Optimally Managing the Impacts of Convergence Tolerance for Distributed Optimal Power Flow

Rachel Harris, *Student Member, IEEE*, Mohannad Alkhraijah, *Student Member, IEEE*,
Daniel K. Molzahn, *Senior Member, IEEE*

***Abstract*—The future power grid may rely on distributed optimization to determine the set-points for huge numbers of distributed energy resources. There has been significant work on applying distributed algorithms to optimal power flow (OPF) problems, which require separate controllers to agree on shared boundary variable values. Looser tolerances for the mismatches in these shared variables generally yield faster convergence at the expense of exacerbating constraint violations, but there is little quantitative understanding of how the convergence tolerance affects solution quality. To address this gap, we first quantify how convergence tolerance impacts constraint violations when the distributed OPF solution is applied to the power system. Using insights from this analysis, we then develop a bound tightening algorithm which guarantees that operating points from distributed OPF algorithms will not result in violations despite the possibility of shared variable mismatches within the convergence tolerance. We also explore how bounding the cumulative shared variable mismatches can prevent unnecessary conservativeness in the bound tightening. While ensuring feasibility, the proposed approach enables control of the trade-off between computational speed, which improves with looser convergence tolerance, and distributed OPF solution cost, which degrades with looser convergence tolerance due to tightened constraints.**

***Index Terms*—Distributed optimization, optimal power flow, convergence tolerance, bound tightening**

I. Introduction

As we transition to low-carbon power systems, distributed energy resources (DERs) such as electric vehicles, battery storage systems, and wind and solar generators will increase by orders of magnitude, motivating the development of new optimization and control methods [1]. Traditional power system optimization approaches where a central operator collects system-wide information and computes optimal dispatches for bulk generation plants may be inadequate for future power systems with widespread DER integration and consumers who desire data privacy. Distributed optimization algorithms can scale to large, complex problems, have the potential to maintain the privacy and autonomy of consumers, and avoid a single point of failure. Such distributed algorithms may be used to coordinate interconnected transmission and distribution networks [2]–[4], optimally manage aggregations of small-scale DERs and dispatchable loads [5]–[7], and leverage parallelization for faster computation of large-scale OPF-constrained problems [8], [9].

Distributed algorithms require further research and development before being applied in practical system operation. Some challenges addressed in previous literature improve convergence performance for non-convex problems [10], [11], enable asynchronous communication [12], [13] and enhance cybersecurity [14], [15], including our recent work on detecting and mitigating malicious manipulation of shared data [16]. This paper focuses on the critical challenge of convergence rate: many distributed optimal power flow (OPF) algorithms take thousands of iterations to converge for large-scale power systems [11], [17], [18]. To use such distributed algorithms to operate future power grids with many rapidly fluctuating DERs, we must reduce computation time. To accelerate distributed algorithms, researchers have proposed adaptive parameter tuning [19]–[21], using machine learning to predict the final boundary variable values [22], [23], or running computations on GPUs for massive parallelization [8].

Distributed algorithms decompose the system into separate regions, each under the control of different local controllers. These controllers solve local optimization problems and share boundary variable values to ensure consistency between regions. The algorithm converges when the norm of the shared variable mismatch values falls below a convergence tolerance ϵ. The authors of [24] provide some guidance on how to select ϵ based on scale of the variables, and most researchers select $\epsilon \in [10^{-5}, 10^{-3}]$. However, to the best of our knowledge, the literature contains no detailed analysis of the impact of convergence tolerance on constraint violations after a distributed OPF solution is applied to the power grid.

One simple way to reduce convergence time is to select a larger convergence tolerance ϵ. As we will demonstrate in this paper, looser tolerances can significantly decrease the number of iterations required to converge and thus reduce computation times. However, before loosening the tolerance, we must ensure the resulting distributed OPF solution provides a safe operating point that will not cause constraint violations. In this paper, we assess the impacts of convergence tolerance on constraint violations and develop a bound tightening algorithm which prevents these violations. We focus on the AC OPF problem solved with the alternating direction method of multipliers (ADMM) distributed algorithm, but our method can be applied without any conceptual changes to other power flow formulations or distributed algorithms.

The main contributions of this paper are as follows:

1) We formulate an optimization problem to find the maximum possible violations of engineering constraints at the system operating point selected by a distributed OPF algorithm terminated at a given convergence tolerance.
2) We develop a bound tightening algorithm to prevent

Support from NSF AI Institute for Advances in Optimization (AI4OPT), #2112533. The authors are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA.

engineering constraint violations in a power system under distributed OPF dispatch, where the distributed algorithm converged to a given tolerance. The operator performing optimization can select a larger convergence tolerance to decrease computation time and run the bound tightening algorithm offline. During real-time operation, distributed OPF using the tightened bounds will not result in any constraint violations once converged to the given tolerance.

3) We present numerical results from several representative test cases and show that running distributed OPF on the bound-tightened cases significantly decreases computation time without resulting in constraint violations.

The remainder of this paper is organized as follows. In Section II, we describe the distributed OPF formulation and discuss how the choice of convergence tolerance impacts computation speed and the result's feasibility. Section IV formulates an optimization problem which finds the worst-case constraint violations that may result from selecting a certain convergence tolerance for the distributed OPF computation. We present a bound tightening algorithm which iteratively solves this optimization problem and tightens constraints until there can be no violations of the original constraints when the distributed OPF solution converged to the given tolerance is applied to the system. The algorithm may be augmented with bounds on cumulative mismatches so that bound tightening is less conservative. In Section V, we present numerical results, including solution costs for constraint-tightened test cases and relationships between cumulative mismatch bounds and violations. We conclude and discuss future work in Section VI.

II. Distributed OPF Formulation

This section provides background material by formulating the OPF problem and reviewing distributed OPF algorithms.

A. Optimal Power Flow

The OPF problem optimizes power system performance subject to engineering constraints and equations that model physical power flows through the system. We formulate OPF using the AC power flow equations and an objective which minimizes generation cost. However, the analysis and methods presented in the paper could be used for other OPF formulations without major conceptual changes.

The OPF problem is

$$\min_{\substack{\boldsymbol{p}^g,\boldsymbol{q}^g,\boldsymbol{p},\\ \boldsymbol{q},\boldsymbol{\theta},\boldsymbol{v}}} \quad \sum_{i\in\mathcal{N}} f_i(p_i^g) \tag{1a}$$

$$\text{s.t.} \quad \theta_i = 0 \text{ for } i \in \mathcal{S}, \tag{1b}$$

$$\forall i \in \mathcal{N},\ \forall (i,j) \in \mathcal{E}:$$

$$p_i^g - p_i^d = \sum_{(i,j)\in\mathcal{E}} p_{ij} + g_i^{sh} v_i^2, \tag{1c}$$

$$q_i^g - q_i^d = \sum_{(i,j)\in\mathcal{E}} q_{ij} - b_i^{sh} v_i^2, \tag{1d}$$

$$p_{ij} = v_i^2 G_{ij} - v_i v_j \left[G_{ij}\cos(\theta_{ij}) + B_{ij}\sin(\theta_{ij})\right], \tag{1e}$$

$$q_{ij} = -v_i^2 (B_{ij}^{sh} + B_{ij}) - v_i v_j \left[G_{ij}\sin(\theta_{ij}) - B_{ij}\cos(\theta_{ij})\right], \tag{1f}$$

$$\underline{P}_i^g \le p_i^g \le \overline{P}_i^g, \quad \underline{Q}_i^g \le q_i^g \le \overline{Q}_i^g, \tag{1g}$$

$$\underline{V}_i \le v_i \le \overline{V}_i, \tag{1h}$$

$$p_{ij}^2 + q_{ij}^2 \le \left(\overline{S}_{ij}\right)^2, \tag{1i}$$

where the set of buses is $\mathcal{N}$ and the set of lines is $\mathcal{E}$. For each line $(i,j) \in \mathcal{E}$, we denote the series conductance and susceptance as G_{ij} and B_{ij}, and the line's thermal limit as $\overline{S}_{ij}$. The angle difference between bus i and bus j is $\theta_{ij} = \theta_i - \theta_j$. The active and reactive power flows from bus i to bus j along line (i,j) are denoted by p_{ij} and q_{ij}, respectively. For bus $i \in \mathcal{N}$, its voltage phasor is $v_i \angle \theta_i$ and its shunt admittance is $g_i^{sh} + j b_i^{sh}$. Each bus $i \in \mathcal{N}$ may also have active power demand p_i^d and reactive power demand q_i^d. If bus i contains a generator, its active power output is p_i^g, its reactive power output is q_i^g, and the cost function is f_i. Also, $\mathcal{S}$ contains the reference bus. This formulation minimizes the generation cost (1a) subject to power balance constraints (1c)–(1d), power flow across lines (1e)–(1f), generator operational limits (1g), voltage magnitude limits (1h), and line thermal limits (1i). Note also that we set the phase angle to 0 at a selected reference bus in (1b).

