Pacific Northwest
NATIONAL LABORATORY

PNNL-37601

DistOPF: Advanced Solutions for Distribution Optimal Power Flow Analysis

DistOPF v0.2 Documentation

March 2025

Nathan Gray
Anamika Dubey
Andrew P Reiman
Rabayet Sadnan

U.S. DEPARTMENT OF ENERGY
Prepared for the U.S. Department of Energy
Under contract DE-AC05-76RL01830

DISCLAIMER

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC05-76RL01830

Printed in the United States of America

Available to DOE and DOE contractors from
the Office of Scientific and Technical Information,
P.O. Box 62, Oak Ridge, TN 37831-0062
www.osti.gov
ph: (865) 576-8401
fox: (865) 576-5728
email: reports@osti.gov

Available to the public from the National Technical Information Service
5301 Shawnee Rd., Alexandria, VA 22312
ph: (800) 553-NTIS (6847)
or (703) 605-6000
email: info@ntis.gov
Online ordering: http://www.ntis.gov

PNNL-37601

DistOPF: Advanced Solutions for Distribution Optimal Power Flow Analysis

DistOPF v0.2 Documentation

March 2025

Nathan Gray
Andrew P Reiman
Anamika Dubey
Rabayet Sadnan

Prepared for
the U.S. Department of Energy
Under Contract DE-AC05-76RL01830

Pacific Northwest National Laboratory
Richland, Washington 99352

Abstract

To achieve an affordable and reliable energy system, research on power distribution system is often focused on integration of distributed generators, energy storage solution, EV charging, smart meters, and other advanced assets that may benefit from or require more advanced control and optimization techniques. Despite this focus on advanced distribution system topics, early researchers and grid scientists often start from scratch when developing optimization programs for power distribution systems. This report introduces DistOPF, a Python package that consolidates years of research into a versatile and modular tool. DistOPF provides researchers with essential capabilities to solve distribution system optimal power flow (OPF) problems using standard network models. Additionally, it offers a platform to benchmark both new and existing algorithms against established test systems.

Executive Summary

DistOPF is a powerful Python package designed to address the growing need for advanced optimization techniques in power distribution systems. This versatile tool consolidates years of research into a modular framework that provides researchers with essential capabilities to solve distribution system OPF problems using standard network models. It serves as both a practical tool for immediate application and a platform for benchmarking new algorithms against established test systems. Additionally, this package helps identifying the required OPF modleling parameters from the network models.

This package provides a robust framework for three-phase unbalanced OPF modeling, supporting common Python solver packages such as CVXPY and SciPy. It enables researchers to develop and benchmark new algorithms using a standardized set of test systems with standard distribution network models, such as OpenDSS, CIM, GridLab-D models, etc. The platform includes an OpenDSS model importer, allowing seamless integration of power system models for OPF solver formulation. To ensure accuracy, model validation is conducted with OpenDSS. Additionally, comprehensive visualization tools facilitate the interpretation of results, and the framework supports flexible mathematical modeling to accommodate various OPF objectives and constraints. The package implementation provides a block-modular codebase that enables users to easily add constraints and objectives, enhancing flexibility in OPF formulation. Thus, the package addresses several key challenges in the field: (i) the need to integrate unbalanced distribution power flow models as constraints in OPF algorithms, (ii) the time-consuming process of creating extensive training data for machine learning approaches, (iii) the inefficiency of researchers repeatedly starting from scratch when developing optimization programs for distribution systems, and (iv) identifying OPF modleing parameters for standard distribution network model.

In this report, section 1.0 introduces the vision, motivation, and overview for DistOPF that describes its role in addressing three-phase unbalanced OPF challenges in distribution systems. The underlying mathematical models and the formulations implemented by DistOP are detailed in section 2.0. Section 3.0-4.0 provides step-by-step guidance on setting up an OPF problem by describing the data input format and parameters required. Additionally, section 4.0 provides corresponding visualizions to interprete the optimal solutions. These sections outlines how to properly format input files, specify network components, define operational constraints, select optimization objectives, and analyze results. These section ensures users can efficiently configure the necessary data for solving a distribution system OPF problem using the DistOPF framework. Section 5.0 validates the accuracy of the model by comparing the DistOPF results with OpenDSS results. Finally, we conclude the findings in section 6.0.

Acronyms

BESS Battery Energy Storage System

DER Distributed Energy Resource

DG Distributed Generation

OPF optimal power flow

Contents

1.0 Introduction

Both the traditional mathematical optimization methods and machine-learning (ML) -based approaches are gaining traction to attain tractable optimal solutions for the scaled power distribution systems [1, 2, 3, 4]. However, when developing any OPF algorithm, it is crucial to integrate unbalanced distribution power flow models as constraints [5, 6, 7]. This represents one of the most challenging aspects of formulating the OPF problem, given the availability of multiple models. Besides, for ML-based approaches, creating extensive training data is time-consuming task that significantly delays the process. Additionally, early researchers and electric power grid scientists often start from scratch when developing optimization programs to solve distribution OPF problems, involving the laborious task of writing OPF constraints such as power flow models and operational limits from the ground up. This repetitive groundwork not only consumes valuable time but also impedes progress and innovation in the field.

To address these challenges, this report introduces DistOPF, a Python package that encapsulates years of research into a versatile and modular tool. This report aims to present state-of-the-art optimization platform to develop and provide grid-support functionality and enhance grid resilience for the power distribution system; and alleviate the above-mentioned issues with developing OPF methods. It details the mathematical formulations for distribution OPFs and guides users on integrating standard network models like CIM and OpenDSS. The block-modular codebase allows adding constraints and objectives, enhancing flexibility in OPF formulation.

1.1 Vision

This Python package serves as an open-source platform for three-phase, unbalanced OPF in distribution systems, designed to support researchers. Our goal is to provide users with a robust tool that offers

1. Asymmetrical 3-Phase OPF model generators usable with common Python solver packages such as CVXPY [8] and SciPy [9];

2. A platform for creating and benchmarking new algorithms on a set of standard test systems;

3. Standard distribution network model importer allowing users to import power system models directly from the standard model format for OPF solver modeling language. This feature ensures seamless integration of existing network data into the OPF framework, enabling efficient and accurate problem formulation for three-phase, unbalanced distribution system analysis.;

4. Validation of the Models with standard network models;

5. Visualization and interpretation of OPF solutions/results, offering intuitive plots and analytical tools to assess voltage profiles, power flows, and control decisions. These tools aid in understanding the impact of optimization on network performance and identifying potential operational challenges..

1.2 Motivation

The motivation for this work stems from the need to integrate unbalanced distribution power flow models as constraints in OPF algorithms, which is a challenging task due to the availability

of multiple models. Furthermore, no existing package offers the flexible and modular design needed for various users: naïve users can utilize it for standard solutions, advanced users can easily modify and customize it for their OPF formulations, and ML-based scientists can generate training data without writing OPF formulation codes themselves, ensuring the models reflect practical scenarios more accurately. Python was chosen as the language for this package because it is very popular for scientific computing and has a low learning curve for new users. Currently, no Python package is designed for optimal power flow for distribution systems with asymmetric phases. The Python package, PandaPower provides tools for simulating distribution systems but requires all lines to be three-phase.

1.3 DistOPF Tool Overview and Components

The tool is composed of four (4) major parts, 1) model input system, 2) optimization model formulation, 3) OPF solver interface, and 4) solution output and visualization. Distribution models are represented using a set of CSV files, which are read into Pandas DataFrames and often converted from standard CIM or OpenDSS network models. Buses are described in one CSV having columns for loads, base units, and voltage limits. Lines, switches, and transformers are described in a CSV having columns for each term in the upper diagonal impedance matrix. Regulators, capacitor banks, and generators each have their own CSV. To aid in model creation and validation, models can be created using OpenDSS/CIM and converted to the tabular CSV format. The tool provides classes and functions to make it easy to formulate and solve the power system problem for new users while being flexible for advanced users to create new models and algorithms. The tool has been used to solve a variety of problems including, conservative voltage reduction, power loss minimization, generation curtailment minimization, where either real or reactive power injections of the generators are controlled. Figure 1 shows the power flow results on the IEEE 123-Bus test system with average voltages and real power flow represented.

