AC Power Flow Feasibility Restoration via a State Estimation-Based Post-Processing Algorithm

Babak Taheri and Daniel K. Molzahn

***Abstract*—This paper presents an algorithm for restoring AC power flow feasibility from solutions to simplified optimal power flow (OPF) problems, including convex relaxations, power flow approximations, and machine learning (ML) models. The proposed algorithm employs a state estimation-based post-processing technique in which voltage phasors, power injections, and line flows from solutions to relaxed, approximated, or ML-based OPF problems are treated similarly to noisy measurements in a state estimation algorithm. The algorithm leverages information from various quantities to obtain feasible voltage phasors and power injections that satisfy the AC power flow equations. Weight and bias parameters are computed offline using an adaptive stochastic gradient descent method. By automatically learning the trustworthiness of various outputs from simplified OPF problems, these parameters inform the online computations of the state estimation-based algorithm to both recover feasible solutions and characterize the performance of power flow approximations, relaxations, and ML models. Furthermore, the proposed algorithm can simultaneously utilize combined solutions from different relaxations, approximations, and ML models to enhance performance. Case studies demonstrate the effectiveness and scalability of the proposed algorithm, with solutions that are both AC power flow feasible and much closer to the true AC OPF solutions than alternative methods, often by several orders of magnitude in the squared two-norm loss function.**

***Index Terms*—Optimal power flow (OPF), state estimation (SE), operating point restoration, machine learning (ML)**

I. Introduction

Optimal power flow (OPF) problems seek steady-state operating points for a power system that minimize a specified objective, such as generation costs, losses, voltage deviations, etc., while satisfying equalities that model power flows and inequalities that impose engineering limits. The OPF problem forms the basis for many power system applications. Algorithmic improvements have the potential to save billions of dollars annually in the U.S. alone [1]. The AC power flow equations accurately model how a power system behaves during steady-state conditions by relating the complex power injections and line flows to the voltage phasors. Optimal power flow problems that utilize an AC power flow model are not easily solvable and are considered computationally challenging (NP-hard) [2].

Since first being formulated by Carpentier in 1962 [3], OPF solution methods have been extensively researched [4], [5]. With formulations based on the Karush-Kuhn-Tucker (KKT) conditions, many solution methods only seek a local optimum due to the nonconvex nature of the OPF problem. This nonconvexity is due to the nonlinearity of the relationships between active and reactive power injections and voltage magnitudes and angles [6]. Challenges from power flow nonconvexities are further compounded when solving OPF problems that consider, e.g., discreteness and uncertainty [7], [8]. To overcome these challenges, it is common to simplify OPF problems to convex formulations via relaxation and approximation techniques, often resulting in semidefinite programs (SDP) [9], second-order cone programs (SOCP) [10], [11], and linear programs [12]; see [13] for a survey. A wide range of emerging machine learning (ML) models are also under intense study [14]–[18]; see [19] for a survey. In this paper, we collectively refer to relaxed, approximated, or ML-based OPF formulations as *simplified* OPF problems.

Relaxed OPF problems bound the optimal objective value, can certify infeasibility, and, when the relaxation is tight, provide globally optimal decision variables [13]. With potential advantages in computational tractability and accuracy when deployed appropriately, power flow approximations are also used in a wide range of operational and design tasks [13]. Whether developed via relaxations or approximations, the convexity of these formulations is valuable in many applications, enabling, for instance, convergence guarantees for distributed optimization algorithms [20] and tractability for robust and chance-constrained formulations that consider uncertainties [21], [22]. Similarly, machine learning models hold substantial promise in certain applications, such as quickly characterizing many power injection scenarios with fluctuating load demands and renewable generator outputs [19].

The computational advantages of simplifying OPF problems via relaxation, approximation, and machine learning techniques come from replacing the nonconvex AC power flow equations with some other model. As a result, all of these simplified OPF problems may suffer from a key deficiency in *accuracy* that is the motivation for this paper. Specifically, the outputs of a simplified OPF problem may not satisfy the AC power flow equations, meaning that the power injections and line flows may be inconsistent with the voltage phasors [8], [13], [22], [23]. This is problematic since many practical applications for OPF problems require solutions that satisfy the power flow equations to high accuracy.[1]

Due to these inaccuracies, there is a need to develop restoration methods that obtain voltage phasors and power

The authors are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA. This research was supported by NSF award #2145564. Provision of data for the machine learning case study results was supported by NSF award #2112533.

[1]Prior research has identified sufficient conditions which ensure the tightness of certain relaxed OPF problems, but the assumptions underlying these conditions make them inapplicable for many practical situations [13], [24].

injections conforming to the AC power flow equations from the outputs of relaxed, approximated, and ML-based OPF models. There are three main types of methods in the literature that address this. The first type adds penalty terms to the objective function of a relaxed OPF problem. Appropriate choices for these penalty terms can result in the relaxation being tight for the penalized problem, yielding feasible and near-optimal solutions for some OPF problems; see, e.g., [25]. However, determining the appropriate penalty parameters can be challenging and is often done in a trial-and-error fashion, which can be time-consuming [26]. The second type updates the power flow relaxations and approximations within an algorithm that tries to find a local optimum; see, e.g., [27], in which the OPF problem is formulated as a difference-of-convex programming problem and iteratively solved by a penalized convex–concave procedure. As another example, the algorithm in [28] utilizes a power flow approximation to generate an initial operating point and then seeks small adjustments to the outputs of a select number of generators to obtain an operating point that satisfies both the equality and inequality constraints of an OPF problem. These methods can find local optima, but they require good starting points and multiple evaluations of the relaxed or approximate problem, which can be computationally expensive. See [13, Ch. 6] for a survey of these first two types of methods.

This paper is most closely related to a third type of method that is faster and more direct. This third type of method fixes selected values from the solution of the simplified OPF problem to formulate a power flow problem that has the same number of variables and equations. Solving this power flow problem then yields values for the remaining quantities that satisfy the AC power flow equations. There are various ways to formulate this power flow problem. For instance, one method fixes active power injections and voltage magnitudes at non-slack generator buses to create a power flow problem that can be solved using traditional Newton-based methods [26]. One could instead fix the active and reactive power injections at non-slack generator buses and then solve a power flow problem. As another alternative, one could directly substitute the voltage magnitudes and angles from the simplified OPF solution into the power flow equations to obtain consistent values for active and reactive power generation. This third type of method for restoring AC power flow feasibility is commonly used in prior literature, usually as part of a larger algorithm [15], [17], [26], [29]–[31]. Note that this third type of method may result in variable values that do not align with the solution of the simplified problem and may also violate the OPF problem's inequality constraints. Nevertheless, quickly obtaining an AC power flow feasible point that is close to the true OPF solution is often of paramount importance.

Our proposed algorithm for restoring power flow solutions is inspired by ideas from state estimation [32] and offers significantly improved accuracy over alternatives from this third type of method. Instead of fixing some subset of values from a simplified OPF solution, we instead formulate an optimization problem that incorporates additional information regarding the voltage phasors, power injections, and line flows from relaxed, approximated, and ML-based solutions. Our algorithm addresses inaccuracies in power flow models similarly to how measurement errors are handled in state estimation. The use of additional information from simplified solutions allows for a more accurate restoration of the true OPF solution while leveraging mathematical machinery from state estimation. Additionally, our algorithm can simultaneously consider outputs from multiple simplified OPF problems to both improve the quality of the restored solution and automatically characterize the accuracy of different power flow simplifications.

Our proposed algorithm also allows for flexibility in selecting weights, which are comparable to the variances of sensor noise levels in state estimation algorithms, and biases. These weights and biases are parameters that are chosen based on inconsistencies in the solutions to the simplified OPF problems. Determining the best values for these weights and biases is challenging as the inconsistencies in the solutions are not known beforehand. Inspired by ML approaches, we solve this issue via offline training of the weights and biases using the solutions to many actual OPF problems and the corresponding outputs of the simplified OPF problems. To do so, we employ an adaptive stochastic gradient descent (ASGD) algorithm to iteratively update the weights and biases based on information from our proposed restoration algorithm. This minimizes a loss function that compares the actual OPF solutions and the restored points. The trained weights and biases are then used during online calculations to recover the solutions. We demonstrate the proposed restoration algorithm using various convex relaxations, an approximation, and an ML-based model. The results show an improvement of several orders of magnitude in accuracy for some instances.

In summary, this paper presents an algorithm that addresses AC power flow infeasibility in simplified OPF solutions by:

- Proposing an AC feasibility restoration algorithm relevant to multiple types of simplified OPF problems (relaxations, approximations, and ML models).
- Jointly considering outputs from multiple simplified OPF problems to exploit more information from each.
- Developing an adaptive stochastic gradient descent method to determine optimal weight and bias parameters.
- Illustrating the effectiveness and scalability of the proposed algorithm through numerical experiments using various power flow relaxations, an approximation, and an ML model on multiple test cases.

We emphasize that our proposed algorithm does not simply apply least-squares methods to solve power flow problems. Rather, by automatically tuning weighting terms, our proposed approach *learns the trustworthiness of the outputs* from a particular power flow approximation, relaxation, or machine learning model for a specific system and operating range of interest. This enables both *several orders of magnitude improvements in accuracy* for the recovered solutions compared to alternate benchmarks and also provides *targeted insights into the performance* of different approximation, relaxation, and machine learning models with respect to various outputs.

The paper is structured as follows. Section II reviews the OPF problem and various simplifications. Section III presents the proposed restoration algorithm. Section IV provides numerical results to demonstrate the algorithm's performance. Section V concludes the paper and discusses potential avenues for future research. Note that a preliminary version of the proposed algorithm was published in [33] and an extended version of this paper with additional derivations is also available [34].

II. OPF Formulation and Simplifications

This section overviews the OPF problem and discusses several common relaxations and approximations as well as emerging machine learning models that simplify the OPF problem to improve tractability at the cost of accuracy. For more detail, see [13] for a survey on power flow relaxations and approximations and [19] for a survey on machine learning.

