Assessing the Impacts of Nonideal Communications on Distributed Optimal Power Flow Algorithms

Mohannad Alkhraijah, *Student Member, IEEE,* Carlos Menendez, *Student Member, IEEE,* and Daniel K Molzahn, *Senior Member, IEEE*

***Abstract*—Power system operators are increasingly looking toward distributed optimization to address various challenges facing electric power systems. To assess their capabilities in environments with nonideal communications, this paper investigates the impacts of data quality on the performance of distributed optimization algorithms. Specifically, this paper compares the performance of the Alternating Direction Method of Multipliers (ADMM), Analytical Target Cascading (ATC), and Auxiliary Problem Principle (APP) algorithms in the context of DC Optimal Power Flow (DC OPF) problems. Using several test systems, this paper characterizes the performance of these algorithms in terms of their convergence rates and solution quality under three data quality nonidealities: (1) additive Gaussian noise, (2) bad data (large error), and (3) intermittent communication failure.**

***Index Terms*—Distributed Optimization, Nonideal Communication, Optimal Power Flow.**

I. Introduction

TRADITIONALLY, power system operations are primarily conducted centrally, where the tasks of modeling the system's components, solving an optimization problem, and dispatching setpoints are performed by the system operator. However, as power systems move toward a more decentralized paradigm with many locally controlled microgrids, scaling these tasks becomes challenging when using centralized methods. Further, modeling connected systems is difficult, especially for microgrids in a distribution network.

Distributed algorithms address these challenges by allowing interconnected systems with local controllers to cooperatively solve large optimization problems. This increases flexibility as each microgrid solves a local subset of the original optimization problem. Distributed algorithms thus have various potential advantages, such as allowing parallel computations, reducing the requisite communication infrastructure, and possibly maintaining the subsystems' autonomy and data privacy.

These advantages motivate solving Optimal Power Flow (OPF) problems in a distributed fashion. OPF problems seek optimal setpoints for a power system. The decision variables are typically the generators' power outputs and the voltage phasors. Several distributed optimization algorithms have been proposed for solving OPF problems [1]. The Alternating Direction Method of Multipliers (ADMM) applies Augmented Lagrangian Relaxation to decompose the centralized problems into smaller problems which permit a distributed implementation [2]. Augmented Lagrangian Relaxation is also used in other distributed OPF solution algorithms, such as Analytical Target Cascading (ATC) [3], Auxiliary Problem Principle (APP) [4], and Dual Decomposition. Other distributed algorithms directly consider the Karush-Kuhn-Tucker (KKT) optimality conditions for OPF problems. These include the Optimality Condition Decomposition and Consensus+Innovation algorithms. Numerical comparisons among these distributed OPF algorithms are presented in [5] and [6]. Further, a numerical analysis presented in [7] shows that distributed optimization algorithms might converge to suboptimal solutions for nonconvex problems, depending on the initialization. These studies analyze the distributed algorithm assuming ideal communication between interconnected systems.

Communication plays a major role in implementing distributed optimization since regions share data with their neighbors [8]. The shared data between interconnected regions can be subject to communication errors and malicious attacks. The authors of [3] investigate the impact of injecting false values into the shared data of the ATC algorithm when applied to the network-constrained unit commitment problem and show that the error injection increases the convergence time. Nevertheless, it is difficult to derive a conclusion about the robustness of the algorithms when considering one type of communication nonideality with a single erroneous value injection. Reference [9] investigates the communication infrastructure requirements for distributed optimization when applied to power system problems. The authors of [9] model the communication network using a simulator and investigate the impact of communication delay on two distributed optimization algorithms. Communication delay is one important communication nonideality, but other nonidealities can impact the performance of distributed optimization algorithms.

In more general distributed optimization settings, the impacts of communication noise due to quantization effects have been investigated in [10]–[12]. The authors of [13] provide analytic upper and lower bounds on the ADMM algorithm's performance in the presence of random errors and validate the results using randomly generated networks. The authors of [14] study the robustness of distributed algorithms based on dual averaging to additive communication noise. The authors of [15] investigate the ADMM algorithm's convergence under additive communication noise and recommend modifications to the underlying optimization problem. Packet loss is another communication issue that has been discussed in the literature. The impact of packet loss on unconstrained convex distributed optimization is investigated in [16]. A relaxed ADMM algorithm is proposed in [17] to optimize integrated electrical and heating systems considering communication packet loss.

In this paper, we compare the performance of three distributed algorithms (ADMM, ATC, and APP) in solving OPF problems in the presence of nonideal communications. We characterize the algorithms' performance in terms of conver-

gence rate and solution quality. To serve as a benchmark, we first compare the performance of the three algorithms with ideal communications. We then consider three noise models that impact the shared data between neighboring regions: (1) additive Gaussian noise, (2) bad data (large error), and (3) intermittent communication loss. We use five test systems to evaluate the performance of the algorithms with nonideal communication. This paper considers the DC OPF formulation which uses the DC power flow linearization [18]. Thus, this paper's main contribution is an extensive empirical study regarding the impacts of nonideal communication on various distributed solution algorithms for DC OPF problems.

We note that the convexity of the DC OPF formulation studied in this paper provides advantages in terms of theoretical convergence guarantees for the distributed algorithms. These guarantees are useful in the context of this paper as they ensure that the distributed algorithms should all converge to the same solution, thus permitting consistent comparisons that primarily focus on the speed and robustness of the algorithms. However, there are applications where the DC power flow approximation is inappropriate, thus requiring alternative power flow representations [19]. This paper forms a basis for future extensions of our analyses for these applications.

The rest of the paper is organized as follows. Section II introduces the mathematical notation and the DC OPF problem. Section III formulates the problem decomposition and the distributed solution algorithms. Section IV describes the noise models we use in the analysis. Section V presents numerical results and discusses the distributed algorithms' performance under nonideal communication. Section VI summarizes the main findings and discusses future work.

Remark on Notation

Throughout the paper, we use bold letters to indicate vectors. We denote a local subproblem and regions with the letter m and let $\mathscr{M}$ indicate the set of all subproblems. The set $\mathscr{N}_s^m$ contains the boundary variables for region m. We use the letters n and c to denote variables from the neighboring regions and the central coordinator. We use $||\cdot||$ to denote the vector l_2-norm. We use the hat in $\hat{x}$ to indicate that the variables x are evaluated with constant values from the previous iteration as received from neighboring regions or obtained from the prior local solution.

II. DC Optimal Power Flow Formulation

OPF problems typically minimize generation costs while satisfying the power flow equations and limits on generator outputs, line flows, etc. The optimization variables are the generators' outputs and the bus voltages. We consider the DC power flow approximation [18] to obtain convergence guarantees for the distributed algorithms, which are discussed in Section III. The DC OPF formulation used in this paper is:

$$\min_{\boldsymbol{\theta},\boldsymbol{p}} \quad \sum_{i\in\mathscr{G}} f_i(p_i) \tag{1a}$$

$$\text{s.t.} \quad p_i - d_i = \sum_{(i,j)\in\mathscr{L}} B_{ij}(\theta_i - \theta_j), \qquad \forall i \in \mathscr{B} \tag{1b}$$

$$P_i^{min} \le p_i \le P_i^{max}, \qquad \forall i \in \mathscr{G} \tag{1c}$$

$$-P_{ij}^{max} \le B_{ij}(\theta_i - \theta_j) \le P_{ij}^{max}, \qquad \forall (i,j) \in \mathscr{L} \tag{1d}$$

$$\theta^{ref} = 0 \tag{1e}$$

where the sets $\mathscr{B}$, $\mathscr{G}$, and $\mathscr{L}$ denote the buses, generators, and lines, respectively. The generation costs are denoted by f_i. The decision variables are $\boldsymbol{\theta}$, the bus voltage angles, and $\boldsymbol{p}$, the generators' active power outputs. We denote the power demands by d, and B denotes the lines' susceptances. Bounds on the power output of generator i are P_i^{max} and P_i^{min}, and P_{ij}^{max} defines the flow limit of the line between buses i and j. The objective function in (1a) minimizes the total cost of the generators' power outputs. Constraint (1b) is the DC approximation of the power flow equations [18]. Constraints (1c) and (1d) limit the generators' power outputs and the line flows. Constraint (1e) sets the reference angle, θ^{ref}.

III. Distributed Optimization Algorithms

This section overviews the distributed algorithms considered in this paper. These algorithms decompose the centralized DC OPF problem into subproblems corresponding to a partition of the original system. In the DC OPF formulation (1), tie-lines between regions couple the power flow equations (1b) and the line flow limits (1d). For each tie-line, we duplicate the boundary variables and assign a local copy to each region as shown in Fig. 1.

