \forall e \in \mathcal{E} \tag{7i}$$

$$\underline{\Delta}\theta_e \le \boldsymbol{\theta}_i - \boldsymbol{\theta}_j \le \overline{\Delta}\theta_e \qquad \forall e = (i,j) \in \mathcal{E} \tag{7j}$$

$$\boldsymbol{\theta}_{\mathrm{ref}} = 0 \tag{7k}$$

$$\underline{\mathbf{p}}_i^{\mathrm{g}} \le \mathbf{p}_i^{\mathrm{g}} \le \overline{\mathbf{p}}_i^{\mathrm{g}} \qquad \forall i \in \mathcal{G} \tag{7l}$$

$$\underline{\mathbf{q}}_i^{\mathrm{g}} \le \mathbf{q}_i^{\mathrm{g}} \le \overline{\mathbf{q}}_i^{\mathrm{g}} \qquad \forall i \in \mathcal{G} \tag{7m}$$

$$\underline{\mathbf{v}_i} \le \mathbf{v}_i \le \overline{\mathbf{v}_i} \qquad \forall i \in \mathcal{N} \tag{7n}$$

$$-\overline{S_e} \le \mathbf{p}_e^{\mathrm{f}} \le \overline{S_e} \qquad \forall e \in \mathcal{E} \tag{7o}$$

$$-\overline{S_e} \le \mathbf{q}_e^{\mathrm{f}} \le \overline{S_e} \qquad \forall e \in \mathcal{E} \tag{7p}$$

$$-\overline{S_e} \le \mathbf{p}_e^{\mathrm{t}} \le \overline{S_e} \qquad \forall e \in \mathcal{E} \tag{7q}$$

$$-\overline{S_e} \le \mathbf{q}_e^{\mathrm{t}} \le \overline{S_e} \qquad \forall e \in \mathcal{E} \tag{7r}$$

The decision variables comprise active and reactive power dispatch $\mathbf{p}^{\mathrm{g}}$ and $\mathbf{q}^{\mathrm{g}}$, active and reactive power flows in the forward and reverse direction $\mathbf{p}^{\mathrm{f}}, \mathbf{q}^{\mathrm{f}}, \mathbf{p}^{\mathrm{t}}, \mathbf{q}^{\mathrm{t}}$, and voltage magnitude and angle $\mathbf{v}$ and $\boldsymbol{\theta}$, respectively. The objective (7a) minimizes the production costs; PGLearn currently supports linear objective functions. Constraints (7b) and (7c) encode the active and reactive power balance physics constraints given by Kirchhoff's current law. Constraints (7d)-(7g) express active and reactive power flow following Ohm's law. Constraints (7h)-(7i) and (7j) enforce the engineering limits of the transmission lines, namely, their thermal capacity and maximum voltage angle difference. Constraint (7k) fixes the reference bus' (slack bus) voltage angle to zero. Finally, constraints (7l) and (7m) encode each generator's minimum and maximum active and reactive power outputs, constraint (7n) enforces the voltage magnitude bounds, and constraints (7o), (7p), (7q), (7r) enforce bounds on the power flow variables.

PGLearn supports any nonlinear optimization solver supported by JuMP. The default configuration solves AC-OPF instances using Ipopt (Biegler and Zavala, 2009) with the MA27 linear solver (Duff and Reid, 1982) via LibHSL (Fowkes et al., 2024). Note that, unless a global optimization solver is used, global optimality of AC-OPF solutions is not guaranteed. Nevertheless, previous experience suggests that solutions obtained by Ipopt are typically close to optimal Gopinath et al. (2020).

A.2.2 SOC Optimal Power Flow

PGLearn also implements the second-order cone relaxation of AC-OPF proposed by Jabr in (Jabr, 2006b; 2007), herein referred to as SOC-OPF. This convex relaxation can be solved in polynomial time using, e.g., an interior-point algorithm, and is exact on radial networks Molzahn and Hiskens (2019).

The SOC-OPF relaxation is obtained by introducing variables

$$\mathbf{w}_i = \mathbf{v}_i^2 \tag{8a}$$

$$\mathbf{w}_e^{\mathrm{re}} = \mathbf{v}_i \mathbf{v}_j \cos(\boldsymbol{\theta}_i - \boldsymbol{\theta}_j) \tag{8b}$$

$$\mathbf{w}_e^{\mathrm{im}} = \mathbf{v}_i \mathbf{v}_j \sin(\boldsymbol{\theta}_i - \boldsymbol{\theta}_j) \tag{8c}$$

together with the valid (non-convex) constraint

$$(\mathbf{w}_e^{\mathrm{re}})^2 + (\mathbf{w}_e^{\mathrm{im}})^2 = \mathbf{w}_i \mathbf{w}_j, \quad \forall e = (i,j) \in \mathcal{E}. \tag{9}$$

Then, constraints (7b), (7c), (7d), (7e), (7f), (7g), (7j), and (7n) are reformulated using these new variables, and constraint (9) is convexified into the so-called Jabr inequality

$$(\mathbf{w}_e^{\mathrm{re}})^2 + (\mathbf{w}_e^{\mathrm{im}})^2 \leq \mathbf{w}_i \mathbf{w}_j, \quad \forall e = (i,j) \in \mathcal{E}. \tag{10}$$

Finally, valid lower and upper bounds for $\mathbf{w}^{\mathrm{re}}, \mathbf{w}^{\mathrm{im}}$ variables are derived from (7n), (7j) and (9), using the same strategy as Coffrin et al. (2018). Model 2 states the resulting SOC-OPF formulation, in conic form.