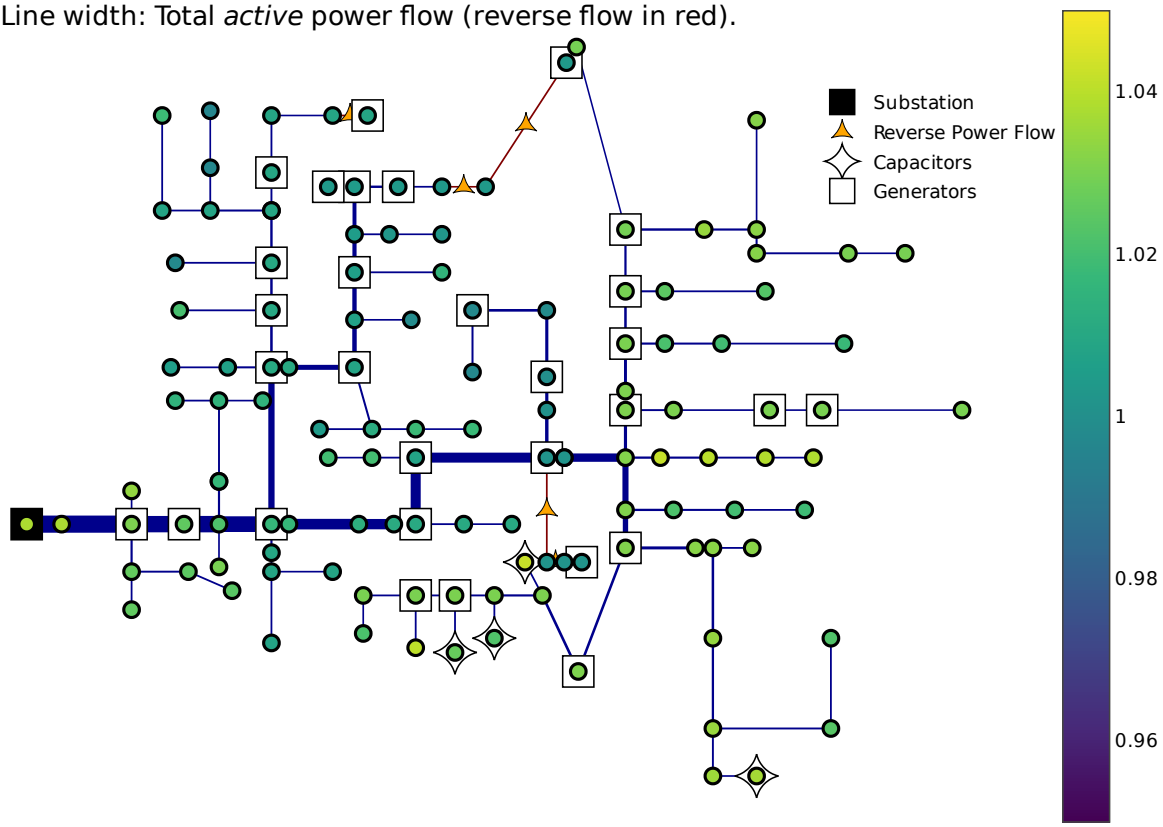

Figure 1. Results of power flow on the IEEE 123-Bus network after being converted from OpenDSS. The average phase voltage on each bus is shown with color and total power flow on each line as the line thickness.

2.0 Mathematical Modeling in DistOPF

The mathematical basis of DistOPF originates from the modeling equations of OPF problems, which are formulated within power distribution systems to address OPF challenges. Similar to other optimization problems, the OPF problem includes an objective function or cost function that is minimized while maintaining the system's governing physics and adhering to physical and operational constraints. The objective or cost function can vary, targeting goals such as minimizing power losses, reducing generation costs, or achieving conservation voltage reduction. The constraints of the OPF problem generally encompass power flow models, grid equipment models, and operational limits. This section describes the optimization constraints and different OPF objectives pertinent to electric power distribution systems.

In this paper, $(\cdot)^T$ represents matrix transpose; $(\cdot)^*$ represents the complex-conjugate; $\overline{(\cdot)}$ & $\underline{(\cdot)}$ denotes the max and min of a variable; $(\cdot)^{(n)}$ represents the n^{th} iteration; $\mathbb{R}, \mathbb{I}$ denotes the real, imaginary part of the complex number, respectively; the superscript p (without parenthesis) denotes the three-phases, i.e., $\{a, b, c\}$ of the system.

2.1 Constraints

The constraints of OPF problems consist of both equality and inequality relations. These constraints are generally composed of power flow models, Distributed Energy Resource (DER) models, and models of various assets such as transformers, regulator taps, and capacitor switches. Additionally, operational and physical limits also form part of the constraints in OPF formulations.

2.1.1 Power Flow Model

This section details the power flow models to formulate the OPF problems for an unbalanced power distribution system. The unbalanced power flow equations are based on our prior work [5]. Let us consider an unbalanced radial power distribution network of n buses where, $\mathcal{N}$ denotes the set of buses in that system and $\mathcal{E}$ denotes the set of edges identifying distribution lines that connect the ordered pair of buses $\{ij\}$, $\forall\, i, j \in \mathcal{N}$. Here, ϕ_j denotes the set of phases in the bus j. Let $v_j^p = |V_j^p|^2$ be the squared magnitude of voltage at bus $j \in \mathcal{N}$ for phase $p \in \phi_j$. Define $\phi_{ij} = \{pq : p \in \phi_i \text{ and } q \in \phi_j \ \forall \{ij\} \in \mathcal{E}\}$. Let $l_{ij}^{pq} = (|I_{ij}^{pp}||I_{ij}^{qq}|)$ be the squared magnitude of the line current flowing in the phase $pq \in \phi_{ij}$ of line (i, j); i.e., the term l_{ij}^{pq} is a mathematical abstraction representing the product of the magnitudes of branch currents in phases p and q. Also, $S_{ij}^{pq} = P_{ij}^{pq} + jQ_{ij}^{pq}$ and $z_{ij}^{pq} = r_{ij}^{pq} + jx_{ij}^{pq}$, where $pq \in \phi_{ij}$. Variable $p_{L,j}^p$ and $q_{L,j}^p$ denote the active and reactive load (respectively) connected at node j of phase $p \in \phi_j$. Similarly, subscript D, j denotes the DER generation at node j; e.g., $p_{D,j}^p$ denotes the active power generation at node j for phase p. δ_{ij}^{pq} is the angle difference between the phase currents. First, we define the nonlinear power flow model and then the linearized model is detailed.

2.1.1.1 Nonlinear Model:

With the approximated phase voltage and branch current angles, [5] developed the nonlinear power flow model for an unbalanced radial power distribution system. The model is defined below in (1). The loads can be modeled as voltage dependent loads as formulated in the Appendix.

$$P_{ij}^{pp} - \sum_{q\in\phi_j} l_{ij}^{pq}\left(r_{ij}^{pq}\cos(\delta_{ij}^{pq}) - x_{ij}^{pq}\sin(\delta_{ij}^{pq})\right) = \sum_{k:j\to k} P_{jk}^{pp} + p_{L,j}^{p} - p_{D,j}^{p} \tag{1a}$$

$$Q_{ij}^{pp} - \sum_{q\in\phi_j} l_{ij}^{pq}\left(x_{ij}^{pq}\cos(\delta_{ij}^{pq}) + r_{ij}^{pq}\sin(\delta_{ij}^{pq})\right) = \sum_{k:j\to k} Q_{jk}^{pp} + q_{L,j}^{p} - q_{D,j}^{p} - q_{C,j}^{p} \tag{1b}$$

$$v_j^p = v_i^p - \sum_{q\in\phi_j} 2\mathbb{R}\left[S_{ij}^{pq}(z_{ij}^{pq})^*\right] + \sum_{q\in\phi_j} z_{ij}^{pq} l_{ij}^{qq} + \sum_{q1,q2\in\phi_j, q1\neq q2} 2\mathbb{R}\left[z_{ij}^{pq1} l_{ij}^{q1q2}\left(\angle(\delta_{ij}^{q1q2})\right)(z_{ij}^{pq2})^*\right] \tag{1c}$$

$$(P_{ij}^{pp})^2 + (Q_{ij}^{pp})^2 = v_i^p l_{ij}^{pp} \tag{1d}$$

$$(l_{ij}^{pq})^2 = l_{ij}^{pp} l_{ij}^{qq} \tag{1e}$$

2.1.1.2 Linear-approximated Model

The computational complexity augmented by the nonlinear power flow models to the existing scalability issues associated with large-scale OPF problems can be reduced using the approximated linearized power flow models. In equation (2), a linear-approximated power flow model – known as the three-phase LinDistFlow model is defined. The assumption is that the line loses are negligible compared to the power flow in the system; however, line impedances are included in the formulation to compute the voltage drops across the lines. [10, 5].

