

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

Rachel Harris, Mohannad Alkhrajah, and Daniel K. Molzahn

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 computing agents 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 generator dispatch 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. The proposed approach enables control of the trade-off between computational speed, which improves as the convergence tolerance increases, and distributed OPF solution cost, which increases with convergence tolerance due to tightened constraints, while ensuring feasibility.

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 keep local information private, and avoid a single point of failure.

Many researchers have applied distributed algorithms to the optimal power flow (OPF) problem. Commonly used distributed algorithms include the alternating direction method of multipliers (ADMM) [2], [3], analytical target cascading (ATC) [4], [5], and auxiliary problem principle (APP) [6], [7]. Distributed algorithms decompose the system into separate

regions, each under the control of different local computing agents. These agents 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 [8] 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.

Many distributed algorithms take thousands of iterations to converge for large-scale systems [9]–[11]. To use such distributed algorithms to operate future power grids with many rapidly fluctuating DERs, we must reduce distributed OPF computation time. To accelerate distributed algorithms, some researchers have proposed adaptive parameter tuning [12]–[14] and using machine learning to predict the converged boundary variable values [15], [16]. 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 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 ADMM distributed algorithm, but our method can be applied without any conceptual changes to other power flow formulations or distributed algorithms. We formulate an optimization problem which, given some convergence tolerance, finds the maximum possible violation for each constraint at the power system operating point under distributed OPF dispatch. Next, we propose an algorithm which iterates between finding these maximum violations and updating bound tightenings to guarantee that distributed OPF with these tightened bounds will not result in any constraint violations. We present numerical results from several representative test cases, showing 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 as well as feasibility and optimality of

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

the final solution. 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 also 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 with the given convergence 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 performance subject to operational limits and physical power flow equations. In this paper, we consider OPF formulations with an AC power flow model and an objective that minimizes generation cost as a quadratic function of real power output from each generator. However, alternative power flow formulations or different objectives could also be used.

Let \mathcal{N} , \mathcal{E} , and \mathcal{G} denote the sets of buses, lines, and generators, respectively. The OPF formulation is

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

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

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

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

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

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

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

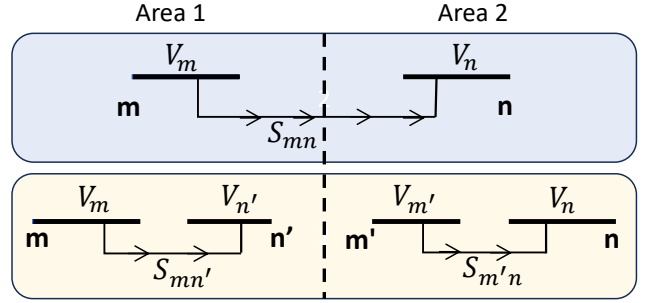
$$\underline{P}_i^g \leq p_i^g \leq \overline{P}_i^g, \quad (1g)$$

$$\underline{Q}_i^g \leq q_i^g \leq \overline{Q}_i^g, \quad (1h)$$

$$\underline{V}_i \leq v_i \leq \overline{V}_i, \quad (1i)$$

$$p_{ij}^2 + q_{ij}^2 \leq (\overline{S}_{ij})^2, \quad (1j)$$

where f_i is the cost function and p_i^g, q_i^g are the real and reactive power outputs, respectively, of generator $g \in \mathcal{G}$ located at bus $i \in \mathcal{N}$. We define $\theta_{ij} = \theta_i - \theta_j$ for $(i, j) \in \mathcal{E}$. The series conductance and susceptance of line $(i, j) \in \mathcal{E}$ are G_{ij} and B_{ij} , while \overline{S}_{ij} denotes the line's thermal limit. The shunt conductance and susceptance at bus $i \in \mathcal{N}$ are



Constraints:

$$V_m = V_{m'}, V_n = V_{n'}, S_{mn'} = S_{m'n}$$

Fig. 1: Decomposition of power network

g_i^{sh} and b_i^{sh} . Each bus $i \in \mathcal{N}$ has a voltage phasor $v_i \angle \theta_i$. We denote the real power demand at bus $i \in \mathcal{N}$ as p_i^d and reactive power demand as q_i^d . Also, \mathcal{S} contains the reference bus. The OPF problem minimizes the generation cost in (1a) subject to the AC power flow equations (1c)–(1f), the voltage limits and generators' power output limits (1g)–(1i), and the lines' thermal limits (1j). Note also that we set the phase angle to 0 at a selected reference bus in (1b).