The implementation in PGLearn slighly differs from Coffrin et al. (2018) in the definition of $\mathbf{w}^{\mathrm{re}}, \mathbf{w}^{\mathrm{im}}$ variables. Namely, in Coffrin et al. (2018), $\mathbf{w}^{\mathrm{re}}, \mathbf{w}^{\mathrm{im}}$ variables are defined per *bus-pair*, defined as a pair of buses (i,j) linked by at least one branch $e = (i,j) \in \mathcal{E}$. In contrast, PGLearn defines $\mathbf{w}^{\mathrm{re}}, \mathbf{w}^{\mathrm{im}}$ variables for each branch; the two formulations are equivalent unless parallel branches are present. This design choice was motivated by the simplicity of the branch-level formulation and the corresponding data formats, and was found to have a marginal impact on the quality of the relaxation.

The dual SOC-OPF model is stated in Model 3. Dual variables $\lambda^{\mathrm{p}}, \lambda^{\mathrm{q}}, \lambda^{\mathrm{pf}}, \lambda^{\mathrm{qf}}, \lambda^{\mathrm{pt}}, \lambda^{\mathrm{qt}}$ are associated to equality constraints (11b), (11c), (11d), (11e), (11f), (11e), and are therefore unrestricted. Dual conic variables $\nu^{\mathrm{f}}, \nu^{\mathrm{t}}$ and ω are associated to conic constraints (11h), (11i) and (11j), respectively. Dual variables $\underline{\mu}^{\theta}$ and $\bar{\mu}^{\theta}$ are associated to the lower and upper side of voltage angle difference constraint (11k). Finally, dual variables $\underline{\mu}^{\mathrm{w}}, \bar{\mu}^{\mathrm{w}}, \underline{\mu}^{\mathrm{pg}}, \bar{\mu}^{\mathrm{pg}}, \underline{\mu}^{\mathrm{qg}}, \bar{\mu}^{\mathrm{qg}}, \underline{\mu}^{\mathrm{pf}}, \bar{\mu}^{\mathrm{pf}}, \underline{\mu}^{\mathrm{qf}}, \bar{\mu}^{\mathrm{qf}}, \underline{\mu}^{\mathrm{pt}}, \bar{\mu}^{\mathrm{pt}}, \underline{\mu}^{\mathrm{qt}}, \bar{\mu}^{\mathrm{qt}}, \underline{\mu}^{\mathrm{wr}}, \bar{\mu}^{\mathrm{wr}}, \underline{\mu}^{\mathrm{wi}}, \bar{\mu}^{\mathrm{wi}}$ are associated to lower and upper bounds on variables $\mathbf{w}$, $\mathbf{p}^{\mathrm{g}}, \mathbf{q}^{\mathrm{g}}$, $\mathbf{p}^{\mathrm{f}}, \mathbf{q}^{\mathrm{f}}, \mathbf{p}^{\mathrm{t}}, \mathbf{q}^{\mathrm{t}}$, $\mathbf{w}^{\mathrm{re}}, \mathbf{w}^{\mathrm{im}}$, respectively.

Note that users need not interact with the dual SOC-OPF problem directly, as interior-point solvers typically report both primal and dual information. The formulation in Model 3 is stated for complete-

Model 2 SOC Optimal Power Flow (SOC-OPF)

$$\min_{\mathbf{p}^{\mathrm{g}},\mathbf{q}^{\mathrm{g}},\mathbf{p}^{\mathrm{f}},\mathbf{q}^{\mathrm{f}},\mathbf{p}^{\mathrm{t}},\mathbf{q}^{\mathrm{t}},\mathbf{w},\mathbf{w}^{\mathrm{re}},\mathbf{w}^{\mathrm{im}}} \quad \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{G}_i} c_j \mathbf{p}^{\mathrm{g}}_j \tag{11a}$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{G}_i} \mathbf{p}^{\mathrm{g}}_j - \sum_{j\in\mathcal{L}_i} \mathbf{p}^{\mathrm{d}}_j - g^{\mathrm{s}}_i \mathbf{w}_i = \sum_{e\in\mathcal{E}_i} \mathbf{p}^{\mathrm{f}}_e + \sum_{e\in\mathcal{E}^R_i} \mathbf{p}^{\mathrm{t}}_e \qquad \forall i \in \mathcal{N} \tag{11b}$$

$$\sum_{j\in\mathcal{G}_i} \mathbf{q}^{\mathrm{g}}_j - \sum_{j\in\mathcal{L}_i} \mathbf{q}^{\mathrm{d}}_j + b^{\mathrm{s}}_i \mathbf{w}_i = \sum_{e\in\mathcal{E}_i} \mathbf{q}^{\mathrm{f}}_e + \sum_{e\in\mathcal{E}^R_i} \mathbf{q}^{\mathrm{t}}_e \qquad \forall i \in \mathcal{N} \tag{11c}$$

$$\mathbf{p}^{\mathrm{f}}_e = g^{\mathrm{ff}}_e \mathbf{w}_i + g^{\mathrm{ft}}_e \mathbf{w}^{\mathrm{re}}_e + b^{\mathrm{ft}}_e \mathbf{w}^{\mathrm{im}}_e \qquad \forall e = (i,j) \in \mathcal{E} \tag{11d}$$

$$\mathbf{q}^{\mathrm{f}}_e = -b^{\mathrm{ff}}_e \mathbf{w}_i - b^{\mathrm{ft}}_e \mathbf{w}^{\mathrm{re}}_e + g^{\mathrm{ft}}_e \mathbf{w}^{\mathrm{im}}_e \qquad \forall e = (i,j) \in \mathcal{E} \tag{11e}$$

