An Online Gradient Algorithm for Optimal Power Flow on Radial Networks

Lingwen Gan and Steven H. Low, *Fellow, IEEE*

***Abstract*—We propose an online algorithm for solving optimal power flow (OPF) problems on radial networks where the controllable devices continuously interact with the network that implicitly computes a power flow solution given a control action. Collectively the controllable devices and the network implement a gradient projection algorithm for the OPF problem in real time. The key design feature that enables this approach is that the intermediate iterates of our algorithm always satisfy power flow equations and operational constraints. This is achieved by explicitly exploiting the network to implicitly solve power flow equations for us in real time at scale. We prove that the proposed algorithm converges to the set of local optima and provide sufficient conditions under which it converges to a global optimum. We derive an upper bound on the suboptimality gap of any local optimum. This bound suggests that any local minimum is almost as good as any strictly feasible point. We explain how to greatly reduce the gradient computation in each iteration by using approximate gradient derived from linearized power flow equations. Numerical results on test networks, ranging from 42-bus to 1990-bus, show a great speedup over a second-order cone relaxation method with negligible difference in objective values.**

***Index Terms*—Branch flow model, distflow equations, interior point method, online optimization algorithm, optimal power flow (OPF).**

I. Introduction

OPTIMAL power flow (OPF) is fundamental in power system operations as it underlies many applications such as economic dispatch, unit commitment, state estimation, stability and reliability assessment, volt/var control, demand response, etc. There has been a great deal of research on OPF since Carpentier's first formulation in 1962 [6]. An early solution appears in [11], [33] and extensive surveys can be found in e.g. [5], [7], [8], [16], [17], [23], [25]–[32]. Almost all algorithms in this literature are offline where one must wait till the iteration has converged to obtain a solution that can be applied to the network because the intermediate iterates of these algorithms do not satisfy power flow equations and therefore are not implementable. In this paper, we propose a different approach, motivated by the need to optimize the operation of a large network of distributed energy resources in distribution systems of the future, such as distributed wind and solar generations, electric vehicles, smart buildings, smart appliances, storage devices, smart inverters and other power electronics.

In this scenario an OPF problem has two set of variables $(x(t), y(t))$ where the independent variables $x(t)$ represents controllable devices and the dependent variables $y(t)$ are determined by power flow equations given an $x(t)$. Most existing algorithms update $(x(t), y(t))$ jointly in each iteration t offline say, using the Newton-Raphson method. These iterates typically do not satisfy power flow equations or operational constraints until the algorithm has converged. In contrast, our algorithm only updates $x(t)$, based on $(x(t-1), y(t-1))$, and applies the control $x(t)$ to the network in each iteration t. The network automatically computes a power flow solution $y(t)$ by implicitly solving the power flow equations. Our algorithm then updates $x(t+1)$ based on measurements of $y(t)$ and the cycle repeats.

One of the key computational challenges in almost all the offline AC OPF algorithms in the literature is implicitly solving power flow equations. Here, *we explicitly exploit the laws of physics to solve power flow equations in real time at scale for free over the network. The key advantage of this approach is that it can be used for continuous feedback optimization to track evolving network conditions in a plug-and-play framework.*

See [4] for a similar approach and [12] for a purely local algorithm, both for volt/var control.

Summary and contributions. We use the branch flow model (DistFlow equations) proposed in [1], [2] for radial networks and formulate the AC OPF problem in Section II. Our algorithm is a first-order gradient algorithm where in each iteration, derivatives of the objective function with respect to the controllable variables x are calculated, based on $(x(t), y(t))$, to compute the next control $x(t+1)$ in the direction of the negative gradient. We must ensure that, throughout the process, the intermediate results $(x(t), y(t))$ both (i) satisfy power flow equations and (ii) satisfy operational constraints (e.g., voltage magnitudes must lie within 5% of their nominal values) so that the control $x(t)$ can be safely applied to the network at each iteration t.

It is useful to treat our algorithm as a discrete-time feedback system:

$$\text{controller:} \qquad x(t+1) = g(x(t), y(t)) \tag{1a}$$

$$\text{network:} \qquad y(t) = f(x(t)) \tag{1b}$$

where (1a) is a gradient descent step and (1b) is a power flow solution. While we can design the function g and actively apply it to the network, the function f is determined by power flow

Manuscript received July 31, 2015; revised November 17, 2015; accepted December 17, 2015. Date of publication February 3, 2016; date of current version March 15, 2016. This work was supported in part by ARPA-E under Grant DE-AR0000226, in part by the DTRA under Grant HDTRA 1-15-1-0003 and in part by Skoltech through a collaboration agreement 1075-MRA.

The authors are with the Engineering and Applied Science Division, California Institute of Technology, Pasadena, CA 91125 USA (e-mail: lgan@caltech.edu; slow@caltech.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSAC.2016.2525598

equations through the implicit function theorem and enforced by the network. This design satisfies the first requirement. To satisfy the second requirement, especially for constraints on the dependent variables y that we do not explicitly control, we add a log-barrier function to the objective to prevent $y(t)$ from violating their constraints. The solution strategy is overviewed in Section III and the algorithm is detailed in Section IV.

We prove in Section V that algorithm (1) always converges to the set of local optima and provide sufficient conditions under which it converges to a global optimum. Moreover we derive an upper bound on the gap between the cost of any local optimum and the cost of an arbitrary feasible point that is a small distance away from the boundary of the feasible set. This bound suggests that any local minimum is almost as good as any strictly feasible point.

We present in Section VI several refinements and extensions to the basic gradient algorithm. For example, due to the implicit power flow solution (1b), the gradient computation in (1a) requires inverting a certain Jacobian matrix. This is inefficient for large networks in terms of both computational effort and communication requirement. We describe how to exploit the tree topology of the network to iteratively compute the gradient in (1a) without the need for matrix inversion. To further reduce the computational effort, we describe how to use linearized power flow equations to derive approximate gradients that avoids both matrix inversion and iterative calculation. These two methods greatly reduce the computational effort in each iteration of (1a), but does not directly address communication requirement. In [19] these algorithms are extended to their distributed versions that require communication only between neighboring buses.

While we discuss our algorithms mostly in the context of a single-phase network for simplicity of exposition, most distribution systems are multiphase unbalanced [10], [21], [24]. We provide a sketch on how these algorithms can be extended to multiphase unbalanced radial networks.

Finally we present in Section VII numerical experiments on 22 test networks with 42 buses to 1,990 buses. While semidefinite relaxation of OPF [25], [26] is able to compute globally optimal solutions, it takes a much longer time. In comparison, the algorithm developed in this paper takes a much shorter time and is able to obtain a close-to-optimal solution. Specifically, for all our test networks, the difference in objective values is below 10^{-5} between these two methods but the speedup is over 70x for large networks. It is therefore promising to further develop the algorithms in this paper for real-time applications.