B. Distributed Optimal Power Flow

In the distributed OPF formulation, the power network is divided into regions, each under the control of a separate computing agent. 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, separate computing agents 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 agents. Agents 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. Let \mathcal{G}_m , \mathcal{N}_m , and \mathcal{E}_m denote the sets of generators, buses, and lines in region m , respectively. We denote the set of shared variables in region m with \mathcal{N}_m^s . In addition, we use the same notation for variables as in (1) but add dots to designate agents' 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 consistency constraints are relaxed with the augmented Lagrangian technique. The vector \mathbf{z}_m contains all shared variables in \mathcal{N}_m^s , and the vector $\bar{\mathbf{z}}_m$ is a "central" variable which accounts for all neighbors' copies of the shared variables. In traditional ADMM, this variable is computed by a central coordinator, but for our formulation it simplifies to the average of the neighboring agents' shared variable values and

is thus entirely separable, as in [17]. At iteration k , agent m solves the following subproblem:

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

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

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

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

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

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

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

$$\underline{P}_i^g \leq \dot{\mathbf{p}}_{i,m}^{g,k} \leq \bar{P}_i^g, \quad (2g)$$

$$\underline{Q}_i^g \leq \dot{\mathbf{q}}_{i,m}^{g,k} \leq \bar{Q}_i^g, \quad (2h)$$

$$\underline{V}_i \leq \dot{\mathbf{v}}_{i,m}^k \leq \bar{V}_i, \quad (2i)$$

$$(\dot{\mathbf{p}}_{ij,m}^k)^2 + (\dot{\mathbf{q}}_{ij,m}^k)^2 \leq (\bar{S}_{ij})^2. \quad (2j)$$

Note that α is a user-defined penalty parameter. After solving (2), each agent m shares the boundary variable values \mathbf{z}_m with their neighbors. Then, each agent m updates the $\bar{\mathbf{z}}_m$ variables. For every neighbor n of agent m , there is a set of variables $\mathcal{N}_{m,n}^s$ that are shared between agents m and n . We denote agent m 's copies of these shared variables as the vector $\mathbf{z}_{m,n}$ and agent n 's copies of these shared variables as the vector $\mathbf{z}_{n,m}$. Agent m updates the average of local shared variables and shared variables received from neighbor n , $\bar{\mathbf{z}}_{m,n}$, as

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

Each agent m updates their Lagrange multipliers as

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

Thus, the iterative algorithm alternates between minimizing the agents' subproblems in (2), updating the average copies of variables shared between agents in (3), and updating dual variables in (4). Typically, the stopping criterion is based on primal and dual residuals [8]. The vector of primal residuals \mathbf{r}^k contains the difference between local and central copies of all boundary variable values:

$$\mathbf{r}^k = [\mathbf{z}_1^T - \bar{\mathbf{z}}_1^T \quad \mathbf{z}_2^T - \bar{\mathbf{z}}_2^T \quad \dots \quad \mathbf{z}_M^T - \bar{\mathbf{z}}_M^T]^T. \quad (5)$$

The dual residual is

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

where we have collected all central copies of boundary variables into one vector $\bar{\mathbf{z}}$. The algorithm terminates when the

primal and dual residual norms fall below the respective primal and dual tolerances:

$$\|\mathbf{r}^k\| \leq \epsilon^{pri}, \quad \|\mathbf{s}^k\| \leq \epsilon^{dual}. \quad (7)$$