$$\mathbf{p}^{\mathrm{t}}_e = g^{\mathrm{tt}}_e \mathbf{w}_j + g^{\mathrm{tf}}_e \mathbf{w}^{\mathrm{re}}_e - b^{\mathrm{tf}}_e \mathbf{w}^{\mathrm{im}}_e \qquad \forall e = (i,j) \in \mathcal{E} \tag{11f}$$

$$\mathbf{q}^{\mathrm{t}}_e = -b^{\mathrm{tt}}_e \mathbf{w}_j - b^{\mathrm{tf}}_e \mathbf{w}^{\mathrm{re}}_e - g^{\mathrm{tf}}_e \mathbf{w}^{\mathrm{im}}_e \qquad \forall e = (i,j) \in \mathcal{E} \tag{11g}$$

$$(\overline{S_e},\ \mathbf{p}^{\mathrm{f}}_e,\ \mathbf{q}^{\mathrm{f}}_e) \in \mathcal{Q}^3 \qquad \forall e \in \mathcal{E} \tag{11h}$$

$$(\overline{S_e},\ \mathbf{p}^{\mathrm{t}}_e,\ \mathbf{q}^{\mathrm{t}}_e) \in \mathcal{Q}^3 \qquad \forall e \in \mathcal{E} \tag{11i}$$

$$(\frac{\mathbf{w}_i}{\sqrt{2}},\ \frac{\mathbf{w}_j}{\sqrt{2}},\ \mathbf{w}^{\mathrm{re}}_e,\ \mathbf{w}^{\mathrm{im}}_e) \in \mathcal{Q}^4_{\mathrm{r}} \qquad \forall e = (i,j) \in \mathcal{E} \tag{11j}$$

$$\tan(\underline{\Delta}\theta_e)\mathbf{w}^{\mathrm{re}}_e \le \mathbf{w}^{\mathrm{im}}_e \le \tan(\overline{\Delta}\theta_e)\mathbf{w}^{\mathrm{re}}_e \qquad \forall e \in \mathcal{E} \tag{11k}$$

$$(\underline{\mathbf{v}_i})^2 \le \mathbf{w}_i \le (\overline{\mathbf{v}_i})^2 \qquad \forall i \in \mathcal{N} \tag{11l}$$

$$\underline{\mathbf{p}^{\mathrm{g}}_i} \le \mathbf{p}^{\mathrm{g}}_i \le \overline{\mathbf{p}^{\mathrm{g}}_i} \qquad \forall i \in \mathcal{G} \tag{11m}$$

$$\underline{\mathbf{q}^{\mathrm{g}}_i} \le \mathbf{q}^{\mathrm{g}}_i \le \overline{\mathbf{q}}^{\mathrm{g}}_i \qquad \forall i \in \mathcal{G} \tag{11n}$$

$$-\overline{S_e} \le \mathbf{p}^{\mathrm{f}}_e \le \overline{S_e} \qquad \forall e \in \mathcal{E} \tag{11o}$$

$$-\overline{S_e} \le \mathbf{q}^{\mathrm{f}}_e \le \overline{S_e} \qquad \forall e \in \mathcal{E} \tag{11p}$$

$$-\overline{S_e} \le \mathbf{p}^{\mathrm{t}}_e \le \overline{S_e} \qquad \forall e \in \mathcal{E} \tag{11q}$$

$$-\overline{S_e} \le \mathbf{q}^{\mathrm{t}}_e \le \overline{S_e} \qquad \forall e \in \mathcal{E} \tag{11r}$$

$$\underline{\mathbf{w}}^{\mathrm{re}}_e \le \mathbf{w}^{\mathrm{re}}_e \le \overline{\mathbf{w}}^{\mathrm{re}}_e \qquad \forall e \in \mathcal{E} \tag{11s}$$

$$\underline{\mathbf{w}}^{\mathrm{im}}_e \le \mathbf{w}^{\mathrm{im}}_e \le \overline{\mathbf{w}}^{\mathrm{im}}_e \qquad \forall e \in \mathcal{E} \tag{11t}$$

ness and to support research on predicting dual solutions. PGLearn supports the use of any JuMP-supported conic solver. By default, PGLearn solves SOC-OPF instances using Clarabel (Goulart and Chen, 2024).

Model 3 Dual of SOC-OPF

$$\max_{\lambda,\mu,\nu,\omega} \quad \sum_{i\in\mathcal{N}}\left(\lambda_i^{\mathrm{p}}\sum_{j\in\mathcal{L}_i}\left(\mathbf{p}_j^{\mathrm{d}}\right)+\lambda_i^{\mathrm{q}}\sum_{j\in\mathcal{L}_i}\left(\mathbf{q}_j^{\mathrm{d}}\right)+\sum_{j\in\mathcal{G}_i}\left(\underline{\mathbf{p}}_j^{\mathrm{g}}\underline{\mu}_j^{\mathrm{pg}}+\overline{\mathbf{p}}_j^{\mathrm{g}}\bar{\mu}_j^{\mathrm{pg}}+\underline{\mathbf{q}}_j^{\mathrm{g}}\underline{\mu}_j^{\mathrm{qg}}+\overline{\mathbf{q}}_j^{\mathrm{g}}\bar{\mu}_j^{\mathrm{qg}}\right)+\underline{\mathrm{v}}_i^2\underline{\mu}_i^{\mathrm{w}}+\overline{\mathrm{v}}_i^2\bar{\mu}_i^{\mathrm{w}}\right)$$