B. Distributed Formulation with ADMM

In the distributed OPF formulation, the power network is divided into multiple regions, each operated by a local controller. When branch terminals are in different regions, we add fictitious buses as shown in Figure 1 and set consistency constraints to ensure that the fictitious variables match the original variables in the neighbor's region.

We can solve the distributed OPF formulation using alternating distributed algorithms (ADAs). In such algorithms, local controllers solve OPF subproblems over their region of the network. They augment their local OPF objective with relaxed consistency constraints using boundary variable values shared from neighboring controllers. Controllers iteratively solve their OPF subproblems and share boundary variable data until the consistency constraints are satisfied. This paper focuses on the ADMM algorithm, although our methods apply directly to other ADAs such as APP and ATC.

We formulate the distributed OPF problem for ADMM as follows. For some region m, we denote the set of buses

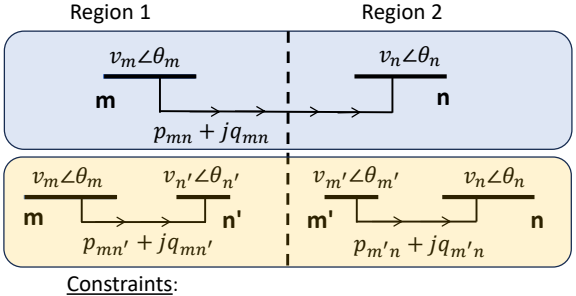


Fig. 1: Decomposition of power network

and lines in the region as $\mathcal{N}_m$ and $\mathcal{E}_m$, respectively. We denote the set of boundary variables which region m shares with its neighbors by $\mathcal{N}_m^s$, and gather the values of shared variables into the vector $\boldsymbol{z}_m$. The ADMM formulation keeps a "central" copy of shared variables, and we denote the values of central variables associated with region m as $\bar{\boldsymbol{z}}_m$. Note that we compute these "central" variables in a distributed manner without requiring a coordinator, as in [25]. The vector $\boldsymbol{y}_m$ contains the dual variables of the consistency constraints for region m. In addition, we use the same notation for variables as in (1) but add dots to designate controllers' copies of variables in their region, so that, e.g., $\dot{p}_{i,m}^g$ denotes region m's copy of the power generation at bus i.

The OPF subproblem for region m at the k-th iteration is shown below:

$$\min_{\substack{\dot{\boldsymbol{p}}^{g,k},\dot{\boldsymbol{q}}^{g,k},\dot{\boldsymbol{p}}^k,\\ \dot{\boldsymbol{q}}^k,\dot{\boldsymbol{\theta}}^k,\dot{\boldsymbol{v}}^k,\boldsymbol{z}_m^k}} \sum_{i\in\mathcal{N}_m} f_i(\dot{p}_{i,m}^{g,k}) + (\boldsymbol{y}_m^{k-1})^T\boldsymbol{z}_m^k + \frac{\alpha}{2}||\boldsymbol{z}_m^k - \bar{\boldsymbol{z}}_m^{k-1}||_2^2 \tag{2a}$$

$$\text{s.t.} \quad \dot{\theta}_i = 0 \text{ for } i \in \mathcal{S}, \tag{2b}$$

$$\forall i \in \mathcal{N}_m, \forall (i,j) \in \mathcal{E}_m:$$

$$\dot{p}_{i,m}^{g,k} - p_i^d = \sum_{(i,j)\in\mathcal{E}_m} \dot{p}_{ij,m}^k + g_i^{sh}(\dot{v}_{i,m}^k)^2, \tag{2c}$$

$$\dot{q}_{i,m}^{g,k} - q_i^d = \sum_{(i,j)\in\mathcal{E}_m} \dot{q}_{ij,m}^k - b_i^{sh}(\dot{v}_{i,m}^k)^2, \tag{2d}$$

$$\dot{p}_{ij,m}^k = (\dot{v}_{i,m}^k)^2 G_{ij} - \dot{v}_{i,m}^k v_j^k \left[G_{ij}\cos(\theta_{ij}^k) + B_{ij}\sin(\dot{\theta}_{ij,m}^k)\right], \tag{2e}$$

$$\dot{q}_{ij,m}^k = -(\dot{v}_{i,m}^k)^2 (B_{ij}^{sh} + B_{ij}) - \dot{v}_{i,m}^k v_j^k \left[G_{ij}\sin(\dot{\theta}_{ij,m}^k) - B_{ij}\cos(\dot{\theta}_{ij,m}^k)\right], \tag{2f}$$

$$\underline{P}_i^g \le \dot{p}_{i,m}^{g,k} \le \overline{P}_i^g, \quad \underline{Q}_i^g \le \dot{q}_{i,m}^{g,k} \le \overline{Q}_i^g, \tag{2g}$$

$$\underline{V}_i \le \dot{v}_{i,m}^k \le \overline{V}_i, \tag{2h}$$

$$(\dot{p}_{ij,m}^k)^2 + (\dot{q}_{ij,m}^k)^2 \le \left(\overline{S}_{ij}\right)^2. \tag{2i}$$

The penalty parameter α is selected by the user. After solving (2), controller m shares the boundary variable values $\boldsymbol{z}_m$ with their neighbors, receives their neighbors' copies of these variables, and then locally updates the "central" variables $\bar{z}_m$:

$$\bar{\boldsymbol{z}}_{m,n}^k = \frac{1}{2}(\boldsymbol{z}_{m,n}^k + \boldsymbol{z}_{n,m}^k), \tag{3}$$

where $\boldsymbol{z}_{m,n}$ denotes region m's copies of variables shared between regions m and n, while $\boldsymbol{z}_{n,m}$ denotes region n's copies of these shared variables.

Finally, controller m updates their dual variables as

$$\boldsymbol{y}_m^k = \boldsymbol{y}_m^{k-1} + \alpha(\boldsymbol{z}_m^k - \bar{\boldsymbol{z}}_m^k). \tag{4}$$

The ADMM algorithm iterations repeat until convergence, where each iteration consists of minimizing local subproblems (2), updating central copies of shared variables (3), and computing the new dual variables (4). Typically, the stopping criterion is based on primal and dual residuals [24]. The vector of primal residuals $\boldsymbol{r}^k$ contains the difference between local and central copies of all boundary variable values:

$$\boldsymbol{r}^k = \begin{bmatrix}\boldsymbol{z}_1^T - \bar{\boldsymbol{z}}_1^T & \boldsymbol{z}_2^T - \bar{\boldsymbol{z}}_2^T & \dots & \boldsymbol{z}_M^T - \bar{\boldsymbol{z}}_M^T\end{bmatrix}^T. \tag{5}$$

The dual residual is

$$\boldsymbol{s}^k = -\alpha(\bar{\boldsymbol{z}}^k - \bar{\boldsymbol{z}}^{k-1}), \tag{6}$$

where we have collected all central copies of boundary variables into one vector $\bar{\boldsymbol{z}}$. The algorithm terminates when the primal and dual residual norms fall below the respective primal and dual tolerances:

$$||\boldsymbol{r}^k|| \le \epsilon^{pri}, \quad ||\boldsymbol{s}^k|| \le \epsilon^{dual}. \tag{7}$$

The next section discusses how these tolerances are selected.