We first establish notation. The sets of buses and lines are represented by $\mathcal{N}$ and $\mathcal{E}$, respectively. Each bus $i \in \mathcal{N}$ has a voltage phasor V_i with phase angle θ_i, a complex power demand S_i^d, a shunt admittance Y_i^S, and a complex power generation S_i^g. Buses without generators are modeled as having zero generation limits. Complex power flows into each terminal of each line $(j,k) \in \mathcal{E}$ are denoted as S_{jk} and S_{kj}. Each line $(j,k) \in \mathcal{E}$ has admittance parameters Y_{jk} and Y_{kj}. The real and imaginary parts of a complex number are denoted as $\Re(\,\cdot\,)$ and $\Im(\,\cdot\,)$, respectively. The complex conjugate of a number is represented by $(\,\cdot\,)^*$ and the transpose of a matrix is represented by $(\,\cdot\,)^T$. Upper and lower bounds are denoted as $\overline{(\cdot)}$ and $\underline{(\cdot)}$, interpreted as separate bounds on real and imaginary parts for complex variables. The OPF problem is:

$$\min \sum_{i \in N} c_{2i} \left(\Re\left(S_i^g\right)\right)^2 + c_{1i} \Re\left(S_i^g\right) + c_{0i} \tag{1a}$$

$$\text{s.t.} \quad (\forall i \in \mathcal{N},\ \forall (j,k) \in \mathcal{E})$$

$$\mathbf{W}_{jk} = V_j V_k^*,\ \mathbf{W}_{kj} = V_k V_j^*,\ \mathbf{W}_{ii} = V_i V_i^* \tag{1b}$$

$$\left(\underline{V}_i\right)^2 \le \mathbf{W}_{ii} \le \left(\overline{V}_i\right)^2 \tag{1c}$$

$$\underline{S}_i^g \le S_i^g \le \overline{S}_i^g \tag{1d}$$

$$|S_{jk}| \le \overline{S}_{jk},\ |S_{kj}| \le \overline{S}_{jk} \tag{1e}$$

$$S_i^g - S_i^d - \left(Y_i^S\right)^* \mathbf{W}_{ii} = \sum_{(i,j) \in \mathcal{E}} S_{ij} + \sum_{(k,i) \in \mathcal{E}} S_{ki} \tag{1f}$$

$$S_{jk} = Y_{jk}^* \mathbf{W}_{jj} - Y_{jk}^* \mathbf{W}_{jk} \tag{1g}$$

$$S_{kj} = Y_{kj}^* \mathbf{W}_{kk} - Y_{kj}^* \mathbf{W}_{kj}^* \tag{1h}$$

$$\tan\left(-\overline{\theta}_{jk}\right) \Re\left(\mathbf{W}_{jk}\right) \le \Im\left(\mathbf{W}_{jk}\right) \le \tan\left(\overline{\theta}_{jk}\right) \Re\left(\mathbf{W}_{jk}\right). \tag{1i}$$

The OPF problem (1) minimizes an objective, in this case, the generation cost as shown by (1a). The objective has quadratic coefficients c_{2i}, c_{1i}, and c_{0i}. The voltage phasor products are collected in a Hermitian matrix $\mathbf{W}$, as described in (1b). The OPF problem also imposes voltage magnitude limits (1c), generator output limits (1d), apparent power flow limits (1e), and complex power balance at each bus (1f). Power flows for each line are defined in (1g) and (1h) and limits on phase angle differences across lines are imposed in (1i).

All the nonconvexities in the OPF problem are associated with the products in (1b), which, in combination with (1f)–(1h), form the power flow equations. Power flow relaxations convexify the power flow equations by replacing (1b) with less stringent conditions. The SDP relaxation requires that $\mathbf{W}$ is positive semidefinite [9], which is implied by (1b). The SOCP relaxation requires that $\mathbf{W}$ has non-negative principal minors [10], which is implied by positive semidefiniteness of $\mathbf{W}$. The QC relaxation strengthens the SOCP relaxation with additional variables and constraints corresponding to phase angle differences that are restricted to convex envelopes [11].

Power flow approximations replace (1b), (1f)–(1h) with alternative (usually linear) expressions relating the line flows and voltage phasors. For instance, the linear-programming approximation (LPAC) linearizes a polar representation of the power flow equations [35]. This is accomplished by linearizing sine functions using a first-order Taylor approximation around zero phase angle differences and replacing cosine functions with lifted variables restricted to a convex polytope.

Some emerging ML approaches for OPF problems replace the power flow equations (1b), (1f)–(1h) with a surrogate model. For instance, the approach in [18] replaces the power flow equations with the piecewise linear function corresponding to a trained neural network with rectified linear unit (ReLU) activation functions to obtain a mixed-integer linear programming formulation. Other ML approaches such as [14]–[17] directly estimate OPF solutions by training neural networks or other ML models to predict values for the optimal voltage phasors, power injections, and/or line flows.

Since these simplified OPF formulations do not enforce the true AC power flow equations (1b), (1f)–(1h), their outputs may have inconsistencies between the power injections, power flows, and voltage phasors. Thus, all would potentially benefit from our proposed restoration algorithm. We next present our proposed restoration algorithm considering a generic simplified OPF problem. In Section IV, we illustrate the method's performance using the SDP, SOCP, and QC relaxations, the LPAC approximation, and the ML model from [14].

III. Restoring AC Power Flow Feasibility

Solutions to any of the simplified OPF problems discussed in Section II may suffer from voltage phasors, power injections, and line flows that do not satisfy the AC power flow equations. To restore operating points that are AC power flow feasible, this section introduces a restoration algorithm inspired by state estimation techniques where the voltage phasors, power injections, and line flows from the simplified solution are analogous to noisy measurements. This algorithm improves on previous methods as it does not fix any variables to specific values, thus enabling the restoration of higher-quality solutions. Note that this algorithm *does not rely on actual measurements* from the physical system. Instead, this algorithm finds the AC power flow feasible voltage phasors that most closely match the voltage phasors, power injections, and line flows resulting from the solution to a simplified OPF problem in a similar manner by which state estimation algorithms resolve inconsistencies among noisy measurements.

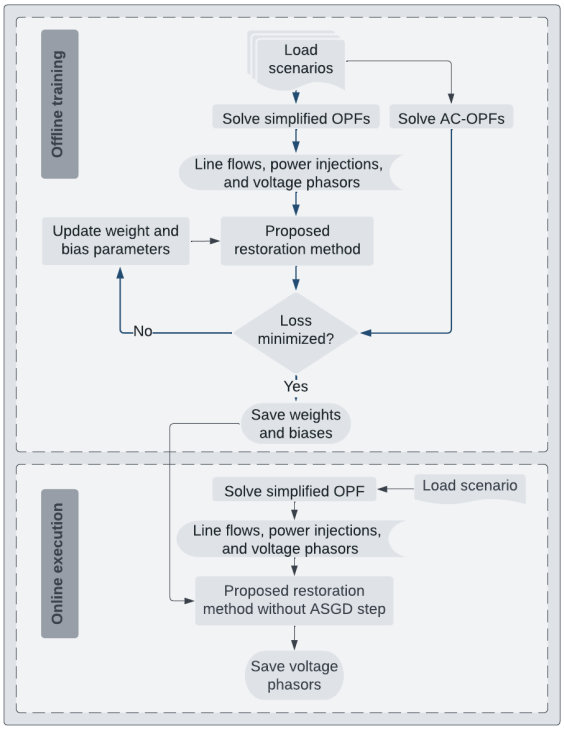


Figure 1. Flowchart of the proposed algorithm.

Analogous to how state estimation algorithms use variations in the amount of sensor noise to weight measured quantities, the proposed algorithm includes weight and bias parameters associated with the outputs of each quantity from the simplified OPF solution. However, unlike state estimation algorithms, these weight parameters are not determined by the physical characteristics of a sensor, but rather by the inconsistencies (with regard to the AC power flow equations) among various quantities in the solution to the simplified OPF problem. To determine the optimal values for these weight and bias parameters, we propose an ASGD-based method that is executed offline, with the results used online for restoring AC power flow feasibility. Fig. 1 shows both the algorithm for determining the weights and biases and the solution restoration algorithm. Furthermore, Table I summarizes the analogy between the proposed algorithm and state estimation.

A. AC Feasibility Restoration Algorithm

In this section, we introduce our proposed algorithm for restoring AC power flow feasible points from the solutions of simplified OPF problems (convex relaxations, approximations, and ML-based models). This method aims to find the voltage phasors that are close to the true OPF solution's voltage phasors based on the voltage phasors, power injections, and line flows from a simplified OPF solution.

This section introduces notation based on the typical representation of state estimation algorithms [32] to show how this mathematical machinery is leveraged in our method. We emphasize that we do not use any actual measurements from physical sensors, but rather use the information from the simplified OPF solution. The goal is to find the voltage

Table I
ANALOGY RELATING THE PROPOSED ALGORITHM AND STATE ESTIMATION

Proposed Algorithm	**State Estimation**
Solutions from relaxed, approximated, or ML-based models	Measurements from physical sensors
Inconsistencies in relaxed, approximated, or ML-based solutions	Noise from physical sensors
Weight parameters	Variance of the measurement noise
Bias parameters	—

magnitudes and angles (denoted as $\mathbf{x}$) that are most consistent with the voltage magnitudes, phase angles, power flows, and power injections from the simplified OPF solution, which we gather into a vector $\mathbf{z}$. We denote the number of these quantities, i.e., the length of $\mathbf{z}$, by m and let n be the number of voltage magnitudes plus the number of (non-slack) voltage angles, i.e., the length of $\mathbf{x}$.