$$+\sum_{e\in\mathcal{E}}\left(-\bar{s}_e\left(\underline{\mu}_e^{\mathrm{pf}}-\bar{\mu}_e^{\mathrm{pf}}+\underline{\mu}_e^{\mathrm{qf}}-\bar{\mu}_e^{\mathrm{qf}}+\underline{\mu}_e^{\mathrm{pt}}-\bar{\mu}_e^{\mathrm{pt}}+\underline{\mu}_e^{\mathrm{qt}}-\bar{\mu}_e^{\mathrm{qt}}+\nu_e^{\mathrm{sf}}+\nu_e^{\mathrm{st}}\right)+\underline{\mathbf{w}}_e^{\mathrm{re}}\underline{\mu}_e^{\mathrm{wr}}+\overline{\mathbf{w}}_e^{\mathrm{re}}\bar{\mu}_e^{\mathrm{wr}}+\underline{\mathbf{w}}_e^{\mathrm{im}}\underline{\mu}_e^{\mathrm{wi}}+\overline{\mathbf{w}}_e^{\mathrm{im}}\bar{\mu}_e^{\mathrm{wi}}\right) \tag{12a}$$

$$\text{s.t.} \quad \lambda_i^{\mathrm{p}}+\underline{\mu}_g^{\mathrm{pg}}+\bar{\mu}_g^{\mathrm{pg}}=c_g \qquad \forall i\in\mathcal{N},\forall g\in\mathcal{G}_i \tag{12b}$$

$$\lambda_i^{\mathrm{q}}+\underline{\mu}_g^{\mathrm{qg}}+\bar{\mu}_g^{\mathrm{qg}}=0 \qquad \forall i\in\mathcal{N},\forall g\in\mathcal{G}_i \tag{12c}$$

$$-\lambda_i^{\mathrm{p}}-\lambda_e^{\mathrm{pf}}+\nu_e^{\mathrm{pf}}+\underline{\mu}_e^{\mathrm{pf}}+\bar{\mu}_e^{\mathrm{pf}}=0 \qquad \forall e=(i,j)\in\mathcal{E} \tag{12d}$$

$$-\lambda_i^{\mathrm{q}}-\lambda_e^{\mathrm{qf}}+\nu_e^{\mathrm{qf}}+\underline{\mu}_e^{\mathrm{qf}}+\bar{\mu}_e^{\mathrm{qf}}=0 \qquad \forall e=(i,j)\in\mathcal{E} \tag{12e}$$

$$-\lambda_j^{\mathrm{p}}-\lambda_e^{\mathrm{pt}}+\nu_e^{\mathrm{pt}}+\underline{\mu}_e^{\mathrm{pt}}+\bar{\mu}_e^{\mathrm{pt}}=0 \qquad \forall e=(i,j)\in\mathcal{E} \tag{12f}$$

$$-\lambda_j^{\mathrm{q}}-\lambda_e^{\mathrm{qt}}+\nu_e^{\mathrm{qt}}+\underline{\mu}_e^{\mathrm{qt}}+\bar{\mu}_e^{\mathrm{qt}}=0 \qquad \forall e=(i,j)\in\mathcal{E} \tag{12g}$$

$$-g_i^s\lambda_i^{\mathrm{p}}+b_i^s\lambda_i^{\mathrm{q}}+\sum_{e\in\mathcal{E}_i^+}\left(g_e^{\mathrm{ff}}\lambda_e^{\mathrm{pf}}-b_e^{\mathrm{ff}}\lambda_e^{\mathrm{qf}}+\frac{\omega_e^{\mathrm{f}}}{\sqrt{2}}\right)+\sum_{e\in\mathcal{E}_i^-}\left(g_e^{\mathrm{tt}}\lambda_e^{\mathrm{pt}}-b_e^{\mathrm{tt}}\lambda_e^{\mathrm{qt}}+\frac{\omega_e^{\mathrm{t}}}{\sqrt{2}}\right)+\underline{\mu}_i^{\mathrm{w}}+\bar{\mu}_i^{\mathrm{w}}=0 \qquad \forall i\in\mathcal{N} \tag{12h}$$

$$g_e^{\mathrm{ft}}\lambda_e^{\mathrm{pf}}+g_e^{\mathrm{tf}}\lambda_e^{\mathrm{pt}}-b_e^{\mathrm{ft}}\lambda_e^{\mathrm{qf}}-b_e^{\mathrm{tf}}\lambda_e^{\mathrm{qt}}-\tan(\underline{\Delta}\theta_e)\underline{\mu}_e^{\theta}+\tan(\overline{\Delta}\theta_e)\bar{\mu}_e^{\theta}+\omega_e^{\mathrm{re}}+\underline{\mu}_e^{\mathrm{wr}}+\bar{\mu}_e^{\mathrm{wr}}=0 \qquad \forall e\in\mathcal{E} \tag{12i}$$

$$b_e^{\mathrm{ft}}\lambda_e^{\mathrm{pf}}-b_e^{\mathrm{tf}}\lambda_e^{\mathrm{pt}}+g_e^{\mathrm{ft}}\lambda_e^{\mathrm{qf}}-g_e^{\mathrm{tf}}\lambda_e^{\mathrm{qt}}+\underline{\mu}_e^{\theta}-\bar{\mu}_e^{\theta}+\omega_e^{\mathrm{im}}+\underline{\mu}_e^{\mathrm{wi}}+\bar{\mu}_e^{\mathrm{wi}}=0 \qquad \forall e\in\mathcal{E} \tag{12j}$$

$$\nu_e^{\mathrm{f}}=(\nu_e^{\mathrm{sf}},\nu_e^{\mathrm{pf}},\nu_e^{\mathrm{qf}})\in\mathcal{Q}^3,\nu_e^{\mathrm{t}}=(\nu_e^{\mathrm{st}},\nu_e^{\mathrm{pt}},\nu_e^{\mathrm{qt}})\in\mathcal{Q}^3 \qquad \forall e\in\mathcal{E} \tag{12k}$$