We conclude in Section VIII. A key challenge to overcome is to minimize the measurement and communication requirements so that these algorithms can be implemented in real time by a large network of distributed energy resources, building on the ideas in [19].

II. Problem Formulation

A. Model

Consider a distribution network modeled as a *directed* (and connected) tree graph (N^+, E) where $N^+ := \{0\} \cup N$, $N := \{1, \dots, n\}$, and $E \subseteq N^+ \times N^+$. We will refer to each $i \in N^+$ as a "bus" or "node" and each $(i, j) \in E$ as a "line" or "link". Let $m := |E| = n$ be the number of lines in E. Let bus 0 be the root of the tree. For convenience only, we assume the graph is oriented such that each line $(i, j) \in E$ points away from the root. We may use (i, j) or $i \rightarrow j$ interchangeably to denote a line. For each $(i, j) \in E$, let $z_{ij} := r_{ij} + \mathbf{i}x_{ij}$ were $r_{ij} > 0$ and $x_{ij} > 0$ are the line resistance and reactance respectively.

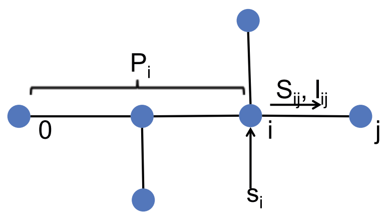


Fig. 1. Some of the notations.

For each bus $i \in N^+$, let V_i be the complex voltage at bus i and $v_i = |V_i|^2$ the square of its magnitude, e.g., if the voltage is $V_i = 1.05\angle 120°$ per unit, then $v_i = 1.05^2$. Bus 0 is the slack or substation bus and we assume as customary that $V_0 = 1\angle 0°$ pu. Let $s_i = p_i + \mathbf{i}q_i$ be the net complex power injection at bus i with p_i and q_i as the real and reactive power injections respectively. Let $\mathbb{P}_i$ denote the unique path from bus 0 to bus i. Since the network is radial (has a tree topology), the path $\mathbb{P}_i$ is well-defined.

For each line $(i, j) \in E$, let I_{ij} be the complex current and $\ell_{ij} = |I_{ij}|^2$ its squared magnitude, e.g., if the current is $I_{ij} = 0.5\angle 10°$, then $\ell_{ij} = 0.5^2$. Let $S_{ij} = P_{ij} + \mathbf{i}Q_{ij}$ be the *sending-end* complex power from buses i to j with P_{ij} and Q_{ij} as the real and reactive power respectively.

We will mainly be using branch flow models in real domain, so we will abuse notation to use s_i to denote either the complex number $p_i + \mathbf{i}q_i$ or the real pair (p_i, q_i) depending on the context; similarly for other variables $z_{ij}, V_i, S_{ij}, I_{ij}$. Some of the notations are summarized in Figure 1.

Let $x := (p_i, q_i, i \in N) \in \mathbb{R}^{2n}$ denote the bus injections[1] and $y := (p_0, q_0, v_i, i \in N; P_{ij}, Q_{ij}, \ell_{ij}, (i, j) \in E) \in \mathbb{R}^{3m+n+2}$ the other dependent variables. We assume x represents controllable devices and y uncontrollable network states. These variables, together with v_0, satisfy the power flow equations:

$$\sum_{k:\, j\to k} P_{jk} = P_{ij} - r_{ij}\ell_{ij} + p_j, \quad j \in N^+ \tag{2a}$$

$$\sum_{k:\, j\to k} Q_{jk} = Q_{ij} - x_{ij}\ell_{ij} + q_j, \quad j \in N^+ \tag{2b}$$

$$v_i - v_j = 2(r_{ij}P_{ij} + x_{ij}Q_{ij}) - |z_{ij}|^2\ell_{ij}, \ i \to j \tag{2c}$$

$$v_i\ell_{ij} = P_{ij}^2 + Q_{ij}^2, \quad i \to j \tag{2d}$$

where i in (2a) and (2b) is the unique bus between bus 0 and bus j. Note that the number $2(m + n + 1) = 4n + 2$ of equations is the same as the number of variables in y. The equations (2), called the DistFlow equations, are first proposed in [1], [2] and

[1]Even though x is also used to denote line reactances, the meaning should be clear from the context.

are valid only for radial networks (see [14] for the generalization to mesh networks). Discrete devices like tap-changers and circuit breakers are not modeled.

B. Optimal Power Flow (OPF)

The OPF problem that we seek to solve is:

$$\min \quad \sum_{i=0}^{n} a_i p_i^2 + b_i p_i \tag{3a}$$

$$\begin{aligned} \text{over} \quad & x := (p_i, q_i, i \in N) \\ & y := (p_0, q_0, v_i, i \in N; P_{ij}, Q_{ij}, \ell_{ij}, (i,j) \in E) \\ \text{s.t.} \quad & (2) \end{aligned}$$

$$\underline{v}_i \le v_i \le \overline{v}_i, \qquad i \in N \tag{3b}$$

$$\underline{p}_i \le p_i \le \overline{p}_i, \underline{q}_i \le q_i \le \overline{q}_i, \qquad i \in N \tag{3c}$$

The objective function (3a) is assumed to be separable, quadratic, and purely a function of real power injections p. Equation (3b) represents the voltage constraints, and (3c) represents the power injection constraints. If there is no bound on an injection x_j then we set $\underline{x}_j = -\infty$ or/and $\overline{x}_j = \infty$. On the other hand if an injection x_j is fixed (e.g. a constant-power load) then we set $\underline{x}_j = \overline{x}_j$ to the specified value.

OPF as defined (3) is a simplified version that ignores other important constraints such as line limits, security constraints, stability constraints, and chance constraints. Some of these (e.g., including shunt elements or line limits on ℓ_{ij}) can be incorporated without much change to the results in this paper.

III. Solution Strategy

We are motivated by the need to optimize the operation of a large network of distributed energy resources in the future, such as distributed wind and solar generations, electric vehicles, smart buildings, smart appliances, storage devices, and power electronics. We model these controllable devices by injections $x := (p_i, q_i, i \in N)$. We will develop a gradient projection algorithm that iteratively solves an approximate version of the OPF problem (3) as follows: at each iteration t,

1) the algorithm applies the current iterate $x(t)$ to the network;
2) the network automatically computes the dependent variables $y(t)$ according to the power flow equations (2);
3) the algorithm computes $x(t+1)$ based on $(x(t), y(t))$ using a gradient projection algorithm; goto 1 until converge.