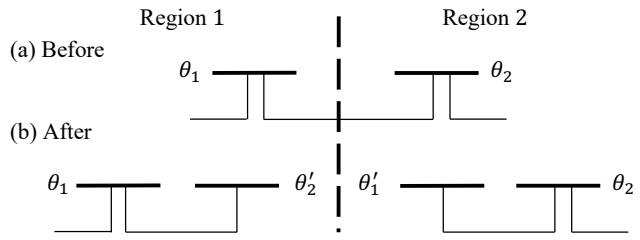

Fig. 1. Tie-line model before (a) and after (b) applying the decomposition.

To obtain a feasible solution to the centralized problem, we need to ensure consistency between the duplicated variables in each subproblem. To accomplish this, we consider algorithms that use an Augmented Lagrangian method to enforce consistency using a penalized objective function:

$$L_\rho(\boldsymbol{p},\boldsymbol{\theta},\boldsymbol{\lambda}) := \sum_{i\in\mathscr{G}} f_i(p_i) + \sum_{i\in\mathscr{N}_s} \lambda_i(\theta_i - \theta_i') + \frac{\rho}{2}||\theta_i - \theta_i'||_2^2, \tag{2}$$

where $\boldsymbol{\lambda}$ are the Lagrange multipliers of the consistency constraints, $\boldsymbol{\theta}'$ are copies of the boundary variables, and ρ is a parameter. The set $\mathscr{N}_s$ contains all copies of the bus voltage angle variables corresponding to the tie-lines between regions.

The three distributed algorithms we consider are similar in their application of an Augmented Lagrangian to decompose the original problem. However, they use different processes for evaluating the objective function and updating the Lagrange

multiplier λ. Generally, the three algorithms use the shared variables received from neighboring regions to update the Lagrange multipliers and evaluate the relaxed consistency constraints in the objective function. This makes the subproblems separable and independent from each other. The algorithms repeat these steps until all regions reach a consensus on the shared variables' values. We use the ℓ_2-norm of the shared variables' mismatch, denoted by $||\Delta\theta||$, to measure the consensus on the shared variables. We next provide more detail for these distributed algorithms.

A. Alternating Direction Method of Multipliers (ADMM)

ADMM is a well-known algorithm for solving large optimization problems [20], [21]. The ADMM algorithm solves the augmented Lagrangian problem in a distributed fashion that is similar to the Gauss-Siedel iterative method. Solving the OPF problem using ADMM in a distributed way involves a central coordinator to decompose the problem. The local problem of region m for iteration $k+1$ is:

$$\min_{\boldsymbol{p}^{k+1},\boldsymbol{\theta}^{m,k+1}} \sum_{i\in\mathscr{G}^m} f_i(p_i^{k+1}) + \sum_{i\in\mathscr{N}_s^m} \lambda_i^k(\hat{\theta}_i^{c,k} - \theta_i^{m,k+1}) + \frac{\alpha}{2}||(\hat{\theta}_i^{c,k} - \theta_i^{m,k+1})||_2^2, \tag{3}$$

subject to the DC OPF constraints (1b)–(1e),

where α is a tuning parameter. The decision variable $\boldsymbol{\theta}^m$ includes the local variables, i.e., the voltage angles for both the local and shared buses. The variable $\hat{\boldsymbol{\theta}}^c$ denotes the shared variables evaluated with values received from the central coordinator, and $\boldsymbol{\lambda}$ denotes the Lagrange multipliers.

After solving the local problem (3), the local controllers share their solutions with the central coordinator. The coordinator then solves an unconstrained optimization problem:

$$\min_{\boldsymbol{\theta}^{c,k+1}} \sum_{m\in\mathscr{M}} \sum_{i\in\mathscr{N}_s^m} \lambda_i(\theta_i^{c,k+1} - \hat{\theta}_i^{m,k+1}) + \frac{\alpha}{2}||(\theta_i^{c,k+1} - \hat{\theta}_i^{m,k+1})||_2^2, \tag{4}$$

where $\boldsymbol{\theta}^c$ are the decision variables and $\hat{\boldsymbol{\theta}}^m$ are the shared variables received from the local controllers. The shared variable values obtained from the central coordinator's problem are then sent to the local controllers to update the Lagrange multipliers according to (5):

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \alpha(\boldsymbol{\theta}^{c,k+1} - \boldsymbol{\theta}^{m,k+1}), \tag{5}$$

where α is the same parameter used in the local optimization problem (3). Note that $\boldsymbol{\theta}^m$ and $\boldsymbol{\theta}^c$ are the solutions of (3) and (4), respectively. The local controllers then use the updated Lagrange multipliers and the central coordinator's solution to update the local problems. This process is repeated until the shared variables agree to within a specified tolerance.

There are several variants of this ADMM implementation that have been proposed for solving OPF problems. These include extensions that eliminate the need for a central coordinator [22], [23]. A proximal massage passing (PMP) method proposed in [24] also permits a fully distributed ADMM implementation. The calculations in this paper use a fully distributed implementation of the ADMM algorithm by replacing the central coordinator's problem with its first-order optimality conditions. In this implementation, each local controller uses the prior local solution and the neighboring regions' solutions to solve the central coordinator's problem locally. We also note that the ADMM algorithm has been proven to converge to the optimal solution if the subproblems are convex. See [2] for more detail regarding ADMM implementations and convergence guarantees.

B. Analytic Target Cascading (ATC)

ATC is another distributed algorithm based on augmented Lagrangian relaxation that solves a large optimization problem via dividing it into hierarchically connected subproblems with multiple levels [25]. Two levels of subproblems are connected if they share coupling variables. To solve the OPF problem, we use a two-level ATC structure. The first level consists of a central coordinator, while the regions' local problem are in the second level. The local problem m for iteration $k+1$ is:

$$\min_{\boldsymbol{p}^{k+1},\boldsymbol{\theta}^{m,k+1}} \sum_{i\in\mathscr{G}^m} f_i(p_i) + \sum_{i\in\mathscr{N}_s^m} \lambda_i(\hat{\theta}_i^{c,k} - \theta_i^{m,k+1}) + ||\beta(\hat{\theta}_i^{c,k} - \theta_i^{m,k+1})||_2^2, \tag{6}$$

subject to DC OPF constraints (1b)–(1e),

where $\boldsymbol{\lambda}$ are the Lagrange multipliers and β is a parameter. We denote the shared variables that are fixed to their values from the central coordinator with $\hat{\boldsymbol{\theta}}^c$. The local controllers communicate the resulting shared variable values with a central coordinator. The coordinator solves an unconstrained optimization problem that minimizes the differences between the boundary variables for neighboring regions:

$$\min_{\boldsymbol{\theta}^{c,k+1}} \sum_{m\in\mathscr{M}} \sum_{i\in\mathscr{N}_s^m} \lambda_i(\theta_i^{c,k+1} - \hat{\theta}_i^{m,k+1}) + ||\beta(\theta_i^{c,k+1} - \hat{\theta}_i^{m,k+1})||_2^2. \tag{7}$$

The coordinator shares the target results $\boldsymbol{\theta}^{c,k+1}$ with the local controllers. Next, the local controllers update the Lagrange multipliers and the parameters β using the target variables:

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + 2(\beta^k)^2(\boldsymbol{\theta}^{c,k+1} - \boldsymbol{\theta}^{m,k+1}), \tag{8a}$$

$$\beta^{k+1} = \alpha\beta^k, \tag{8b}$$

where β is the same parameter used in the local optimization problem (6), and α is a tuning parameter. After updating the Lagrange multipliers, the local controllers and the central coordinator repeat the process until the shared variables are within a specified tolerance.

Variants of ATC use different functions besides the Augmented Lagrangian to relax the consistency constraints [26]. Further, the ATC variant proposed in [3] is fully distributed, eliminating the need for a central coordinator. Similar to the ADMM algorithm, we use a fully distributed implementation, where each local controller uses the first-order optimality conditions of the central coordinator's problem to solve the local subproblem. The ATC algorithm is proven to converge for convex problems if the interaction is limited to subproblems in different levels [25].