$$P_{ij}^{pp} = \sum_{k:j\to k} P_{jk}^{pp} + p_{L,j}^{p} - p_{D,j}^{p} \tag{2a}$$

$$Q_{ij}^{pp} = \sum_{k:j\to k} Q_{jk}^{pp} + q_{L,j}^{p} - q_{D,j}^{p} - q_{C,j}^{p} \tag{2b}$$

$$v_j^p = v_i^p - \sum_{q\in\phi_j} 2\mathbb{R}\left[S_{ij}^{pq}(z_{ij}^{pq})^*\right] \tag{2c}$$

2.1.2 Voltage dependent loads

The voltage dependent loads are modeled here using the CVR factor developed in [5].This approach linearizes the load modeling, which aids in mitigating computational issues associated with solving OPFs. Let CVR_p and CVR_q be the CVR factor that determines the voltage dependencies of real and reactive power loads, respectively; also, 0 subscript denotes the nominal load value at 1 p.u. voltage. Then the voltage dependent loads can be modeled in the OPF formulation as equation (3). For more details of CVR factors, please refer to [5].

$$p_{L,j}^{p} = p_{j,0}^{p} + \mathrm{CVR}_p \frac{p_{i,0}^{p}}{2}(v_j^p - 1) \tag{3a}$$

$$q_{L,j}^{p} = q_{j,0}^{p} + \mathrm{CVR}_q \frac{q_{i,0}^{p}}{2}(v_j^p - 1) \tag{3b}$$

2.1.3 DG Modeling

We define a general DG model for different network objectives in equation (4)-(6). This DGs are designed as a general model, and can accommodate any DERs, such as Battery Energy Storage System (BESS), EVs, etc. For more please refer to [11]. The general DER model is defined by equation (4) at node j for phase $p \in \phi_j$. If $S^p_{DR,j}$ is defined as the nominal rating of the DG at node j for phase $p \in \phi_j$, then the reactive power generation ($q^p_{D,j}$) and the active power generation ($p^p_{D,j}$) of the DG are constrained by the nominal rating of the DG. When $q^p_{D,j}$ is modeled as the decision variable, $p^p_{D,j}$ is assumed to measured and known. For this case, the DG model evolves from a quadratic inequality (4) to a linear inequality (5) constraint. On the contrary, when the active power generation, $p^p_{D,j}$ is set as decision variables, we assume $q^p_{D,j} = 0$ **THIS IS NO LONGER TRUE. Q can be specified.**, and the DG model is defined by the linear inequality (6).

$$\left(p^p_{D,j}\right)^2 + \left(q^p_{D,j}\right)^2 \leq \left(S^p_{DR,j}\right)^2 \tag{4}$$

$$-\sqrt{(S^p_{DR,j})^2 - (p^p_{D,j})^2} \leq q^p_{D,j} \leq \sqrt{(S^p_{DR,j})^2 - (p^p_{D,j})^2} \tag{5}$$

$$0 \leq p^p_{D,j} \leq S^p_{DR,j} \tag{6}$$

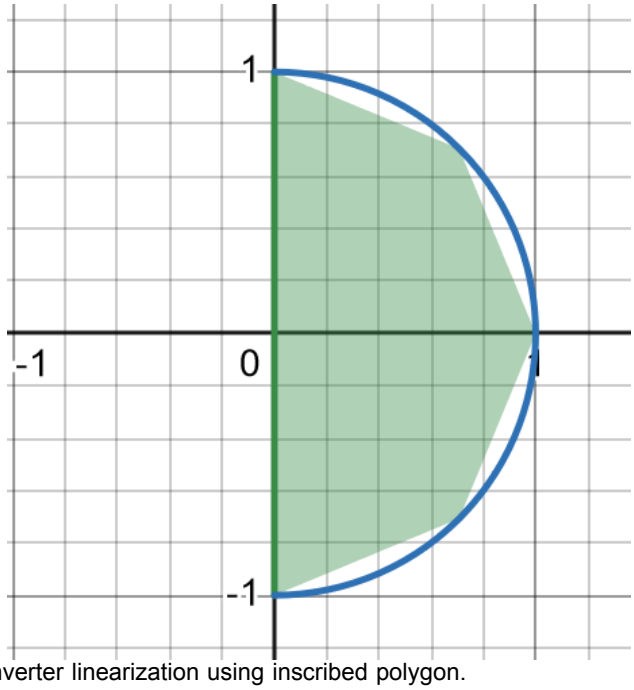

Figure 2. Visualization of inverter linearization using inscribed polygon.

However, for both real and reactive power control, DGs can be linearly approximated. The limits of active and reactive power injection of each DG can be described as a circle with radius $s^{rated}_{D,j}$ on the complex plane. It may also be limited to only produce active power and not absorb it, limiting its operation to the right half plane. To create linear limits we inscribe a polygon in the circle. In this instance we have used an octagon with vertices on the real and imaginary axis and at $\pm 45°$. This is described by (7) and illustrated by Fig. 2.

$$\sqrt{2}p^p_{D,j} + (\sqrt{2}-2)q^p_{D,j} \leq \sqrt{2}s^{rated}_{D,j} \tag{7a}$$
$$\sqrt{2}p^p_{D,j} - (\sqrt{2}-2)q^p_{D,j} \leq \sqrt{2}s^{rated}_{D,j} \tag{7b}$$
$$(-1+\sqrt{2})p^p_{D,j} + q^p_{D,j} \leq s^{rated}_{D,j} \tag{7c}$$
$$(-1+\sqrt{2})p^p_{D,j} - q^p_{D,j} \leq s^{rated}_{D,j} \tag{7d}$$
$$p^p_{D,j} \geq 0 \tag{7e}$$

2.1.4 Regulator Taps

Voltage regulator taps in OPF models are represented through discrete variables that adjust the tap settings to maintain desired voltage levels across the network [11]. The regulator taps are modeled here with the help of a set of binary variables. Let a^p_j be the turn ratio for the voltage regulator on phase p between node i and j. Let $u^p_{tap,k,j} \in 0, 1 \forall k \in \{1, 2, ..., 33\}$ be the binary variable for the regulator on phase p between node i and j; let $b_k \in \{0.9, 0.90625, ..., 1.1\}$. Then, the regulator can be modeled in the OPF formulation as equation (8).

$$B_k = b^2_k, A_j = a^2_i, V^p_j = a^p_j V^p_i, v^p_j = A^p_j v^p_i \tag{8a}$$
$$A^p_j = \sum_{k=1}^{33} B_k u^p_{tap,k,j} \tag{8b}$$
$$\sum_{k=1}^{33} u^p_{tap,k} = 1 \tag{8c}$$

2.1.5 Capacitor switch

Similar to the modeling of the regulator taps, here the capacitor switches are modeled using binary variables that represents the status of the capacitor banks For more please refer to [11, 5]. Let $u^p_{cap,j}$ be the binary variable denoting the switch status of the capacitor bank at phase p of node j; then the capacitor can be modeled in the OPF formulation as equation (9).

$$q^p_{C,j} = u^p_{cap,j} q^{rated,p}_{cap,j} v^p_j \tag{9}$$

2.1.6 Operational Limits

Besides the grid equipment and physics modeling, OPF constraints include the operational bounds as well. Such as thermal limits of the branches and the nodal voltage bounds. The models are expressed below in (10).

$$l^p_{ij} \leq \left(I^{rated}_{ij}\right)^2 \tag{10a}$$
$$\underline{v} \leq v^p_j \leq \overline{v} \tag{10b}$$

2.2 Problem Objective

In this section, a few examples of OPF objectives, specific for distribution systems are presented. Please note that these examples are not exhaustive; using the available OPF

variables, a wide range of objectives or cost functions ($f(x)$, where x is the optimization variable) can be formulated to suit specific requirements.