Hence we explicitly exploit the law of physics, modeled by power flow equations (2), to carry out part of the gradient projection algorithm to solve approximately our OPF problem. The key advantage of this approach is that, by applying intermediate iterates $(x(t), y(t))$ to the network at each t, it can be used in real time for continuous feedback control to track evolving network conditions. This is in stark contrast to most traditional OPF algorithms where intermediate iterates $(x(t), y(t))$ do not satisfy power flow equations and therefore cannot be implemented until the algorithms have converged. We will comment on the communication requirements to implement this strategy in Section VI.

We now describe, in two steps the approximate OPF problem that we solve.

A. Injection Optimization

We first transform (3) into one where the optimization variable is only x. To this end, let

$$X := \{ (p_i, q_i) \mid \underline{p}_i \le p_i \le \overline{p}_i, \underline{q}_i \le q_i \le \overline{q}_i, \ i \in N \} \tag{4}$$

Write the power flow equations (2), in terms of a continuously differentiable function $F : X \to \mathbb{R}^{2(m+n+1)}$, as:

$$F(x, y) = 0 \tag{5}$$

We make the following assumption throughout the paper:

A1: Given any $\tilde{x} \in X$ (and v_0), there is a unique $\tilde{y}$ that solves the power flow equation (5) and satisfies the voltage constraints (3b). Moreover the Jacobian matrix $\partial_y F(\tilde{x}, \tilde{y})$ at $(\tilde{x}, \tilde{y})$ is nonsingular.

A1 is widely believed to hold in practice for radial networks and a rigorous proof for some special cases are provided in [9].

Equation (5) hence defines implicitly a function $y = y(x)$ over X:

$$\begin{aligned} & p_0 := p_0(x), q_0 := q_0(x); \\ & v_i := v_i(x), && i \in N \\ & P_{ij} := P_{ij}(x), Q_{ij} := Q_{ij}(x), \ell_{ij} = \ell_{ij}(x), && i \to j \end{aligned}$$

such that $F(x, y(x)) = 0$. Then the OPF problem (3) can be written in terms of x:

$$\min \quad a_0 p_0^2(x) + b_0 p_0(x) + \sum_{i=1}^{n} (a_i p_i^2 + b_i p_i) \tag{6a}$$

$$\text{over} \quad x := (p_i, q_i, i \in N) \in X \tag{6b}$$

$$\text{s.t.} \quad \underline{v}_i \le v_i(x) \le \overline{v}_i, \qquad i \in N \tag{6c}$$

where X is defined in (4). While (6) is equivalent to (3), (6) has much fewer optimization variables and is therefore potentially more efficient to compute. Note however that while (3b) is linear in v_i, (6c) is generally nonlinear in x.

B. Modified OPF

The nonlinear voltage constraints (6c) couple the variables $x = (p_i, q_i, i \in N)$. To further simplify the feasible set to facilitate a distributed algorithm (see [19]) where each bus i updates its own (p_i, q_i) locally, we replace the hard constraints (6c) by a log-barrier function in the objective that prevents the voltages from violating (6c):

$$\begin{aligned} L(x; \mu) := {} & a_0 p_0^2(x) + b_0 p_0(x) + \sum_{i=1}^{n} (a_i p_i^2 + b_i p_i) \\ & - \underline{\mu} \sum_{i=1}^{n} \ln(v_i(x) - \underline{v}_i) - \overline{\mu} \sum_{i=1}^{n} \ln(\overline{v}_i - v_i(x)) \end{aligned} \tag{7}$$

where $\mu = (\underline{\mu}, \overline{\mu}) > 0$ componentwise. Since

$$\lim_{t \downarrow \underline{v}_i} -\underline{\mu} \ln(t - \underline{v}_i) = \infty, \quad \lim_{t \uparrow \overline{v}_i} -\overline{\mu} \ln(\overline{v}_i - t) = \infty, \quad i \in N$$

minimizing L ensures $v_i(x)$ stays in $(\underline{v}_i, \overline{v}_i)$ as long as $\mu > 0$. Moreover L converges to the original objective function as μ tends to zero:

$$\lim_{\mu \downarrow 0} L(x; \mu) = a_0 p_0^2(x) + b_0 p_0(x) + \sum_{i=1}^{n} (a_i p_i^2 + b_i p_i)$$

and therefore solving OPF is similar to minimizing $L(x; \mu)$ with small $\mu > 0$.

To summarize, our goal is to solve

$$\begin{aligned} \textbf{OPF}(\mu): \quad & \min \quad L(x; \mu) & \text{(8a)} \\ & \text{over} \quad x := (p_i, q_i, i \in N) \in X & \text{(8b)} \end{aligned}$$

for a decreasing sequence of $\mu > 0$, where the objective L is defined in (7) and the feasible set X in (4).

IV. Gradient Projection Algorithm

A. Solving OPF(μ) for Fixed μ

For each $\mu > 0$ our algorithm is a gradient projection algorithm that takes the form

$$x(t+1) = [x(t) - \eta(t) \partial_x L(x(t); \mu)]_X$$

where $\partial_x L$ denotes the partial derivative of L with respect to x, $\eta(t) > 0$ is a stepsize at iteration t, and $[x]_X$ projects x onto X. Under assumption A1, the implicit function theorem implies that the Jacobian matrix $\partial_x y(x)$ of $y(x)$ with respect to x exists in X and is continuous. It is given by:

$$\partial_x y(x) = -\left[\partial_y F(x, y(x))\right]^{-1} \partial_x F(x, y(x))$$

Writing the objective $L(x; \mu) = L(x, y(x); \mu)$, we have

$$\begin{aligned} \partial_x L &= \partial_x L(x, y(x); \mu) \; + \; \partial_y L(x, y(x); \mu) \, \partial_x y(x) \\ &= \partial_x L(x, y(x); \mu) \; - \; \partial_y L(x, y(x); \mu) \\ & \quad \left[\partial_y F(x, y(x))\right]^{-1} \partial_x F(x, y(x)) \qquad (9) \end{aligned}$$

For a gradient projection algorithm that uses the network to compute the dependent variables $y(t)$, it is practical to use a constant stepsize $\eta(t) \equiv \eta$, especially if we wish the algorithm to track slowly evolving network conditions. (See Section VI-D for time-varying stepsizes.) As long as η is small enough, the iterates $(x(t), y(t))$ will converge to a local minimum, as we now explain.

Definition 1: A point $x = (p, q) \in X$ is a *local minimum* of L if

$$\left\langle \partial_{x_i} L, \, \tilde{x}_i - x_i \right\rangle \geq 0, \quad \tilde{x}_i \in (\underline{x}_i, \overline{x}_i), i \in N$$

Let $\mathbb{O}^* \subseteq X$ denote the set of local minima of L.