$$\omega_e=\left(\omega_e^{\mathrm{f}},\omega_e^{\mathrm{t}},\omega_e^{\mathrm{re}},\omega_e^{\mathrm{im}}\right)\in\mathcal{Q}_r^4 \qquad \forall e\in\mathcal{E} \tag{12l}$$

$$\underline{\mu}^{\mathrm{pg}},\underline{\mu}^{\mathrm{qg}},\underline{\mu}^{\mathrm{w}},\underline{\mu}^{\theta},\underline{\mu}^{\mathrm{pf}},\underline{\mu}^{\mathrm{qf}},\underline{\mu}^{\mathrm{pt}},\underline{\mu}^{\mathrm{qt}}\geq 0 \tag{12m}$$

$$\bar{\mu}^{\mathrm{pg}},\bar{\mu}^{\mathrm{qg}},\bar{\mu}^{\mathrm{w}},\bar{\mu}^{\theta},\bar{\mu}^{\mathrm{pf}},\bar{\mu}^{\mathrm{qf}},\bar{\mu}^{\mathrm{pt}},\bar{\mu}^{\mathrm{qt}}\leq 0 \tag{12n}$$

A.2.3 DC Optimal Power Flow

The DC-OPF is a popular linear approximation of AC-OPF, which underlies most electricity markets. The DC approximation is motivated by several assumptions, whose validity mainly holds for transmission systems. Namely, the DC approximation assumes that all voltage magnitudes are fixed to one per-unit, voltage angles are assumed to be small (i.e. $\sin(\theta) \approx \theta$), and that reactive power and power losses can be neglected. The reader is referred to Molzahn and Hiskens (2019) for additional background on the DC approximation.

Model 4 states the DC-OPF formulation used in PGLearn. Constraint (13b) enforces nodal power balance through Kirchhoff's current law. Constraint (13c) expresses active power flows on each branch according to Ohm's law, and constraint (13d) restricts the voltage angle difference between each branch's endpoints. Constraint (13e) fixes the reference bus' voltage angle to zero. Finally, constraints (13f) and (13g) enforce lower and upper limits on active power generation and active power flows.

Model 4 DC Optimal Power Flow (DC-OPF)

$$\min_{\mathbf{p}^{\mathrm{g}},\mathbf{p}^{\mathrm{f}},\boldsymbol{\theta}} \quad \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{G}_i} c_j \mathbf{p}^{\mathrm{g}}_j \tag{13a}$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{G}_i} \mathbf{p}^{\mathrm{g}}_j - \sum_{e\in\mathcal{E}_i} \mathbf{p}^{\mathrm{f}}_e + \sum_{e\in\mathcal{E}^R_i} \mathbf{p}^{\mathrm{f}}_e = \sum_{j\in\mathcal{L}_i} \mathbf{p}^{\mathrm{d}}_j + g^{\mathrm{s}}_i \qquad \forall i \in \mathcal{N} \tag{13b}$$

$$-b_e(\boldsymbol{\theta}_i - \boldsymbol{\theta}_j) - \mathbf{p}^{\mathrm{f}}_e = 0 \qquad \forall e = (i,j) \in \mathcal{E} \tag{13c}$$

$$\underline{\Delta}\theta_e \le \boldsymbol{\theta}_i - \boldsymbol{\theta}_j \le \overline{\Delta}\theta_e \qquad \forall e = (i,j) \in \mathcal{E} \tag{13d}$$

$$\boldsymbol{\theta}_{\text{ref}} = 0 \tag{13e}$$

$$\underline{\mathbf{p}^{\mathrm{g}}_i} \le \mathbf{p}^{\mathrm{g}}_i \le \overline{\mathbf{p}^{\mathrm{g}}_i} \qquad \forall i \in \mathcal{G} \tag{13f}$$

$$-\overline{S_e} \le \mathbf{p}^{\mathrm{f}}_e \le \overline{S_e} \qquad \forall e \in \mathcal{E} \tag{13g}$$

The dual DC-OPF problem is stated in Model 5. Dual variables λ^{p} and λ^{pf} are associated to equality constraints (13b) and (13c), respectively. Dual variables $\underline{\mu}^{\theta}, \bar{\mu}^{\theta}$ are associated to lower and upper sides of the voltage angle difference constraint (13d). Finally, dual variables $\underline{\mu}^{\mathrm{pg}}, \bar{\mu}^{\mathrm{pg}}$, $\underline{\mu}^{\mathrm{pf}}, \bar{\mu}^{\mathrm{pf}}$ are associated to lower and upper bounds on variables $\mathbf{p}^{\mathrm{g}}$ and $\mathbf{p}^{\mathrm{f}}$.

Model 5 Dual of DC-OPF

$$\max_{\lambda,\mu} \quad \sum_{i\in\mathcal{N}} \lambda^{\mathrm{p}}_i (g^{\mathrm{s}}_i + \sum_{j\in\mathcal{L}_i} \mathbf{p}^{\mathrm{d}}_j) + \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{G}_i} \left(\underline{\mathbf{p}^{\mathrm{g}}_j}\underline{\mu}^{\mathrm{pg}}_j + \overline{\mathbf{p}^{\mathrm{g}}_j}\bar{\mu}^{\mathrm{pg}}_j\right)$$
$$+ \sum_{e\in\mathcal{E}} \left(\underline{\Delta}\theta_e \underline{\mu}^{\theta}_e + \overline{\Delta}\theta_e \bar{\mu}^{\theta}_e - \bar{s}_e \underline{\mu}^{\mathrm{pf}}_e + \bar{s}_e \bar{\mu}^{\mathrm{pf}}_e\right) \tag{14a}$$