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, [8], 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 [2], [12], [14], [18]. Other papers use the ℓ_∞ - or ℓ_2 -norm of the dual residuals only [19], [20], while yet other publications use the ℓ_∞ - or ℓ_2 -norm of the primal residuals [9], [21], [22]. Most of the above works select a tolerance in the range of $[10^{-5}, 10^{-3}]$, although [8] proposes a method to define tolerances based on the scale of the variables:

$$\epsilon^{pri} = \sqrt{p} \epsilon^{abs} + \epsilon^{rel} \max\{\|\mathbf{A}\mathbf{x}^k\|_2, \|\mathbf{B}\mathbf{z}^k\|_2, \|\mathbf{c}\|_2\}, \quad (8)$$

$$\epsilon^{dual} = \sqrt{n} \epsilon^{abs} + \epsilon^{rel} \|\mathbf{A}^T \mathbf{y}^k\|_2,$$

where ϵ^{abs} , ϵ^{rel} are user-selected absolute and relative tolerances, respectively, and the notation is for a general ADMM formulation which minimizes a function $f(\mathbf{x}) + g(\mathbf{z})$ subject to the coupling constraint $\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c}$. In (8), p is the number of constraints and n is the number of shared variables.

Our analysis will focus on ϵ^{pri} , and we will determine convergence based on the primal residuals alone. The requirement for small primal residuals, $\|\mathbf{r}^k\| \leq \epsilon^{pri}$, results in near feasibility of the final solution by satisfying consistency constraints. The requirement for small dual residuals, $\|\mathbf{s}^k\| \leq \epsilon^{dual}$, 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 the 500-bus test case from the PGLib-OPF archive [23] divided into 8 regions for distributed optimization. We use the PowerModelsADA library [24] 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 distributed OPF generator dispatch and 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

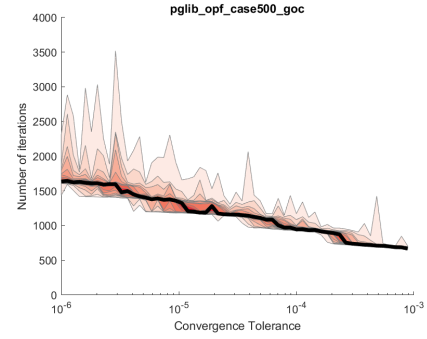
$$\frac{1}{N_v} \sum_{i \in \mathcal{C}} \frac{\max\{x_i^{AC-PF} - x_i^{max}, x_i^{min} - x_i^{AC-PF}, 0\}}{x_i^{max} - x_i^{min}},$$

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 $\epsilon^{pri} = 4 \times 10^{-5}$.

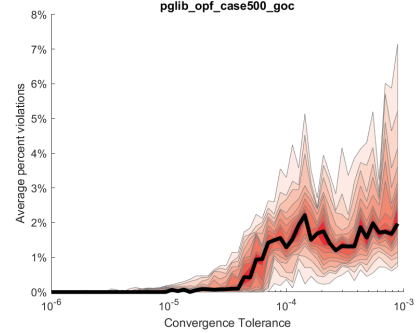
IV. ANALYSIS AND BOUND TIGHTENING ALGORITHM

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.

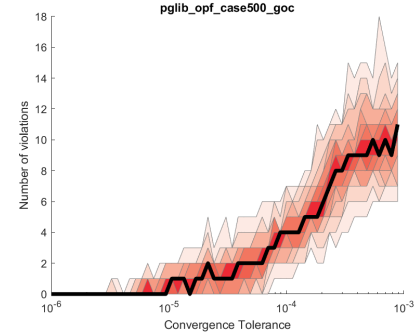
The worst-case violation analysis and constraint-tightening algorithm is useful for OPF problems solved repeatedly, with a constant network model and loads varying with each run. The proposed algorithm requires the ability to perform offline calculations where information regarding the entire system is available. Offline, we formulate an optimization problem



(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

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 consider a network model with buses in \mathcal{N} , generators in \mathcal{G} and lines in \mathcal{E} . We use the same notation for system variables as in Section II-A. Note that while the active and reactive power demands p_i^d and q_i^d at each bus i are fixed for the OPF formulation (1) in Section II-A, this worst-case violation problem has the power demands as variables. We allow the