We also define a length-m vector of bias parameters $\mathbf{b}$ for the simplified solution.[2] While measurement errors in state estimation are typically only characterized by their variations, the errors in simplified OPF solutions may be *biased*, i.e., consistently overestimate or underestimate the true values of some quantities. We account for this using a bias term $\mathbf{b}$ that represents the systematic errors in the simplified OPF solutions. We also have an error term $\mathbf{e}$ that captures variations that are modeled as being random. By considering both bias and error terms, our model is better equipped to handle both systematic and random deviations from the true values. We use an AC power flow model denoted as $\mathbf{h}(\mathbf{x})$ to relate $\mathbf{x}$ and $\mathbf{z}$, parameterized by the bias $\mathbf{b}$:

$$z_i + b_i = h_i(\mathbf{x}) + e_i, \quad i = 1, \ldots, m, \tag{2}$$

where $\mathbf{h}(\mathbf{x}) = [\mathbf{V}(\mathbf{x})^T\ \mathbf{P}(\mathbf{x})^T\ \mathbf{Q}(\mathbf{x})^T\ \mathbf{P}^f(\mathbf{x})^T\ \mathbf{Q}^f(\mathbf{x})^T\ \boldsymbol{\theta}(\mathbf{x})^T]^T$ denotes the AC power flow equations relating the vectors of voltage magnitudes $\mathbf{V}$, active and reactive power injections $\mathbf{P}$ and $\mathbf{Q}$, active and reactive line flows $\mathbf{P}^f$ and $\mathbf{Q}^f$, and voltage angles $\boldsymbol{\theta}$ to the vector $\mathbf{x} = [\boldsymbol{\theta}^T\ \mathbf{V}^T]^T$. The first and last entries of $\mathbf{h}(\mathbf{x})$, namely, $\mathbf{V}(\mathbf{x})$ and $\boldsymbol{\theta}(\mathbf{x})$, are obtained via the identity function. The remaining entries of the $\mathbf{h}(\mathbf{x})$ are:

$$P_i = \sum_{(i,j)\in\mathcal{E}} P_{ij}^f, \qquad Q_i = \sum_{(i,j)\in\mathcal{E}} Q_{ij}^f, \tag{3a}$$

$$\begin{aligned} P_{ij}^f =& V_i^2 \left(\Re(Y_{ij}) + \Re(Y_{ij}^{sh})\right) - V_i V_j \Re(Y_{ij}) \cos(\theta_i - \theta_j) \\ & - V_i V_j \Im(Y_{ij}) \sin(\theta_i - \theta_j), \end{aligned} \tag{3b}$$

$$\begin{aligned} Q_{ij}^f =& - V_i^2 \left(\Im(Y_{ij}) + \Im(Y_{ij}^{sh})\right) - V_i V_j \Re(Y_{ij}) \sin(\theta_i - \theta_j) \\ & + V_i V_j \Im(Y_{ij}) \cos(\theta_i - \theta_j). \end{aligned} \tag{3c}$$

Hence, the error $\mathbf{e}$ is the difference between the simplified solution (offset by the bias parameter) and the value corresponding to the restored point $\mathbf{x}$. As we will discuss below in Section III-B, the bias $\mathbf{b}$ is computed offline based on the

[2]As we will discuss in Section IV-D, we can generalize our algorithm to jointly consider solutions from multiple simplified OPF problems. In this case, the elements of $\mathbf{z}$ and $\mathbf{b}$ correspond to quantities from multiple simplified OPF problems stacked together, but the size of $\mathbf{x}$ remains the same.

characteristics of many simplified OPF solutions to reflect the systematic offsets of the simplified solutions from the true values. After determining the bias $\mathbf{b}$, the error $\mathbf{e}$ represents the remaining inconsistencies that are not accounted for by the systematic bias.

To address these remaining inconsistencies, our proposed restoration algorithm uses a weighted least squares formulation similar to typical state estimation algorithms. The goal is to choose the voltage magnitudes and angles in $\mathbf{x}$ that minimize a cost function, denoted as $J(\mathbf{x})$, that is the sum of the squared inconsistencies between the simplified OPF solution (offset by $\mathbf{b}$) and the true OPF solution, i.e., the difference between $\mathbf{h}(\mathbf{x})$ and $\mathbf{z} + \mathbf{b}$. These inconsistencies are represented by the vector $\mathbf{e}$ and are weighted by a specified diagonal matrix $\mathbf{\Sigma}$ with weight parameters associated with the measurements from the simplified OPF solution:

$$\min\ J(\mathbf{x}) = \mathbf{e}^T \mathbf{\Sigma} \mathbf{e}. \tag{4}$$

In a state estimation application, $\mathbf{\Sigma}$ would be the covariance matrix for the sensor noise. Conversely, we permit $\mathbf{\Sigma}$ to be any diagonal matrix with values computed offline using the algorithm described in Section III-B.

We solve (4) by considering the optimality conditions:

$$\mathbf{g}(\mathbf{x}) = \frac{\partial J(\mathbf{x})}{\partial \mathbf{x}} = -\mathbf{H}(\mathbf{x})^T \mathbf{\Sigma}(\mathbf{z} + \mathbf{b} - \mathbf{h}(\mathbf{x})) = \mathbf{0}, \tag{5}$$

where the Jacobian matrix $\mathbf{H}(\mathbf{x})$ of AC power flow equations $\mathbf{h}(\mathbf{x})$ is $\mathbf{H}(\mathbf{x}) = \frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}$:

$$\mathbf{H}(\mathbf{x}) = \begin{bmatrix} \mathbf{0} & \frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}} & \frac{\partial \mathbf{Q}}{\partial \boldsymbol{\theta}} & \frac{\partial \mathbf{P^f}}{\partial \boldsymbol{\theta}} & \frac{\partial \mathbf{Q^f}}{\partial \boldsymbol{\theta}} & \mathbf{1} \\ \mathbf{1} & \frac{\partial \mathbf{P}}{\partial \mathbf{V}} & \frac{\partial \mathbf{Q}}{\partial \mathbf{V}} & \frac{\partial \mathbf{P^f}}{\partial \mathbf{V}} & \frac{\partial \mathbf{Q^f}}{\partial \mathbf{V}} & \mathbf{0} \end{bmatrix}^T. \tag{6}$$

To compute the solution to (5), we apply the Newton-Raphson method described in Algorithm 1 that solves, at the k-th iteration, the following linear system:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - (\mathbf{G}(\mathbf{x}))^{-1} \mathbf{g}(\mathbf{x}), \tag{7}$$

where

$$\mathbf{G}(\mathbf{x}) = \frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}} = \mathbf{H}(\mathbf{x})^T \mathbf{\Sigma} \mathbf{H}(\mathbf{x}) \tag{8}$$

Algorithm 1 uses a convergence tolerance of ϵ and takes a user-specified initialization of $\mathbf{x}^{(0)}$. When available, the voltage magnitudes and angles from the simplified OPF solution often provide reasonable initializations $\mathbf{x}^{(0)}$. Otherwise, a flat start provides an alternative initialization. The output of this algorithm is the restored solution, denoted as $\mathbf{x}_R$.

B. Determining the Weight Parameters

The weight parameters $\mathbf{\Sigma}$ play a crucial role in determining the accuracy of the solution obtained from the Algorithm 1. Ideally, larger values of $\mathbf{\Sigma}_{ii}$ should be chosen for the quantities $\mathbf{z}_i$ from the simplified OPF solution that more closely represent the solution to the true OPF problem. However, it is not straightforward to predict or estimate the accuracy of a particular $\mathbf{z}_i$ in relation to the true OPF solution. Choosing values for the bias parameters $\mathbf{b}$ poses similar challenges. As shown in Fig. 1, we therefore develop an approach inspired by the training of machine learning models to determine these parameters. This approach involves solving a set of randomly generated OPF problems along with the corresponding simplified OPF problems to create a training dataset. As presented in Algorithm 2, we then employ an ASGD method that iteratively solves the proposed restoration algorithm and updates the weight parameters in a way that minimizes the difference between the restored solution and the true OPF solution across the training dataset. In this offline training, solutions to each of the true and simplified OPF problems are computed in parallel.

Algorithm 1: Newton-Raphson Algorithm for AC Power Flow Feasibility Restoration

Input : Simplified OPF solution $\mathbf{z}$, Initialization $\mathbf{x}^{(0)}$, Parameters $\mathbf{\Sigma}$, $\mathbf{b}$, ϵ
Output: Restored AC power flow feasible solution $\mathbf{x}_R$

Initialize $k \leftarrow 0$
while $||\mathbf{\Delta x}^{(k)}|| > \epsilon$ **do**
  $k \leftarrow k + 1$;
  Calculate $\mathbf{h}(\mathbf{x}^{(k)})$, $\mathbf{H}(\mathbf{x}^{(k)})$, $\mathbf{G}(\mathbf{x}^{(k)})$, *and* $\mathbf{g}(\mathbf{x}^{(k)})$ *using* $\mathbf{\Sigma}$, $\mathbf{z}$, $\mathbf{b}$, *and* $\mathbf{x}^{(k)}$;
  $\mathbf{\Delta x}^{(k)} = \mathbf{G}(\mathbf{x}^{(k)})^{-1} \mathbf{H}(\mathbf{x}^{(k)})^T \mathbf{\Sigma} \left(\mathbf{z} + \mathbf{b} - \mathbf{h}(\mathbf{x}^{(k)})\right)$;
  $\mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)} + \mathbf{\Delta x}^{(k)}$;
  $\mathbf{x}_R = \mathbf{x}^{(k+1)}$

The ASGD method relies on the sensitivities of the restored point $\mathbf{x}_R$ with respect to the parameters $\mathbf{\Sigma}$, i.e., $\frac{\partial \mathbf{x}_R}{\partial \mathbf{\Sigma}}$:

$$\frac{\text{vec}(\partial \mathbf{x}_R)}{\text{vec}(\partial \mathbf{\Sigma})} = \Bigg((\mathbf{z} + \mathbf{b} - \mathbf{h}(\mathbf{x})) - \Big(\mathbf{H}(\mathbf{x}) (\mathbf{H}(\mathbf{x})^T \mathbf{\Sigma} \mathbf{H}(\mathbf{x}))^{-1} \times \mathbf{H}(\mathbf{x})^T \mathbf{\Sigma} (\mathbf{z} + \mathbf{b} - \mathbf{h}(\mathbf{x})) \Big) \Bigg) \otimes \Big((\mathbf{H}(\mathbf{x})^T \mathbf{\Sigma} \mathbf{H}(\mathbf{x}))^{-1} \mathbf{H}(\mathbf{x})^T \Big)^T. \tag{9}$$

The expression (9) gives the sensitivities of the restored point $\mathbf{x}_R$ with respect to the weight parameters $\mathbf{\Sigma}$, where $\otimes$ denotes the Kronecker product and vec($\cdot$) denotes the vectorization of a matrix. With length-m vectors $\mathbf{z}$ and $\mathbf{h}(\mathbf{x})$ and an $m \times n$ matrix $\mathbf{H}$, the sensitivities $\frac{\partial \mathbf{x}_R}{\partial \mathbf{\Sigma}}$ are represented by a $n \times m^2$ matrix. Appendix A provides the derivation of (9).

Note that directly applying (9) can be computationally expensive for large-scale systems since this expression computes sensitivities with respect to all entries of $\mathbf{\Sigma}$. While possibly relevant for variants of the proposed formulation, sensitivities for the off-diagonal terms in $\mathbf{\Sigma}$ are irrelevant in our formulation since these terms are fixed to zero due to the diagonal structure of $\mathbf{\Sigma}$. Exploiting this structure can enable faster computations; see Appendix B for further details.