$$\text{s.t.} \quad \lambda^{\mathrm{p}}_i + \underline{\mu}^{\mathrm{pg}}_g + \bar{\mu}^{\mathrm{pg}}_g = c_g \qquad \forall i \in \mathcal{N}, \forall g \in \mathcal{G}_i \tag{14b}$$

$$-\lambda^{\mathrm{p}}_i + \lambda^{\mathrm{p}}_j - \lambda^{\mathrm{pf}}_e + \underline{\mu}^{\mathrm{pf}}_e + \bar{\mu}^{\mathrm{pf}}_e = 0 \qquad \forall e = (i,j) \in \mathcal{E} \tag{14c}$$

$$\sum_{e\in\mathcal{E}^+_i} \left(\underline{\mu}^{\theta}_e - b_e \lambda^{\mathrm{pf}}_e\right) + \sum_{e\in\mathcal{E}^-_i} \left(\bar{\mu}^{\theta}_e + b_e \lambda^{\mathrm{pf}}_e\right) = 0 \qquad \forall i \in \mathcal{N} \tag{14d}$$

$$\underline{\mu}^{\theta}, \underline{\mu}^{\mathrm{pg}}, \underline{\mu}^{\mathrm{pf}} \ge 0 \tag{14e}$$

$$\bar{\mu}^{\theta}, \bar{\mu}^{\mathrm{pg}}, \bar{\mu}^{\mathrm{pf}} \le 0 \tag{14f}$$

PGLearn supports any linear programming solver supported by JuMP. By default, PGLearn solves DC-OPF instances using HiGHS (Huangfu and Hall, 2018).

B DATASET FORMAT

This section contains tables describing the format for the case file, the input data files, and the metadata, primal solution, and dual solution files for each of the formulations. Besides the JSON case file, all data is stored in the HDF5 format (The HDF Group, 2024). Each dataset within PGLearn is structured following the diagram below, where `<case>` refers to the snapshot name, `<split>` is "train", "test", or "infeasible", and `<formulation>` is "ACOPF", "DCOPF", or "SOCOPF":

```
<case>
| - case.json
| - <split>
|   | - input.h5
|   | - <formulation>
|   |   | - primal.h5
|   |   | - dual.h5
|   |   | - meta.h5
```

The main data in the `input.h5` files are stored under the `data` key, with metadata (seed numbers and the configuration file used to generate the dataset) stored under the `meta` key. The structure of the input data tables is given in Table 2. The structure of the metadata for each formulation (stored in the `meta.h5` files), is described in Table 3. The reference case data stored in `case.json` is described in Table 7.

Table 2: Input Data Format

Key	Shape	Meaning
`pd`	$(N, \lvert\mathcal{L}\rvert)$	$\mathbf{p}^{\mathrm{d}}$ – Active power demand
`qd`	$(N, \lvert\mathcal{L}\rvert)$	$\mathbf{q}^{\mathrm{d}}$ – Reactive power demand
`branch_status`	$(N, \lvert\mathcal{E}\rvert)$	0 if branch is disabled, 1 otherwise
`gen_status`	$(N, \lvert\mathcal{G}\rvert)$	0 if generator is disabled, 1 otherwise

Table 3: Metadata Format

Key	Size	Meaning
`formulation`	$(N, 1)$	Formulation name
`termination_status`	$(N, 1)$	`MOI.TerminationStatusCode`
`primal_status`	$(N, 1)$	`MOI.ResultStatusCode` for primal solution
`dual_status`	$(N, 1)$	`MOI.ResultStatusCode` for dual solution
`solve_time`	$(N, 1)$	Time spent in solver
`build_time`	$(N, 1)$	Time spent building JuMP model
`extract_time`	$(N, 1)$	Time spent extracting solution
`primal_objective_value`	$(N, 1)$	Primal objective value
`dual_objective_value`	$(N, 1)$	Dual objective value
`seed`	$(N, 1)$	`MersenneTwister` seed

Table 4: AC-OPF Data Format

Primal Key	Size	Variable
pg	$(N, \lvert\mathcal{G}\rvert)$	$\mathbf{p}^{\mathrm{g}}$
qg	$(N, \lvert\mathcal{G}\rvert)$	$\mathbf{q}^{\mathrm{g}}$
vm	$(N, \lvert\mathcal{N}\rvert)$	$\mathbf{v}$
va	$(N, \lvert\mathcal{N}\rvert)$	$\boldsymbol{\theta}$
pf	$(N, \lvert\mathcal{E}\rvert)$	$\mathbf{p}^{\mathrm{f}}$
qf	$(N, \lvert\mathcal{E}\rvert)$	$\mathbf{q}^{\mathrm{f}}$
pt	$(N, \lvert\mathcal{E}\rvert)$	$\mathbf{p}^{\mathrm{t}}$
qt	$(N, \lvert\mathcal{E}\rvert)$	$\mathbf{q}^{\mathrm{t}}$