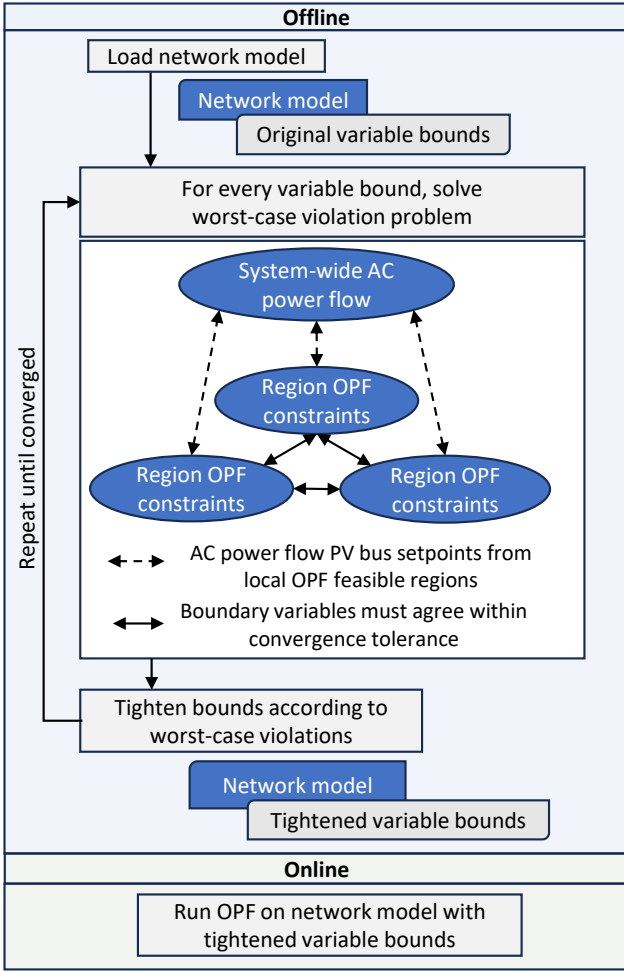


Fig. 3: Bound tightening algorithm overview

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 agents 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 agents controlling regions in the distributed OPF problem and \mathcal{A}_m denotes the neighbors of agent m . Also, the vector $\mathbf{z}_{m,n}$ contains agent m 's copies of all boundary variables shared between agents 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 λ_{V_i} , $\lambda_{\bar{V}_i}$ for lower and upper bounds on the voltage at bus i , λ_{Q_i} , $\lambda_{\bar{Q}_i}$ for lower and upper bounds on reactive power generation at bus i , and

$\lambda_{\bar{S}_{ij}}$ for the upper bound on apparent power flow across line (i, j) . For instance, with a constraint tightening of $\lambda_{\bar{V}_i}$, the upper voltage limit (2i) in an agent's subproblem becomes $\dot{v}_{i,m}^k \leq \bar{V}_i - \lambda_{\bar{V}_i}$. We collect the amount of bound tightening on all variables into one vector λ .

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 λ . Therefore, we denote the worst-case violations on upper and lower bounds on voltage magnitudes at bus i as $\bar{W}_{v_i}(\epsilon^{pri}, \lambda)$ and $W_{v_i}(\epsilon^{pri}, \lambda)$, respectively; on upper and lower bounds on reactive power generation at bus i as $\bar{W}_{q_i}(\epsilon^{pri}, \lambda)$ and $W_{q_i}(\epsilon^{pri}, \lambda)$, respectively; and on upper bounds on line apparent power flows at line (i, j) as $\bar{W}_{s_{ij}}(\epsilon^{pri}, \lambda)$. We next describe an optimization formulation for calculating the worst-case constraint violations for a given convergence tolerance ϵ^{pri} and bound tightening λ .

B. Worst-Case Violation Formulation

We next 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 the behavior of the distributed OPF algorithm. The variables kept by local regions (recall that these are 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:

$$\forall m \in \mathcal{M} :$$

$$(2c)-(2g) \tag{9a}$$

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

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

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

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

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

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

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

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

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

$$p_i^g - p_i^d = \sum_{(i,j) \in \mathcal{E}} p_{ij} + g_i^{sh} v_i^2, \tag{9j}$$

$$q_i^g - q_i^d = \sum_{(i,j) \in \mathcal{E}} q_{ij} - b_i^{sh} v_i^2, \quad (9k)$$

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

$$q_{ij} = -v_i^2 (B_{ij}^{sh} + B_{ij}) - v_i v_j [G_{ij} \sin(\theta_{ij}) - B_{ij} \cos(\theta_{ij})], \quad (9m)$$

$$v_i \geq \underline{V}. \quad (9n)$$

Constraint (9a) ensures that the solution from each agent'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 agents 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 system 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 under the distributed OPF dispatch, where the active power injection and voltage magnitudes at PV buses 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 λ , we first solve

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

The worst-case violation is then

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

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

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

¹Note 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.