III. Selecting Convergence Tolerances

We terminate the distributed OPF algorithm when the primal and dual residuals are sufficiently small. The most widely referenced work on ADMM, [24], suggests using the ℓ_2-norm of the primal and dual residuals as the stopping criterion. Many papers on distributed AC OPF also use the ℓ_2-norm of both primal and dual residuals [19], [21], [24], [26], [27]. Other papers use the ℓ_∞- or ℓ_2-norm of the dual residuals only [28], [29], while yet other publications use the ℓ_∞- or ℓ_2-norm of the primal residuals [11], [30], [31]. Most of the above works select a tolerance in the range of $[10^{-5}, 10^{-3}]$, although [24] proposes a method to define tolerances based on the scale of the variables:

$$\begin{aligned}\epsilon^{pri} &= \sqrt{p}\epsilon^{abs} + \epsilon^{rel}\max\{||\boldsymbol{A}\boldsymbol{x}^k||_2, ||\boldsymbol{B}\boldsymbol{z}^k||_2, ||\boldsymbol{c}||_2\},\\ \epsilon^{dual} &= \sqrt{n}\epsilon^{abs} + \epsilon^{rel}||\boldsymbol{A}^T\boldsymbol{y}^k||_2,\end{aligned} \tag{8}$$

where ϵ^{abs}, ϵ^{rel} are user-selected absolute and relative tolerances, respectively. The notation is for a general ADMM formulation which minimizes a function $f(\boldsymbol{x}) + g(\boldsymbol{z})$ subject to the coupling constraint $\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} = \boldsymbol{c}$, with shared variables $\boldsymbol{x} \in \mathbb{R}^n$ and dual variables $\boldsymbol{y} \in \mathbb{R}^p$.

Our analysis will focus on ϵ^{pri}, and we will determine convergence based on the primal residuals alone. The requirement for small primal residuals, $||\boldsymbol{r}^k|| \le \epsilon^{pri}$, results in near feasibility of the final solution by satisfying consistency constraints. The requirement for small dual residuals, $||\boldsymbol{s}^k|| \le \epsilon^{dual}$, is related to optimality of the final solution. This paper's analysis is primarily concerned with feasibility, and our methods are designed to ensure feasible solutions for a given choice of ϵ^{pri}. However, our numerical results demonstrate that in practice, given appropriate choice of penalty parameter α, setting the stopping criterion based on primal residuals results in solutions that are both nearly optimal and nearly feasible.

We choose the ℓ_∞ norm as the convergence criterion in our analyses for two reasons. First, the ℓ_∞ norm of the shared variable mismatches is immediately interpretable as the maximum variable mismatch and has units of p.u. for voltage magnitudes and power flows and radians for voltage angles. Second, it allows for simple linear constraints in the worst-case violation optimization problem we formulate in Section IV. Extensions of the algorithm we will propose in this paper to other norms are conceptually straightforward.

Changing the convergence tolerance ϵ^{pri} impacts the speed, feasibility, and optimality of distributed optimization algorithms. We provide an illustrative example using case500 from the PGLib-OPF archive [32] divided into 8 regions

for distributed optimization. We use the PowerModelsADA library [33] to solve the distributed OPF problem using the ADMM algorithm. We run the distributed OPF algorithm 2000 times, sweeping the convergence tolerance ϵ^{pri} from 10^{-6} to 10^{-3}, and each time randomly perturbing loads by selecting values between 70%–130% of nominal. Once the distributed OPF algorithm terminates, we run an AC power flow on the system using the *control values* from the distributed OPF solution, which are active power injections and voltage magitudes at PV buses. We determine if the results violate any bounds on voltage magnitudes, reactive power generation, or line flows. The results are shown in Figure 2, where the shaded red bands around the median line in black show every fifth percentile of the results. Figure 2a shows that the number of iterations required to reach convergence decrease significantly as ϵ^{pri} increases. Figure 2b shows the average percent violation for the constraint violations that occur, where we define the average percent violations as

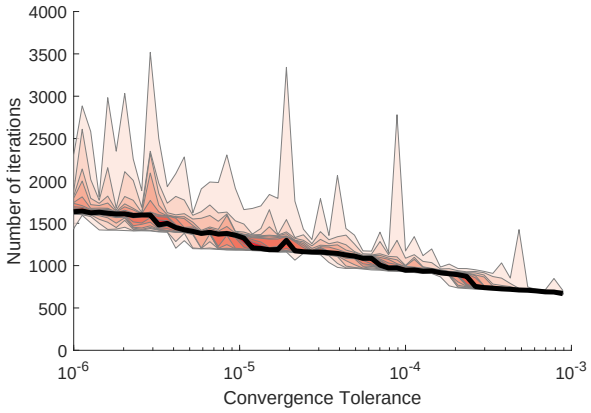
$$\frac{1}{N_v}\sum_{i\in\mathcal{C}}\frac{\max\{x_i^{AC-PF}-x_i^{max},x_i^{min}-x_i^{AC-PF},0\}}{x_i^{max}-x_i^{min}},$$

where N_v is the number of violated constraints and $\mathcal{C}$ contains indices of all variables representing voltage magnitudes, reactive power injections, and line flows. We denote the value of the i-th variable computed by the AC power flow as x_i^{AC-PF}, and its minimum and maximum values as x_i^{min} and x_i^{max}. The median number of violations per run is shown in Figure 2c. As the maximum shared variable mismatches approach 10^{-4}, the power flow solution from the distributed OPF operating point starts to have non-negligible constraint violations, which increase with larger tolerances ϵ^{pri}. This behavior is exactly what we would expect, since as ϵ^{pri} becomes sufficiently large, the consistency constraints for boundary variables are not satisfied and the distributed OPF solution may not be feasible. Note that while the computation time decreases at an approximately linear rate, there is a sudden steep increase in the average percent violations at about $\epsilon^{pri} = 4 \times 10^{-5}$.

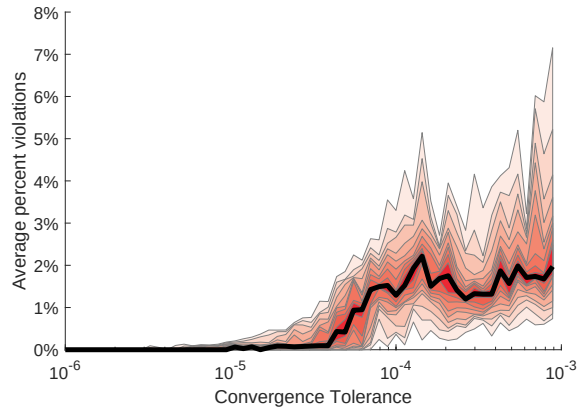
IV. Analysis and Bound Tightening Algorithm

As shown by the example in the prior section, sufficiently loose convergence tolerances may lead to non-negligible constraint violations. This motivates the development of techniques for bounding the worst-case constraint violations and mitigating their impacts on the resulting solutions. We develop a method to determine the worst-case constraint violations that may occur from applying the distributed OPF solution converged to a given tolerance ϵ^{pri} to the system. We first formulate an optimization problem which finds the worst-case constraint violations for a given maximum boundary variable mismatch ϵ^{pri}. Next, we propose a bound tightening algorithm which alternates between finding the worst-case violations and subsequently tightening the constraints to mitigate those violations. Provided that the true worst-case violation is found for each constraint, the distributed OPF algorithm run on the bound-tightened case will not violate any original constraints once applied to the system.

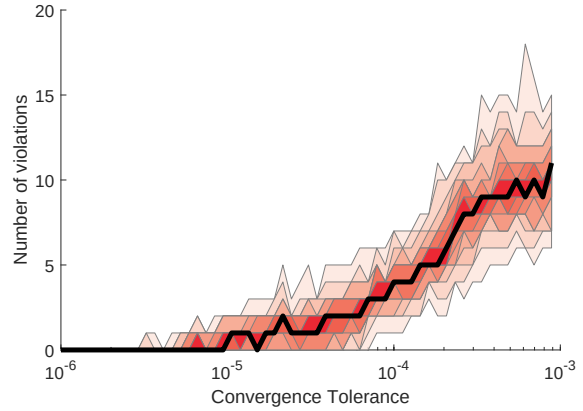
The worst-case violation analysis and constraint-tightening algorithm is useful for OPF problems solved repeatedly, with



(a) Num. iterations to converge vs. convergence tolerance



(b) Average percent violation vs. convergence tolerance



(c) Num. constraint violations vs. convergence tolerance

Fig. 2: Impact of convergence tolerance for case500 from the PGLib-OPF archive. As the convergence tolerance becomes larger, the algorithm converges more quickly, but non-negligible violations of engineering constraints begin to occur. While the number of iterations to converge decreases with the convergence tolerance, the average percent violation and the total number of constraint violations across the system both increase. For each plot, the black line is the average across 2000 runs with randomly varying load demands (70% to 130% of nominal), with lighter red shading representing percentiles of the distribution of outcomes in increments of 5%.