2.2.1 Example 1: Loss Minimization

The first example of an OPF objective presented here is loss minimization problem where the line loss is reduced by optimal control set-points. The cost function for loss minimization objective is defined in (11a). However, for the linear-approximated models, where the branch current flow variable is not present, the cost function is approximated by (11b). Here all the nodal voltages are assumed as 1.00 p.u. to approximate the objective function (active power line loss) by a convex cost function.

$$\min \sum_{p\in\phi_j, j:i\to j} l_{ij}^{pp} r_{ij}^{pp} \tag{11a}$$

$$\min \sum_{p\in\phi_j, j:i\to j} \Big(\left(P_{ij}^{pp}\right)^2 + \left(Q_{ij}^{pp}\right)^2 \Big) r_{ij}^{pp} \tag{11b}$$

2.2.2 Example 2: Maximize Generators Output

Another OPF objective could be maximizing the generator outputs. The objective to maximize the output of distributed generators aims to fully utilize the generation capacity. The function is defined in (12).

$$\max \sum_{p\in\phi_j,\ \forall j\in\mathcal{N}} p_{D,j}^{p} \tag{12}$$

2.2.3 Example 3: Minimize Total Substation Active Power Load

The third OPF objective example is to minimize the total substation active power load seeks to reduce the active power drawn from the substation by optimally set the control variables. The OPF objective is defined in (13).

$$\min \sum_{p\in\phi_{sub}} p_{sub}^{p} \tag{13}$$

2.2.4 Example 4: Minimize DER/Generator Curtailment

The objective of minimizing DER/Generator curtailment focuses on reducing the power that needs to be curtailed from distributed energy resources and generators. In the DER curtailment minimization problem, active power generations from the DERs are expressed as control variables. Let $\overline{p_{D,j}^{p}}$ denotes the maximum available power generation at node j for phase p, then (14) defines the associated cost function.

$$\min \sum_{\forall j\in\mathcal{N}} \sum_{p\in\phi_j} (\overline{p_{D,j}^{p}} - p_{D,j}^{p})^2 \tag{14}$$

2.2.5 Example 5: Substation Load Tracks Total Active Power

The final example of OPF objectives presented here involves the substation load tracking a predetermined total active power target. This helps in ensuring that the overall active power demand at the substation aligns with the target, aiding in load balancing and efficient power distribution. This problem is quadratic, and defined by (15)

$$\min \left(\left(\sum_{p \in \phi_{sub}} p_{sub}^{p} \right) - P_{sub}^{target} \right)^2 \tag{15}$$

In conclusion, the OPF objectives described here are just a few examples tailored for distribution systems. It is important to note that various other cost functions can be designed to meet different operational goals and requirements.

2.3 OPF Problem

In this section, we define the OPF problems with developed constraint models for any objective function $f(x)$ for an unbalanced radial distribution system, where x represent the optimization variable.

$$\min \quad f(x) \tag{16a}$$

s.t.

Power Flow Model:	equation (1) or (2), (3)	(16b)
DG Model:	equation (4), or (5), or (6), or (7)	(16c)
Asset Model:	equation (8) - (9)	(16d)
Operational Limits:	equation (10)	(16e)

In this section, the OPF problems for distribution systems have been defined by developing constraint models and appropriate objective or cost functions within an unbalanced radial distribution system. A comprehensive formulation of the OPF problem has been presented, incorporating models for power flow, voltage-dependent loads, DGs, various assets, and operational limits. These detailed constraint models provide a robust framework for optimizing the operation of distribution systems, ensuring that different operational objectives can be effectively achieved.

3.0 Inputs For Optimal Power Flow

Given the OPF problem formulation in (16), a case will require that all the necessary parameters be defined. The for the load model in (3), nominal active ($p^p_{j,0}$) and reactive power ($q^p_{j,0}$) and corresponding voltage sensitivity (CVR_p and CVR_q) must be specified for each phase of each bus. Generators are specified with an apparent power rating and active and reactive power. Since the equations used depend on the OPF configuration, these parameters required also depend on the OPF configuration. If the OPF is using generator active power as a decision variable then the specified active power will be used as an active power limit. Otherwise it is used as a constant. The specified reactive power is only used if generator reactive power is a constant. The capacitor equation, (9), require the nominal reactive power injection (at 1 p.u. nominal voltage) is specified. Regulator equations, (8) require the active tap position for each phase, A^p_j, if they are constant. If mixed integer regulator control is used then the OPF will determine the tap positions. Finally, the power flow equations in (2) only require the impedance matrix for each line.

3.1 Modeling Parameters: Native Input Format

With the goal of creating easily human readable case files, DistOPF ingests this data in the form of five CSV files as a native input:

1. branch_data.csv: Branch data including r (resistance) and x (reactance) values. See Table 1.
2. bus_data.csv: General bus data and load information. See Table 2.
3. gen_data.csv: Generator data. See Table 3.
4. cap_data.csv: Capacitor bank data. See Table 4.
5. reg_data.csv: Tap changing voltage regulator data. See Table 5.

The CSVs are comma delimited. The first row contains the names of each column. The column names must not have spaces between them since they will be interpreted as part of the column name. The order of the columns does not matter. If more information is required for a new model it is simple to add a new column to the appropriate CSV with the additional data.

The CSV files have columns for the required parameters discussed above as well as base unit values and bus names as well as other useful information. The branch_data.csv file also has a column for tracking line type information and status for keeping track of switches. The bus_data.csv has columns for bus latitude and longitude which is necessary for network visualizations. The gen_data.csv file allows the user to specify additional reactive power limits which is useful when only positive or only negative reactive power injection is desired. It also has a column which allows the user to specify the OPF control variables independently on each generator.

The columns, fb, tb, and id, refer to the bus id which will need to be generated specifically for DistOPF. The bus ids are integer values starting at 1 and 1 is always the SWING bus. The remainder of the buses are sorted with depth-first-search and numbered accordingly.

Table 1. branch_data.csv

CSV/DataFrame Column	Description
fb	From-bus id number
tb	To-bus id number
from_name	From-bus name
to_name	To-bus name
name	name of line
raa, rab, rac, rbb, rbc, rcc	resistance in p.u. (lower triagular matrix)
xaa, xab, xac, xbb, xbc, xcc	reactance in p.u. (lower triagular matrix)
type	overhead_line, switch, transformer, etc.
status	(for switches) “OPEN” or “CLOSED”
s_base	base VA per phase
v_ln_base	base line-to-neutral voltage
z_base	base impedance
phases	phases present

Table 2. bus_data.csv

CSV/DataFrame Column	Description
id	unique id for each bus (integer starting at 1)
name	bus name
pl_a, ql_a, pl_b, ql_b, pl_c, ql_c	active and reactive loads (p.u.)
bus_type	SWING or PQ; SWING bus is voltage source
v_a, v_b, v_c	voltage magnitude (p.u.); input parameter for SWING bus. Other not used as input)
v_ln_base	base line-to-neutral voltage (V)
s_base	base power (VA)
v_min, v_max	voltage magnitude limits (p.u.)
cvr_p, cvr_q	conservative voltage reduction parameters; alternative to ZIP model for voltage dependant loads. (set to 0 for no voltage dependence) (see. [?])
phases	phases at bus (e.g.‘abc’, ‘a’, ‘ab’, etc.)
latitude	node latitude; useful for plotting
longitude	node longitude; useful for plotting

Table 4. cap_data.csv

CSV/DataFrame Column	Description
id	bus id (integer)
name	capacitor name
q_a, q_b, q_c	nominal reactive power (p.u.)
phases	phases (e.g.'abc', 'a', 'ab', etc.)