²The 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).

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

Similarly, we maximize and minimize the variable q_i^g 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 $p_{ij}^2 + q_{ij}^2$ and then compute $W_{\bar{s}_{ij}}(\epsilon^{pri}, \lambda) = \sqrt{(p_{ij}^*)^2 + (q_{ij}^*)^2} - \bar{S}_{ij}$.

C. Discussion

Our formulation neglects the fact that the generator dispatch $\hat{p}_{m,i}^g, \hat{v}_{m,i}$ from the distributed OPF would optimally solve (2) for each agent m . Instead of requiring that the distributed OPF variables are *optimal* for their region's OPF problem with relaxed consistency constraints, we require only that the distributed OPF variables are *feasible* for the region's OPF problem. Thus, our formulation may be conservative, i.e., return worst-case violations larger than would actually be produced by the distributed optimization algorithm. An alternative formulation that enforces optimality of each region's OPF problems would lead to a computationally challenging bilevel problem. Our future work includes leveraging the optimality of $\hat{p}_{m,i}^g, \hat{v}_{m,i}$ to find less conservative worst-case violations.

We also note that 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 [25]. 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 [26, Section XI] for further discussion.

D. Bound Tightening

As demonstrated for a representative test case in Figure 2a, choosing a larger convergence tolerance ϵ^{pri} can dramati-

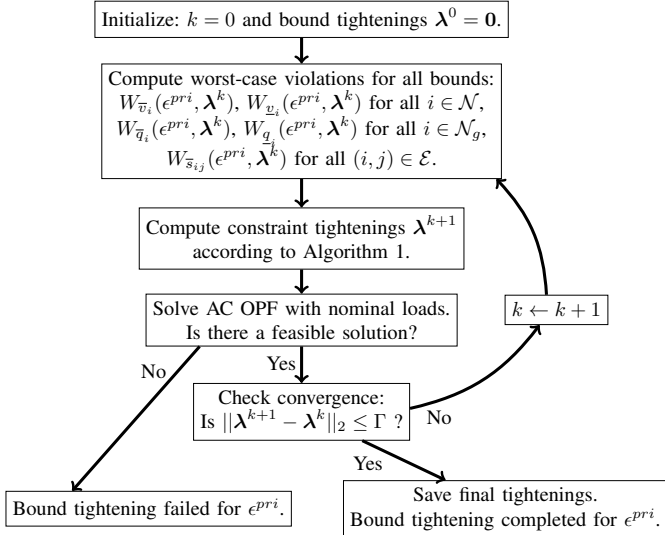


Fig. 4: Alternating algorithm for robust AC OPF problems

cally 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 [27], [28], 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 $\lambda^0 = \mathbf{0}$. Second, we compute worst-case violations of all bounds. Third, we compute updated tightening values λ^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 $\lambda_{\bar{v}_i}$, $\lambda_{\bar{q}_i}$, $\lambda_{\bar{s}_{ij}}$ and tightening of lower bounds λ_{v_i} , λ_{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 λ^{k+1} to make sure that the updated tightenings do not make the