Dual Key	Size	Constraint
slack_bus	$(N, 1)$	(7k)
kcl_p	$(N, \lvert\mathcal{N}\rvert)$	(7b)
kcl_q	$(N, \lvert\mathcal{N}\rvert)$	(7c)
ohm_pf	$(N, \lvert\mathcal{E}\rvert)$	(7d)
ohm_qf	$(N, \lvert\mathcal{E}\rvert)$	(7e)
ohm_pt	$(N, \lvert\mathcal{E}\rvert)$	(7f)
ohm_qt	$(N, \lvert\mathcal{E}\rvert)$	(7g)
sm_fr	$(N, \lvert\mathcal{E}\rvert)$	(7h)
sm_to	$(N, \lvert\mathcal{E}\rvert)$	(7i)
va_diff	$(N, \lvert\mathcal{E}\rvert)$	(7j)
pg_lb / pg_ub	$(N, \lvert\mathcal{G}\rvert)$	(7l)
qg_lb / qg_ub	$(N, \lvert\mathcal{G}\rvert)$	(7m)
vm_lb / vm_ub	$(N, \lvert\mathcal{N}\rvert)$	(7n)
pf_lb / pf_ub	$(N, \lvert\mathcal{E}\rvert)$	(7o)
qf_lb / qf_ub	$(N, \lvert\mathcal{E}\rvert)$	(7p)
pt_lb / pt_ub	$(N, \lvert\mathcal{E}\rvert)$	(7q)
qt_lb / qt_ub	$(N, \lvert\mathcal{E}\rvert)$	(7r)

Table 5: SOC-OPF Data Format

Primal Key	Size	Variable
pg	$(N, \lvert\mathcal{G}\rvert)$	$\mathbf{p}^{\mathrm{g}}$
qg	$(N, \lvert\mathcal{G}\rvert)$	$\mathbf{q}^{\mathrm{g}}$
w	$(N, \lvert\mathcal{N}\rvert)$	$\mathbf{w}$
wr	$(N, \lvert\mathcal{E}\rvert)$	$\mathbf{w}^{\mathrm{re}}$
wi	$(N, \lvert\mathcal{E}\rvert)$	$\mathbf{w}^{\mathrm{im}}$
pf	$(N, \lvert\mathcal{E}\rvert)$	$\mathbf{p}^{\mathrm{f}}$
pt	$(N, \lvert\mathcal{E}\rvert)$	$\mathbf{p}^{\mathrm{t}}$
qf	$(N, \lvert\mathcal{E}\rvert)$	$\mathbf{q}^{\mathrm{f}}$
qt	$(N, \lvert\mathcal{E}\rvert)$	$\mathbf{q}^{\mathrm{t}}$

Dual Key	Size	Constraint
kcl_p	$(N, \lvert\mathcal{N}\rvert)$	(11b)
kcl_q	$(N, \lvert\mathcal{N}\rvert)$	(11c)
ohm_pf	$(N, \lvert\mathcal{E}\rvert)$	(11d)
ohm_qf	$(N, \lvert\mathcal{E}\rvert)$	(11e)
ohm_pt	$(N, \lvert\mathcal{E}\rvert)$	(11f)
ohm_qt	$(N, \lvert\mathcal{E}\rvert)$	(11g)
sm_fr	$(N, \lvert\mathcal{E}\rvert, 3)$	(11h)
sm_to	$(N, \lvert\mathcal{E}\rvert, 3)$	(11i)
jabr	$(N, \lvert\mathcal{E}\rvert, 4)$	(11j)
va_diff_lb / va_diff_ub	$(N, \lvert\mathcal{E}\rvert)$	(11k)
w_lb / w_ub	$(N, \lvert\mathcal{N}\rvert)$	(11l)
wr_lb / wr_ub	$(N, \lvert\mathcal{E}\rvert)$	(11s)
wi_lb / wi_ub	$(N, \lvert\mathcal{E}\rvert)$	(11t)
pg_lb / pg_ub	$(N, \lvert\mathcal{G}\rvert)$	(11m)
qg_lb / qg_ub	$(N, \lvert\mathcal{G}\rvert)$	(11n)
pf_lb / pf_ub	$(N, \lvert\mathcal{E}\rvert)$	(11o)
qf_lb / qf_ub	$(N, \lvert\mathcal{E}\rvert)$	(11p)
pt_lb / pt_ub	$(N, \lvert\mathcal{E}\rvert)$	(11q)
qt_lb / qt_ub	$(N, \lvert\mathcal{E}\rvert)$	(11r)

Table 6: DC-OPF Data Format

Primal Key	Size	Variable
pg	$(N, \lvert\mathcal{G}\rvert)$	$\mathbf{p}^{\mathrm{g}}$
va	$(N, \lvert\mathcal{N}\rvert)$	$\boldsymbol{\theta}$
pf	$(N, \lvert\mathcal{E}\rvert)$	$\mathbf{p}^{\mathrm{f}}$

Dual Key	Size	Constraint
slack_bus	$(N, 1)$	(13e)
kcl	$(N, \lvert\mathcal{N}\rvert)$	(13b)
ohm	$(N, \lvert\mathcal{E}\rvert)$	(13c)
va_diff	$(N, \lvert\mathcal{E}\rvert)$	(13d)
pg_lb / pg_ub	$(N, \lvert\mathcal{G}\rvert)$	(13f)
pf_lb / pf_ub	$(N, \lvert\mathcal{E}\rvert)$	(13g)