Consider the sequence $(x(t), y(t))$ generated by the **gradient projection algorithm**:

$$\begin{aligned} x(t+1) &= [x(t) - \eta \, \partial_x L(x(t); \mu)]_X & \text{(10a)} \\ y(t) &= y(x(t)) & \text{(10b)} \end{aligned}$$

where $\partial_x L$ is given by (9). We will interchangeably refer to the sequence by $x(t)$ or $(x(t), y(t)) := (x(t), y(x(t)))$. A point x^* or equivalently (x^*, y^*) is called a *limit point* of $x(t)$ or equivalently $(x(t), y(t))$ if there is a subsequence $x(t_k)$ or equivalently $(x(t_k), y(t_k))$ such that $x(t_k) \to x^*$ and $y(t_k) \to y^*$ as $k \to \infty$. To estimate a bound on the stepsize η, note that since F in (2) is twice continuously differentiable, the implicit function theorem implies that $y(x)$ is also twice continuously differentiable over X and hence the Hessian matrix $\partial_{xx} L(x; \mu)$ exists and is continuous (see below for an explicit expression). Therefore, since X is compact, $\partial_{xx} L(x; \mu)$ is bounded uniformly on X. This implies that $\partial_x L$ is Lipschitz over X, i.e., there exists an K such that

$$\|\partial_x L(x'; \mu) - \partial_x L(x; \mu)\|_2 \leq \; K \|x' - x\|_2, \quad x, x' \in X$$

Theorem 1 (Local optimality): Suppose assumption A1 holds and the stepsize $0 < \eta < 2/K$.

1) Every limit point x^* of $x(t)$ is a local minimum.
2) The sequence $x(t)$ converges to the set $\mathbb{O}^*$ of local minima, i.e., $\min_{x \in \mathbb{O}^*} \|x(t) - x\|_2 \to 0$ as $t \to \infty$.
3) If there are only finitely many local minima, then the sequence $x(t)$ itself converges to a local minimum.

The theorem is proved in Appendix A

B. Iterating Over μ

To approximately solve (6), we solve OPF(μ) in (8) with different values of μ. Specifically, let $\mu_1, \mu_2, \ldots, \mu_K$ denote a sequence of $\mu > 0$ that approaches 0. Given a feasible initial point $x^{(0)}$, we solve OPF(μ_1), using the gradient projection algorithm (10), with initial point $x^{(0)}$ to obtain $x^{(1)}$, then solves OPF(μ_2) with initial point $x^{(1)}$ to obtain $x^{(2)}$, …, and finally solves OPF(μ_K) with initial point $x^{(K-1)}$ to obtain the final solution $x^{(K)}$, which solves (6) approximately.

V. Optimality Analysis

The gradient projection algorithm (10) may converge to a local minimum (Theorem 1). In this section, we discuss conditions under which it converges to a global minimum and bound the suboptimality when it does not.

A. Global Optimality and Convexity

The condition in Definition 1 is necessary for an x to be a global minimum of OPF(μ). If L were convex, it would also be sufficient. Then Theorem 1 would have implied that any limit point of the gradient projection algorithm (10) would be globally optimal. L however is in general nonconvex, but we now show that it is "nearly" convex in the sense that the Hessian matrix $H(x; \mu) := \partial_{xx} L(x; \mu)$ is positive semidefinite on a large portion of the feasible set of (6).

1) Computing Hessian H: From (7) we have, for $i \in N$,

$$\partial_{x_i} L = (2a_0 p_0(x) + b_0)\partial_{x_i} p_0(x) + (2a_i p_i + b_i)\mathbb{1}(x_i = p_i) - \sum_{j=1}^{n}\left(\frac{\underline{\mu}}{v_j(x) - \underline{v}_j} + \frac{\overline{\mu}}{v_j(x) - \overline{v}_j}\right)\partial_{x_i} v_j(x) \qquad (11)$$

where $\mathbb{1}(\alpha) = 1$ if α is true and 0 otherwise. Let $\mathrm{Diag}(a)$ denote the diagonal matrix with diagonal entries $(a_1, a_2, \ldots, a_n)$. Let

$$c_0(x) := 2a_0 p_0(x) + b_0 \qquad (12a)$$

be the marginal cost of bus 0 power $p_0(x)$ and for $k \in N$

$$\alpha_k(x) := \frac{\underline{\mu}}{(v_k(x) - \underline{v}_k)^2} + \frac{\overline{\mu}}{(v_k(x) - \overline{v}_k)^2} \qquad (12b)$$

$$\beta_k(x) := \frac{\underline{\mu}}{v_k(x) - \underline{v}_k} + \frac{\overline{\mu}}{v_k(x) - \overline{v}_k} \qquad (12c)$$

Then, for $i, j \in N$,

$$\begin{aligned}\frac{\partial^2 L}{\partial p_i \partial p_j} &= 2a_0 \frac{\partial p_0}{\partial p_i}\frac{\partial p_0}{\partial p_j} + (2a_0 p_0 + b_0)\frac{\partial^2 p_0}{\partial p_i \partial p_j} + 2a_i \mathbb{1}(i = j) \\ &\quad + \sum_{k=1}^{n}\left[\frac{\underline{\mu}}{(v_k - \underline{v}_k)^2} + \frac{\overline{\mu}}{(v_k - \overline{v}_k)^2}\right]\frac{\partial v_k}{\partial p_i}\frac{\partial v_k}{\partial p_j} \\ &\quad - \sum_{k=1}^{n}\left[\frac{\underline{\mu}}{v_k - \underline{v}_k} + \frac{\overline{\mu}}{v_k - \overline{v}_k}\right]\frac{\partial^2 v_k}{\partial p_i \partial p_j} \\ &= 2a_0\Big[\partial_p p_0 \partial_p^T p_0\Big]_{ij} + c_0(x)\big[\partial_{pp} p_0\big]_{ij} + 2\left[\mathrm{Diag}(a)\right]_{ij} \\ &\quad + \sum_{k=1}^{n}\alpha_k(x)\left[\partial_p v_k \partial_p^T v_k\right]_{ij} - \sum_{k=1}^{n}\beta_k(x)\left[\partial_{pp} v_k\right]_{ij}\end{aligned}$$

Hence

$$\partial_{pp} L = 2\,\mathrm{Diag}(a) + 2a_0\,\partial_p p_0 \partial_p^T p_0 + c_0(x)\,\partial_{pp} p_0 + \sum_{k=1}^{n}\alpha_k(x)\,\partial_p v_k \partial_p^T v_k - \sum_{k=1}^{n}\beta_k(x)\,\partial_{pp} v_k$$