C. Auxiliary Problem Principle (APP)

The APP algorithm is also based on augmented Lagrangian decomposition [27]. The APP algorithm solves a sequence of auxiliary problems in a distributed fashion without the need for a central coordinator. In contrast to ADMM and ATC which directly use the Augmented Lagrangian, APP linearizes the quadratic term in the augmented Lagrangian around the previous iteration and introduces a regularization term in the objective function [28]. Using the APP algorithm, the OPF formulation for region m and iteration $k+1$ is:

$$\min_{\boldsymbol{p}^{m,k+1},\boldsymbol{\theta}^{k+1}} \quad \sum_{i\in\mathscr{G}} f_i(p_i^{k+1}) + \sum_{i\in\mathscr{N}_s^m} \frac{\beta}{2}||\theta_i^{m,k+1} - \hat{\theta}_i^{n,k}||_2^2 + \gamma\theta_i^{m,k+1}(\hat{\theta}_i^{m,k} - \hat{\theta}_i^{n,k}) + \lambda\,\theta_i^{m,k+1}, \qquad (9)$$

subject to DC OPF constraints (1b)–(1e),

where α, β, and γ are tuning parameters. We define $\hat{\boldsymbol{\theta}}^{m,k}$ as the value of the shared variable obtained by region m in the previous iteration, and $\hat{\boldsymbol{\theta}}^{n,k}$ denotes the values of the shared variables received from neighboring regions. After each region's local controller solves its associated problem and exchanges the results with the neighboring regions, each local controller updates the values of the Lagrange multipliers as follows:

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^{k} + \alpha(\boldsymbol{\theta}^{m,k+1} - \boldsymbol{\theta}^{n,k+1}). \qquad (10)$$

To summarize, each region iteratively solves (9), shares the results with neighboring regions, and updates the Lagrange multiplier using (10) until reaching consensus on the shared variables. If the local problems are convex and differentiable, selecting parameters satisfying the condition $\alpha < 2\gamma < \beta$ guarantees that the APP algorithm will converge [4].

IV. Nonideal Communications Models

The communication requirements for a distributed algorithm depend on the shared variables. During each iteration, each region shares the results of its local optimization by communicating with the neighboring regions. Since communication networks are not ideal, the shared data may suffer from data quality issues or interruptions that impact the distributed algorithms' performance. In this section, we introduce three models for nonideal communications: additive Gaussian noise, bad data (large errors), and intermittent loss of communication.

A. Additive Gaussian Noise

The data shared between connected regions may be subject to noise resulting from imperfect communication. This noise could be due to quantization error [29] or added to enforce data privacy requirements [30]. To model noisy shared data, we inject additive Gaussian noise into the shared variables:

$$\boldsymbol{\theta}_{noisy} = \boldsymbol{\theta}_{noiseless} + \boldsymbol{N}(0, \sigma_{noise}) \qquad (11)$$

where $\boldsymbol{\theta}_{noisy}$ is the data that is actually communicated to the neighbors, $\boldsymbol{\theta}_{noiseless}$ is the true data, and $\boldsymbol{N}(0,\sigma_{noise})$ is a vector of normally distributed random numbers with zero mean and standard deviation of σ_{noise}.

B. Bad Data

Neighboring regions may occasionally receive "bad data", i.e., data with large errors. Bad data may be due to an instantaneous bit error [31] or a malicious adversarial agent [32]. We model a random injection of bad data at a specified occurrence probability as shown in the following model:

$$\theta_{noisy} = \theta_{noiseless} + 2\,R\,r,$$
$$\text{where} \quad r = \begin{cases} U_1 - 0.5 & \text{if } U_2 < p, \\ 0 & \text{otherwise.} \end{cases} \qquad (12)$$

In (12), R is the maximum magnitude of the error and r is the error multiplier that randomly selects the error magnitude. The variables U_1 and U_2 are uniformly distributed random numbers between $[0,1]$, and p is the probability of bad data occurrence per iteration.

C. Intermittent Communication Loss

Communications between the agents may occasionally fail entirely for multiple iterations. For instance, communication transmission collisions or instability of the communication link can cause packet loss preventing the agents from sharing data for a number of iterations [16]. We consider a simple two-state model with success and fail states to represent the loss of communication. If the communication channel is in a success state, then the data will be transmitted while the fail state means the data will be lost. The transition from one state to another occurs with a constant probability. This model is similar to Gilbert-Elliott Erasure channel used to model pocket loss [33]. Although communication loss can have more complicated occurrence behaviour, this model is used to model pocket loss due to its tractability and reasonable behaviour with respect to experimental data [34].

To model intermittent communication loss, we define the *failure probability*, denoted λ_f, as the transition probability of the communication channel from success to fail states per iteration given that the channel was in success state during the previous iteration. Similar to the failure probability, we define the *repair probability*, denoted λ_r, as the transition probability from fail to success state. We also introduce a state variable s for each communication channel connecting two regions, where $s=1$ is a success state and $s=0$ is a fail state. We use an indicator function $\mathbf{1}\{x\}$, which equals 1 if x is true and 0 otherwise. We model the state of the communication channel by sampling from a uniformly distributed random number U at each iteration and compare it with the failure (repair) probability to decide the channel state. If the controller detects an interruption from a neighboring region, it uses the value from the last successful data transmitted from this region. The following pseudocode describes our intermittent communication loss model:

Intermittent Communication Loss Model

generate a random number $U \in [0,1]$
if $s=1$ **then** set $s = 1 - \mathbf{1}\{U < \lambda_f\}$
else set $s = \mathbf{1}\{U < \lambda_r\}$ **end if**
if $s=0$ **then** set $\boldsymbol{\theta}_{shared}^{k+1} = \boldsymbol{\theta}_{shared}^{k}$ **end if**

V. Performance Analyses

This section presents the numerical results of solving the DC OPF problem using the three distributed algorithms. We first present the results of each algorithm with ideal communication. We then compare the performance of the distributed algorithms with the nonideal communication models in Section IV. We use five test systems: the 5-bus system "WB5" from [35] and the IEEE 14-bus, IEEE 118-bus, IEEE 300-bus, and RTS GMLC 73-bus systems from MATPOWER [36]. The first two systems are decomposed into two regions, while the latter three systems are decomposed into three regions. We model the distributed algorithm using a combination of JuMP [37] and PowerModels [38] libraries in Julia programming language [39], and use Gurobi to solve the subproblems.

A. Performance with Ideal Communications

We use the distributed algorithms assuming each region shares the exact results with neighboring regions at each iteration. We use the two-norm of the mismatches between the values of the shared variables to measure the consensus and set the stopping criteria. We use the results of the ideal communication to tune the parameters and evaluate the convergence rates of the algorithms.

1) Alternating Direction Method of Multipliers (ADMM): The ADMM algorithm's performance depends on the value selected for the parameter α. Fig. 2 shows the convergence of the ADMM algorithm for the IEEE 118-bus system. For this test system, we observe that the regions reach consensus on the shared variables the fastest when the value of α is 1×10^6 at the first 50 iterations. However, we obtain more stable convergence when we set α equal 1×10^5 and the convergence became faster after 260 iterations. For the WB5, IEEE 14-bus, RTS GMLC, and IEEE 300-bus systems, we tune the value of α to be 1×10^2, 1×10^3, 1×10^5 and 1×10^5, respectively. Selecting smaller values for α increases the number of iterations to achieve consensus, while selecting significantly larger values cause oscillations that reduces the convergence rate.

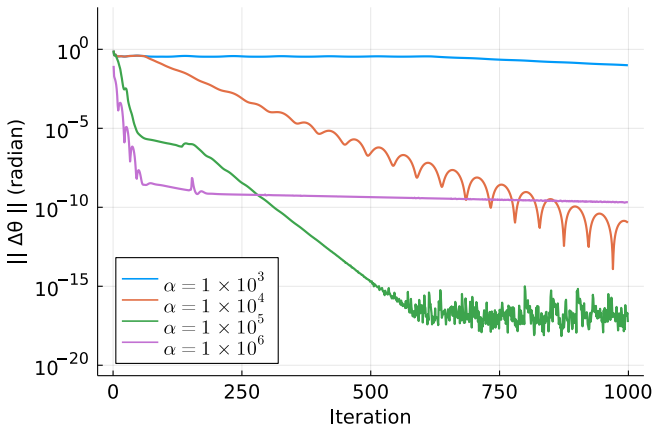

Fig. 2. ADMM convergence with different parameters for the IEEE 118-bus system with ideal communications.

2) Analytical Target Cascading (ATC): Similar to ADMM, the ATC algorithm requires tuning one parameter (α). The convergence of the shared variables for the IEEE 118-bus system is shown in Fig. 3. We observe that setting the parameter $\alpha = 1.1$ increases the convergence rate by a factor of two compared to $\alpha = 1.05$. We see similar behaviour for the other systems. Furthermore, the solver fails to find an optimal solution to the local problems when the number of iterations increases as shown in Fig. 3. This behaviour occurs due to the parameter update criteria (8b), which exponentially increases the penalty on the shared variable consistency term as the number of iterations increases.