C. Determining the Bias Parameters

Like the weight parameters $\mathbf{\Sigma}$, the bias parameters $\mathbf{b}$ significantly impact the accuracy of Algorithm 1's outputs. Similar to the weight parameters $\mathbf{\Sigma}$, Algorithm 2 also computes the bias parameters $\mathbf{b}$ via an ASGD method. The sensitivities of the restored point $\mathbf{x}_R$ with respect to the bias parameters $\mathbf{b}$ are:

$$\frac{\partial \mathbf{x}_R}{\partial \mathbf{b}} = \Big(\mathbf{H}(\mathbf{x})^T \mathbf{\Sigma} \mathbf{H}(\mathbf{x}) \Big)^{-1} \mathbf{H}(\mathbf{x})^T \mathbf{\Sigma}. \tag{10}$$

D. Loss Function

To evaluate the accuracy of the restored solution obtained from our proposed algorithm, we need to define a quantitative measure, i.e., a *loss function*, that compares the restored solution to the true solution of the OPF problem. There are several possible ways to define a loss function in this context, such as comparing the voltage magnitudes, phase angles, power injections, line flows, etc. from the restored solution to those from the true OPF solution.

Following typical approaches for training ML models, we formulate a loss function as the squared difference between the voltage magnitudes and angles from the restored solution $\mathbf{x}_R$ and the true solution $\mathbf{x}_{AC}$. To achieve this, we introduce new vectors $\mathbf{X}_R = [{\mathbf{x}_R^{(1)}}^T, {\mathbf{x}_R^{(2)}}^T, \ldots, {\mathbf{x}_R^{(S)}}^T]^T$ and $\mathbf{X}_{AC} = [{\mathbf{x}_{AC}^{(1)}}^T, {\mathbf{x}_{AC}^{(2)}}^T, \ldots, {\mathbf{x}_{AC}^{(S)}}^T]^T$, where $\mathbf{x}_R^{(i)}$ and $\mathbf{x}_{AC}^{(i)}$ denote the vectors of voltage magnitudes and angles at each bus for the restored and actual OPF solutions of the i-th sampled load scenario and S represents the number of scenarios. Consequently, we define the loss function across samples as:

$$F(\mathbf{\Sigma}, \mathbf{b}) = \frac{1}{n} \|\mathbf{X}_R(\mathbf{\Sigma}, \mathbf{b}) - \mathbf{X}_{AC}\|_2^2, \tag{11}$$

where $\frac{1}{n}$ normalizes this function based on the system size.

E. Adaptive Stochastic Gradient Descent (ASGD) Algorithm

The optimal weight and bias parameters, $\mathbf{\Sigma}$ and $\mathbf{b}$, are obtained using the ASGD method described in Algorithm 2. After the offline execution of Algorithm 2, the resulting weights are applied online to restore AC power flow feasibility for a particular problem via Algorithm 1 (see Fig. 1.)

To compute optimal weight and bias parameters, Algorithm 2 first creates a set of sampled load scenarios representing the range of conditions expected during real-time operations. Next, the algorithm solves (in parallel) both the actual and simplified OPF problems and saves the results in $\mathbf{x}_{AC}^{(i)}$ and $\mathbf{z}^{(i)}$, respectively, for each load scenario $i = 1, \ldots, S$. Using the information from the simplified solutions and Algorithm 1, the algorithm computes the restored solutions $\mathbf{x}_R^{(i)}$ for each load scenario $i = 1, \ldots, S$. The algorithm then iteratively updates the weight and bias parameters based on the discrepancies between the actual and restored solutions along with their respective partial derivatives in order to minimize the loss function. The optimal weight and bias parameters, $\mathbf{\Sigma}^{opt}$ and $\mathbf{b}^{opt}$, are returned as outputs after reaching a maximum number of iterations or satisfying some other termination criteria (e.g., negligible changes from one iteration to the next).

The ASDG algorithm uses the gradient of the loss function with respect to the weight parameters, denoted as $\mathbf{q}^{var}$:

$$\mathbf{q}^{var} = \frac{2}{n} \sum_{i=1}^{S} \left. \frac{\partial \mathbf{x}_R}{\partial \mathbf{\Sigma}} \right|_{\mathbf{x}_R^{(i)}} \left(\mathbf{x}_R^{(i)}(\mathbf{\Sigma}, \mathbf{b}) - \mathbf{x}_{AC}^{(i)} \right). \tag{12}$$

There are many variants of gradient descent algorithms, such as batch gradient, momentum, AdaGrad, Adam, etc., each of which has their own advantages and disadvantages. We use the Adam algorithm since we empirically found it to perform best for this application [33]. The Adam algorithm is commonly used for training machine learning models and involves the following steps at each iteration [36]:

$$\mathbf{m} \leftarrow \beta_1 \mathbf{m} + (1 - \beta_1)\mathbf{q}, \tag{13a}$$

$$\boldsymbol{\tau} \leftarrow \beta_2 \boldsymbol{\tau} + (1 - \beta_2)(\mathbf{q})^2, \tag{13b}$$

$$\hat{\mathbf{m}} \leftarrow \frac{\mathbf{m}}{1 - \beta_1^k}, \tag{13c}$$

$$\mathbf{\Gamma} \leftarrow \frac{\boldsymbol{\tau}}{1 - \beta_2^k}, \tag{13d}$$

$$\mathbf{\Sigma} \leftarrow \mathbf{\Sigma} - \eta \frac{\hat{\mathbf{m}}}{\sqrt{\mathbf{\Gamma}} + \epsilon}, \tag{13e}$$

where $\mathbf{m}$ and $\boldsymbol{\tau}$ are the first and second moments of the gradients at iteration k, η is a learning rate (step size), β_1 and β_2 are exponentially decaying hyperparameters for the first and second moments, and ϵ is a small constant.

In addition, the gradient of the objective function with respect to the bias parameters is represented by $\mathbf{q}^{bias}$:

$$\mathbf{q}^{bias} = \frac{2}{n} \sum_{i=1}^{S} \left. \frac{\partial \mathbf{x}_R^{(i)}}{\partial \mathbf{b}} \right|_{\mathbf{x}_R^{(i)}} \left(\mathbf{x}_R^{(i)}(\mathbf{\Sigma}, \mathbf{b}) - \mathbf{x}_{AC}^{(i)} \right). \tag{14}$$

Using this gradient, one can find the optimal bias parameters $\mathbf{b}$ using the Adam algorithm in the same fashion as in (13).

IV. Experimental Results and Discussion

This section evaluates the proposed algorithm's performance using numerical results from restoring AC power flow feasibility for solutions obtained from the SOCP [10], QC [11], and SDP [9] relaxations, the LPAC approximation [35], and the ML-based OPF model from [14].

A. Experiment Setup

We generated 10,000 scenarios (8,000 for training and 2,000 for testing) for each of the `PJM 5-bus`, `IEEE 14-bus`, `IEEE 57-bus`, `IEEE 118-bus`, `Illinois 200-Bus` [37], `IEEE 300-bus`, and `Pegase 1354-bus` systems [38]. These scenarios were created by multiplying the nominal load demands by a normally distributed random variable with zero mean and standard deviation of 10%. We set the convergence tolerance ϵ to 10^{-6} in Algorithm 1. Solutions to the OPF problems and the relaxations and approximations were computed using `PowerModels.jl` [39] with the solvers Ipopt [40] and Mosek on a computing node of the Partnership for an Advanced Computing Environment (PACE) cluster at Georgia Tech. This computing node has a 24-core CPU and 32 GB of RAM. We also imported the ML results for available test cases from [14]. The restoration algorithm was implemented in Python 3.0 using a Jupyter Notebook. The solution times of the AC-OPF model for given test cases are reported in Table II using `PowerModels.jl`.

B. Benchmarking Approach

We consider three alternate restoration methods as comparisons to our proposed algorithm. The first method simply compares the voltage magnitudes and angles from the relaxed,

Algorithm 2: Computing Weight and Bias Parameters

Input: $\eta, \epsilon, \beta_1, \beta_2, \mathbf{m}, \boldsymbol{\tau}$, *batch size*, *max_iter*: Adaptive stochastic gradient descent parameters
$\boldsymbol{\Sigma}^{init}$: Initial weight parameters
$\mathbf{b}^{init}$: Initial bias parameters
Output: $\boldsymbol{\Sigma}^{opt}$: Optimal weight parameters
$\mathbf{b}^{opt}$: Optimal bias parameters
Data: OPF problem data