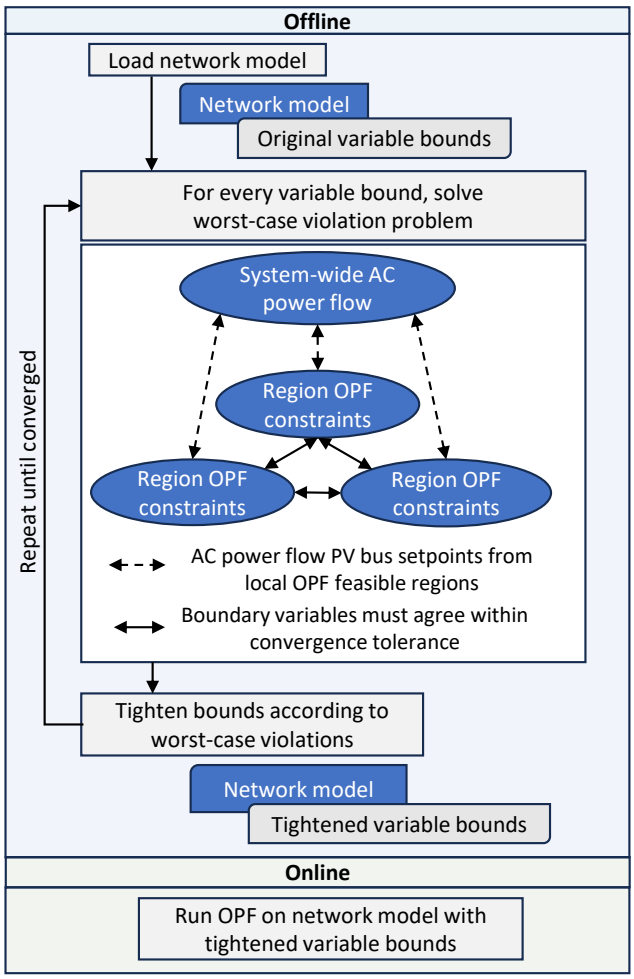


Fig. 3: Bound tightening algorithm overview

a constant network model and loads varying with each run. The proposed algorithm requires the ability to perform offline calculations where information regarding the entire system is available. Offline, we formulate an optimization problem which finds the worst-case violation, allowing the loads to take any values within a specified range, given some ϵ^{pri}. We iterate between solving the worst-case violation problem for all variable bounds and tightening the bounds according to the worst-case violations until the algorithm converges. We show an overview of the full bound tightening algorithm in Figure 3 and next provide the formulation and algorithm details.

A. Notation and Modeling Choices

We propose a method to determine the worst-case constraint violations that may occur for a convergence tolerance ϵ^{pri}. We use the same notation for system variables as in Section II-A. Note that for the worst-case violation problem, the active and reactive power demands p_i^d and q_i^d are variables. We allow the power demands to vary by a factor r; for example, if $r = 0.5$, then the active and reactive power demands may take any value between 50%–150% of their nominal values, denoted as $p_i^{d,nom}$ and $q_i^{d,nom}$. We keep a constant power factor by modeling consistent perturbations to both active and reactive power at a given bus by the same factor r. With this approach, we can perform offline computations for the tightened variable bounds without needing information regarding the exact values of loads that would only be available to local controllers in real-time calculations, as shown in Figure 3. We denote $\mathcal{N}_g$ as the set of buses with generators and $\mathcal{S}$ as the slack bus.

We use the same notation for variables contained in local regions as in Section II-B. We put a dot over variables belonging to local systems to distinguish them from the central system variables. Again, $\mathcal{M}$ denotes the set of local controllers in the distributed OPF problem and $\mathcal{A}_m$ denotes the neighbors of controller m. Also, the vector $\boldsymbol{z}_{m,n}$ contains controller m's copies of all boundary variables shared between controllers m and n, which includes voltage magnitudes and angles for boundary buses and active and reactive power flows on boundary lines. We denote the amount by which original bounds have been tightened by $\lambda_{\underline{V}_i}, \lambda_{\overline{V}_i}$ for lower and upper bounds on the voltage at bus i, $\lambda_{\underline{Q}_i}, \lambda_{\overline{Q}_i}$ for lower and upper bounds on reactive power generation at bus i, and $\lambda_{\overline{S}_{ij}}$ for the upper bound on apparent power flow across line (i,j). For instance, with a constraint tightening of $\lambda_{\overline{V}_i}$, the upper voltage limit (2h) in an controller's subproblem becomes $\dot{v}_{i,m}^k \leq \overline{V}_i - \lambda_{\overline{V}_i}$. We collect the amount of bound tightening on all variables into one vector $\boldsymbol{\lambda}$.

We note that the worst-case violation on any variable bound depends on the choice of convergence tolerance ϵ^{pri} and on the amount of bound tightening $\boldsymbol{\lambda}$. Therefore, we denote the worst-case violations on upper and lower bounds on voltage magnitudes at bus i as $W_{\overline{v}_i}(\epsilon^{pri}, \boldsymbol{\lambda})$ and $W_{\underline{v}_i}(\epsilon^{pri}, \boldsymbol{\lambda})$, respectively; on upper and lower bounds on reactive power generation at bus i as $W_{\overline{q}_i}(\epsilon^{pri}, \boldsymbol{\lambda})$ and $W_{\underline{q}_i}(\epsilon^{pri}, \boldsymbol{\lambda})$, respectively; and on upper bounds on line apparent power flows at line (i,j) as $W_{\overline{s}_{ij}}(\epsilon^{pri}, \boldsymbol{\lambda})$. We next describe an optimization formulation for calculating the worst-case constraint violations for a given convergence tolerance ϵ^{pri} and bound tightening $\boldsymbol{\lambda}$.

B. Worst-Case Violation Formulation

We formulate an optimization problem that computes the worst-case constraint violations for a given range of load variation and convergence tolerance. To formulate this problem, we begin with constraints that belong to two categories:

1) *Distributed OPF constraints* which represent distributed OPF algorithm behavior. The variables kept by local regions (marked with a dot) must satisfy OPF constraints within that region. In addition, consistency constraints require that the differences between neighboring regions' copies of shared variables are no more than ϵ^{pri}.
2) *System-wide AC power flow constraints* which represent the physical behavior of the system under a distributed OPF solution dispatch. These constraints involve variables representing the physical system (which are not marked with a dot) and are the traditional AC power flow equations. The setpoints for PV buses in the AC power flow come from the distributed OPF variable values.

To compute worst-case violations, we will form optimization problems that have the following constraints:

$$\forall m \in \mathcal{M}: \quad \text{(2c)–(2g)} \tag{9a}$$

$$\underline{V}_i + \lambda_{\underline{V}_i} \leq \dot{v}_{i,m} \leq \overline{V}_i - \lambda_{\overline{V}_i}, \quad \forall i \in \mathcal{N}_m, \tag{9b}$$

$$\underline{Q}^g_i + \lambda_{\underline{Q}_i} \leq \dot{q}^g_{i,m} \leq \overline{Q}^g_i - \lambda_{\overline{Q}_i}, \quad \forall i \in \mathcal{N}_m, \tag{9c}$$

$$(\dot{p}_{ij,m})^2 + (\dot{q}^k_{ij,m})^2 \leq \left(\overline{S}_{ij} - \lambda_{\overline{S}_{ij}}\right)^2, \ \forall (i,j) \in \mathcal{E}_m, \tag{9d}$$

$$||\boldsymbol{z}_{m,n} - \boldsymbol{z}_{n,m}||_\infty \leq \epsilon^{pri}, \quad \forall n \in \mathcal{A}_m, \tag{9e}$$

$$p^d_i = p^{d,nom}_i + \tilde{p}_i, \quad |\tilde{p}_i| \leq r \cdot p^{d,nom}_i, \quad \forall i \in \mathcal{N}, \tag{9f}$$

$$q^d_i = q^{d,nom}_i + \tilde{q}_i, \quad |\tilde{q}_i| \leq r \cdot q^{d,nom}_i, \quad \forall i \in \mathcal{N}, \tag{9g}$$

$$p^g_i = \dot{p}^g_{m,i}, \quad v_i = \dot{v}_{m,i}, \quad \forall i \in \mathcal{N}_g, \tag{9h}$$

$$v_i = \dot{v}_{m,i}, \quad \theta_i = \dot{\theta}_{m,i}, \quad \text{for } i \in \mathcal{S}, \tag{9i}$$