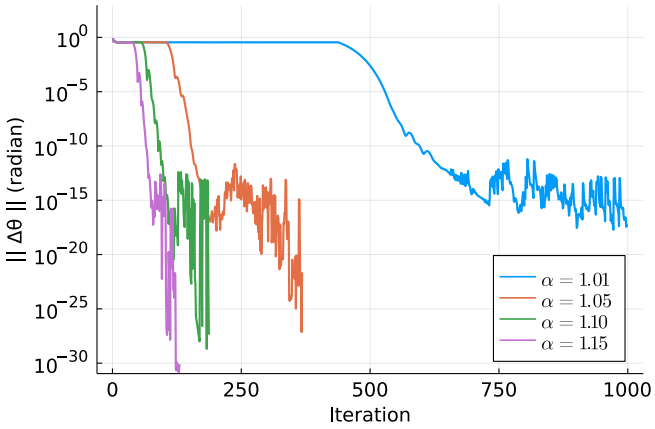

Fig. 3. ATC convergence with different parameters for the IEEE 118-bus system with ideal communications.

3) Auxiliary Problem Principal (APP): Unlike ADMM and ATC, the APP algorithm contains three parameters that need to be tuned. We adopt the condition $\alpha = \gamma = \frac{1}{2}\beta$ from prior literature [4] in order to simplify the parameter tuning. For the IEEE 118-bus system with three regions, the APP algorithm converges when $\alpha = 1 \times 10^4$ as shown in Fig. 4. The algorithm does not converge with the value of $\alpha = 1 \times 10^3$ after 1000 iterations. For the WB5, IEEE 14-bus, RTS GMLC, and IEEE 300-bus systems, we find a value of α equal to 1×10^2, 1×10^3, 1×10^5, and 1×10^5, respectively, yield a fast convergence rate. While selecting larger parameter values might improve the convergence rate, this also degrades the mismatch error in the shared variables achieved by the algorithm.

4) Remark about the Algorithms' Performance: Along with the convergence rate, the quality of the final solution is another metric for selecting parameters. To evaluate solution quality, we use the *Relative Gap* (RG), defined as the absolute value of the difference between the generation cost of the distributed and centralized solutions divided by the centralized solution's cost. Table I summarizes the convergence rate results obtained from the three algorithms after reaching consensus on the shared variables with tolerance equal to 1×10^{-4} radians (5.7×10^{-3} degrees). The convergence rate of the distributed algorithms depends on the selection of the tuning parameters. Comparing the convergence rate of the three algorithms with the tuning parameters that we select, no algorithm outperforms the others, and the fastest algorithm varies depending on the test system.

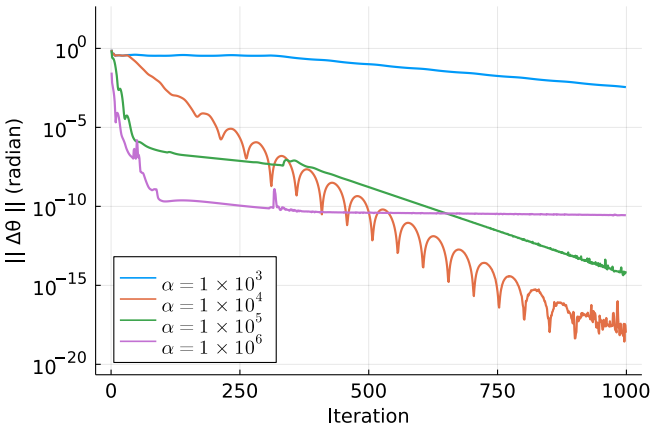

Fig. 4. APP convergence with different parameters for the IEEE 118-bus system with ideal communications.

TABLE I
CONVERGENCE RESULTS WITH IDEAL COMMUNICATION

System	Measure	ADMM	ATC	APP
WB5	α	10^3	1.5	10^2
	Iterations	61	22	34
	Time (s)	0.1631	0.0642	0.0894
14-Bus	α	10^4	1.04	10^5
	Iterations	116	149	107
	Time (s)	0.3134	0.4064	0.2897
RTS	α	10^7	1.3	10^7
	Iterations	33	35	33
	Time (s)	0.8871	1.0216	0.9153
118-Bus	α	10^6	1.1	10^6
	Iterations	55	87	62
	Time (s)	0.4973	0.8381	0.5842
300-Bus	α	10^7	1.2	10^7
	Iterations	130	62	100
	Time (s)	2.1788	1.0833	1.7360

B. Performance with Additive Gaussian noise

We next assume the regions send shared variable information with additive Gaussian noise as described in (11). To compare the performance of the algorithms, we vary the standard deviation of the noise σ_{noise}. The consensus on the voltage angles achieved by the three algorithms for the IEEE 118-bus system with standard deviation $\sigma = 1 \times 10^{-3}$ radians is shown in Fig. 5. The results indicate that small noise levels do not significantly impact the convergence rate of the algorithms. Further, the three distributed algorithms converge to a similar level of accuracy as the level of the injected noise. However, the ATC algorithm can exhibit numerical instability preventing the solver from solving the subproblems when the algorithm is not terminated, which we attribute to the parameter update rule (8b).

We run the distributed algorithms 100 times and calculate the mean and the standard deviation of the shared variables mismatch. The mean and the standard deviation of the shared variables' mismatches in the final solution, $\mu_{||\Delta\theta||}$ and $\sigma_{||\Delta\theta||}$, for the three algorithms with three noise levels are shown in Table II. To visualize the performance differences, Fig. 6- 10 show the mean of the shared variable mismatches, $\mu_{||\Delta\theta||}$, in the final solutions for the test systems as the noise standard

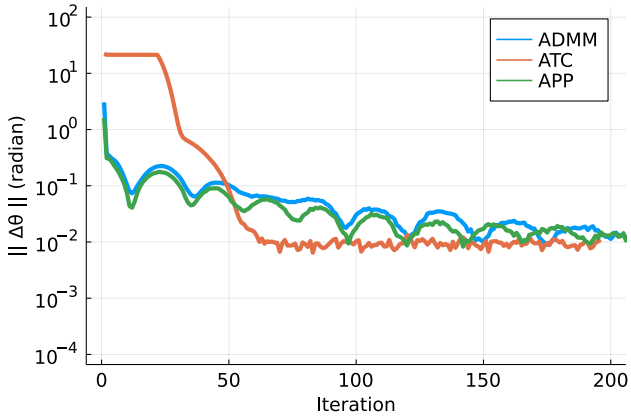

Fig. 5. The three algorithms' convergence for the IEEE 118-bus system with additive Gaussian noise ($\sigma_{noise} = 1 \times 10^{-3}$).

deviation, σ_{noise}, varies from 1×10^{-6} to 1×10^{-3}. The three algorithms achieve consensus on the shared variables with an error that increases approximately linearly with the standard deviation of the added noise, with a slightly lower mismatch observed for ATC compared to ADMM and APP. This is specially noticeable for the IEEE 300-bus case with low noise standard deviating.

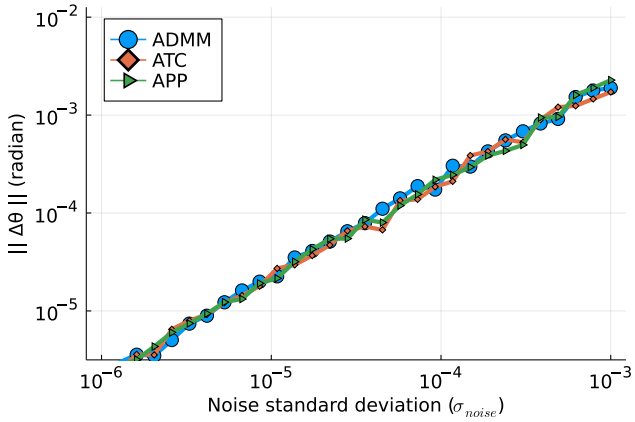

Fig. 6. Mean of the mismatch in the final solution for the WB5 system with different noise levels.

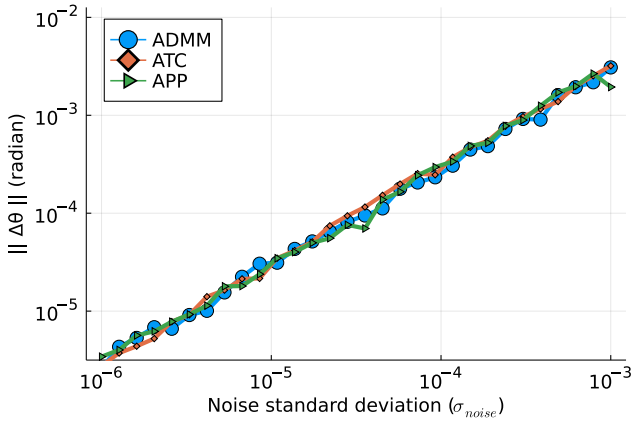

Fig. 7. Mean of the mismatch in the final solution for the IEEE 14-bus system with different noise levels.