Generate load scenarios s_i for $i = 1, \ldots, S$
Solve OPF problem (1) for each scenario s_i, $i = 1, \ldots, S$, and store the results in $\mathbf{x}_{AC}^{(i)}$
Solve simplified OPF problems for each scenario s_i, $i = 1, \ldots, S$, and store the results in $\mathbf{z}^{(i)}$
$\boldsymbol{\Sigma} \leftarrow \boldsymbol{\Sigma}^{init}$, $\mathbf{b} \leftarrow \mathbf{b}^{init}$
$\mathbf{m}^{var} \leftarrow \mathbf{m}$, $\mathbf{m}^{bias} \leftarrow \mathbf{m}$, $\boldsymbol{\tau}^{var} \leftarrow \boldsymbol{\tau}$, $\boldsymbol{\tau}^{bias} \leftarrow \boldsymbol{\tau}$, $k \leftarrow 1$
while $k \leq max\_iter$ **do**
  $\mathbf{X}_R \leftarrow [\,]$, $\mathbf{X}_{AC} \leftarrow [\,]$, $\mathbf{q}^{var} \leftarrow [\,]$, $\mathbf{q}^{bias} \leftarrow [\,]$
  for $i \in$ *random.sample*$(\{1, \ldots, S\})$, *batch size*) **(in parallel) do**
    Run the restoration method in Algorithm 1 for scenario s_i with simplified OPF solution $\mathbf{z}^{(i)}$, weights $\boldsymbol{\Sigma}$, and bias $\mathbf{b}$ to an accuracy of ε and store the solution in $\mathbf{x}_R^{(i)}$
    $\mathbf{q}^{var} \leftarrow \mathbf{q}^{var} + \frac{2}{n} \left.\frac{\partial \mathbf{x}_R}{\partial \boldsymbol{\Sigma}}\right|_{\mathbf{x}_R^{(i)}} (\mathbf{x}_R^{(i)} - \mathbf{x}_{AC}^{(i)})$
    $\mathbf{q}^{bias} \leftarrow \mathbf{q}^{bias} + \frac{2}{n} \left.\frac{\partial \mathbf{x}_R}{\partial \mathbf{b}}\right|_{\mathbf{x}_R^{(i)}} (\mathbf{x}_R^{(i)} - \mathbf{x}_{AC}^{(i)})$
    $\mathbf{X}_{AC} \leftarrow \text{append}(\mathbf{X}_{AC}, \mathbf{x}_{AC}^{(i)})$
    $\mathbf{X}_R \leftarrow \text{append}(\mathbf{X}_R, \mathbf{x}_R^{(i)})$
  $\mathbf{m}^{var} \leftarrow \beta_1 \mathbf{m}^{var} + (1 - \beta_1)\mathbf{q}^{var}$
  $\boldsymbol{\tau}^{var} \leftarrow \beta_2 \boldsymbol{\tau}^{var} + (1 - \beta_2)\mathbf{q}^{var}$
  $\hat{\mathbf{m}}^{var} \leftarrow \frac{\mathbf{m}^{var}}{1-\beta_1^k}$
  $\boldsymbol{\Gamma}^{var} \leftarrow \frac{\boldsymbol{\tau}^{var}}{1-\beta_2^k}$
  $\boldsymbol{\Sigma} \leftarrow \boldsymbol{\Sigma} - \eta \frac{\hat{\mathbf{m}}^{var}}{\sqrt{\boldsymbol{\Gamma}^{var} + \epsilon}}$
  } Adam method for weights $\boldsymbol{\Sigma}$
  $\mathbf{m}^{bias} \leftarrow \beta_1 \mathbf{m}^{bias} + (1 - \beta_1)\mathbf{q}^{bias}$
  $\boldsymbol{\tau}^{bias} \leftarrow \beta_2 \boldsymbol{\tau}^{bias} + (1 - \beta_2)(\mathbf{q}^{bias})^2$
  $\hat{\mathbf{m}}^{bias} \leftarrow \frac{\mathbf{m}^{bias}}{1-\beta_1^k}$
  $\boldsymbol{\Gamma}^{bias} \leftarrow \frac{\boldsymbol{\tau}^{bias}}{1-\beta_2^k}$
  $\mathbf{b} \leftarrow \mathbf{b} - \eta \frac{\hat{\mathbf{m}}^{bias}}{\sqrt{\boldsymbol{\Gamma}^{bias} + \epsilon}}$
  } Adam method for biases $\mathbf{b}$
  $k \leftarrow k + 1$
$\boldsymbol{\Sigma}^{opt} \leftarrow \boldsymbol{\Sigma}$, $\mathbf{b}^{opt} \leftarrow \mathbf{b}$

Table II
AVERAGE EXECUTION TIME PER SCENARIO FOR AC-OPF

Test Case	5	14	57	118	200	300	1354
Time (s)	0.113	0.127	0.174	0.403	0.449	0.929	6.654

approximated, or ML-based solution directly to the OPF solution. However, it is important to note that this method typically does not yield an AC power flow feasible point and is thus unsuitable for many practical applications. Additionally, this method is not applicable to the SDP and SOCP relaxations as they do not have variables corresponding to the voltage phase angles. The second method, referred to as the "benchmark" method, solves the power flow problem obtained from fixing the voltage magnitudes at all generator buses and the active power injections at non-slack generator buses to the outputs of the simplified OPF problem as discussed in [26] and used in a variety of papers such as [15], [17], [26], [29]–[31]. The third method is the proposed restoration algorithm with the initial weight and bias parameters $\boldsymbol{\Sigma}_{ii} = 1$ and $\mathbf{b}_i = 0$, $i = 1, \ldots, m$. The fourth method is the proposed restoration algorithm with weight and bias parameters computed using Algorithm 2.

C. Performance Evaluation

Fig. 2 shows the weight parameters (i.e., the diagonal elements of $\boldsymbol{\Sigma}$) obtained by applying Algorithm 2 to the 5-bus system with the SOCP, QC, and SDP relaxations as well as the LPAC approximation. Observe that certain quantities receive significantly higher weights than others. For instance, in this test case, the algorithm allocates more weight to voltage magnitudes at buses 1 and 5, suggesting that these quantities are superior predictors of the actual OPF solutions. Larger weights imply that the algorithm considers these quantities more reliable when reconstructing the AC feasible points.

Moreover, Fig. 3 shows a geographic representation of the weight parameters $\boldsymbol{\Sigma}$ for the voltage magnitudes in the Illinois 200-bus system. The SOCP, QC, and SDP relaxations and the LPAC approximation each assign different weights to various parts of the system, with some clustering evident. Additionally, the QC relaxation has larger weights on the voltage magnitudes overall compared to the other OPF simplifications. These distinct weight assignments will be leveraged later in Section IV-D to combine multiple simplified OPF solutions for improved accuracy and performance, as our proposed algorithm can exploit the strengths of each method while compensating for individual inaccuracies.

We evaluate the efficacy of the suggested restoration algorithm using the test dataset of 2,000 scenarios that were not used during the calculation of weight and bias parameters in Algorithm 2. Table III presents the loss function values for each solution recovery method. As shown in Table III, the proposed restoration algorithm successfully produces high-quality AC power flow feasible points from simplified OPF solutions. The loss functions resulting from the proposed algorithm are considerably smaller than those of other methods, including the benchmark approach. Furthermore, the application of optimized weight and bias parameters substantially enhances the performance of the loss function compared to using the initial weight and bias parameters $\boldsymbol{\Sigma}_{ii} = 1$ and $\mathbf{b}_i = 0$. Note that incorporating bias parameters into the updated algorithm further improves our initial findings presented in [33].

Additionally, we compare the restoration methods by analyzing the difference in density distributions, which represent how frequently a particular voltage magnitude or angle value is observed in a restored solution relative to how often it is observed in the true OPF solution for the 5-bus system, considering all $2,000$ samples in the test dataset. Fig. 4 demonstrates the performance of our proposed algorithm versus the benchmark, highlighting the superiority of the proposed algorithm when solving SDP relaxations of OPF problems. The figure has two subplots: (a) for voltage magnitudes and (b) for voltage angles. The vertical axes represent the difference in density relative to the true OPF solution, with positive values indicating higher density and negative values indicating lower density. Good performance is indicated by a line that is nearly

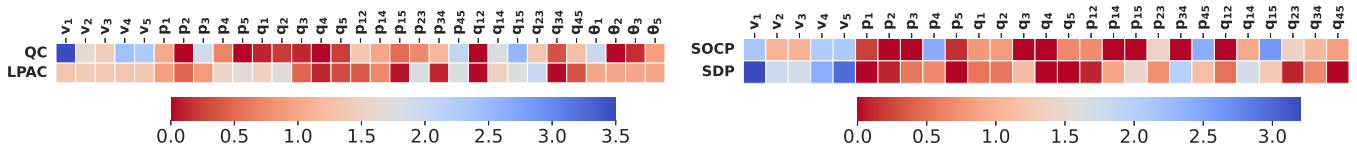
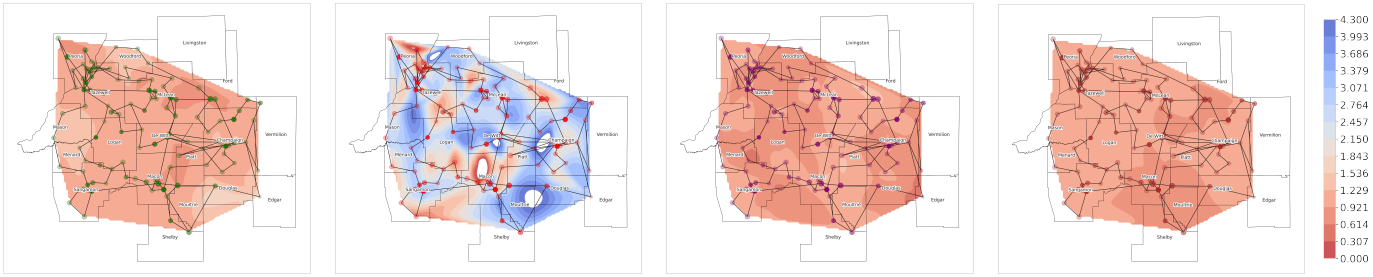


Figure 2. Diagonal elements of the weight parameters Σ for the 5-bus system as obtained using Algorithm 2. Higher values (in blue) indicate more trustworthy quantities for reconstructing AC feasible points. (a) QC and LPAC (left). (b) SOCP and SDP (right).



(a) SOCP (b) QC (c) SDP (d) LPAC

Figure 3. Contour plot of weight parameters for voltage magnitudes in the Illinois 200-bus system for SOCP, QC, SDP, and LPAC.

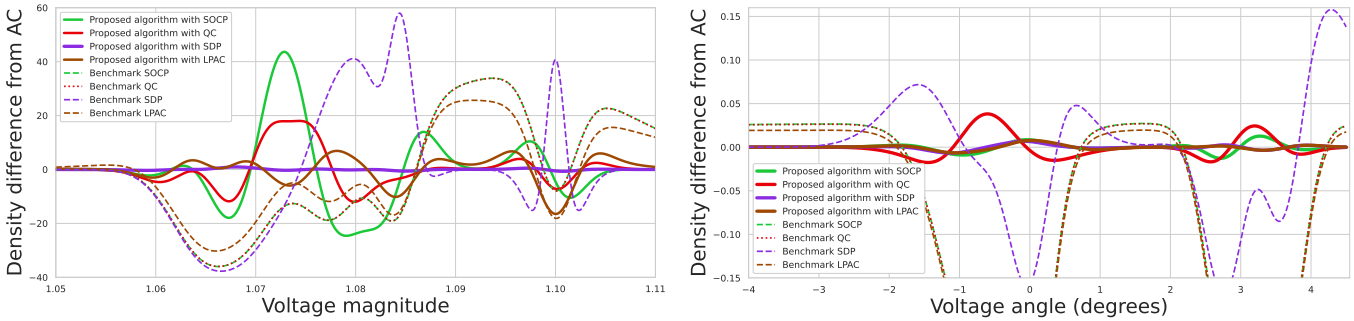


Figure 4. Differences in the density distributions for the voltage magnitudes and angles for the restored points relative to the true OPF solutions. The vertical axis represents the difference in density from the true OPF solution, with positive values indicating higher density and negative values indicating lower density compared to true OPF solution for the 5-bus system over the test set of 2000 scenarios. (a) Voltage magnitudes. (b) Voltage angles (in degrees).

horizontal at zero, which suggests that the restoration method accurately represents the true OPF solution across all voltage magnitudes and angles over $2,000$ test samples. This plot is useful for evaluating the overall performance of the restoration method, complementing the aggregate metrics in Table III by assessing performance across voltage magnitudes and angles. We observe that the restored solutions to SDP relaxations (denoted by the solid purple line) outperform all other methods, including SOCP, QC, and LPAC, for this problem. Moreover, the proposed algorithm with SOCP, QC, SDP, and LPAC exhibits better performance than their respective benchmark counterparts, as indicated by the smaller differences relative to the true OPF solution.