$\forall i \in \mathcal{N}, \forall (i,j) \in \mathcal{E}$:

$$p^g_i - p^d_i = \sum_{(i,j)\in\mathcal{E}} p_{ij} + g^{sh}_i v^2_i, \tag{9j}$$

$$q^g_i - q^d_i = \sum_{(i,j)\in\mathcal{E}} q_{ij} - b^{sh}_i v^2_i, \tag{9k}$$

$$p_{ij} = v^2_i G_{ij} - v_i v_j \big[G_{ij}\cos(\theta_{ij}) + B_{ij}\sin(\theta_{ij})\big], \tag{9l}$$

$$q_{ij} = -v^2_i (B^{sh}_{ij} + B_{ij}) - v_i v_j \big[G_{ij}\sin(\theta_{ij}) - B_{ij}\cos(\theta_{ij})\big], \tag{9m}$$

$$v_i \geq \underline{V}. \tag{9n}$$

Constraint (9a) ensures that the solution from each controller's region satisfies the power balance and line power flow constraints in that region. Constraints (9b)–(9d) are the voltage magnitude, reactive power injection, and line apparent power flow bounds imposed on variables in each region's OPF problem. Constraint (9e) ensures that the maximum boundary variable mismatch is not greater than ϵ^{pri} to model the controllers reaching their convergence tolerances. Constraints (9f)–(9g) set the amount by which loads may vary as described in Section IV-A.[1] Constraint (9h) sets the control values (active power injection and voltage magnitude variables for PV buses) to the setpoints from the distributed OPF solution. Constraint (9i) sets the slack bus voltage angle to 0 and the voltage magnitude to the result from the distributed OPF solution. Constraints (9j)–(9m) are the traditional AC power balance and line flow constraints for the system. These represent physical system behavior where the control values are set to the results of the distributed OPF computation. Constraint (9n) is designed to prevent the solver from finding a low-voltage solution to the AC power flow equations in (9i)–(9m) by providing a lower bound for the voltage magnitudes.[2]

We add an appropriate objective to (9) to find the worst-case violations of bounds on voltage magnitudes, reactive power injections, and line apparent power flows. For example, to compute the worst-case violation of the upper voltage limit at bus i for a given convergence tolerance of ϵ^{pri} and bound tightening values $\boldsymbol{\lambda}$, we first solve

$$\overline{v}^*_i = \max\, v_i \text{ subject to (9).}$$

The worst-case violation is then

$$W_{\overline{v}_i}(\epsilon^{pri}, \boldsymbol{\lambda}) = \overline{v}^*_i - \overline{V}_i.$$

Similarly, for the lower bound on voltage magnitude at bus i, we first solve

$$\underline{v}^*_i = \min\, v_i \text{ subject to (9).}$$

The worst-case violation of the lower voltage bound is then

$$W_{\underline{v}_i}(\epsilon^{pri}, \boldsymbol{\lambda}) = \underline{V}_i - \underline{v}^*_i.$$

Note that if we find $W_{\overline{v}_i}(\epsilon^{pri}, \boldsymbol{\lambda}), W_{\underline{v}_i}(\epsilon^{pri}, \boldsymbol{\lambda}) \leq 0$, then even in the worst case there is no violation of the bound constraint.

Similarly, we maximize and minimize the variable q^g_i at bus i to compute worst-case violations of reactive power generation limits. For worst-case violations of apparent power flow limits on line (i,j), we maximize $p^2_{ij} + q^2_{ij}$ and then compute $W_{\overline{s}_{ij}}(\epsilon^{pri}, \boldsymbol{\lambda}) = \sqrt{(p^*_{ij})^2 + (q^*_{ij})^2} - \overline{S}_{ij}$.

C. Discussion

The non-convex nature of the worst-case violation constraints means that a solver may find a local, rather than global, solution and thus not identify the actual largest possible violation. Alternatively, one could form a variant of (9) with relaxed AC power flow constraints [34]. The violation obtained by optimizing over a convex relaxation of the AC power flow equations will be equal to or greater than the actual largest possible violation. We chose to use the nonlinear AC power flow equations despite the possibility of local optima because problems constrained by convex relaxations may be slower to solve and may require careful implementation to ensure the relaxation is tight enough to avoid overly conservative bounds. We demonstrate via our results in Section V that although a nonlinear programming solver may occasionally return a local solution, we observe no violations in practice when running distributed OPF on test cases with bounds tightened using the formulation with AC power flow equations. This suggests that local solvers perform well for our purposes.

We also assume that there is at most one relevant solution to the AC power flow equations (9j)–(9m) for all power injections within the specified range. Although there may be many "low-voltage" solutions, typically there is only one "high-voltage" solution with near-nominal voltage magnitudes, and this high-voltage solution is the one we desire to find. We add constraint (9n) to screen out low-voltage power flow solutions. We note that the modeling challenges associated with nonconvexities and low-voltage power flow solutions are similar to those faced in stochastic and robust optimization problems; see [35, Section XI] for further discussion.

[1] Note that (9a)–(9e) enforce additional implicit constraints on the loads since some loading conditions within the variability allowed by r may not be feasible given each region's OPF constraints and the requirement that neighboring regions' shared variables agree to within a tolerance of ϵ^{pri}. If a loading condition is not feasible for (9), then it is not feasible for the original OPF problem (1), so it is acceptable for (9) to exclude these infeasible loading points.

[2] The value of $\underline{V}$ is chosen to be much lower than the lowest anticipated voltage (e.g., 0.7 per unit) so that the only effect of (9n) is avoiding a low-voltage power flow solution for (9j)–(9m).

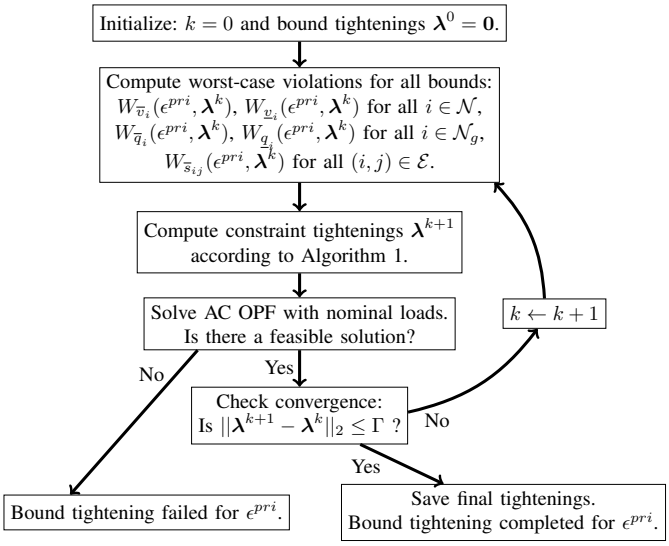


Fig. 4: Alternating algorithm for robust AC OPF problems

D. Bound Tightening

As demonstrated for a representative test case in Figure 2a, choosing a larger convergence tolerance ϵ^{pri} can dramatically decrease the number of iterations for the distributed optimization algorithm. However, larger tolerances may also result in constraint violations due to the inconsistency between neighboring regions' copies of boundary variables. We propose a method to tighten constraints such that the dispatch from the distributed OPF algorithm when converged to a given ϵ^{pri} is guaranteed not to violate the original constraints. Although the setting and application is different, our alternating algorithm is conceptually similar to those proposed in [36], [37], which use constraint tightening to make AC OPF problems robust to uncertainty in power demand or generation.