TABLE II
MEAN ($\mu_{||\Delta\theta||}$) AND STANDARD DEVIATION ($\sigma_{||\Delta\theta||}$) OF THE MISMATCH IN THE FINAL SOLUTION WITH DIFFERENT VALUES OF NOISE (σ_{noise})

Algorithm		ADMM			ATC			APP		
System	σ_{noise}	10^{-5}	10^{-4}	10^{-3}	10^{-5}	10^{-4}	10^{-3}	10^{-5}	10^{-4}	10^{-3}
WB5	$\mu_{\|\|\Delta\theta\|\|}$	4.8×10^{-5}	4.3×10^{-4}	4.6×10^{-3}	4.4×10^{-5}	3.9×10^{-4}	4.4×10^{-3}	5.0×10^{-5}	4.4×10^{-4}	4.7×10^{-3}
	$\sigma_{\|\|\Delta\theta\|\|}$	1.5×10^{-5}	1.2×10^{-4}	1.2×10^{-3}	1.1×10^{-5}	8.9×10^{-5}	9.2×10^{-4}	1.3×10^{-5}	1.1×10^{-4}	1.1×10^{-3}
14-Bus	$\mu_{\|\|\Delta\theta\|\|}$	5.7×10^{-5}	5.0×10^{-4}	5.7×10^{-3}	6.1×10^{-5}	5.2×10^{-4}	5.4×10^{-3}	5.8×10^{-5}	5.1×10^{-4}	5.4×10^{-3}
	$\sigma_{\|\|\Delta\theta\|\|}$	1.3×10^{-5}	1.1×10^{-4}	1.5×10^{-3}	1.5×10^{-5}	1.1×10^{-4}	1.3×10^{-3}	1.4×10^{-5}	1.1×10^{-4}	1.3×10^{-3}
RTS	$\mu_{\|\|\Delta\theta\|\|}$	1.3×10^{-4}	1.1×10^{-3}	1.0×10^{-2}	8.0×10^{-5}	7.7×10^{-4}	8.0×10^{-3}	1.2×10^{-4}	1.1×10^{-3}	1.0×10^{-2}
	$\sigma_{\|\|\Delta\theta\|\|}$	3.6×10^{-5}	3.5×10^{-4}	2.9×10^{-3}	1.4×10^{-5}	1.2×10^{-4}	1.3×10^{-3}	3.2×10^{-5}	3.4×10^{-4}	2.4×10^{-3}
118-Bus	$\mu_{\|\|\Delta\theta\|\|}$	1.2×10^{-4}	1.0×10^{-3}	1.1×10^{-2}	1.1×10^{-4}	8.9×10^{-4}	9.7×10^{-3}	1.2×10^{-4}	1.0×10^{-3}	1.1×10^{-2}
	$\sigma_{\|\|\Delta\theta\|\|}$	2.3×10^{-5}	2.0×10^{-4}	2.0×10^{-3}	1.5×10^{-5}	1.4×10^{-4}	1.2×10^{-3}	2.0×10^{-5}	1.8×10^{-4}	1.8×10^{-3}
300-Bus	$\mu_{\|\|\Delta\theta\|\|}$	1.2×10^{-3}	2.3×10^{-3}	1.9×10^{-2}	1.5×10^{-4}	1.2×10^{-3}	1.1×10^{-2}	1.1×10^{-3}	1.9×10^{-3}	1.8×10^{-2}
	$\sigma_{\|\|\Delta\theta\|\|}$	1.6×10^{-4}	1.1×10^{-3}	7.2×10^{-3}	3.3×10^{-4}	1.8×10^{-4}	1.2×10^{-3}	1.6×10^{-4}	6.8×10^{-4}	6.7×10^{-3}

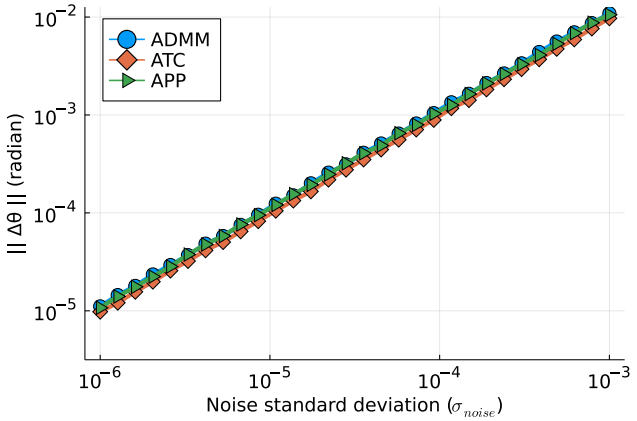

Fig. 8. Mean of the mismatch in the final solution for the IEEE 118-bus system with different noise levels.

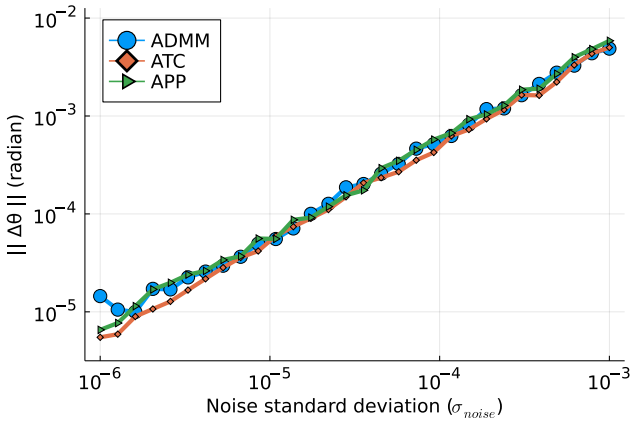

Fig. 9. Mean of the mismatch in the final solution for the RTS GMLC system with different noise levels.

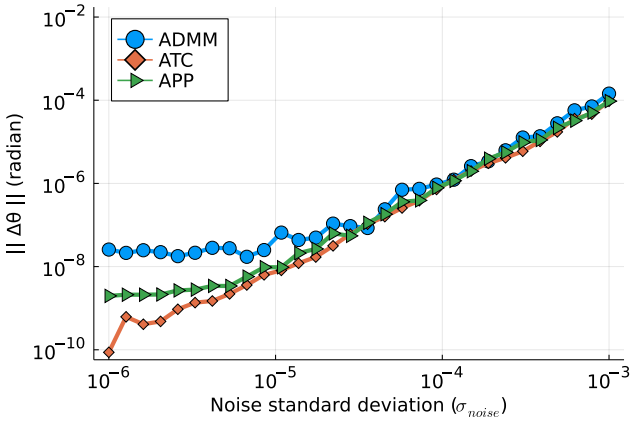

Fig. 10. Mean of the mismatch in the final solution for the IEEE 300-bus system with different noise levels.

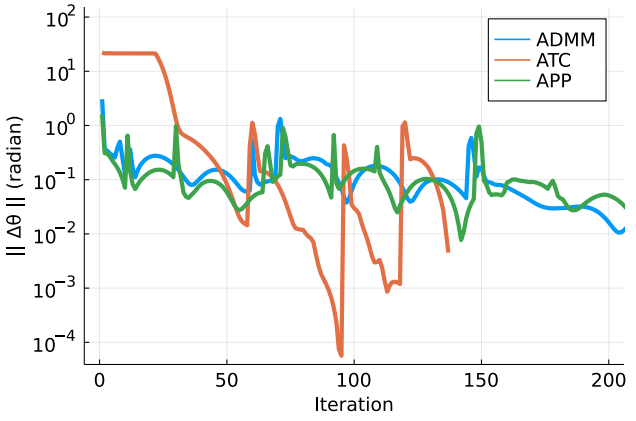

Fig. 11. The three algorithms convergence for the IEEE 118-bus system with bad data ($p = 0.1\%$).

C. Performance with Bad Data

For the second type of noise, we inject bad data into the shared variables as described in (12). We set the value for the bad data magnitude $R = 2$ radians and vary the probability of the injected errors. Fig. 11 shows the mismatches of the shared variables with probabilities of bad data occurrence $p = 0.1\%$ for the IEEE 118-bus system.