Table 3. gen_data.csv

CSV/DataFrame Column	Description
id	bus id (integer)
name	generator name
pa, pb, pc	active power output (p.u.)
qa, qb, qc	reactive power output (p.u.)
s_base	base power (VA)
sa_max, sb_max, sc_max	rated maximum apparent power output (VA)
phases	generator phases (abc string) (e.g.'abc', 'a', 'ab', etc.)
qa_max, qb_max, qc_max	maximum reactive power output (p.u.)
qa_min, qb_min, qc_min	minimum reactive power output (p.u.)
control_variable	Which generator variables are control variables. "PQ", "P", "Q" or ""

Table 5. reg_data.csv

CSV/DataFrame Column	Description
fb	From-bus id number (integer)
tb	To-bus id number (integer)
from_name	From-bus name
to_name	To-bus name
name	regulator name
tap_a, tap_b, tap_c	tap position (p.u.) -16 to +16; 0 is no tap change

3.2 Standard Network Models: Converting from Other Formats

To convert from any other standard network model format, the model data described above must be extracted and then formatted as Pandas DataFrames and, optionally, saved as CSVs according to Tables 1 and 2 and, as needed, Tables 3, 4, and 5. Before converting a model, it is important to check if that the model is compatible. In the current version, the model must be radial as that is a fundamental assumption. Only wye-wye transformers are supported. Also, models with split-phase secondary lines are not supported. If a model has loads or other assets on split-phase secondary lines they may be reflected onto the primary of the secondary transformers before converting.

3.2.1 OpenDSS Network Model

OpenDSSDirect.py is a Python that can be used to parse OpenDSS models in python. It can be used to aid conversion from OpenDSS. An OpenDSS converter is supplied with DistOPF to allow easy use of OpenDSS models as input. The converter populates the CSV tables required to create the OPF model.

3.2.2 CIM Equipment Model

Extracting a model stored using the CIM standard can be accomplished in Python with the CIMantic Graphs Library using the “cimhub_2023” profile. This process involves extracting the necessary data, i.e., modeling parameters from the CIM model to populate the native CSV table format required for the OPF solver modeling language. By generating these CSV tables from the CIM data, user can efficiently run DistOPF.

4.0 Using DistOPF

This section will briefly describe how to use the basic features of DistOPF version 0.2.0. DistOPF is actively being developed. For the most up to data documentation visit `github.com/Scalelab-wsu/distopf.git`.

4.1 Installation

To get started with DistOPF, the package needs to be installed. This can be achieved quickly and easily using Python's package installer, 'pip'.

4.1.1 pip install

Simply copy and paste the below command line in the terminal:

```
pip install distopf
```

4.1.2 Developer Installation

To install the latest version from github:

1. From the directory you want to keep your distopf files, run:
   `git clone https://github.com/Scalelab-wsu/distopf.git`
3. Create or activate the python environment you want to use.
4. From the directory where the distopf package is stored, run:
   `pip install -e .`

This installs your local distopf package the python environment you activated. The `-e` option enables editable mode, which allows you to directly edit the package and see changes immediately reflected in your environment without reinstalling.

4.2 Basic Example

This section builds up a basic example describing each block of code and the alternatives functions or classes that may be used.

4.2.1 Loading Case Data

First we import the DistOPF library and as the Pandas Library. Then we load the CSVs from the file system using Pandas Please note, besides the native model inputs, i.e., CSVs, user can use standard models such as OpenDSS or CIM, with associated appropriate converters. Several cases are provided with the package including cases formated as CSVs and OpenDSS models. These are stored in `distopf/cases/`. For convenience you can access this path in DistOPF with `distopf.cases.CASES_DIR`. The cases are separated CSV models and OpenDSS models by two subdirectories `csv` and `dss`.

```
import pandas as pd
import distopf as opf
```

```
branch_data = pd.read_csv("branch_data.csv", header=0)
bus_data = pd.read_csv("bus_data.csv", header=0)
gen_data = pd.read_csv("gen_data.csv", header=0)
cap_data = pd.read_csv("cap_data.csv", header=0)
reg_data = pd.read_csv("reg_data.csv", header=0)
```

4.2.2 Instantiating the Model

The model is instantiated using each of the DataFrames loaded above. In this example we are using `LinDistModel`, however other models may be used depending on the features and performance requirements. `LinDistModel` is recommended since it is has most features and good performance. If control of capacitor switching is desired, then `LinDistModelCapMI` which provides a mixed integer formulation can be used. `LinDistModelCapacitorRegulatorMI` should be used if regulator tap control is required.

```
model = opf.LinDistModel(
    branch_data=branch_data,
    bus_data=bus_data,
    gen_data=gen_data,
    cap_data=cap_data,
    reg_data=reg_data,
)
```

4.2.3 Solving the OPF

Once the model is instantiated, it can be solved. In this example the `lp_solve` function is used to solve the model with the `gradient_load_min` objective.

```
result = opf.lp_solve(model, opf.gradient_load_min(model))
```

The `lp_solve` function is used when the objective function is linear and requires the model and the objective function gradient vector be supplied. Additionally, `cvxpy_solve` can be used which uses the CVXPY python package. Instead of a gradient, a function is passed as the second argument (just the function name without parenthesis).

```
result = opf.cvxpy_solve(model, opf.cp_obj_loss)
```

When using `LinDistModelCapMI` the `cvxpy_mi_solve` must be used and when using `LinDistModelCapacitorRegulatorMI`, then the `solve` method of that class must be used (`model.solve(f)`).

The solve functions described above all return a object which includes various attributes including the resulting vector of the variables, `x`, and the objective value, `fun`. Each of the model classes includes methods for parsing the values from `x` into easy to understand Pandas DataFrames with a column for bus names and each phase.

```
# Parse results
v = model.get_voltages(result.x)
s = model.get_apparent_power_flows(result.x)
p_gens = model.get_p_gens(result.x)
q_gens = model.get_q_gens(result.x)
```

4.2.4 Visualizing Results

Visualizations are produced using the Python Plotly package. Because we are using Plotly, all the visualizations are interactive and easy to explore. Functions are provided for plotting voltages, power flows, and generators. Additionally, `plot_network` can be used to create a network plot which makes it easy to quickly understand how power flows in the network and how voltages are distributed geographically (Fig. 3). Using the mouse, the user can hover over nodes to get exact values of power flow, loads, generation, and capacitor reactive power for each node. An example of plots from each of the four plotting functions used in the example are shown in figures 3, 4, 5, and 6.

```
# Visualize network and power flows
opf.plot_network(model, v=v, s=s, p_gen=p_gens, q_gen=q_gens).show()
opf.plot_voltages(v).show()
opf.plot_power_flows(s).show()
opf.plot_gens(p_gens, q_gens).show()
```