Table 7: Case JSON Format

Key	Size	Description
`case`	–	Snapshot name
`N`	1	Number of buses ($\lvert\mathcal{N}\rvert$)
`E`	1	Number of edges ($\lvert\mathcal{E}\rvert$)
`L`	1	Number of loads ($\lvert\mathcal{L}\rvert$)
`G`	1	Number of generators ($\lvert\mathcal{G}\rvert$)
`ref_bus`	1	Index of reference/slack bus (1-based)
`base_mva`	1	Base MVA to convert from per-unit
`vnom`	$\lvert\mathcal{N}\rvert$	Nominal voltage
`pd`	$\lvert\mathcal{L}\rvert$	Reference active power load ($\mathbf{p}^{\mathrm{d}}$)
`qd`	$\lvert\mathcal{L}\rvert$	Reference reactive power load ($\mathbf{q}^{\mathrm{d}}$)
`A`	($\lvert\mathcal{E}\rvert$, $\lvert\mathcal{N}\rvert$)	Branch incidence matrix in COO format
`Ag`	($\lvert\mathcal{N}\rvert$, $\lvert\mathcal{G}\rvert$)	Generator incidence matrix in COO format
`bus_arcs_fr`	$\lvert\mathcal{N}\rvert$	Indices of branches leaving each bus ($\mathcal{E}_i$)
`bus_arcs_to`	$\lvert\mathcal{N}\rvert$	Indices of branches entering each bus ($\mathcal{E}_i^R$)
`bus_gens`	$\lvert\mathcal{N}\rvert$	Indices of generators at each bus ($\mathcal{G}_i$)
`bus_loads`	$\lvert\mathcal{N}\rvert$	Indices of loads at each bus ($\mathcal{L}_i$)
`gs`	$\lvert\mathcal{N}\rvert$	Nodal shunt conductance (g^{s})
`bs`	$\lvert\mathcal{N}\rvert$	Nodal shunt susceptance (b^{s})
`vmin`	$\lvert\mathcal{N}\rvert$	Voltage magnitude lower bound ($\underline{\mathbf{v}}$)
`vmax`	$\lvert\mathcal{N}\rvert$	Voltage magnitude upper bound ($\overline{\mathbf{v}}$)
`dvamin`	$\lvert\mathcal{E}\rvert$	Minimum voltage angle difference ($\underline{\Delta}\theta$)
`dvamax`	$\lvert\mathcal{E}\rvert$	Maximum voltage angle difference ($\overline{\Delta}\theta$)
`smax`	$\lvert\mathcal{E}\rvert$	Branch thermal limit ($\overline{S}$)
`pgmin`	$\lvert\mathcal{G}\rvert$	Minimum active power generation ($\mathbf{p}^{\mathrm{g}}$)
`pgmax`	$\lvert\mathcal{G}\rvert$	Maximum active power generation ($\overline{\mathbf{p}^{\mathrm{g}}}$)
`qgmin`	$\lvert\mathcal{G}\rvert$	Minimum reactive power generation ($\mathbf{q}^{\mathrm{g}}$)
`qgmax`	$\lvert\mathcal{G}\rvert$	Maximum reactive power generation ($\overline{\mathbf{q}^{\mathrm{g}}}$)
`c1`	$\lvert\mathcal{G}\rvert$	Linear cost coefficient
`gen_bus`	$\lvert\mathcal{G}\rvert$	Bus index of each generator (1-based)
`load_bus`	$\lvert\mathcal{L}\rvert$	Bus index of each load (1-based)
`bus_fr`	$\lvert\mathcal{E}\rvert$	From bus index for each branch (i) (1-based)
`bus_to`	$\lvert\mathcal{E}\rvert$	To bus index for each branch (j) (1-based)
`g`	$\lvert\mathcal{E}\rvert$	Branch conductance
`b`	$\lvert\mathcal{E}\rvert$	Branch susceptance
`gff`	$\lvert\mathcal{E}\rvert$	From-side branch conductance (g^{ff})
`gft`	$\lvert\mathcal{E}\rvert$	From-to branch conductance (g^{ft})
`gtf`	$\lvert\mathcal{E}\rvert$	To-from branch conductance (g^{tf})
`gtt`	$\lvert\mathcal{E}\rvert$	To-side branch conductance (g^{tt})
`bff`	$\lvert\mathcal{E}\rvert$	From-side branch susceptance (b^{ff})
`bft`	$\lvert\mathcal{E}\rvert$	From-to branch susceptance (b^{ft})
`btf`	$\lvert\mathcal{E}\rvert$	To-from branch susceptance (b^{tf})
`btt`	$\lvert\mathcal{E}\rvert$	To-side branch susceptance (b^{tt})

C Data Loading Tutorial

PGLearn is compatible with the `datasets` Python library enabling powerful features such as streaming and compatibility with many major ML frameworks. For example, to use the `1354_pegase` dataset, one can run the following code:

```python
from datasets import load_dataset

ds = load_dataset(
    "PGLearn/PGLearn-Medium-1354_pegase",
    split="test",     # train or test
    streaming=True,   # optional; download samples on-demand
    columns=[         # optional; only download some columns
      "input/pd", "ACOPF/primal/pg",
    ]
)

sample = next(iter(ds))
print(len(sample["input/pd"]))          # 673
print(len(sample["ACOPF/primal/pg"]))   # 260

# example torch usage with torch.utils.data.DataLoader
import torch
dl = torch.utils.data.DataLoader(
    ds.with_format("torch"),
    batch_size=8
)
batch = next(iter(dl))
print(batch["ACOPF/primal/pg"].shape)   # torch.Size([8, 260])
```

The HDF5 files can also be downloaded directly from the `script` revision:

```python
from huggingface_hub import snapshot_download

# download the compressed HDF5 files
snapshot_download(
    "PGLearn/PGLearn-Small-14_ieee",
    local_dir="./data",
    repo_type="dataset", revision="script",
    # optional; filter what files to download
    allow_patterns=[
      "*/DCOPF/*", "*input*"
    ],
    ignore_patterns=[
      "*dual.h5.gz", "infeasible/*"
    ],
)

# optional; pre-decompress all files using gzip
from pathlib import Path
import gzip, shutil
for src in Path("./data").rglob("*.h5.gz"):
    dest = src.with_suffix("")
    with gzip.open(src, "rb") as fsrc, open(dest, "wb") as fdest:
        shutil.copyfileobj(fsrc, fdest)
    src.unlink()  # optional; delete the compressed files

# read using h5py
import h5py
h5py.File("./data/train/DCOPF/primal.h5")['pg'].shape  # (756205, 5)
```