Algorithm 1 Updating constraint tightenings λ_s^k

```

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

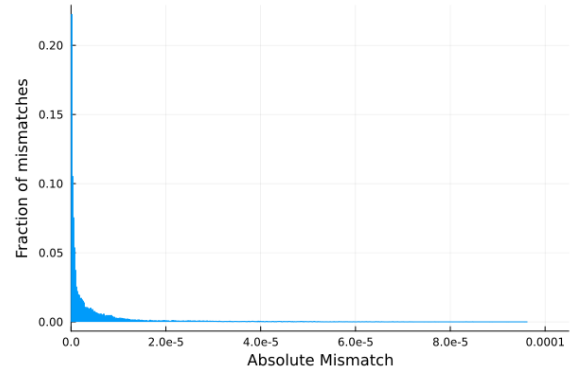


Fig. 5: Distribution of mismatches for case500 divided into eight regions, with distributed OPF converged to $\epsilon^{pri} = 10^{-4}$

problem infeasible. Last, we evaluate the change in λ since the last iteration (as measured by the 2-norm) and return to Step 2 if this change is above a specified threshold Γ . Otherwise, the algorithm ends.

E. Budget Uncertainty Set

In the formulation (9), we allow every shared variable to take on its maximum possible mismatch ϵ^{pri} . However, in practice, agents reach consensus on some boundary variables more quickly than others. By the time the algorithm converges with a maximum mismatch below ϵ^{pri} , many of the other 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 [29], 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:

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

where \mathcal{P} is the set of all neighboring agent pairs (m, n) , and the set $\mathcal{I}_{m,n}$ contains indices for the specific variables shared between agents m and n . Here, $z_{m,n}^i$ is agent m 's copy of the i -th boundary variable shared between agents 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 λ . 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 [30] to formulate the worst-case violation optimization problems. We run distributed OPF to evaluate violations on the bound-tightened test cases using PowerModelsADA [24], and we run centralized OPF problems using PowerModels [31]. Our test cases are the 14-bus, 118-bus and 500-bus network models selected from the PGLib-OPF archive [23]. We divide the 14-bus and 118-bus cases into 3 regions and the 500-bus case 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 the suboptimality of the solution. 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

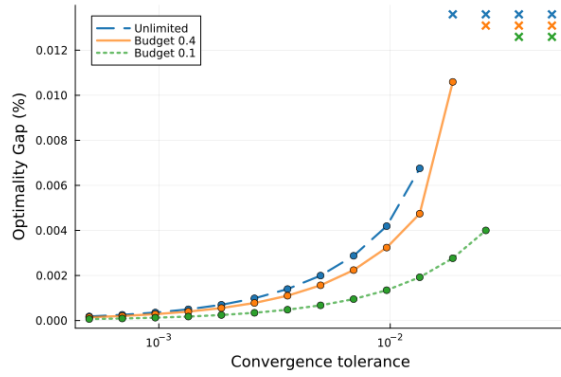
very slightly (by less than 0.05% for our test cases). More significantly, using a smaller budget can sometimes result in feasible tightened cases for values of ϵ^{pri} at which tightening with a larger budget or no budget causes infeasibility; see, e.g., convergence tolerance values greater than 10^{-2} for the 14-bus case 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 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.

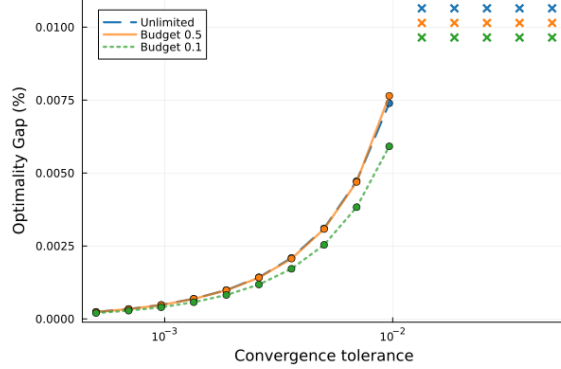
In addition to evaluating the optimality of bound-tightened cases, we also assess whether the bound tightening algorithm prevents constraint violations once the distributed OPF solution is applied to the system. We expect distributed OPF on cases tightened without any mismatch budget ($\beta = 1$) to have no constraint violations. As mentioned above, there is one caveat: due to the non-convex nature of the AC power flow equations in the worst-case violation problems, the solver may find local solutions. However, we find that in practice the bound tightening algorithm using 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 the generator dispatch 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.

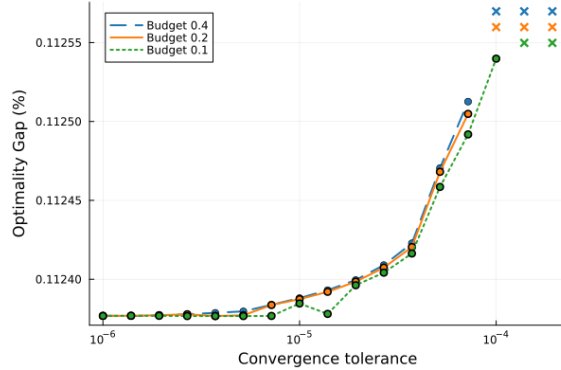
We note that one motivation for bound tightening is to reduce the number of iterations to convergence. Bound tightening allows us to increase ϵ^{pri} without risking constraint violations once the distributed OPF solution is applied to the grid. We showed an example of the impacts of increasing ϵ^{pri} 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



(a) case14



(b) case118



(c) case500

Fig. 6: Cost vs. convergence tolerance ϵ^{pri}

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_{orig}^{pri} , 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_{tight}^{pri} , 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 $(\nu_{\epsilon_{orig}^{pri}} - \nu_{\epsilon_{tight}^{pri}}) / \nu_{\epsilon_{orig}^{pri}}$, 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 decrease the number of iterations (by