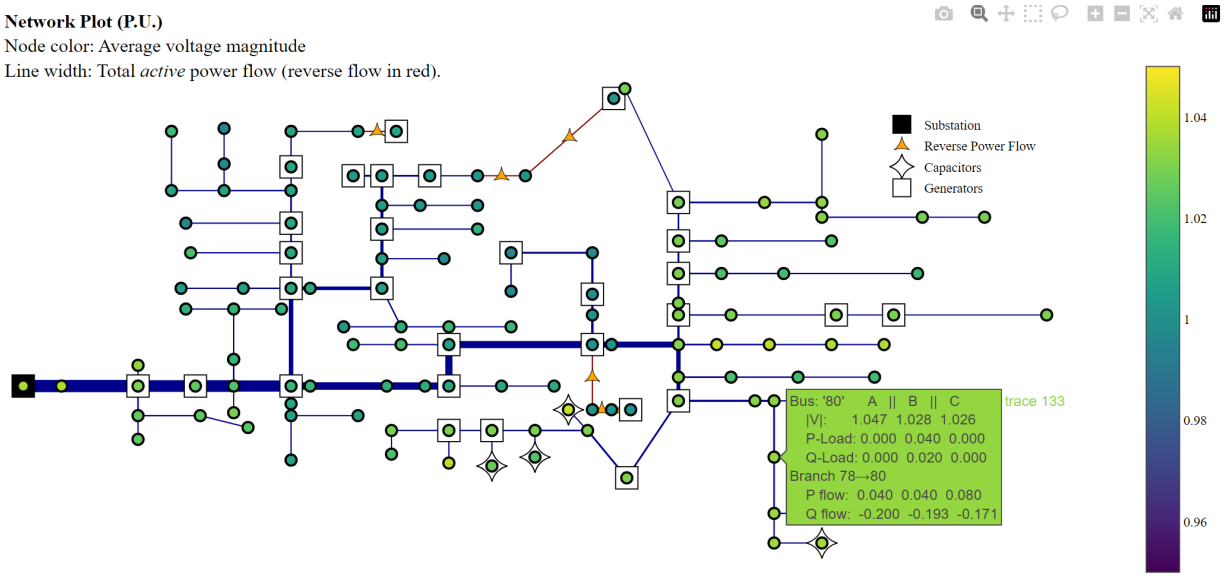

Figure 3. Result of running `opf.plot_network` after solving the IEEE 123 bus system with 30 DERs. Additional information is available by hovering over any of the buses. In this example hover-data is shown for bus "80".

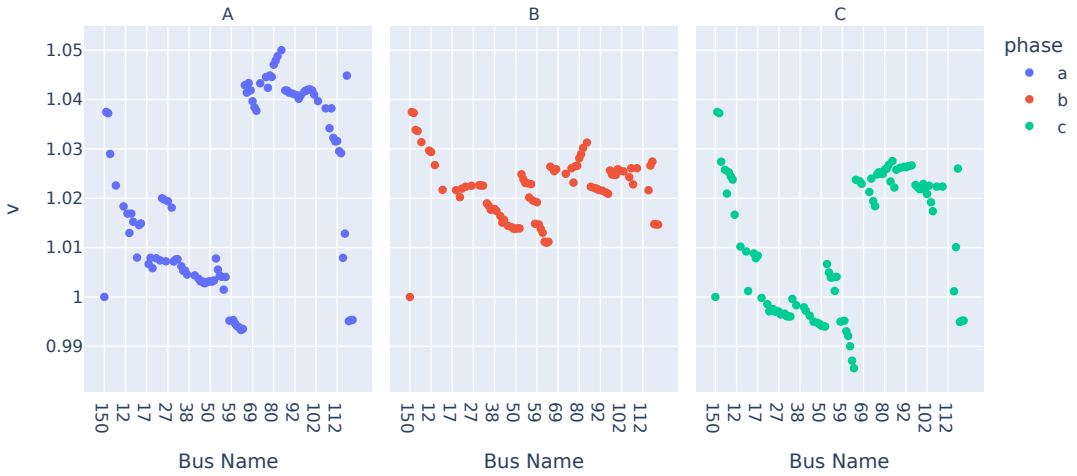

Figure 4. Result of running `opf.plot_voltages` showing nodal voltages after solving the IEEE 123 bus system with 30 DERs.

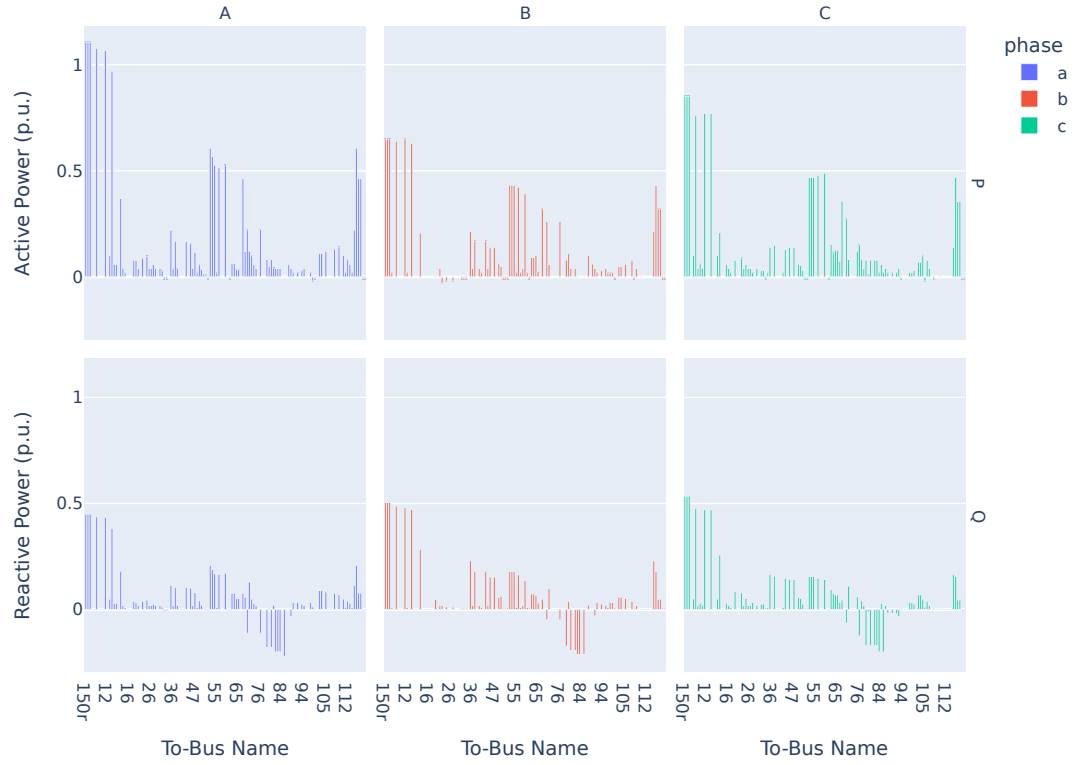

Figure 5. Result of running `opf.plot_power_flows` showing active and reactive power flows (branch flows) after solving the IEEE 123 bus system with 30 DERs.