Similarly one can compute

$$\begin{aligned}\partial_{qq} L &= 2a_0\,\partial_q p_0 \partial_q^T p_0 + c_0(x)\,\partial_{qq} p_0 + \sum_{k=1}^{n}\alpha_k(x)\,\partial_q v_k \partial_q^T v_k - \sum_{k=1}^{n}\beta_k(x)\,\partial_{qq} v_k \\ \partial_{qp} L &= 2a_0\,\partial_q p_0 \partial_p^T p_0 + c_0(x)\,\partial_{qp} p_0 + \sum_{k=1}^{n}\alpha_k(x)\,\partial_q v_k \partial_p^T v_k - \sum_{k=1}^{n}\beta_k(x)\,\partial_{qp} v_k\end{aligned}$$

Hence the Hessian matrix is

$$\begin{aligned}H(x;\mu) &= \begin{bmatrix}\partial_{pp} L & \partial_{pq} L \\ \partial_{qp} L & \partial_{qq} L\end{bmatrix} \\ &= c_0(x)\,\partial_{xx} p_0 + 2\begin{bmatrix}\mathrm{Diag}(a) & \\ & 0\end{bmatrix} + 2a_0\big[\partial_x p_0\big]\big[\partial_x p_0\big]^T \\ &\quad + \sum_{k=1}^{n}\alpha_k(x)\,[\partial_x v_k]\,[\partial_x v_k]^T - \sum_{k=1}^{n}\beta_k(x)\,\partial_{xx} v_k \qquad (13)\end{aligned}$$

2) Conditions for Convexity: Consider:

C1) The quadratic coefficients are nonnegative, i.e., $a_i \geq 0$ for $i = 0, 1, \ldots, n$;

C2) The marginal cost at bus 0 is nonnegative, i.e., $c_0(x) := 2a_0 p_0(x) + b_0 \geq 0$ on X.

C3) $p_0(x)$ is convex on X;

C4) $v_k(x)$ is concave on X for $k = 1, 2, \ldots, n$.

Conditions C1 and C2 are typically satisfied in practice. We will justify C3 and C4 below after we have explained the implication of these conditions. The first term on the right-hand side of (13) is positive semidefinite (psd) under C2 and C3, the second and third terms are psd under C1, and the fourth term is psd since $\alpha(x)$ is positive. Hence, under C1–C4, the Hessian $H(x;\mu)$ is psd at $x \in X$ where $\beta(x) \geq 0$ (the set A in Lemma 2). We summarize.

Lemma 2: Assume C1–C4 hold. Then $H(x;\mu) \succeq 0$ on

$$A := \left\{x \in X \,|\, v(x) \leq \frac{\underline{\mu}}{\overline{\mu} + \underline{\mu}}\overline{v} + \frac{\overline{\mu}}{\overline{\mu} + \underline{\mu}}\underline{v}\right\}.$$

In particular, $H(x;\mu) \succeq 0$ on X if $\mu := (\underline{\mu}, \overline{\mu}) = 0$ or $\overline{v} = \infty$.

Lemma 2 implies that $L(x;\mu)$ is convex in x over A. This, together with Theorem 1 directly implies

Theorem 3 (Global optimality): Suppose conditions A1, C1–C4 hold. Suppose the stepsize of the gradient projection algorithm (10) satisfies $0 < \eta < 2/K$. Let $x(t)$ be a sequence generated by the algorithm. If A contains all local optima, then:

1) Every limit point x^* of $x(t)$ is a global minimum of OPF(μ) in (8).
2) If there are only finitely many local minima, then the sequence $x(t)$ itself converges minimum.

We make two remarks on the region A in Lemma 2 over which $H(x;\mu) \succeq 0$ or equivalently $L(x)$ is convex. First, μ is equivalently 0 if voltage constraints do not bind, and that $\overline{v}$ is equivalently ∞ if voltage upper bound constraints do not bind. Hence, if voltage upper bound constraints do not bind, then $H(x;\mu) \succeq 0$ on $A = X$. Second, let

$$\tilde{X} := \{x \in X \,|\, \underline{v}_i \leq v_i(x) \leq \overline{v}_i,\ i \in N\} \qquad (14)$$

denote the (generally nonconvex) feasible set of (6). If the sequence $\mu_1, \mu_2, \ldots, \mu_k, \ldots$ of μ is chosen according to

$$\underline{\mu}_k = \delta_k, \overline{\mu}_k = \delta_k^2, \quad k = 1, 2, \ldots$$

where $\lim_{k\to\infty}\delta_k = 0$, then the difference set $\tilde{X}\backslash A$ vanishes as $k \to \infty$. We emphasize however that Lemma 2 does not guarantee $L(x;\mu)$ to be convex over $\tilde{X}$, though the set A can be arbitrarily close to $\tilde{X}$ with carefully chosen μ.

3) Justification of C3–C4: To justify condition C3, consider the following second-order cone program (SOCP) for each $x := (p, q) \in X$:

$$\begin{aligned}\textbf{SOCP}-p_0(x): \quad &\min \quad p_0 \\ &\text{over} \quad y := (p_0, q_0, v, P, Q, \ell) \\ &\text{s.t.} \quad (2a)\text{–}(2c) \\ &\qquad v_i \ell_{ij} \geq P_{ij}^2 + Q_{ij}^2,\ i \to j \in E \qquad (15)\end{aligned}$$

Replacing the quadratic equality in the power flow equations (2d) by inequality in (15) relaxes the (nonconvex) set of power flow solutions to a superset that is a (convex) second-order cone, making SOCP-$p_0(x)$ a convex program. The idea of SOCP relaxation of the DistFlow equations is first proposed in [13], [14] and used to convexify OPF problems. We say that SOCP-$p_0(x)$ is *exact* if every of its optimal solutions attains equality in (15) and therefore satisfies the power flow equations (2). In that case, one can obtain a globally optimal solution of a nonconvex OPF problem by solving its SOCP relaxation. Even though SOCP relaxations may not be exact in general, there are various sufficient conditions that guarantee the exactness of SOCP relaxations for radial networks, e.g., [13], [14], [20]; see also [25], [26] for extensive references. Moreover SOCP relaxations for practical radial networks are often exact even when these sufficient conditions are not satisfied. This is important for us because of the following result that justifies condition C3. It is proved in Appendix B.

Theorem 4 (Convexity of p_0): Let C be convex. If SOCP-$p_0(x)$ is exact for $x \in C$, then $p_0(x)$ is convex on C.

For C4, consider the following second-order cone program (SOCP) for each $x := (p, q) \in X$ and each $k \in N$:

$$\begin{aligned} \textbf{SOCP}-v_k(x): \quad & \max \quad v_k \\ & \text{over} \quad y := (p_0, q_0, v, P, Q, \ell) \\ & \text{s.t.} \quad (2a)-(2c) \\ & \qquad v_i \ell_{ij} \geq P_{ij}^2 + Q_{ij}^2, \; i \to j \in E \end{aligned} \tag{16}$$

Similarly the following result justifies condition C4: if SOCP-$v_k(x)$ is exact on X, then $v_k(x)$ is concave on X. Its proof is similar to that of Theorem 4 and omitted.