We observe that the mismatches return to the same convergence pattern after a large error is injected in all three algorithms. To quantitatively compare the algorithms' performance, we estimate the probability of achieving the optimal solution using 100 runs of the algorithm for varying probabilities of bad data occurrence. We consider an algorithm to have achieved the optimal solution if the value of the relative gap is below 1% within a maximum of 1000 iterations. Table III on the next page summarizes the results with probabilities of bad data equal to 0.1% and 1%. The results show that the distributed algorithms are highly susceptibility to the bad data and they may fail to attain the optimal solution even when the

TABLE III
PERFORMANCE OF THE DISTRIBUTED ALGORITHMS WITH BAD DATA

Algorithm		ADMM		ATC		APP	
Probability of Bad Data		0.1%	1.0%	0.1%	1.0%	0.1%	1.0%
WB5	Success rate (%)	100	100	100	100	100	76
	Avg. Iterations	32	41	16	17	25	31
14-Bus	Success rate (%)	100	85	99	70	100	88
	Avg. Iterations	32	59	44	48	21	43
RTS	Success rate (%)	72	19	83	29	56	19
	Avg. Iterations	96	1000	66	109	104	1000
118-Bus	Success rate (%)	30	0	83	8	30	0
	Avg. Iterations	123	NC	60	66	95	NC
300-Bus	Success rate (%)	62	0	92	3	55	0
	Avg. Iterations	34	NC	62	65	34	NC

NC: Not converged.

bad data probability is as low as 0.1%. Further, Fig. 12-16 show the success rates when varying the bad data probability from 0.1% to 1% for the test systems. Comparing the three algorithms, the ATC algorithm shows a better performance to bad data errors. For the IEEE 118-bus, IEEE 300-bus, and RTS GMLC test systems, the ATC algorithm attains the optimal solution more than 80% of the time when the probability of the bad data is around 0.1%, while the ADMM and APP fail 40% of the time in the same cases.

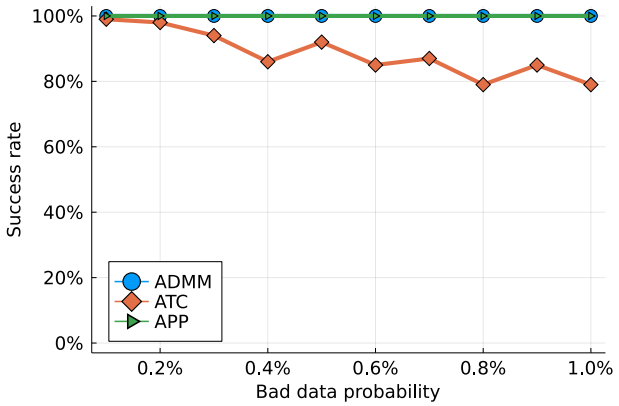

Fig. 12. Success rates for the WB5 system with different rates of bad data injection.

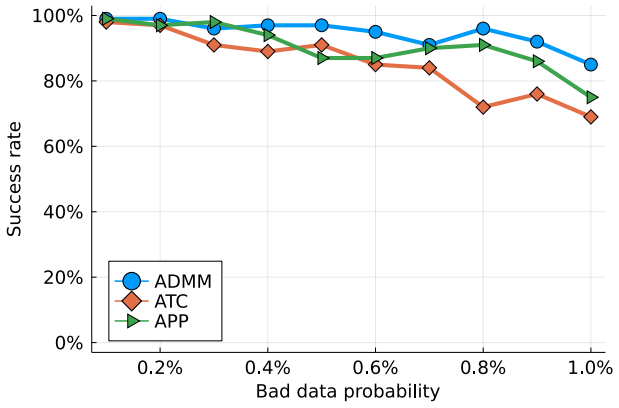

Fig. 13. Success rates for the IEEE 14-bus system with different rates of bad data injection.

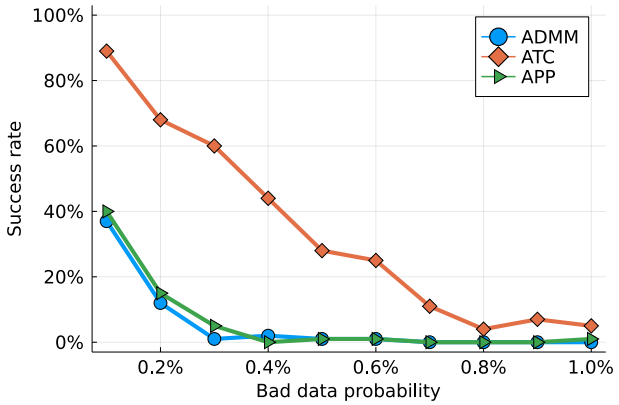

Fig. 14. Success rates for the IEEE 118-bus system with different rates of bad data injection.

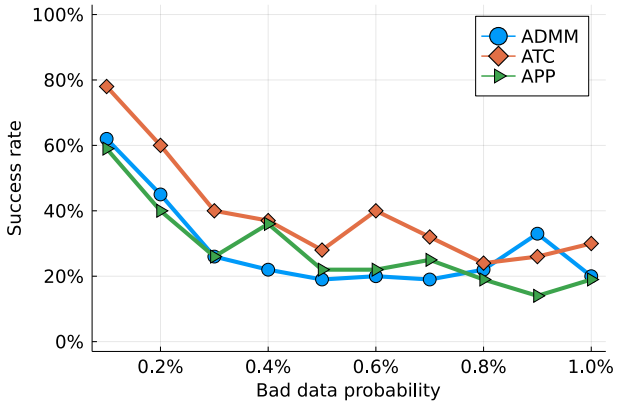

Fig. 15. Success rates for the RTS GMLC system with different rates of bad data injection.

D. Performance with Intermittent Communication Loss

This section compares the distributed algorithms' performance under the intermittent communication loss model in Section IV-C. Fig. 17 shows the convergence characteristics of the shared variable mismatches for the three algorithms with failure probability $\lambda_f = 10\%$ and repair probability $\lambda_r = 10\%$.

The results indicate that the distributed algorithms achieve consensus on the values of the shared variables and the mismatch almost always decreases as the number of iterations

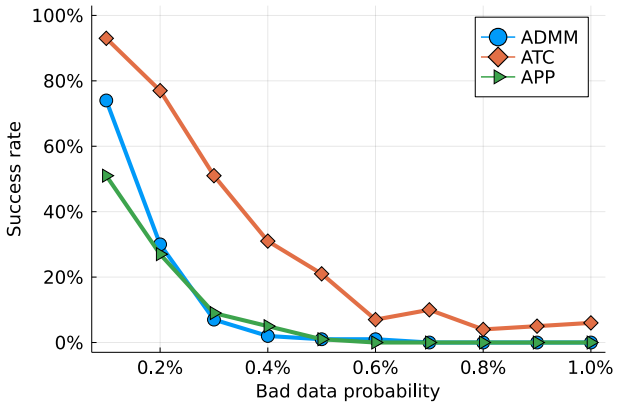

Fig. 16. Success rates for the IEEE 300-bus system with different rates of bad data injection.

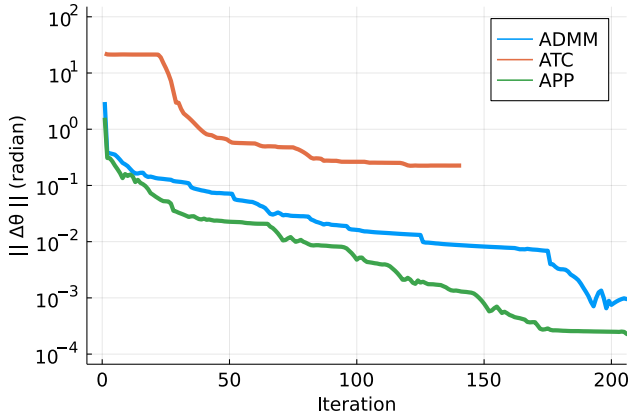

Fig. 17. The three algorithms convergence for the IEEE 118-bus system with intermittent communication loss ($\lambda_f = 10\%$, $\lambda_r = 10\%$).

increases for all cases. However, the final solutions can be far from optimal with high values of the relative gap. To qualitatively compare algorithmic performance during intermittent communication loss, Table IV describes how the probability of achieving the optimal solution changes with failure probability equal to 1% and 5% per iteration. Each of the results in this table are computed from 100 runs of the algorithm with a constant repair rate of 10% per iteration. We again consider an algorithm to have achieved the optimal solution if the relative gap is below 1% within a maximum of 1000 iterations.

The results show that the three algorithm performance is impacted by the communication loss values to varying degrees. With a low probability of failure, the algorithms manage to archive the optimal solution most of the time. Comparing the three algorithms, the ATC algorithm is the most susceptible algorithm to communication loss. The ATC algorithm fails around half the time to obtain the optimal solution for the largest three test system when the failure probability is 1%, and almost all the time when the failure probability is 5%.