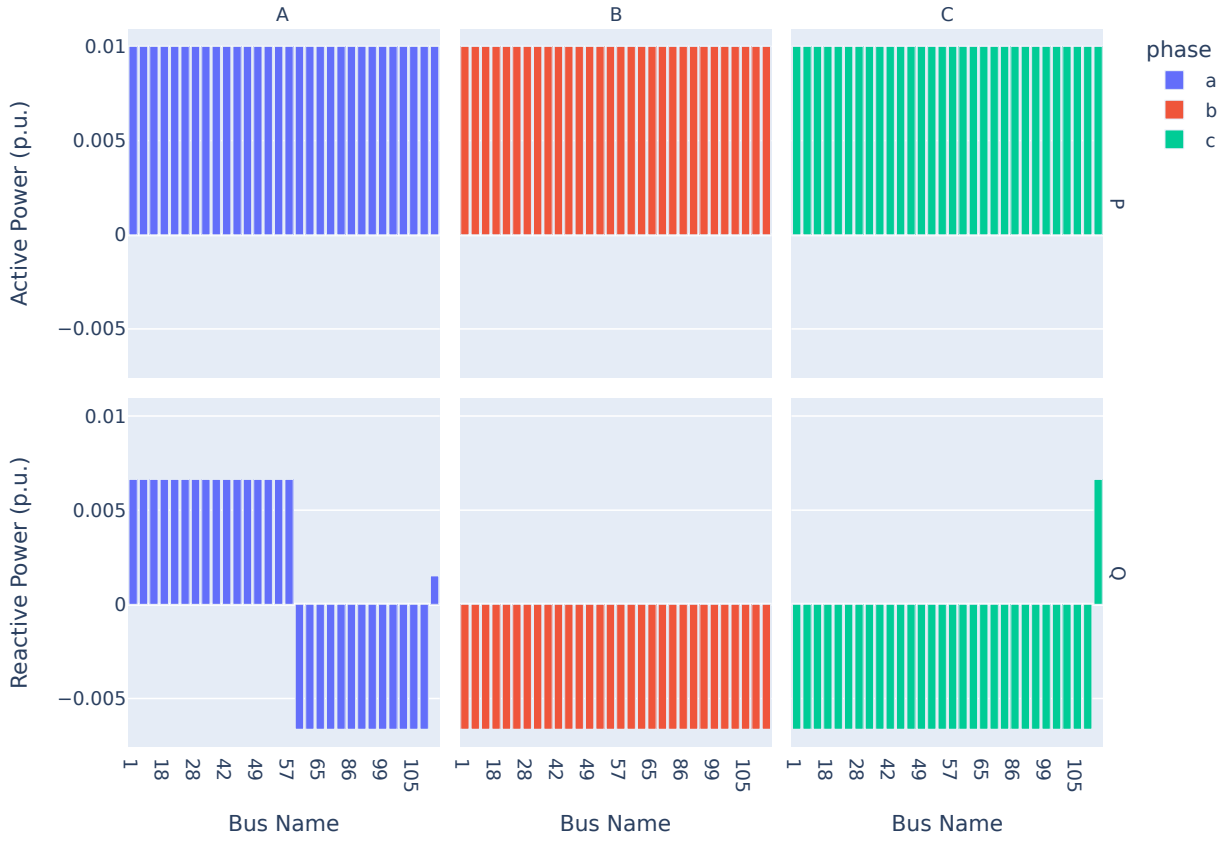

Figure 6. Result of running `opf.plot_gens` showing optimal generations after solving the IEEE 123 bus system with 30 DERs.

4.3 Advanced Options

The advanced options of the DistOPF package offer enhanced flexibility and customization for power flow optimization. Users can write custom objective functions, specify control variables, and set precise voltage limits, enabling tailored solutions for diverse operational requirements and constraints. These features empower users to fine-tune the optimization process to meet specific goals and conditions.

4.3.1 Writing Objective Functions

DistOPF provides a variety of objective functions built in, however, it does not aim to provide all possible objective functions that users may want. This section aims to provide users with an understanding of how an objective function may be formulated.

There are two types of objective functions supported as of the version 0.2.0 release, linear function gradients for use with `distopf.lp_solve` and convex functions for use with `distopf.cvxpy_solve`.

4.3.1.1 Linear Functions

Linear functions take the model and any optional keyword arguments as inputs and return a 1-dimensional array, c, representing the gradient of the objective function and having the same length as x. The solve function, `distopf.lp_solve`, will minimize $c \cdot x$.

The load minimization example below implements (13). A value of 1 is placed in the gradient array at the index for each of the variables corresponding to active power flow out of the swing bus. Here `model.phase_exists` is used to prevent trying to access a variable for a phase that doesn't exist. Also, `model.idx` is used to retrieve the indices of the variables for active power flow out of the swing bus. The first argument of model.idx gets the name of the variable, the second gets the index of the bus, j, and the last gets the phase "a", "b" or "c". It is important to note that j is 0-indexed and corresponds to the bus-id minus one; therefore the swing bus which should have an id of 1 has in index of 0.

```
def gradient_load_min(model: LinDistBase, *args, **kwargs) -> np.ndarray:
    c = np.zeros(model.n_x)
    for ph in "abc":
        if not model.phase_exists(ph):
            continue
        c[model.idx("pjk", model.swing_bus, ph)] = 1
    return c
```

4.3.1.2 Convex Functions

Convex functions are designed to be compatible with the CVXPY Python Package. They take the model, a CVXPY `cvxpy.Variable` object and optional keyword arguments. It returns a CVXPY expression. For the best performance, it is best to formulate the expressions in a vectorized form. You must also use disciplined convex programming (DCP) as described by the CVXPY documentation.

The example below implements (14) by collecting the indices for all the variables corresponding to active power generation on each phase. The parameter, `model.pg_map`, is a dictionary of Pandas Series for each phase containing the indices for each active power generation variable. The code below concatenates the indices for each phase into `all_pg_idx` which is used to formulate the equation in a vectorized form.

```
def cp_obj_curtail(model: LinDistBase, xk: cp.Variable, **kwargs) -> cp.
↪Expression:
    all_pg_idx = np.array([])
    for a in "abc":
        if not model.phase_exists(a):
            continue
        all_pg_idx = np.r_[all_pg_idx, model.pg_map[a].to_numpy()]
    all_pg_idx = all_pg_idx.astype(int)
    return cp.sum((model.x_max[all_pg_idx] - xk[all_pg_idx]) ** 2)
```

4.3.2 Specifying Control Variables

The control variables used in the OPF include active and reactive power injection from generators (and capacitor switching and regulator tap positions for mixed integer models). Control variables can be specified for each generator by listing “PQ”, “P”, “Q”, (or nothing if no control is desired) in the `control_variable` column of `gen_data.csv`.

4.3.3 Specifying Voltage Limits

Voltage limits are specified for each node using the `v_min` and `v_max` columns in the `bus_data.csv`.

5.0 Validation

This section focuses on validating the results obtained from simulations using Distopf against those generated by OpenDSS. Validation is crucial for ensuring the reliability and accuracy of our power flow models and conversion methods.

The results for both the IEEE 13-Bus and IEEE 123-Bus systems are presented in Table 6. For validation, we analyze power flow parameters including voltages (V), real power (P), and reactive power (Q). The goal is to confirm that the converted models yield consistent and credible results across different platforms. Detailed comparisons of the voltage and power flows for both networks are shown in figures 8, 9, 10, 11 and 12.

We use several error metrics to quantify discrepancies between the results from DistOPF and OpenDSS:

1. Maximum relative voltage error ($\%V$ Error)
2. Maximum voltage error in per-unit (p.u.)
3. Real power mismatch at the substation compare to the system's size
4. Reactive power mismatch at the substation compare to the system's size

Table 6. Power flow validation with OpenDSS for several models.

Model	%V Error	V Error (p.u.)	Real Power (MW)		Reactive Power (MVAR)	
			ΔP	Size (P)	ΔQ	Size (Q)
IEEE 13-Bus	0.809%	0.00844	0.113	3.45	0.32	2.10
IEEE 123-Bus	0.499%	0.00516	0.097	3.51	0.196	1.93

The linear approximation utilized in the modeling process does not account for power losses, which causes expected errors to vary depending on the power losses present in each system. To examine this relationship, we adjusted the load multiplier from 0 to 1 and plotted the voltage error relative to the power loss. Figure 7 illustrates this effect.

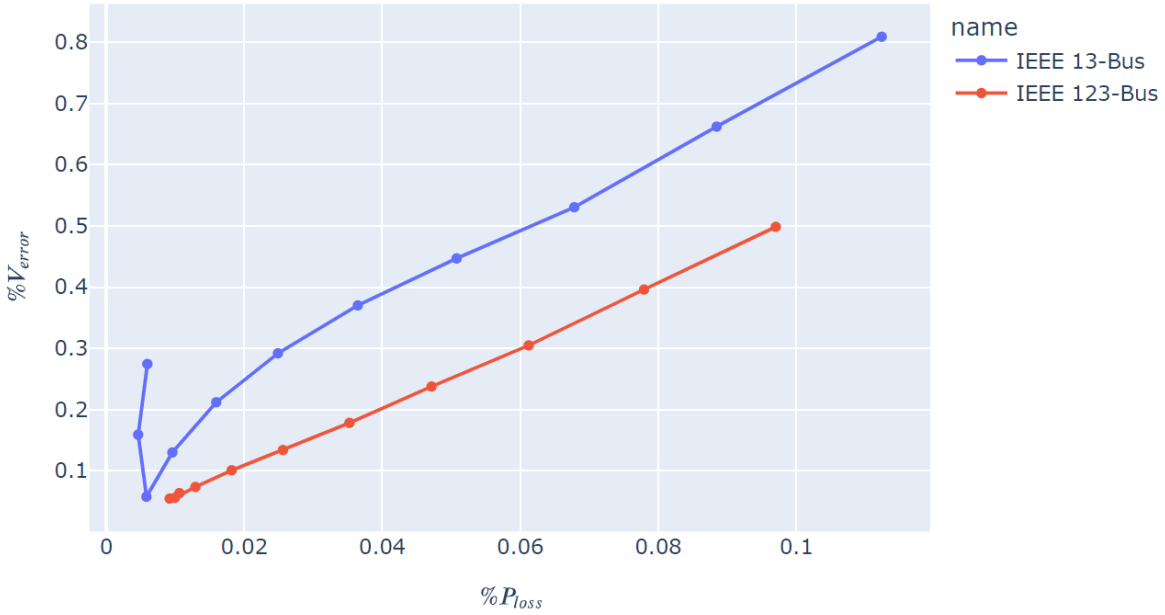

Figure 7. Maximum voltage error relative to OpenDSS solution shows a linear relationship to active power loss calculated by OpenDSS.