Theorem 5 (Convexity of v_k): Let C be convex and $k \in N$. If SOCP-$v_k(x)$ is exact for $x \in C$, then $v_k(x)$ is concave on C.

In summary, C3 and C4 hold under various conditions for the exactness of SOCP relaxations of OPF problems; see [13], [14], [20], [26].

B. Suboptimality Bound

When Theorem 3 does not hold, i.e., when a limit point x^* generated by the gradient projection algorithm (10) is not in A, x^* may not be a global minimum of OPF(μ) in (8). Theorem 1 still guarantees that x^* is a local minimum. We now bound the gap between the cost $L(x^*; \mu)$ of such a local minimum x^* and the cost $L(\tilde{x}; \mu)$ of an arbitrary feasible point $\tilde{x} \in \tilde{X}$ of the original problem (6), where $\tilde{X}$ is defined in (14). It suggests that any local minimum x^* is almost as good as any strictly feasible point of (6).

Theorem 6 (Suboptimality bound): Assume C1–C4 hold. Let $x^* = (p^*, q^*)$ be a local minimum of $L(x; \mu)$ and $\tilde{x} = (\tilde{p}, \tilde{q}) \in \tilde{X}$ be feasible for (6). Let $\Delta x := \tilde{x} - x^*$. Define for $\theta \in (0, 1)$,

$$\partial_q v := \begin{bmatrix} \partial_q v_1 & \partial_q v_2 & \cdots & \partial_q v_n \end{bmatrix}$$

$$r(\theta) := \frac{1}{2} \left[\partial_q v(x^*) \right]^{-1} \begin{bmatrix} \Delta x^T \cdot \partial_{xx} v_1(x^* + \theta \Delta x) \cdot \Delta x \\ \Delta x^T \cdot \partial_{xx} v_2(x^* + \theta \Delta x) \cdot \Delta x \\ \vdots \\ \Delta x^T \cdot \partial_{xx} v_n(x^* + \theta \Delta x) \cdot \Delta x \end{bmatrix}$$

If $\tilde{q} + r(\theta) \in \left(\underline{q}, \overline{q} \right)$ for $\theta \in (0, 1)$, then

$$L(x^*; \mu) - L(\tilde{x}; \mu) \leq \left(2a_0 p_0(x^*) + b_0 \right) \cdot \partial_q p_0(x^*) \cdot r(\theta^*) - a_0 (p_0(x^*) - p_0(\tilde{x}))^2$$

for some $\theta^* \in (0, 1)$.

The theorem bounds the gap between the cost $L(x^*; \mu)$ of any local minimum x^* and the cost $L(\tilde{x}; \mu)$ of any feasible point $\tilde{x} \in \tilde{X}$ of (6) that is $r(\theta)$ away from the boundary of X (not $\tilde{X}$). In practice, the term $r(\theta)$ is small for all $\theta \in (0, 1)$. Hence, $\tilde{x}$ can be taken from a large portion of $\tilde{X}$. The derivative $\partial_q p_0$ is typically small and therefore the upper bound on the gap is approximately zero. Hence Theorem 6 roughly says that *any local minimum x^* is no worse than almost any strictly feasible point $\tilde{x}$ of* (6). It is proved in Appendix C.

VI. Refinements and Extensions

In this section we describe several refinements to the basic gradient projection algorithm (10) discussed in Section IV.

A. Overview

First it is difficult to apply the basic algorithm for real-time continuous feedback optimization because it needs to invert the Jacobian matrix $\partial_y F$ in each iteration (see (9)). A natural implementation takes a centralized approach that uses global information for matrix inversion. This is inefficient for large networks in terms of both computational effort and communication requirement. We partially address this in Sections VI-B and VI-C. In Section VI-B we describe a method that exploits the tree topology of the network to iteratively compute the gradient $\partial_x L$ without having to invert the Jacobian matrix $\partial_y F$. In Section VI-C we use linearized power flow equations to derive approximate gradients $\partial_x \hat{L}$ that avoids both matrix inversion and iterative calculation. These two methods greatly reduce the computational effort in each iteration, but does not directly address communication requirement. In [19] these algorithms are extended to their distributed versions that require communication only between neighboring buses.

Second the basic algorithm of Section IV uses a constant stepsize η. The bound on η in Theorem 1 is typically conservative and there is no easy way to determine a constant η a priori that is not conservative but guarantees convergence. In Section VI-D we describe a gradient algorithm that uses time-varying stepsize $\eta(t)$ obtained through line search. It can be proved that it retains the convergence properties of the basic algorithm [19].

While we discuss our algorithms mostly in the context of a single-phase network in this paper, most distribution systems are multiphase unbalanced. In Section VI-E we provide a sketch on how these algorithms can be extended to multiphase unbalanced networks based on the model in [21] for multiphase radial networks.

B. Gradient Computation $\partial_x L$

We present an iterative algorithm to compute $\partial_x L$ in (9). It exploits the tree topology to avoid inverting the matrix

$\partial_y F(x, y(x))$. It is similar to backward-forward sweep in the literature to compute power flow solutions and similarly enjoys fast convergence. In our experience, the algorithm for computing $\partial_x L$ typically converges in just a few iterations, even though there is no formal proof of convergence for general backward-forward sweep to our knowledge.

Recall $\partial_x L$ from (11). We first express $\partial_x p_0(x)$ in terms of $(\partial_x v, \partial_x P, \partial_x Q, \partial_x \ell)$. Sum up (2a) for $j \in N^+$ to obtain

$$\sum_{i=0}^{n} p_i = \sum_{i\to j} r_{ij}\ell_{ij} = \sum_{i\to j} r_{ij}\frac{P_{ij}^2 + Q_{ij}^2}{v_i}.$$

Hence for $i \in N$

$$\begin{aligned}\partial_{x_i} p_0 &= -\mathbb{1}(x_i = p_i) + \sum_{k\to l} r_{kl}\,\partial_{x_i}\left(\frac{P_{kl}^2 + Q_{kl}^2}{v_k}\right)\\ &= -\mathbb{1}(x_i = p_i) + \sum_{k\to l} r_{kl}\left(\frac{2P_{kl}}{v_k}\,\partial_{x_i}P_{kl}\right.\\ &\qquad \left. + \frac{2Q_{kl}}{v_k}\,\partial_{x_i}Q_{kl} - \frac{\ell_{kl}}{v_k}\,\partial_{x_i}v_k\right) \qquad (17)\end{aligned}$$