D. Combining Solutions

The results presented above show that the restoration algorithm's outcomes vary with the choice of relaxation, approximation, or ML model, with none consistently dominating the others in every aspect. This suggests that there may be advantages in simultaneously considering *multiple* simplified OPF solutions by combining their outputs using our proposed algorithm. With the flexibility to individually assign weights and biases for each quantity in a merged set of simplified OPF solutions, our proposed algorithm can naturally exploit the most accurate aspects of each simplified OPF solution while counterbalancing their individual inaccuracies. In other words, merging solutions from multiple simplified OPF problems supplies additional information for reconstructing even higher quality solutions via our proposed algorithm. For example, if we simultaneously use the results of the SOCP, QC, and SDP relaxations as well as the LPAC approximation, Algorithm 2 will consider $\mathbf{z} = [\mathbf{V}_{SOCP}^T \mathbf{V}_{QC}^T \mathbf{V}_{SDP}^T \mathbf{V}_{LPAC}^T \mathbf{P}_{SOCP}^T \mathbf{P}_{QC}^T \mathbf{P}_{SDP}^T \mathbf{P}_{LPAC}^T \mathbf{Q}_{SOCP}^T \mathbf{Q}_{QC}^T \mathbf{Q}_{SDP}^T \mathbf{Q}_{LPAC}^T \mathbf{P}^{f\,T}_{SOCP} \mathbf{P}^{f\,T}_{QC} \mathbf{P}^{f\,T}_{SDP} \mathbf{P}^{f\,T}_{LPAC} \mathbf{Q}^{f\,T}_{SOCP} \mathbf{Q}^{f\,T}_{QC} \mathbf{Q}^{f\,T}_{SDP} \mathbf{Q}^{f\,T}_{LPAC} \boldsymbol{\theta}_{QC}^T \boldsymbol{\theta}_{LPAC}^T]^T$. Algorithm 2 automatically identifies the accuracy for each quantity in all simplified OPF solutions by suitably assigning the corresponding $\boldsymbol{\Sigma}_{ii}$ and $\mathbf{b}_i$ values to optimally utilize all available information, thus enabling recovery of high-quality solutions even when individual simplified OPF solutions might falter. For instance, the loss function for the 5-bus system with all simplified OPF solutions improves by an order of magnitude relative to the restoration achievable with the best individual solution (1×10^{-5} for the merged solutions versus 1×10^{-4} for the solution restored from the SDP solution alone).

Fig. 5 compares the absolute errors in voltage magnitudes and angles, expressed as $|\mathbf{X}_{AC} - \mathbf{X}_R|$, for the 5-bus system over $2,000$ test samples. The horizontal axis denotes the

Table III
LOSS FUNCTION EVALUATED USING TEST DATASET (2,000 SCENARIOS) FOR DIFFERENT TEST CASES USING VARIOUS SIMPLIFIED OPF PROBLEMS

Test Case	$\|\mathcal{N}\|$	$\|\mathcal{L}\|$	$\|\mathcal{G}\|$	$\|\mathcal{E}\|$	Method	SOCP	QC	SDP	LPAC	ML [14]
PJM 5-Bus	5	5	3	6	Initial solution	—	0.6709	—	0.4996	—
					Benchmark	0.6077	0.6069	0.1279	0.4748	—
					SE with $\mathbf{\Sigma}^{init}$	0.2355	0.2886	1.0840	0.2697	—
					SE with $\mathbf{\Sigma}^{opt}$	**0.0055**	**0.0041**	**0.0001**	**0.0033**	—
IEEE 14-Bus	14	11	5	20	Initial solution	—	3.7926	—	0.5655	—
					Benchmark	0.0010	0.0009	**0.000003**	0.1937	—
					SE with $\mathbf{\Sigma}^{init}$	0.2457	0.2284	0.2540	0.6110	—
					SE with $\mathbf{\Sigma}^{opt}$	**0.0002**	**0.00008**	0.00007	**0.0012**	—
IEEE 57-Bus	57	42	7	80	Initial solution	—	1.8558	—	0.5538	—
					Benchmark	0.0566	0.0567	0.0463	0.7968	—
					SE with $\mathbf{\Sigma}^{init}$	1.4155	1.3544	1.0713	2.4205	—
					SE with $\mathbf{\Sigma}^{opt}$	**0.0099**	**0.0097**	**0.0091**	**0.0214**	—
IEEE 118-Bus	118	99	54	186	Initial solution	—	6.1651	—	4.8066	—
					Benchmark	0.2056	0.2051	0.0113	5.0810	—
					SE with $\mathbf{\Sigma}^{init}$	7.3201	5.2822	6.8255	4.5119	—
					SE with $\mathbf{\Sigma}^{opt}$	**0.0116**	**0.0106**	**0.0078**	**0.0910**	—
Illinois 200-Bus	200	108	49	245	Initial solution	—	1.5836	—	10.8926	—
					Benchmark	0.1455	0.1492	0.1461	12.0743	—
					SE with $\mathbf{\Sigma}^{init}$	0.0024	1.3533	0.0020	10.9523	—
					SE with $\mathbf{\Sigma}^{opt}$	**0.0001**	**0.0004**	**0.0004**	**0.0053**	—
IEEE 300-Bus	300	201	69	411	Initial solution	—	4.7658	—	10.5747	0.5284
					Benchmark	10.5438	8.8866	37.7240	19.9149	6.9820
					SE with $\mathbf{\Sigma}^{init}$	35.7829	19.3754	31.9334	10.2737	0.7901
					SE with $\mathbf{\Sigma}^{opt}$	**0.1891**	**0.1602**	**0.8809**	**0.9242**	**0.1702**
Pegase 1354-Bus	1354	673	260	1991	Initial solution	—	10.7125	—	10.3606	0.0568
					Benchmark	3.7920	3.7415	58.4361	20.2537	0.1561
					SE with $\mathbf{\Sigma}^{init}$	9.2313	8.9304	16.3797	22.3679	0.0502
					SE with $\mathbf{\Sigma}^{opt}$	**0.3602**	**0.3442**	**0.9623**	**1.9174**	**0.0291**

Note that the loss function values in this table correspond to the error of the restored solution with respect to the true OPF solution as defined in (11), *not* the convergence tolerance. All results (except for the initial solution) satisfy AC power flow feasibility to a tolerance $\epsilon = 10^{-6}$.

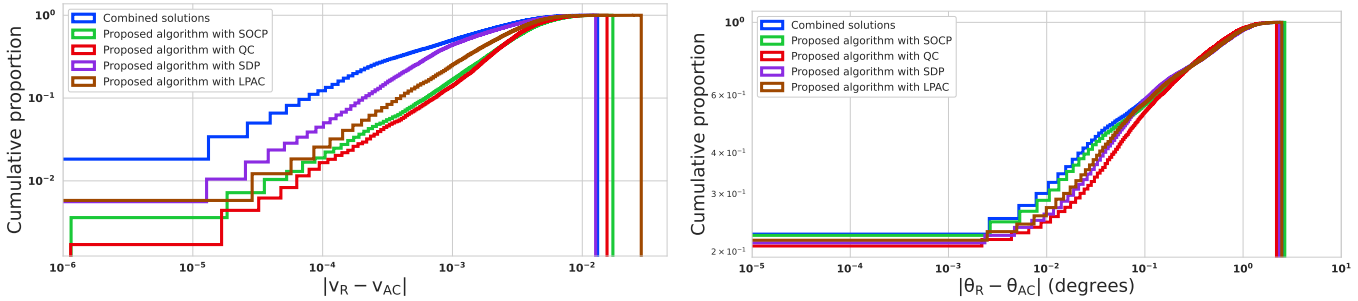


Figure 5. Cumulative proportion of the absolute error in the voltage magnitudes and angles for various restoration methods for the 5-bus system over 2,000 test scenarios. The vertical axis displays the cumulative proportion of the absolute error and the horizontal axis shows differing levels of the absolute error. Both the horizontal and vertical axes are on logarithmic scales. Each curve shows the cumulative proportion of errors up to a certain level, with higher curves thus indicating a larger proportion of smaller errors (i.e., better performance). (a) Voltage magnitudes. (b) Voltage angles (in degrees).

absolute error and the vertical axis represents the cumulative proportion of errors less than or equal to the respective value on the horizontal axis. Ideal performance, i.e., a high proportion of small errors, would correspond to a curve in the upper-left corner. With the steepest rise in Fig. 5, using multiple relaxations and approximations consistently leads to superior performance relative to using any single relaxation or approximation.

Table IV gives online execution per-scenario run time with optimized weights and biases. The online execution is comparable to a power flow solve. For instance, `PowerModels.jl` performs the power flow calculations used by the benchmark restoration method in an average of 1.18 seconds for each scenario with the Pegase 1354-bus system, which is similar to the 1.02 to 1.21 seconds for our online execution.

Table IV
AVERAGE EXECUTION TIME PER SCENARIO (SECONDS)

Test Case	SOCP	QC	SDP	LPAC
PJM 5-Bus	0.001	0.002	0.001	0.002
IEEE 14-Bus	0.011	0.008	0.013	0.009
IEEE 57-Bus	0.022	0.019	0.023	0.021
IEEE 118-Bus	0.151	0.078	0.158	0.081
Illinois 200-Bus	0.230	0.119	0.243	0.154
IEEE 300-Bus	0.310	0.191	0.322	0.217
Pegase 1354-Bus	1.132	1.093	1.214	1.021

V. CONCLUSION

In this paper, we present a solution restoration algorithm that significantly improves AC power flow feasibility restoration

from the solutions of simplified (relaxed, approximated, and ML-based) OPF problems. Our algorithm, which is based on state estimation techniques with the adjustment of weight and bias parameters through an ASGD method, can be several orders of magnitude more accurate than alternate methods.