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
	0.06	1	0.0006
	0.10	0	0
case118 (10^{-2})	0.01	3	0.487
	0.03	1	0.029
	0.06	0	0
	0.10	0	0
case500 (10^{-4})	0.01	6	1.04
	0.03	1	0.085
	0.06	0	0.0001
	0.10	0	0

TABLE II: Reduction in Iterations

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

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%.

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. For example, any bound not violated for the original test case will not be violated after other bounds are tightened and can thus be ignored after the first iteration of the bound tightening.

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 agents 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] T. Erseghe, "Distributed optimal power flow using ADMM," *IEEE Transactions on Power Systems*, vol. 29, no. 5, pp. 2370–2380, 2014.
- [3] A. X. Sun, D. T. Phan, and S. Ghosh, "Fully decentralized AC optimal power flow algorithms," in *IEEE Power & Energy Society General Meeting*, 2013.
- [4] 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.
- [5] A. K. Marvasti, Y. Fu, S. DorMohammadi, and M. Rais-Rohani, "Optimal operation of active distribution grids: A system of systems framework," *IEEE Transactions on Smart Grid*, vol. 5, no. 3, pp. 1228–1237, 2014.
- [6] D. Hur, J. Park, and B. Kim, "Evaluation of convergence rate in the auxiliary problem principle for distributed optimal power flow," *IEEE Proceedings-Generation, Transmission and Distribution*, no. 5, pp. 525–532, 2002.
- [7] L. Cao, Y. Sun, X. Cheng, B. Qi, and Q. Li, "Research on the convergent performance of the auxiliary problem principle based distributed and parallel optimization algorithm," in *IEEE International Conference on Automation and Logistics*, 2007, pp. 1083–1088.
- [8] 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.
- [9] 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.
- [10] 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.
- [11] 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.
- [12] 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.
- [13] 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.
- [14] 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. 108546, 2022, presented at the 22nd Power Systems Computation Conference (PSCC 2022).
- [15] 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.
- [16] T. W. Mak, M. Chatzos, M. Tanneau, and P. V. Hentenryck, "Learning regionally decentralized AC optimal power flows with ADMM," to appear in *IEEE Transactions on Smart Grid*, 2023.
- [17] 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.
- [18] 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.
- [19] 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.
- [20] 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.
- [21] 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.
- [22] 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. 108297, 2022, presented at the 22nd Power Systems Computation Conference (PSCC 2022).
- [23] IEEE PES PGLib-OPF Task Force, "The power grid library for benchmarking AC optimal power flow algorithms," Aug. 2019, arXiv:1908.02788.
- [24] M. Alkhraijah, R. Harris, C. Coffrin, and D. K. Molzahn, "PowerModelsADA: A framework for solving optimal power flow using distributed algorithms," to appear in *IEEE Transactions on Power Systems*, 2023.
- [25] 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, Feb. 2019.
- [26] 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, Jan. 2023, presented at the 22nd Power Systems Computation Conference (PSCC 2022).
- [27] 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.
- [28] 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.
- [29] 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.
- [30] I. Dunning, J. Huchette, and M. Lubin, "JuMP: A modeling language for mathematical optimization," *SIAM Review*, vol. 59, no. 2, 2017.
- [31] 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.