Next, we derive $(\partial_x v, \partial_x P, \partial_x Q, \partial_x \ell)$ from (2): for all $i \to j$

$$\begin{aligned}\partial_x P_{ij} &= r_{ij}\partial_x\ell_{ij} - \partial_x p_j + \sum_{k:\, j\to k}\partial_x P_{jk}\\ \partial_x Q_{ij} &= x_{ij}\partial_x\ell_{ij} - \partial_x q_j + \sum_{k:\, j\to k}\partial_x Q_{jk}\\ \partial_x v_j &= \partial_x v_i - 2\left(r_{ij}\partial_x P_{ij} + x_{ij}\partial_x Q_{ij}\right) + |z_{ij}|^2\partial_x\ell_{ij}\\ \partial_x\ell_{ij} &= \frac{2P_{ij}}{v_i}\partial_x P_{ij} + \frac{2Q_{ij}}{v_i}\partial_x Q_{ij} - \frac{\ell_{ij}}{v_i}\partial_x v_i\end{aligned}$$

Let I denote the 2×2 identity matrix. Eliminate $\partial_x\ell_{ij}$ to obtain for $i \to j$

$$\begin{pmatrix}\partial_x P_{ij}\\ \partial_x Q_{ij}\end{pmatrix} = \left[I - \frac{2}{v_i}\begin{pmatrix}r_{ij}\\ x_{ij}\end{pmatrix}\begin{pmatrix}P_{ij} & Q_{ij}\end{pmatrix}\right]^{-1} \left[\sum_{k:\, j\to k}\begin{pmatrix}\partial_x P_{jk}\\ \partial_x Q_{jk}\end{pmatrix} - \begin{pmatrix}\partial_x p_j\\ \partial_x q_j\end{pmatrix} - \begin{pmatrix}r_{ij}\\ x_{ij}\end{pmatrix}\frac{\ell_{ij}}{v_i}\partial_x v_i\right] \qquad (18a)$$

$$\begin{aligned}\partial_x v_j = {}&\left(1 - |z_{ij}|^2\frac{\ell_{ij}}{v_i}\right)\partial_x v_i - 2\left(r_{ij} - |z_{ij}|^2\frac{P_{ij}}{v_i}\right)\partial_x P_{ij}\\ &- 2\left(x_{ij} - |z_{ij}|^2\frac{Q_{ij}}{v_i}\right)\partial_x Q_{ij} \qquad (18b)\end{aligned}$$

Since the network is radial, (18) defines a recursion that allows us to compute $\partial_x(p_0, q_0, v, P, Q)$ efficiently using a backward-forward sweep, as shown in Algorithm 1. Then the gradient $\partial_x L$ can be computed from (11) and (17) using the output $\partial_x p_0(x)$ and $\partial_x v(x)$ of Algorithm 1.

C. Approximate Gradient $\partial_x \hat{L}$

To further simplify computation, we can approximate gradient using a linearization of the power flow equations (2). The terms ℓ in (2a)–(2c) are typically much smaller than the other

Algorithm 1. Compute partial derivatives

```
Input: network graph $(N^+, E)$, power flow solution
    $(p, q, v, P, Q, \ell)$, stopping criterion $\epsilon$.
Output: $\partial_x(p_0, q_0, v, P, Q)$ where $x = (p_i, q_i, i \in N)$.
 1: Initialization:  $\partial_x v_i \leftarrow 0$ for $i = 0, 1, \ldots, n$;
 2: Backward sweep: from the leafs towards the root, compute
      $\begin{pmatrix}\partial_x P_{ij}\\ \partial_x Q_{ij}\end{pmatrix} \leftarrow \left[I - \frac{2}{v_i}\begin{pmatrix}r_{ij}\\ x_{ij}\end{pmatrix}\begin{pmatrix}P_{ij} & Q_{ij}\end{pmatrix}\right]^{-1}$
      $\left[\sum_{k:\, j\to k}\begin{pmatrix}\partial_x P_{jk}\\ \partial_x Q_{jk}\end{pmatrix} - \begin{pmatrix}\partial_x p_j\\ \partial_x q_j\end{pmatrix} - \begin{pmatrix}r_{ij}\\ x_{ij}\end{pmatrix}\frac{\ell_{ij}}{v_i}\partial_x v_i\right]$
 3: Forward sweep: from the root towards the leafs, compute
      $\partial_x v_j \leftarrow \left(1 - |z_{ij}|^2\frac{\ell_{ij}}{v_i}\right)\partial_x v_i - 2\left(r_{ij} - |z_{ij}|^2\frac{P_{ij}}{v_i}\right)\partial_x P_{ij}$
      $- 2\left(x_{ij} - z_{ij}^2\frac{Q_{ij}}{v_i}\right)\partial_x Q_{ij}$
 4: if update in $\partial_x(P, Q, v) > \epsilon$
        go to 2;
    end
 5: Return value: $\partial_x v$ and
      $\begin{pmatrix}\partial_x p_0\\ \partial_x q_0\end{pmatrix} \leftarrow \sum_{k:\, 0\to k}\left[I - \frac{2}{v_0}\begin{pmatrix}r_{0k}\\ x_{0k}\end{pmatrix}\begin{pmatrix}P_{0k} & Q_{0k}\end{pmatrix}\right]^{-1}$
      $\left[\sum_{l:\, k\to l}\begin{pmatrix}\partial_x P_{kl}\\ \partial_x Q_{kl}\end{pmatrix} - \begin{pmatrix}\partial_x p_k\\ \partial_x q_k\end{pmatrix} - \begin{pmatrix}r_{0k}\\ x_{0k}\end{pmatrix}\frac{\ell_{0k}}{v_0}\partial_x v_0\right]$
```

terms. Ignoring ℓ in (2) leads to linearized power flow equations (called *simplified DistFlow* equations in [2]):

$$\begin{aligned}\sum_{i:\, i\to j}\hat{P}_{ij} + p_j &= \sum_{k:\, j\to k}\hat{P}_{jk}, & j \in N^+\\ \sum_{i:\, i\to j}\hat{Q}_{ij} + q_j &= \sum_{k:\, j\to k}\hat{Q}_{jk}, & j \in N^+\\ \hat{v}_i - \hat{v}_j &= 2(r_{ij}\hat{P}_{ij} + x_{ij}\hat{Q}_{ij}), & i \to j \in E\end{aligned}$$

Hence

$$\begin{aligned}\partial_x\hat{P}_{ij} &= \sum_{k:\, j\to k}\partial_x\hat{P}_{jk} - \partial_x p_j, & i \to j\\ \partial_x\hat{Q}_{ij} &= \sum_{k:\, j\to k}\partial_x\hat{Q}_{jk} - \partial_x q_j, & i \to j\\ \partial_x\hat{v}_j &= \partial_x\hat{v}_i - 2r_{ij}\partial_x\hat{P}_{ij} - 2x_{ij}\partial_x\hat{Q}_{ij}, & i \to j\end{aligned}$$