In future research, we plan to utilize the insights gained from the trained weights and biases to improve the accuracy of power flow relaxations, approximations, and machine learning models. We also intend to incorporate the proposed restoration process into the training of machine learning models, thus closing the loop between training ML models and restoring AC feasible solutions. Additionally, future research will aim to devise related self-supervised restoration techniques that do not depend on the availability of accurate OPF solutions. Instead, these techniques would instead compute the sensitivities of a solution quality metric with respect to the weight and bias parameters without requiring an optimal solution. The ability to restore high-quality AC power flow feasible operating points without the need for true OPF solutions would further increase the range of practical applications for the proposed algorithm.

Acknowledgement

The authors thank M. Klamkin and P. Van Hentenryck for sharing the outputs of the ML models in [14].

References

[1] M. B. Cain, R. P. O'Neill, and A. Castillo, "History of optimal power flow and formulations (OPF Paper 1)," *Federal Energy Regulatory Commission*, Dec. 2012.

[2] D. Bienstock and A. Verma, "Strong NP-hardness of AC power flows feasibility," *Oper. Res. Lett.*, vol. 47, no. 6, pp. 494–501, 2019.

[3] J. Carpentier, "Contribution to the economic dispatch problem," *Bull. Soc. Franc. Elect.*, vol. 3, no. 8, pp. 431–447, 1962.

[4] J. Momoh, R. Adapa, and M. El-Hawary, "A review of selected optimal power flow literature to 1993. I. Nonlinear and quadratic programming approaches," *IEEE Trans. Power Syst.*, vol. 14, no. 1, pp. 96–104, 1999.

[5] J. Momoh, M. El-Hawary, and R. Adapa, "A review of selected optimal power flow literature to 1993. II. Newton, linear programming and interior point methods," *IEEE Trans. Power Syst.*, vol. 14, no. 1, pp. 105–111, 1999.

[6] I. A. Hiskens and R. J. Davy, "Exploring the power flow solution space boundary," *IEEE Trans. Power Syst.*, vol. 16, no. 3, pp. 389–395, Aug. 2001.

[7] C. Barrows, S. Blumsack, and P. Hines, "Correcting optimal transmission switching for AC power flows," in *47th Hawaii Int. Conf. Syst. Sci.*, Jan. 2014, pp. 2374–2379.

[8] L. A. Roald, D. Pozo, A. Papavasiliou, D. K. Molzahn, J. Kazempour, and A. Conejo, "Power systems optimization under uncertainty: A review of methods and applications," *Elect. Power Syst. Res.*, vol. 214, p. 108725, 2023, presented at the *22nd Power Syst. Comput. Conf. (PSCC)*.

[9] J. Lavaei and S. H. Low, "Zero duality gap in optimal power flow problem," *IEEE Trans. Power Syst.*, vol. 27, no. 1, pp. 92–107, 2011.

[10] R. A. Jabr, "Radial distribution load flow using conic programming," *IEEE Trans. Power Syst.*, vol. 21, no. 3, pp. 1458–1459, 2006.

[11] C. Coffrin, H. L. Hijazi, and P. Van Hentenryck, "The QC relaxation: A theoretical and computational study on optimal power flow," *IEEE Trans. Power Syst.*, vol. 31, no. 4, pp. 3008–3018, 2015.

[12] C. Coffrin and P. Van Hentenryck, "A linear-programming approximation of AC power flows," *INFORMS J. Comput.*, vol. 26, no. 4, pp. 718–734, 2014.

[13] D. K. Molzahn and I. A. Hiskens, "A survey of relaxations and approximations of the power flow equations," *Found. Trends Electr. Energy Syst.*, vol. 4, no. 1-2, pp. 1–221, 2019.

[14] M. Klamkin, M. Tanneau, T. W. Mak, and P. Van Hentenryck, "Active bucketized learning for ACOPF optimization proxies," *arXiv:2208.07497*, 2022.

[15] X. Pan, M. Chen, T. Zhao, and S. H. Low, "DeepOPF: A feasibility-optimized deep neural network approach for AC optimal power flow problems," *IEEE Syst. J.*, vol. 17, no. 1, pp. 673–683, 2023.

[16] M. Chatzos, T. W. K. Mak, and P. Van Hentenryck, "Spatial network decomposition for fast and scalable AC-OPF learning," *IEEE Trans. Power Syst.*, vol. 37, no. 4, pp. 2601–2612, 2022.

[17] A. S. Zamzam and K. Baker, "Learning optimal solutions for extremely fast AC optimal power flow," in *IEEE Int. Conf. Commun., Control, Comput. Tech. Smart Grids (SmartGridComm)*, 2020.

[18] A. Kody, S. Chevalier, S. Chatzivasileiadis, and D. K. Molzahn, "Modeling the AC power flow equations with optimally compact neural networks: Application to unit commitment," *Elect. Power Syst. Res.*, vol. 212, p. 108282, 2022, presented at the *22nd Power Syst. Comput. Conf. (PSCC)*.

[19] L. Duchesne, E. Karangelos, and L. Wehenkel, "Recent developments in machine learning for energy systems reliability management," *Proc. IEEE*, vol. 108, no. 9, pp. 1656–1676, 2020.

[20] D. K. Molzahn, F. Dörfler, H. Sandberg, S. H. Low, S. Chakrabarti, R. Baldick, and J. Lavaei, "A survey of distributed optimization and control algorithms for electric power systems," *IEEE Trans. Smart Grid*, vol. 8, no. 6, pp. 2941–2962, Nov. 2017.

[21] D. K. Molzahn and L. A. Roald, "Towards an AC optimal power flow algorithm with robust feasibility guarantees," in *20th Power Syst. Comput. Conf. (PSCC)*, 2018.

[22] A. Venzke, L. Halilbasic, U. Markovic, G. Hug, and S. Chatzivasileiadis, "Convex relaxations of chance constrained AC optimal power flow," *IEEE Trans. Power Syst.*, vol. 33, no. 3, pp. 2829–2841, 2018.

[23] K. Bestuzheva, H. Hijazi, and C. Coffrin, "Convex relaxations for quadratic on/off constraints and applications to optimal transmission switching," *INFORMS J. Comput.*, vol. 32, no. 3, pp. 682–696, 2020.

[24] S. H. Low, "Convex relaxation of optimal power flow–Part II: Exactness," *IEEE Trans. Control Netw. Syst.*, vol. 1, no. 2, pp. 177–189, 2014.

[25] R. Madani, S. Sojoudi, and J. Lavaei, "Convex relaxation for optimal power flow problem: Mesh networks," *IEEE Trans. Power Syst.*, vol. 30, no. 1, pp. 199–211, 2014.

[26] A. Venzke, S. Chatzivasileiadis, and D. K. Molzahn, "Inexact convex relaxations for AC optimal power flow: Towards AC feasibility," *Elect. Power Syst. Res.*, vol. 187, p. 106480, 2020.

[27] Z. Tian and W. Wu, "Recover feasible solutions for SOCP relaxation of optimal power flow problems in mesh networks," *IET Gen., Trans. & Dist.*, vol. 13, no. 7, pp. 1078–1087, 2019.

[28] X. Fang, Z. Yang, J. Yu, and Y. Wang, "AC feasibility restoration in market clearing: Problem formulation and improvement," *IEEE Trans. Ind. Inform.*, vol. 18, no. 11, pp. 7597–7607, 2021.

[29] M. Li, Y. Du, J. Mohammadi, C. Crozier, K. Baker, and S. Kar, "Numerical comparisons of linear power flow approximations: Optimality, feasibility, and computation time," in *IEEE PES Soc. General Meeting (PESGM)*, 2022.

[30] L. Bobo, A. Venzke, and S. Chatzivasileiadis, "Second-order cone relaxations of the optimal power flow for active distribution grids: Comparison of methods," *Int. J. Elect. Power & Energy Syst.*, vol. 127, p. 106625, 2021.

[31] M. Vanin, H. Ergun, R. D'hulst, and D. Van Hertem, "Comparison of linear and conic power flow formulations for unbalanced low voltage network optimization," *Elect. Power Syst. Res.*, vol. 189, p. 106699, 2020, presented at the *21st Power Syst. Comput. Conf. (PSCC)*.

[32] A. Abur and A. Gómez Expósito, *Power System State Estimation: Theory and Implementation*. Marcel Dekker, 2004.

[33] B. Taheri and D. K. Molzahn, "Restoring AC power flow feasibility for solutions to relaxed and approximated optimal power flow problems," in *American Control Conf. (ACC)*, May 2023.

[34] ——, "AC power flow feasibility restoration via a state estimation-based post-processing algorithm," *arXiv:2304.11418*, 2023.

[35] C. Coffrin and P. Van Hentenryck, "A linear-programming approximation of AC power flows," *INFORMS J. Comput.*, vol. 26, no. 4, pp. 718–734, 2014.

[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd Int. Conf. Learning Representations (ICLR)*, 2015.

[37] A. B. Birchfield, T. Xu, K. M. Gegner, K. S. Shetye, and T. J. Overbye, "Grid structural characteristics as validation criteria for synthetic networks," *IEEE Trans. Power Syst.*, vol. 32, no. 4, pp. 3258–3265, 2016.

[38] IEEE PES Task Force on Benchmarks for Validation of Emerging Power System Algorithms, "The Power Grid Library for benchmarking AC optimal power flow algorithms," Aug. 2019, *arXiv:1908.02788*.

[39] C. Coffrin, R. Bent, K. Sundar, Y. Ng, and M. Lubin, "PowerModels.jl: An open-source framework for exploring power flow formulations," in *20th Power Syst. Comput. Conf. (PSCC)*, 2018.
[40] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Prog.*, vol. 106, no. 1, pp. 25–57, 2006.