Fig. 18- 22 present a comparison between the algorithms' performance when subjected to intermittent communication loss by varying the failure probability from 0.5% to 15%, while fixing the repair probability to 10%. The ADMM algorithm shows a slightly better performance than the APP algorithm for the five test systems. Overall, the results suggest that the ADMM and APP algorithms outperform the ATC algorithm with the intermittent communication loss model for all test systems we considered in this study.

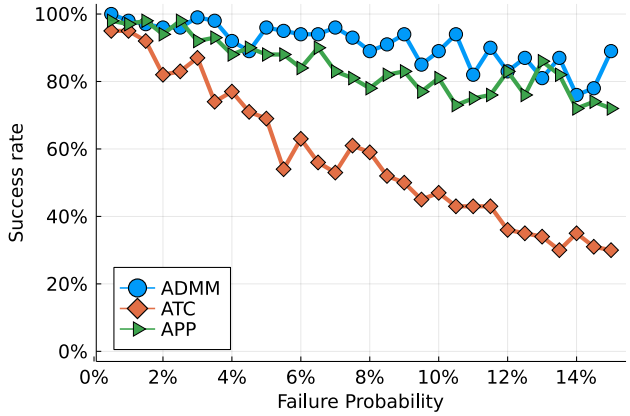

Fig. 18. Success rates for the WB5 system with different communication failure probability. Communication repair probability $\lambda_r = 10\%$.

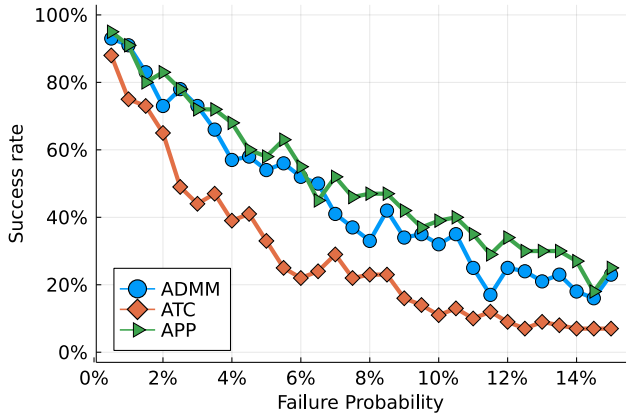

Fig. 19. Success rates for the IEEE 14-bus system with different communication failure probability. Communication repair probability $\lambda_r = 10\%$.

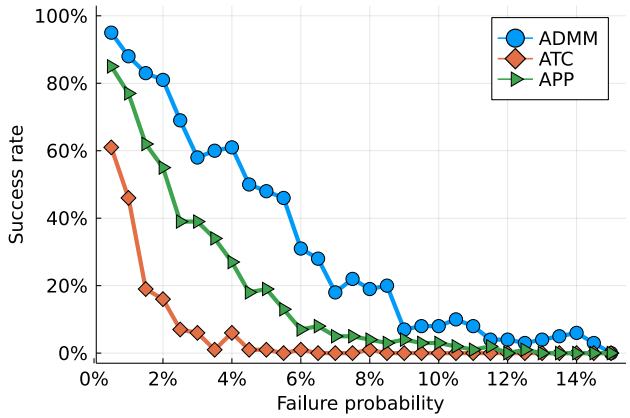

Fig. 20. Success rates for the IEEE 118-bus system with different communication failure probability. Communication repair probability $\lambda_r = 10\%$.

TABLE IV
PERFORMANCE OF THE DISTRIBUTED ALGORITHMS WITH INTERMITTENT COMMUNICATION LOSS

Algorithm		ADMM		ATC		APP	
Failure Probability		1%	5%	1%	5%	1%	5%
WB5	Success rate (%)	97	88	92	71	99	89
	Avg. Iterations	36	45	17	24	27	37
14-Bus	Success rate (%)	84	69	82	38	94	57
	Avg. Iterations	32	34	43	45	22	26
RTS	Success rate (%)	70	28	33	7	58	29
	Avg. Iterations	81	91	69	124	71	94
118-Bus	Success rate (%)	88	48	46	1	77	19
	Avg. Iterations	89	77	60	54	75	61
300-Bus	Success rate (%)	99	95	65	6	99	89
	Avg. Iterations	37	71	65	75	35	71

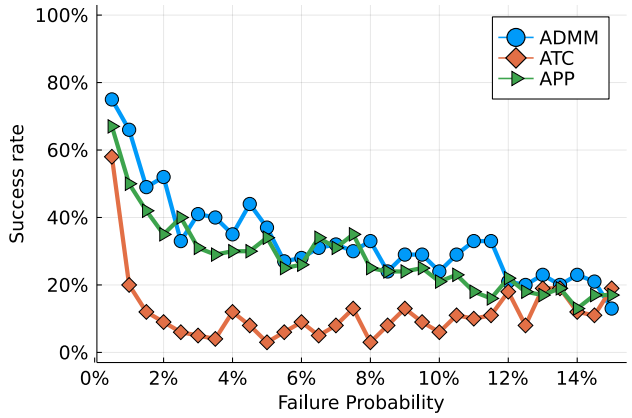

Fig. 21. Success rates for the RTS GMLC system with different communication failure probability. Communication repair probability $\lambda_r = 10\%$.

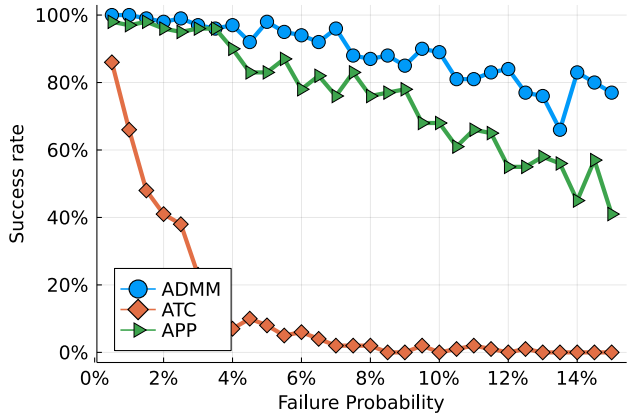

Fig. 22. Success rates for the IEEE 300-bus system with different communication failure probability. Communication repair probability $\lambda_r = 10\%$.

E. Discussion and Comparison

Parameter tuning plays a major role in the performance of distributed algorithms as it strongly impacts the convergence rate. The parameters in the selected algorithms are associated with the penalty terms for the relaxed consistency constraints. We observe that all three algorithms converge for a certain range of parameter values. Generally speaking, selecting large parameter values will prevent the algorithm from achieving the optimal solution and, in some cases, large values might cause the algorithm to diverge. On the other hand, small values reduce the convergence rate and, in extreme cases, the algorithm can diverge. We also observe that different systems and cost functions might require repeating the parameter tuning step.

The results shown in this paper indicate the importance of data integrity on the performance of the distributed algorithms. We observe various responses from the distributed algorithms to the error models. With Gaussian communication noise, all three algorithms converge to an accuracy that is proportional to the standard deviation of the noise. We observed a slightly better performance when using the ATC algorithm compared to the other two algorithms. Among the three algorithms considered in this paper, the ATC algorithm has the best performance with the presence of bad data, while the ADMM and APP algorithms have the best performance when there is a high intermittent communication loss probability.

Both the ADMM and APP algorithms have a very similar performance pattern for the three noise models with slightly better performance observed when using the ADMM algorithm. The ATC algorithm on the other hand has a different performance pattern. Further, the ATC algorithm's final solution can have a high relative gap from the optimal solution or lead to numerical instability in the optimization solver if consensus is not achieved, i.e., the stopping criteria are not met after many iterations. This happens due to the parameter update for the ATC algorithm (8b), which exponentially increases the penalty on the consistency term in the objective as the number of iterations increases. This suggests that reliable performance of the ATC algorithm in the presence of noise requires either using another stopping criterion or modifying the update step in the algorithm.

VI. CONCLUSION AND FUTURE WORK

Distributed algorithms have many attractive features for solving power system optimization problems, especially for systems with many independent microgrids. Distributed algorithms allow interconnected systems to cooperatively solve large optimization problems while maintaining their autonomy. However, the performance of a distributed algorithm strongly depends on the quality of the shared data. In this paper, we numerically show distributed algorithms' responses to data quality issues. We evaluate the performance of ADMM, ATC, and APP distributed algorithms using three noise models.