We now present the bound tightening algorithm. We show the steps of the algorithm in Figure 4. First, we initialize the tightening for each constraint to 0 by setting $\boldsymbol{\lambda}^0 = \mathbf{0}$. Second, we compute worst-case violations of all bounds. Third, we compute updated tightening values $\boldsymbol{\lambda}^k$ based on these violations as shown in Algorithm 1. Note that we use s as a generic variable index and observe that the update of λ_s follows the same logic for tightening of upper bounds $\lambda_{\overline{V}_i}$, $\lambda_{\overline{Q}_i}$, $\lambda_{\overline{S}_{ij}}$ and tightening of lower bounds $\lambda_{\underline{V}_i}$, $\lambda_{\underline{Q}_i}$. For a positive worst-case violation W_r, we increase the amount of tightening by W_r. We also check for unnecessary tightening: if the worst-case violation W_r is negative (that is, the variable is within its bound) and there has already been some tightening so that $\lambda_r > 0$, we reduce the amount of tightening by W_r or until $\lambda_r = 0$. Fourth, we solve an AC OPF problem on the system with nominal loads and bounds tightened by $\boldsymbol{\lambda}^{k+1}$ to make sure that the updated tightenings do not make the problem infeasible. Last, we evaluate the change in $\boldsymbol{\lambda}$ since the last iteration and return to Step 2 if this change is above a specified threshold Γ. Otherwise, the algorithm ends.

Algorithm 1 Updating constraint tightenings λ_s^k

if $W_s(\epsilon^{pri}, \boldsymbol{\lambda}^{k-1}) > 0$ **then**
  $\lambda_s^k = \lambda_s^{k-1} + W_s(\epsilon^{pri}, \boldsymbol{\lambda}^{k-1})$
else if $\lambda_s^{k-1} > 0$ **then**
  $\lambda_s^k = \lambda_s^{k-1} - \min\{-W_s(\epsilon^{pri}, \boldsymbol{\lambda}^{k-1}), \lambda_s^{k-1}\}$
end if

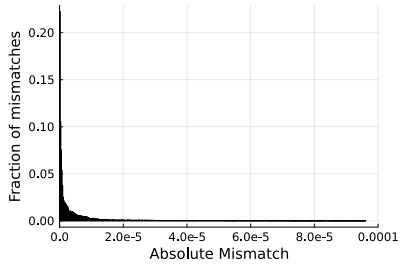


Fig. 5: Distribution of mismatches for case500 divided into eight regions, with distributed OPF converged to $\epsilon^{pri} = 10^{-4}$

E. Budget Uncertainty Set

In the formulation (9), every shared variable may reach the maximum possible mismatch ϵ^{pri}. However, in practice, controllers reach consensus on some boundary variables more quickly than others. When the algorithm converges with maximum mismatch below ϵ^{pri}, most mismatches are much smaller than ϵ^{pri}. We show a representative case in Figure 5 and observe that most of the mismatches are much smaller than the convergence tolerance of $\epsilon^{pri} = 10^{-4}$.

This motivates introducing the concept of budget uncertainty, which allows us to bound the total mismatch across the system and thus make less conservative predictions of the worst-case constraint violations. The budget uncertainty concept we use is similar to that used in [38], although our "uncertainty" regards the mismatch in shared variable values in a mathematical distributed optimization problem, rather than coming from renewable power fluctuations. To add the uncertainty budget to our problem, we choose the budget size β and augment (9) with the following constraint:

$$\sum_{(m,n)\in\mathcal{P}} \sum_{i\in\mathcal{I}_{m,n}} |z^i_{m,n} - z^i_{n,m}| \leq \beta N_b \epsilon^{pri} \qquad (10)$$

where $\mathcal{P}$ is the set of all neighboring controller pairs (m,n), and the set $\mathcal{I}_{m,n}$ contains indices for the specific variables shared between controllers m and n. Here, $z^i_{m,n}$ is controller m's copy of the i-th boundary variable shared between controllers m and n. The total number of boundary variables in the system is $N_b = \sum_{(m,n)\in\mathcal{P}} |\mathcal{I}_{m,n}|$.

Without adding (10), the constraints (9) allow for the total mismatch in the system, i.e., the sum of all boundary variable mismatches, to reach $N_b\epsilon^{pri}$, because every boundary variable can reach a mismatch of ϵ^{pri}. We add the bounds on the total mismatch in (10) so that the sum of absolute mismatches across the system is no more than a fraction β of $N_b\epsilon^{pri}$. Note that it is straightforward to reformulate (10) as a set of linear inequalities, which is how we implemented this constraint.

The choice of parameter β allows us to control the conservativeness of the constraint tightenings $\boldsymbol{\lambda}$. With $\beta < 1$, we cannot guarantee finding the true worst-case violations and thus the tightenings are not robust to all possible mismatches for

which the distributed optimization algorithm could terminate. Hence, the distributed OPF solution could violate constraints even after applying the bound tightening algorithm. However, choosing $\beta < 1$ allows the tightened bounds to be less conservative. This may lead to more optimal distributed OPF solutions. In addition, when the fully robust ($\beta = 1$) bound tightening algorithm leads to infeasibility of the resulting AC OPF problem, an appropriately selected uncertainty budget allows for less conservative bound tightening and may result in feasible AC OPF problems. Our empirical results in the following section indicate that β can be made fairly small in practice without introducing significant constraint violations. Thus, we can use larger convergence tolerances to substantially reduce the number of distributed OPF iterations, while achieving negligible constraint violations and only minor suboptimality compared to the OPF problem without tightened bounds.

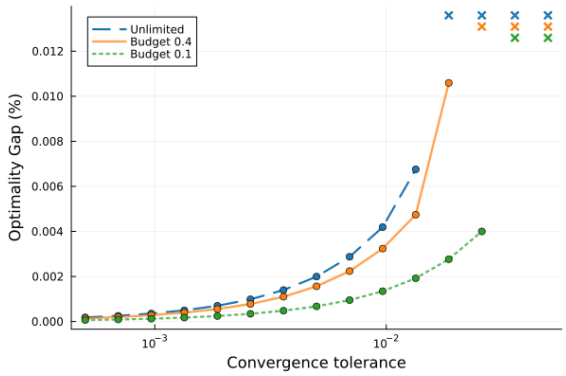
V. Numerical Results

We use Julia with the optimization modeling package JuMP [39] to formulate the worst-case violation optimization problems. We run distributed OPF to evaluate violations on the bound-tightened test cases using PowerModelsADA [33], and we run centralized OPF problems using PowerModels [40]. Our test cases are the case14, case118, and case500 test systems from the PGLib-OPF archive [32]. We divide case14 and case118 into 3 regions and case500 into 8 regions for distributed optimization.

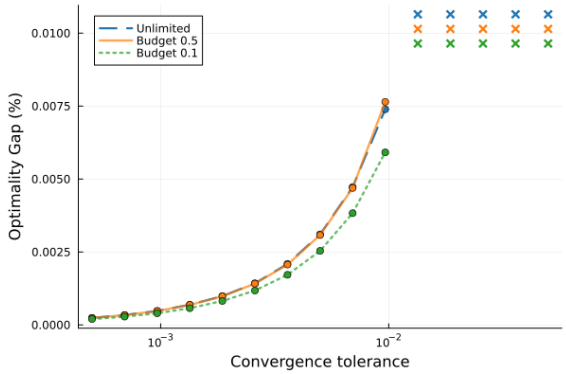
We first examine the relationship between convergence tolerance ϵ^{pri} and optimality of the bound-tightened cases. To do so, we sweep ϵ^{pri} across $[5 \times 10^{-4}, 5 \times 10^{-2}]$ for case14 and case118 and across $[10^{-6}, 10^{-4}]$ for case500. We choose smaller values of ϵ^{pri} for case500 because violations begin to appear with smaller ϵ^{pri} for this case. We run the bound tightening algorithm for each value of ϵ^{pri} and solve a centralized AC OPF problem on the bound-tightened test case with nominal loads. Figure 6 shows the cost percent difference for bound-tightened cases compared to original cases across multiple budgets β. We compute the cost percent difference as $(\tilde{f} - f^*)/f^*$, where $\tilde{f}$ is the AC OPF objective value for the bound-tightened case and f^* is the objective value for the original case. When ϵ^{pri} is sufficiently large, the bounds are tightened until the resulting test case is not feasible. We mark tolerances that result in infeasible test cases with $\times$.

As expected, for every test case, the amount of bound tightening increases with ϵ^{pri}, worsening solution suboptimality. However, the bound-tightened cases' costs are no more than 0.2% above optimal for all ϵ^{pri} at which the bounds can be tightened without causing AC OPF infeasibility. Decreasing the budget parameter allows for less conservative bound tightening, which may improve optimality very slightly (by less than 0.05% for our test cases). More significantly, using a smaller budget may result in feasible tightened cases for values of ϵ^{pri} at which tightening with a larger budget or no budget causes infeasibility; see, e.g., convergence tolerance values greater than 10^{-2} for the case14 in Figure 6a.