APPENDIX A
DERIVATION OF SENSITIVITIES

The sensitivities of the voltage phasors $\mathbf{x}_R$ obtained from the state estimation-inspired algorithm in relation to the weight matrix $\mathbf{\Sigma}$ are calculated using (9), which is derived by taking the derivative of (7) with respect to $\mathbf{\Sigma}$ as follows:

$$Y = (\mathbf{H}^T\mathbf{\Sigma}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{\Sigma}(\mathbf{z}+\mathbf{b}-\mathbf{h}), \tag{15}$$

$$dY = \overbrace{d\Big(\left(\mathbf{H}^T\mathbf{\Sigma}\mathbf{H}\right)^{-1}\mathbf{H}^T\Big)\mathbf{\Sigma}\left(\mathbf{z}+\mathbf{b}-\mathbf{h}\right)}^{I} + \underbrace{\left(\mathbf{H}^T\mathbf{\Sigma}\mathbf{H}\right)^{-1}\mathbf{H}^T d\Big(\mathbf{\Sigma}\left(\mathbf{z}+\mathbf{b}-\mathbf{h}\right)\Big)}_{U}, \tag{16}$$

$$U = \left(\mathbf{H}^T\mathbf{\Sigma}\mathbf{H}\right)^{-1}\mathbf{H}^T d\mathbf{\Sigma}\left(\mathbf{z}+\mathbf{b}-\mathbf{h}\right), \tag{17}$$

$$I = d\Big(\left(\mathbf{H}^T\mathbf{\Sigma}\mathbf{H}\right)^{-1}\Big)\mathbf{H}^T\mathbf{\Sigma}\left(\mathbf{z}+\mathbf{b}-\mathbf{h}\right), \tag{18}$$

$$I = \Big(-\left(\mathbf{H}^T\mathbf{\Sigma}\mathbf{H}\right)^{-1} d\left(\mathbf{H}^T\mathbf{\Sigma}\mathbf{H}\right)\left(\mathbf{H}^T\mathbf{\Sigma}\mathbf{H}\right)^{-1}\Big) \times\Big(\mathbf{H}^T\mathbf{\Sigma}\left(\mathbf{z}+\mathbf{b}-\mathbf{h}\right)\Big), \tag{19}$$

$$I = \Big(-\left(\mathbf{H}^T\mathbf{\Sigma}\mathbf{H}\right)^{-1}\mathbf{H}^T d\mathbf{\Sigma}\mathbf{H}\left(\mathbf{H}^T\mathbf{\Sigma}\mathbf{H}\right)^{-1}\Big) \times\Big(\mathbf{H}^T\mathbf{\Sigma}\left(\mathbf{z}+\mathbf{b}-\mathbf{h}\right)\Big), \tag{20}$$

$$dY = -\left(\mathbf{H}^T\mathbf{\Sigma}\mathbf{H}\right)^{-1}\mathbf{H}^T d\mathbf{\Sigma}\mathbf{H}\left(\mathbf{H}^T\mathbf{\Sigma}\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{\Sigma}(\mathbf{z}+\mathbf{b}-\mathbf{h}) + \left(\mathbf{H}^T\mathbf{\Sigma}\mathbf{H}\right)^{-1}\mathbf{H}^T d\mathbf{\Sigma}\left(\mathbf{z}+\mathbf{b}-\mathbf{h}\right), \tag{21}$$

$$\mathrm{vec}(dY) = \mathrm{vec}\Big[-\left(\mathbf{H}^T\mathbf{\Sigma}\mathbf{H}\right)^{-1}\mathbf{H}^T d\mathbf{\Sigma}\mathbf{H}\left(\mathbf{H}^T\mathbf{\Sigma}\mathbf{H}\right)^{-1} \times\mathbf{H}^T\mathbf{\Sigma}\left(\mathbf{z}+\mathbf{b}-\mathbf{h}\right)\Big] + \mathrm{vec}\Big[\left(\mathbf{H}^T\mathbf{\Sigma}\mathbf{H}\right)^{-1} \times\mathbf{H}^T d\mathbf{\Sigma}\left(\mathbf{z}+\mathbf{b}-\mathbf{h}\right)\Big], \tag{22}$$

$$\mathrm{vec}(dY) = \Big[-\Big(\mathbf{H}\left(\mathbf{H}^T\mathbf{\Sigma}\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{\Sigma}\left(\mathbf{z}+\mathbf{b}-\mathbf{h}\right)\Big)^T \otimes\Big(\left(\mathbf{H}^T\mathbf{\Sigma}\mathbf{H}\right)^{-1}\mathbf{H}^T\Big)\Big]\mathrm{vec}(d\mathbf{\Sigma}) + \Big[(\mathbf{z}+\mathbf{b}-\mathbf{h})^T \otimes\Big(\left(\mathbf{H}^T\mathbf{\Sigma}\mathbf{H}\right)^{-1}\mathbf{H}^T\Big)\Big]\mathrm{vec}(d\mathbf{\Sigma}), \tag{23}$$

$$\mathrm{vec}(dY) = \Big[(\mathbf{z}+\mathbf{b}-\mathbf{h})^T \otimes\Big(\left(\mathbf{H}^T\mathbf{\Sigma}\mathbf{H}\right)^{-1}\mathbf{H}^T\Big) -\Big(\mathbf{H}\left(\mathbf{H}^T\mathbf{\Sigma}\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{\Sigma}\left(\mathbf{z}+\mathbf{b}-\mathbf{h}\right)\Big)^T \otimes\Big(\left(\mathbf{H}^T\mathbf{\Sigma}\mathbf{H}\right)^{-1}\mathbf{H}^T\Big)\Big]\mathrm{vec}(d\mathbf{\Sigma}), \tag{24}$$

$$\frac{\mathrm{vec}(\partial Y)}{\mathrm{vec}(\partial\mathbf{\Sigma})} = \Big((\mathbf{z}+\mathbf{b}-\mathbf{h}) - (\mathbf{H}(\mathbf{H}^T\mathbf{\Sigma}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{\Sigma} \times(\mathbf{z}+\mathbf{b}-\mathbf{h})\Big) \otimes\Big((\mathbf{H}^T\mathbf{\Sigma}\mathbf{H})^{-1}\mathbf{H}^T\Big)^T. \tag{25}$$

APPENDIX B
EXPLOITING THE DIAGONAL STRUCTURE OF Σ

We observe that the computation of (4) can be optimized by leveraging the diagonal structure of the $\mathbf{\Sigma}$ matrix. Instead of using a matrix, we can represent the weight terms as a vector $\boldsymbol{\sigma}$ with $\boldsymbol{\sigma}_i = \mathbf{\Sigma}_{ii}$, $i = 1,\ldots,m$. This approach allows us to focus on computing sensitivities exclusively for the diagonal entries, leading to a more efficient calculation. We will next demonstrate this approach and its computational advantages.

First, we rewrite (4) as:

$$J(\mathbf{x}) = (\mathbf{z}+\mathbf{b}-\mathbf{h}(\mathbf{x}))^T\mathbf{\Sigma}(\mathbf{z}+\mathbf{b}-\mathbf{h}(\mathbf{x})), \tag{26a}$$
$$= (\mathbf{z}+\mathbf{b}-\mathbf{h}(\mathbf{x}))^T\boldsymbol{\sigma}\odot(\mathbf{z}+\mathbf{b}-\mathbf{h}(\mathbf{x})), \tag{26b}$$

where $\odot$ denotes the Hadamard (element-wise) product. Following the same procedure as in (5)–(7), we first compute the derivative of (26b) with respect to $\mathbf{x}$ as follows:

$$\mathbf{g}(\mathbf{x}) = \frac{\partial J(\mathbf{x})}{\partial\mathbf{x}} = -\Big(\boldsymbol{\sigma}\odot\mathbf{H}(\mathbf{x})\Big)^T(\mathbf{z}+\mathbf{b}-\mathbf{h}(\mathbf{x})), \tag{27}$$

where $\mathbf{H}(\mathbf{x}) = \frac{\partial\mathbf{h}(\mathbf{x})}{\partial\mathbf{x}}$ is the Jacobian matrix of the function $\mathbf{h}(\mathbf{x})$. The derivative of $\mathbf{g}(\mathbf{x})$ with respect to $\mathbf{x}$ is:

$$\mathbf{G}(\mathbf{x}) = \frac{\partial\mathbf{g}(\mathbf{x})}{\partial\mathbf{x}} = \Big(\frac{\partial\mathbf{h}(\mathbf{x})}{\partial\mathbf{x}}\Big)^T\boldsymbol{\sigma}\odot\mathbf{H}(\mathbf{x}). \tag{28}$$

To solve $\mathbf{g}(\mathbf{x}) = 0$, we apply the Newton-Raphson method described in Algorithm 1 with modified $\mathbf{g}(\mathbf{x})$ and $\mathbf{G}(\mathbf{x})$ that performs, at the k-th iteration, the following steps:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - (\mathbf{G}(\mathbf{x}))^{-1}g(\mathbf{x}), \tag{29a}$$
$$\Delta\mathbf{x}^k = \left(\mathbf{H}^T\boldsymbol{\sigma}\odot\mathbf{H}\right)^{-1}\Big(\boldsymbol{\sigma}\odot\mathbf{H}\Big)^T(\mathbf{z}+\mathbf{b}-\mathbf{h}), \tag{29b}$$
$$\mathbf{x}^{k+1} = \mathbf{x}^k + \Delta\mathbf{x}^k. \tag{29c}$$

Now, we can derive the sensitivities of the state vector $\mathbf{x}_R$ with respect to the $\boldsymbol{\sigma}$ in the same fashion as in Appendix A:

$$\frac{\partial\mathbf{x}_R}{\partial\boldsymbol{\sigma}} = \Big((\mathbf{z}+\mathbf{b}-\mathbf{h}) - (\mathbf{H}(\mathbf{H}^T\boldsymbol{\sigma}\odot\mathbf{H})^{-1}\mathbf{H}^T\boldsymbol{\sigma} \odot(\mathbf{z}+\mathbf{b}-\mathbf{h}))\Big) \odot\Big((\mathbf{H}^T\boldsymbol{\sigma}\odot\mathbf{H})^{-1}\mathbf{H}^T\Big)^T. \tag{30}$$

The expression (30) gives the sensitivities of the restored point $\mathbf{x}_R$ with respect to the vector of weight parameters $\boldsymbol{\sigma}$. With length-m vectors $\mathbf{z}$ and $\mathbf{h}(\mathbf{x})$ and an $m\times n$ matrix $\mathbf{H}(\mathbf{x})$, the sensitivities $\frac{\partial\mathbf{x}_R}{\partial\boldsymbol{\sigma}}$ are represented by a $n\times m$ matrix. Accordingly, the size of the sensitivity matrix in (9) reduces from $n\times m^2$ to $n\times m$; therefore, this approach results in a more efficient implementation than considering the sensitivities of all entries of the $\mathbf{\Sigma}$ matrix.