Figures 8, 9, 10, 11, and 12 further validates the solutions by comparing all the nodal voltages and branch power flows for IEEE 13-Bus and IEEE 123-Bus test system.

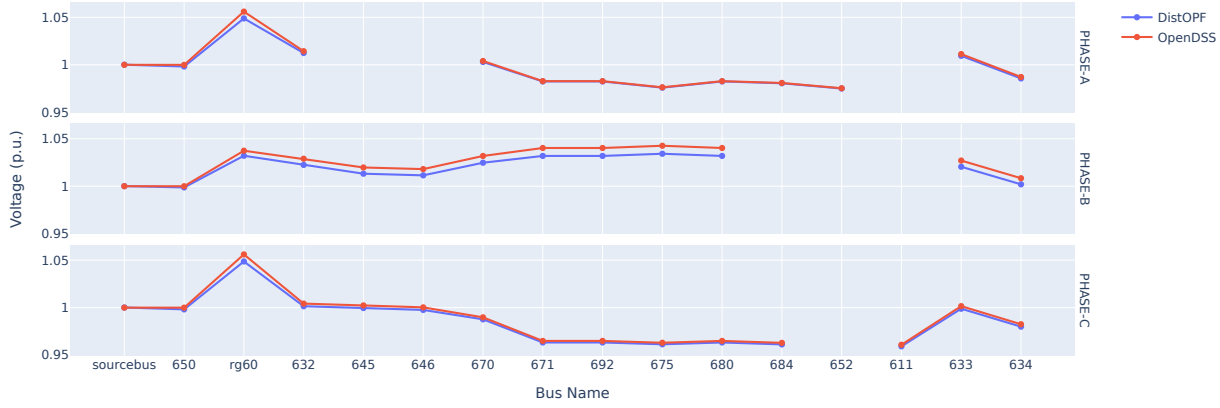

Figure 8. DistOPF voltage comparison with OpenDSS solutions for IEEE 13-Bus.

Figure 9. DistOPF branch power flow comparison with OpenDSS solutions for IEEE 13-Bus.

Figure 10. DistOPF voltage comparison with OpenDSS solutions for IEEE 123-Bus.

Figure 11. DistOPF branch active power flow comparison with OpenDSS solutions for IEEE 123-Bus.

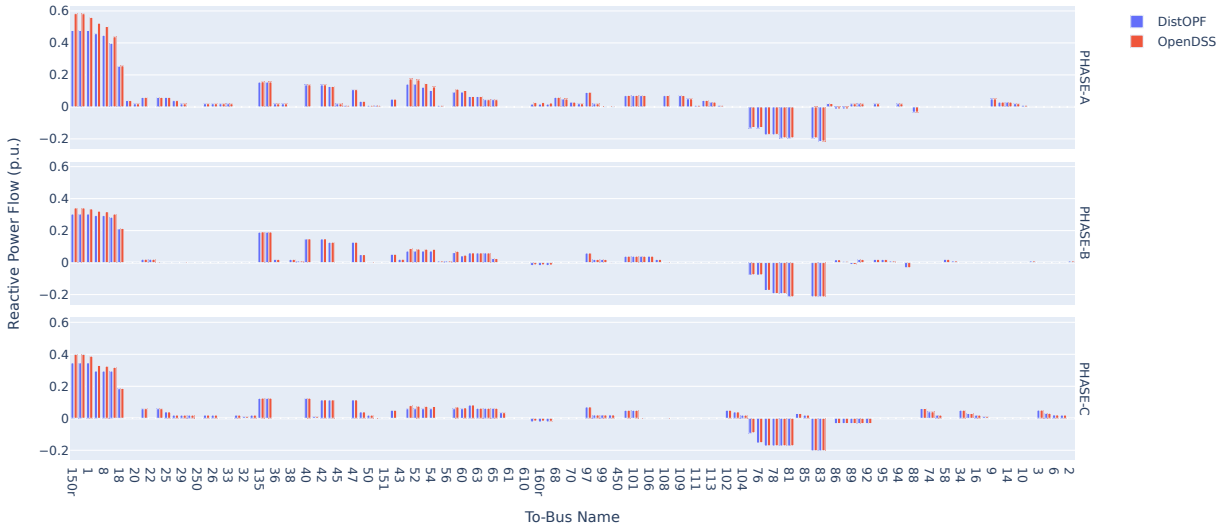

Figure 12. DistOPF branch reactive power flow comparison with OpenDSS solutions for IEEE 123-Bus.

6.0 Conclusion

In conclusion, DistOPF significantly enhances the efficiency and effectiveness of power distribution system optimization by providing compatibility with various standard network models, such as OpenDSS and CIM equipment models. It equips researchers with detailed insights into OPF modeling, rooted in solid mathematical principles, and outlines essential modeling parameters for formulating the OPF and the OPF modeling language. The tool demonstrates how to use and navigate different OPF problems, specifies the necessary modeling parameters for OPF solver languages, interprets results through useful plots, and validates these solutions against OpenDSS power flow results to ensure feasibility. By reducing the development time for optimization programs, DistOPF enables a quicker advancement in power distribution system research. Researchers can customize the tool to meet their specific needs, benefiting from comprehensive benchmarking options that ultimately contribute to the creation of a more resilient and efficient energy system.

References

[1] Shiva Poudel et al. “Fairness-Aware Distributed Energy Coordination for Voltage Regulation in Power Distribution Systems”. In: *IEEE Transactions on Sustainable Energy* (2023).

[2] Jeffrey D Taft, Paul De Martini, and Rick Geiger. *Ultra-large-scale power system control and coordination architecture: A strategic framework for integrating advanced grid functionality*. Tech. rep. Pacific Northwest National Lab.(PNNL), Richland, WA (United States), 2014.

[3] Daniel K Molzahn et al. “A survey of distributed optimization and control algorithms for electric power systems”. In: *IEEE Transactions on Smart Grid* 8.6 (2017), pp. 2941–2962.

[4] Kevin P Schneider et al. *Modern grid initiative distribution taxonomy final report*. Tech. rep. Pacific Northwest National Lab.(PNNL), Richland, WA (United States), 2008.

[5] Rahul Ranjan Jha et al. “Bi-level volt-var optimization to coordinate smart inverters with voltage control devices”. In: *IEEE Transactions on Power Systems* 34.3 (2019), pp. 1801–1813.

[6] Kevin P Schneider, BA Mather, Pal, et al. “Analytic considerations and design basis for the IEEE distribution test feeders”. In: *IEEE Transactions on power systems* 33.3 (2017), pp. 3181–3188.

[7] Patrick Panciatici et al. “Advanced optimization methods for power systems”. In: *2014 Power Systems Computation Conference*, pp. 1–18.

[8] Steven Diamond and Stephen Boyd. “CVXPY: A Python-embedded modeling language for convex optimization”. In: *Journal of Machine Learning Research* 17.83 (2016), pp. 1–5.

[9] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[10] Lingwen Gan and Steven H Low. “Convex relaxations and linear approximation for optimal power flow in multiphase radial networks”. In: *2014 Power Systems Computation Conference*. IEEE. 2014, pp. 1–9.

[11] Rabayet Sadnan et al. “Scaling Distributed Optimal Renewable Energy Coordination in Unbalanced Distribution Systems”. In: *IEEE Transactions on Sustainable Energy* (2024).