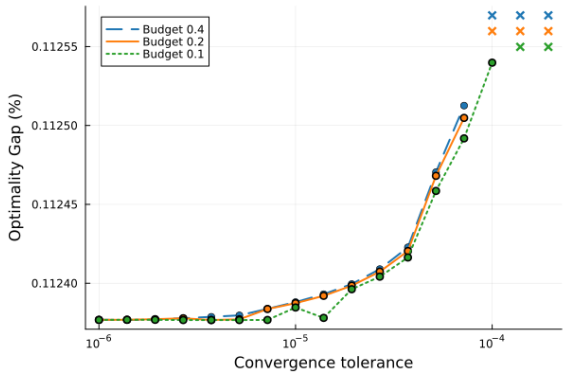
There is some unexpected behavior in these results: for case118 at $\epsilon^{pri} = 9.7 \times 10^{-3}$, the cost is slightly higher for a budget of $\beta = 0.5$ compared to an unlimited budget. This is



(a) case14



(b) case118



(c) case500

Fig. 6: Cost vs. convergence tolerance ϵ^{pri}. The $\times$ mark indicates a tolerance and budget for which OPF on the test case after bound tightening was infeasible.

because the bounds are tightened less for the unlimited budget due to an instance in which the solver for the unlimited budget found a local solution, rather than the true global optimum, to one of the optimization problems used to compute the bound tightenings. As described in Section IV-C, since we use the non-convex AC power flow equations in our optimization formulation, we cannot guarantee that the solver will find the global solution to these worst-case violation problems.

In addition to evaluating the optimality of bound-tightened cases, we also assess whether the bound tightening algorithm prevents constraint violations once the distributed OPF solution is applied to the system. We expect distributed OPF on cases tightened without any mismatch budget ($\beta = 1$) to have no constraint violations. Although we use local solutions to the

TABLE I: Violations vs. budget

Test Case (Tolerance)	Budget β	Median Number of Violations	Median Percent Violation
case14 (10^{-2})	0.01	1	0.484
	0.03	1	0.016
	0.10	0	0
case118 (10^{-2})	0.01	3	0.487
	0.03	1	0.029
	0.10	0	0
case500 (10^{-4})	0.01	6	1.04
	0.03	1	0.085
	0.10	0	0

non-convex worst-case violation problems, the results indicate that the bound tightening algorithm with the AC power flow formulation does not result in constraint violations once the distributed OPF solution is applied to the system. We do expect that when the mismatch budget becomes small enough, the constraints will not be tightened sufficiently. Thus, we may see that distributed OPF on test cases tightened with very small mismatch budgets result in constraint violations on the system.

To assess this, we run distributed OPF computations on cases tightened across a range of values for ϵ^{pri} and across a range of budgets. Each time we run distributed OPF, we vary the loads by up to 50% from nominal for case14 and case118 and by up to 30% from nominal for case500. The perturbation for each load is randomly selected from a uniform distribution across this range. Once the distributed OPF converges to a tolerance of ϵ^{pri}, we solve an AC power flow using control values from the distributed OPF solution. We record the average percent violation of any constraints violated and the total number of violations as described in Section III. The results are shown in Table III. For every test case, bound tightening with very small budgets ($\beta < 0.05$) may result in a few small violations, but tightening with a budget of at least $\beta = 0.1$ achieves negligible constraint violations.

One motivation for bound tightening is to reduce the number of iterations to convergence. Bound tightening allows us to increase ϵ^{pri} without risking constraint violations once the distributed OPF solution is applied to the grid. We showed an example of the impacts of increasing ϵ^{pri} in Section III for case500. Here, we show in Table II the median percent reduction in iterations to convergence when we increase ϵ^{pri} to the maximum value at which we can feasibly tighten bounds. Just as in Section III, we run distributed OPF repeatedly, perturbing the loads each time by up to 50% for case14 and case118 and by up to 30% for case500. For each case, we find ϵ^{pri}_{orig}, the greatest value of ϵ^{pri} which results in no violations under distributed OPF for the original test case, which is 5×10^{-4} for case14 and case118, and 5×10^{-6} for case500. Then, we find ϵ^{pri}_{tight}, the greatest value of ϵ^{pri} for which we can feasibly run a bound tightening algorithm with a budget of 10% ($\beta = 0.1$) or higher, which is 10^{-2} for case14 and case118 and 10^{-4} for case500. We measure the percent reduction as $(\nu_{\epsilon^{pri}_{orig}} - \nu_{\epsilon^{pri}_{tight}})/\nu_{\epsilon^{pri}_{orig}}$, where $\nu_{\epsilon^{pri}}$ is the median number of iterations required to converge to a tolerance of ϵ^{pri} in our experiments. That is, the percent reduction in iterations in Table II indicates the amount by which bound tightening allows us to decrease the number of iterations (by increasing ϵ^{pri}) without resulting in constraint violations. For these representative test cases, bound tightening can reduce the number of iterations by over 35% without increasing the cost by more than 0.2%.

TABLE II: Reduction in Iterations

Test case	case14	case118	case500
Reduction in Iterations	53.9%	85.2%	36.9%

We also provide a brief discussion on computation time. While collecting these results, we ran the bound tightening algorithm on the test cases for many different values of ϵ^{pri} and for several different budgets. We record in Table III the minimum, median and maximum times required to run the bound tightening algorithm on each test case. We ran the experiments on Georgia Tech's PACE cluster, where each node had a 16-core 2.7 GHz processor and 64 GB RAM. Recall that all bound tightening occurs offline. To speed up offline bound tightening, we parallelize the computation of worst-case bound violations and adaptively determine which bounds are at risk for violations to reduce the number of problems to be solved.

During real-time operation, when running distributed OPF on a bound-tightened test case, the computation time for solving ADMM subproblems at each iteration is no different from the computation time for subproblems on the original test case. However, a bound-tightened test case allows for selecting a larger convergence tolerance, resulting in fewer iterations required to converge, without risking constraint violations.

TABLE III: Bound Tightening Time in Minutes

Test case	Minimum	Median	Maximum
case14	0.17	0.18	0.20
case118	1.75	2.25	2.49
case500	66.2	120.8	229.7

VI. Conclusion

Distributed optimization algorithms provide several advantages, including scalability, flexibility, and privacy, for operating power systems with widespread distributed energy resources. Such algorithms require separate computing controllers to reach consensus, up to some convergence tolerance, on the values of shared boundary variable values. Increasing the convergence tolerance generally reduces the number of iterations to convergence, which is a key challenge for distributed algorithms, but may also lead to constraint violations with respect to the original problem. In this paper, we first formulate an optimization problem which finds the worst-case constraint violations that result from applying a distributed OPF solution converged to a given tolerance to the power system. Next, we propose a bound tightening algorithm which, provided that global solutions are found for worst-case violation problems, guarantees that the distributed OPF solution will not cause constraint violations on the real power system. We also introduce a "budget uncertainty" method to bound cumulative boundary variable mismatches in the worst-case violation problem, allowing for less conservative bound tightening. Our numerical results demonstrate that the bound

tightening algorithm increases suboptimality only slightly, while allowing for a significant reduction in distributed OPF iterations without causing constraint violations.

For sufficiently large convergence tolerances, the algorithm tightens bounds to the point that OPF is no longer feasible. Our future work is to increase the range of convergence tolerances for which the bound tightening algorithm maintains OPF feasibility. To do so, we plan to analyze distributions of boundary variable mismatches, explore chance-constrained variants of the worst-case violation problems, and leverage the optimality of solutions to regions' OPF subproblems, which may yield less conservative worst-case violations.

REFERENCES

[1] D. K. Molzahn, F. Dörfler, H. Sandberg, *et al.*, "A survey of distributed optimization and control algorithms for electric power systems," *IEEE Transactions on Smart Grid*, vol. 8, no. 6, pp. 2941–2962, 2017.

[2] N. Huebner, Y. Rink, M. Suriyah, and T. Leibfried, "Distributed AC-DC optimal power flow in the European transmission grid with ADMM," in *55th International Universities Power Engineering Conference (UPEC)*, 2020.