The results show that the three algorithms perform well with additive Gaussian noise as long as the stopping criteria and the required solution accuracy are lower than the error standard deviation. The results also show that the bad data errors have a severe impact on the quality of the distributed algorithms' solutions even with low error probability. Moreover, the impacts of intermittent communication loss might not be visible by the local controllers, as the distributed algorithms might reach a consensus on the shared variables corresponding to an operating point that is far from optimal.

As extensions to this work, there are other communication and data integrity issues requiring detailed investigations. For power systems applications, these include asynchronous data sharing between neighboring regions and communication latency. We further plan to use hardware-in-the-loop testing with an actual communication network to investigate the performance of distributed algorithms in practical setups. Another direction for future work is studying how different power flow representations affect the convergence rates for problems where the DC approximation is inapplicable.

REFERENCES

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

[2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2010.

[3] A. Kargarian, M. Mehrtash, and B. Falahati, "Decentralized implementation of unit commitment with analytical target cascading: A parallel approach," *IEEE Trans. Power Syst.*, vol. 33, no. 4, pp. 3981–3993, 2018.

[4] B. H. Kim and R. Baldick, "Coarse-grained distributed optimal power flow," *IEEE Trans. Power Syst.*, vol. 12, no. 2, pp. 932–939, 1997.

[5] B. H. Kim and R. Baldick, "A comparison of distributed optimal power flow algorithms," *IEEE Trans. Power Syst.*, vol. 15, no. 2, pp. 599–604, 2000.

[6] A. Kargarian, M. Mehrtash, and B. Falahati, "Decentralized implementation of unit commitment with analytical target cascading: A parallel approach," *IEEE Trans. Power Syst.*, vol. 33, no. 4, pp. 3981–3993, 2017.

[7] I. Murzakhanov, A. Malakhov, and E. Gryazina, "Suboptimality of decentralized methods for OPF," in *IEEE Milan PowerTech*, 2019.

[8] J. Mohammadi, G. Hug, and S. Kar, "Role of communication on the convergence rate of fully distributed DC optimal power flow," in *IEEE Int. Conf. Smart Grid Commun. (SmartGridComm)*, 2014, pp. 43–48.

[9] J. Guo, G. Hug, and O. K. Tonguz, "On the role of communications plane in distributed optimization of power systems," *IEEE Trans. Industr. Inform.*, vol. 14, no. 7, pp. 2903–2913, 2018.

[10] H. Li, C. Huang, G. Chen, X. Liao, and T. Huang, "Distributed consensus optimization in multiagent networks with time-varying directed topologies and quantized communication," *IEEE Trans. Cybern.*, vol. 47, no. 8, pp. 2044–2057, 2017.

[11] A. Nedic, A. Olshevsky, A. Ozdaglar, and J. N. Tsitsiklis, "Distributed subgradient methods and quantization effects," in *47th IEEE Conf. Decis. Control (CDC)*, 2008, pp. 4177–4184.

[12] D. Yuan, S. Xu, H. Zhao, and L. Rong, "Distributed dual averaging method for multi-agent optimization with quantized communication," *Syst. Control Lett.*, vol. 61, no. 11, pp. 1053–1061, 2012.

[13] L. Majzoobi, F. Lahouti, and V. Shah-Mansouri, "Analysis of distributed ADMM algorithm for consensus optimization in presence of node error," *IEEE Trans. Signal Process.*, vol. 67, no. 7, pp. 1774–1784, 2019.

[14] C. Shi and G. Yang, "Distributed optimization under unbalanced digraphs with node errors: Robustness of surplus-based dual averaging algorithm," *IEEE Trans. Control Netw. Syst.*, pp. 1–1, 2020.

[15] H. Li, B. Jin, and W. Yan, "Distributed model predictive control for linear systems under communication noise: Algorithm, theory and implementation," *Automatica*, vol. 125, p. 109422, 2021.

[16] S. Chen, W. Lan, J. Ma, and X. Yu, "Distributed optimization design of multi-agent systems with packet losses," in *IEEE 16th Int. Conf. Control Autom.*, 2020, pp. 1241–1246.

[17] X. Liang, Z. Li, W. Huang, Q. H. Wu, and H. Zhang, "Relaxed alternating direction method of multipliers for hedging communication packet loss in integrated electrical and heating system," *J. Mod. Power Syst. Clean Energy*, vol. 8, no. 5, pp. 874–883, 2020.

[18] B. Stott, J. Jardim, and O. Alsac, "DC power flow revisited," *IEEE Trans. Power Syst.*, vol. 24, no. 3, pp. 1290–1300, 2009.

[19] D. K. Molzahn and I. A. Hiskens, "A Survey of Relaxations and Approximations of the Power Flow Equations," *Found. Trends Electric Energy Syst.*, vol. 4, no. 1-2, pp. 1–221, February 2019.

[20] D. Gabay and B. Mercier, "A dual algorithm for the solution of nonlinear variational problems via finite element approximation," *Comput. Math. Appl.*, vol. 2, no. 1, pp. 17–40, 1976.

[21] R. Glowinski and A. Marroco, "Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires," *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique*, vol. 9, no. R2, pp. 41–76, 1975.

[22] A. X. Sun, D. T. Phan, and S. Ghosh, "Fully decentralized AC optimal power flow algorithms," in *IEEE Power & Energy Society General Meeting*, 2013, pp. 1–5.

[23] T. Erseghe, "Distributed optimal power flow using ADMM," *IEEE Trans. Power Syst.*, vol. 29, no. 5, pp. 2370–2380, 2014.

[24] M. Kraning, E. Chu, J. Lavaei, and S. Boyd, "Dynamic network energy management via proximal message passing," *Found. Trends Optimiz.*, vol. 1, no. 2, pp. 70–122, 2013.

[25] N. Michelena, H. Park, and P. Y. Papalambros, "Convergence properties of analytical target cascading," *AIAA J.*, vol. 41, no. 5, pp. 897–905, 2003.

[26] S. Tosserams, L. Etman, P. Papalambros, and J. Rooda, "An augmented Lagrangian relaxation for analytical target cascading using the alternating direction method of multipliers," *Struct. Multidiscip. Optimiz.*, vol. 31, no. 3, pp. 176–189, 2006.

[27] G. Cohen, "Auxiliary problem principle and decomposition of optimization problems," *J. Optimiz. Theory App.*, vol. 32, no. 3, pp. 277–305, 1980.

[28] L. Zhao, D. Zhu, and B. Jiang, "Auxiliary problem principle of augmented Lagrangian with varying core functions for large-scale structured convex problems," *arXiv:1512.04175*, 2015.

[29] D. Marco and D. L. Neuhoff, "The validity of the additive noise model for uniform scalar quantizers," *IEEE Trans. Inf. Theory*, vol. 51, no. 5, pp. 1739–1755, 2005.

[30] T. Zhang and Q. Zhu, "Dynamic differential privacy for ADMM-based distributed classification learning," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 1, pp. 172–187, 2017.

[31] M. Jeruchim, "Techniques for estimating the bit error rate in the simulation of digital communication systems," *IEEE J. Sel. Areas Commun.*, vol. 2, no. 1, pp. 153–170, 1984.

[32] F. Pasqualetti, R. Carli, and F. Bullo, "A distributed method for state estimation and false data detection in power networks," in *IEEE Int. Conf. Smart Grid Commun. (SmartGridComm)*, 2011, pp. 469–474.

[33] G. Haßlinger and O. Hohlfeld, "The gilbert-elliott model for packet loss in real time services on the internet," in *14th GI/ITG Conference-Measurement, Modelling and Evalutation of Computer and Communication Systems*. VDE, 2008, pp. 1–15.

[34] E. Martinian and C. Sundberg, "Decreasing distortion using low delay codes for bursty packet loss channels," *IEEE Trans. Multimedia*, vol. 5, no. 3, pp. 285–292, 2003.

[35] W. A. Bukhsh, A. Grothey, K. I. M. McKinnon, and P. A. Trodden, "Local solutions of the optimal power flow problem," *IEEE Trans. Power Syst.*, vol. 28, no. 4, pp. 4780–4788, 2013.

[36] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 12–19, 2011.

[37] I. Dunning, J. Huchette, and M. Lubin, "JuMP: A modeling language for mathematical optimization," *SIAM Rev.*, vol. 59, no. 2, pp. 295–320, 2017.

[38] C. Coffrin, R. Bent, K. Sundar, Y. Ng, and M. Lubin, "PowerModels.jl: An open-source framework for exploring power flow formulations," in *Power Systems Computation Conference (PSCC)*, 2018.

[39] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM Rev.*, vol. 59, no. 1, pp. 65–98, 2017.