[3] A. Garcia, R. Khatami, C. Eksin, and F. Sezer, "An incentive compatible iterative mechanism for coupling electricity markets," *IEEE Transactions on Power Systems*, vol. 37, no. 2, pp. 1241–1252, 2022.

[4] L. Yang, J. Luo, Y. Xu, Z. Zhang, and Z. Dong, "A distributed dual consensus ADMM based on partition for DC-DOPF with carbon emission trading," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 1858–1872, 2020.

[5] A. R. Malekpour, A. Pahwa, and B. Natarajan, "Hierarchical architecture for integration of rooftop PV in smart distribution systems," *IEEE Transactions on Smart Grid*, vol. 9, no. 3, pp. 2019–2029, 2018.

[6] T. AlSkaif and G. van Leeuwen, "Decentralized optimal power flow in distribution networks using blockchain," in *International Conference on Smart Energy Systems and Technologies (SEST)*, 2019.

[7] B. Wang, C. Zhang, C. Li, G. Yang, and Z. Y. Dong, "Transactive energy sharing in a microgrid via an enhanced distributed adaptive robust optimization approach," *IEEE Transactions on Smart Grid*, vol. 13, no. 3, pp. 2279–2293, 2022.

[8] Y. Kim and K. Kim, "Accelerated computation and tracking of AC optimal power flow solutions using GPUs," in *51st International Conference on Parallel Processing*, New York, NY, USA: Association for Computing Machinery, 2023.

[9] Y. Kim, F. Pacaud, K. Kim, and M. Anitescu, "Leveraging GPU batching for scalable nonlinear programming through massive Lagrangian decomposition," 2021, arXiv:2106.14995.

[10] K. Sun and X. A. Sun, "A two-level ADMM algorithm for AC OPF with global convergence guarantees," *IEEE Transactions on Power Systems*, vol. 36, no. 6, pp. 5271–5281, 2021.

[11] J. Guo, G. Hug, and O. K. Tonguz, "A case for nonconvex distributed optimization in large-scale power systems," *IEEE Transactions on Power Systems*, vol. 32, no. 5, pp. 3842–3851, 2017.

[12] J. Xu, H. Sun, and C. J. Dent, "ADMM-based distributed OPF problem meets stochastic communication delay," *IEEE Transactions on Smart Grid*, vol. 10, no. 5, pp. 5046–5056, 2019. DOI: 10.1109/TSG.2018.2873650.

[13] A. Mohammadi and A. Kargarian, "Learning-aided asynchronous ADMM for optimal power flow," *IEEE Transactions on Power Systems*, vol. 37, no. 3, pp. 1671–1681, 2022. DOI: 10.1109/TPWRS.2021.3120260.

[14] Z. Cheng and M.-Y. Chow, "Resilient collaborative distributed AC optimal power flow against false data injection attacks: A theoretical framework," *IEEE Transactions on Smart Grid*, vol. 13, no. 1, pp. 795–806, 2022. DOI: 10.1109/TSG.2021.3113287.

[15] J. Duan, W. Zeng, and M.-Y. Chow, "Resilient distributed DC optimal power flow against data integrity attack," *IEEE Transactions on Smart Grid*, vol. 9, no. 4, pp. 3543–3552, 2018. DOI: 10.1109/TSG.2016.2633943.

[16] R. Harris and D. K. Molzahn, "Detecting and mitigating data integrity attacks on distributed algorithms for optimal power flow using machine learning," in *57th Hawaii International Conference on System Sciences (HICSS)*, 2024.

[17] Y. Wang, S. Wang, and L. Wu, "Distributed optimization approaches for emerging power systems operation: A review," *Electric Power Systems Research*, vol. 144, pp. 127–135, 2017.

[18] A. Kargarian, J. Mohammadi, J. Guo, *et al.*, "Toward distributed/decentralized DC optimal power flow implementation in future electric power systems," *IEEE Transactions on Smart Grid*, vol. 9, no. 4, pp. 2574–2594, 2018.

[19] S. Mhanna, G. Verbič, and A. C. Chapman, "Adaptive ADMM for distributed AC optimal power flow," *IEEE Transactions on Power Systems*, vol. 34, no. 3, pp. 2025–2035, 2019.

[20] A. Mohammadi and A. Kargarian, "Accelerated and robust analytical target cascading for distributed optimal power flow," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 12, pp. 7521–7531, 2020.

[21] S. Zeng, A. Kody, Y. Kim, K. Kim, and D. K. Molzahn, "A reinforcement learning approach to parameter selection for distributed optimization in power systems," *Electric Power Systems Research*, vol. 212, p. 108 546, 2022, presented at the *22nd Power Systems Computation Conference (PSCC 2022)*.

[22] D. Biagioni, P. Graf, X. Zhang, A. S. Zamzam, K. Baker, and J. King, "Learning-accelerated ADMM for distributed DC optimal power flow," in *American Control Conference (ACC)*, 2021, pp. 576–581.

[23] T. W. K. Mak, M. Chatzos, M. Tanneau, and P. V. Hentenryck, "Learning regionally decentralized ac optimal power flows with admm," *IEEE Transactions on Smart Grid*, vol. 14, no. 6, pp. 4863–4876, 2023.

[24] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[25] Y. Wang, L. Wu, and S. Wang, "A fully-decentralized consensus-based ADMM approach for DC-OPF with demand response," *IEEE Transactions on Smart Grid*, vol. 8, no. 6, pp. 2637–2647, 2017.

[26] T. Erseghe, "Distributed optimal power flow using ADMM," *IEEE Transactions on Power Systems*, vol. 29, no. 5, pp. 2370–2380, 2014.

[27] Q. Peng and S. H. Low, "Distributed optimal power flow algorithm for radial networks, I: Balanced single phase case," *IEEE Transactions on Smart Grid*, vol. 9, no. 1, pp. 111–121, 2018.

[28] B. A. Robbins and A. D. Domínguez-García, "Optimal reactive power dispatch for voltage regulation in unbalanced distribution systems," *IEEE Transactions on Power Systems*, vol. 31, no. 4, pp. 2903–2913, 2016.

[29] J. Li, C. Zhang, Z. Xu, J. Wang, J. Zhao, and Y.-J. A. Zhang, "Distributed transactive energy trading framework in distribution networks," *IEEE Transactions on Power Systems*, vol. 33, no. 6, pp. 7215–7227, 2018.

[30] R. Baldick, B. Kim, C. Chase, and Y. Luo, "A fast distributed implementation of optimal power flow," *IEEE Transactions on Power Systems*, vol. 14, no. 3, pp. 858–864, 1999.

[31] M. Alkhraijah, C. Menendez, and D. K. Molzahn, "Assessing the impacts of nonideal communications on distributed optimal power flow algorithms," *Electric Power Systems Research*, vol. 212, p. 108 297, 2022, presented at the *22nd Power Systems Computation Conference (PSCC 2022)*.

[32] IEEE PES PGLib-OPF Task Force, "The power grid library for benchmarking AC optimal power flow algorithms," 2019, arXiv:1908.02788.

[33] M. Alkhraijah, R. Harris, C. Coffrin, and D. K. Molzahn, "Powermodelsada: A framework for solving optimal power flow using distributed algorithms," *IEEE Transactions on Power Systems*, vol. 39, no. 1, pp. 2357–2360, 2024.

[34] D. K. Molzahn and I. A. Hiskens, "A survey of relaxations and approximations of the power flow equations," *Foundations and Trends® in Electric Energy Systems*, vol. 4, no. 1-2, pp. 1–221, 2019.

[35] 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," *Electric Power Systems Research*, vol. 214, no. 108725, 2023, presented at the 22nd Power Systems Computation Conference (PSCC 2022).

[36] L. Roald and G. Andersson, "Chance-constrained AC optimal power flow: Reformulations and efficient algorithms," *IEEE Transactions on Power Systems*, vol. 33, no. 3, pp. 2906–2918, 2018.

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

[38] Á. Lorca and X. A. Sun, "The adaptive robust multi-period alternating current optimal power flow problem," *IEEE Transactions on Power Systems*, vol. 33, no. 2, pp. 1993–2003, 2018.

[39] I. Dunning, J. Huchette, and M. Lubin, "JuMP: A modeling language for mathematical optimization," *SIAM Review*, vol. 59, no. 2, 2017.

[40] 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 Systems Computation Conference (PSCC)*, 2018.