Let $i \wedge j$ denote the joint of buses i and j for $i, j \in N^+$ (i.e., the bus that is farthest away from bus 0 and on the paths from bus 0 both to bus i and to bus j). Let $R_i := \sum_{(j,k)\in\mathbb{P}_i} r_{jk}$ denote the total resistance from bus 0 to bus i for $i \in N$ ($\mathbb{P}_i$ denotes the path from bus 0 to bus i). See Figure 2 for an example. Then $\partial_x(\hat{P}, \hat{Q}, \hat{v})$ has the following closed-form expression: for $k \in N$, $i \to j \in E$,

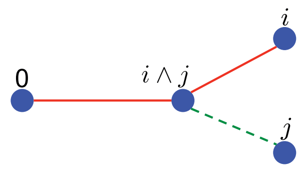


Fig. 2. The bus $i \wedge j$ denotes the joint of bus i and bus j. The resistance R_i denotes the sum of resistances on the red solid line segment.

$$\partial_{p_k} \hat{P}_{ij} = -\mathbb{1}(j \in \mathbb{P}_k), \qquad \partial_{q_k} \hat{P}_{ij} = 0 \tag{19a}$$

$$\partial_{q_k} \hat{Q}_{ij} = -\mathbb{1}(j \in \mathbb{P}_k), \qquad \partial_{p_k} \hat{Q}_{ij} = 0 \tag{19b}$$

$$\partial_{p_k} v_i = 2R_{i\wedge k}, \qquad \partial_{q_k} v_i = 2X_{i\wedge k} \tag{19c}$$

Note that $\partial_x(\hat{P}, \hat{Q}, \hat{v})$ is constant that does not depend on (P, Q, v), and therefore can be pre-computed just once.

Recall $\partial_x L$ from (11) reproduced here:

$$\partial_{x_i} L = c_0(x)\, \partial_{x_i} p_0(x) + (2a_i p_i + b_i)\mathbb{1}(x_i = p_i) - \sum_{j=1}^{n} \beta_k(x)\, \partial_{x_i} v_j(x) \tag{20}$$

where $c_0(x)$ and $\beta_k(x)$ are defined in (12). We will approximate the gradient $\partial_x L$ by approximating the partial derivatives $\partial_{x_i} p_0(x)$ and $\partial_x(P, Q, v)$ in (20) by $\partial_x(\hat{P}, \hat{Q}, \hat{v})$ in (19). In particular, using (17), we approximate $\partial_{x_i} p_0(x)$ by

$$\partial_{x_i} \hat{p}_0 := -\mathbb{1}(x_i = p_i) + \sum_{k\to l} r_{kl} \left(\frac{2P_{kl}}{v_k} \partial_{x_i}\hat{P}_{kl} + \frac{2Q_{kl}}{v_k} \partial_{x_i}\hat{Q}_{kl} - \frac{\ell_{kl}}{v_k} \partial_{x_i}\hat{v}_k \right)$$

for $i \in N$. Note that we do not ignore ℓ in (17), but only substitute the approximate gradient $\partial_x(\hat{P}, \hat{Q}, \hat{v})$ which does not contain ℓ. Hence, using (20), we approximate the gradient $\partial_x L(x; \mu)$ by: for $i \in N$

$$\begin{aligned}\partial_{p_i} \hat{L} &:= c_0(x) \sum_{k\to l} r_{kl} \left(\frac{2P_{kl}}{v_k} \partial_{p_i}\hat{P}_{kl} + \frac{2Q_{kl}}{v_k} \partial_{p_i}\hat{Q}_{kl} - \frac{\ell_{kl}}{v_k} \partial_{p_i}\hat{v}_k \right) \\ &\quad - \sum_{k=1}^{n} \beta_k(x)\, \partial_{p_i}\hat{v}_k + (2a_i p_i + b_i - c_0(x)) \\ &= -c_0(x) \sum_{k\to l} 2r_{kl} \left(\frac{P_{kl}}{v_k} \mathbb{1}(l \in \mathbb{P}_i) + \frac{\ell_{kl}}{v_k} R_{i\wedge k} \right) \\ &\quad - \sum_{k=1}^{n} 2\beta_k(x)\, R_{i\wedge k} + (2a_i p_i + b_i - c_0(x)) \end{aligned} \tag{21a}$$

$$\begin{aligned}\partial_{q_i} \hat{L} &:= c_0(x) \sum_{k\to l} r_{kl} \left(\frac{2P_{kl}}{v_k} \partial_{q_i}\hat{P}_{kl} + \frac{2Q_{kl}}{v_k} \partial_{q_i}\hat{Q}_{kl} - \frac{\ell_{kl}}{v_k} \partial_{q_i}\hat{v}_k \right) \\ &\quad - \sum_{k=1}^{n} \beta_k(x)\, \partial_{q_i}\hat{v}_k \\ &= -c_0(x) \sum_{k\to l} 2r_{kl} \left(\frac{Q_{kl}}{v_k} \mathbb{1}(l \in \mathbb{P}_i) + \frac{\ell_{kl}}{v_k} X_{i\wedge k} \right) \\ &\quad - \sum_{k=1}^{n} 2\beta_k(x)\, X_{i\wedge k} \end{aligned} \tag{21b}$$

Algorithm 2. Compute next iterate using line search

```
Input: back-off parameter $\hat{\alpha} \in (0,1)$, linearization-criterion
    parameter $\hat{\beta} \in (0,1)$, slow-progress parameter $\epsilon \ll 1$, cur-
    rent iterate $x := (p, q)$, bounds $\left(\underline{p}, \overline{p}, \underline{q}, \overline{q}, \underline{v}, \overline{v}\right)$, gradient
    $\partial_x L$.
Output: next iterate $x' := (p', q')$, stopping flag stopFlag.
 1: $\eta = 1$, stopFlag $= 0$;
 2: $x' \leftarrow [x - \eta\, \partial_x L(x; \mu)]_X$;
 3: solve (2) for a power flow solution $(v', p_0', q_0')$ given $x$ (and
    $v_0$), e.g., using a backward-forward sweep method;
 4: if $v' \notin [\underline{v}, \overline{v}]$
        $\eta \leftarrow \hat{\alpha}\eta$, go to 2;
    end
 5: $\Delta x \leftarrow x' - x$;
 6: if $\|\Delta x\| \le \epsilon$
        stopFlag $= 1$;
    else if $L(x'; \mu) > L(x; \mu) + \hat{\beta} \cdot \partial_x L(x; \mu)\, \Delta x$
        $\eta \leftarrow \hat{\alpha}\eta$, go to 2;
    end
 7: if $L(x') > L(x)$
        $x' \leftarrow x$;
    end
```

It can be verified that it will take $O(n^3)$ time to compute $\partial_x L$ by brute force. The approximate gradient in (21) can be computed in $O(n)$ time in a distributed manner; see [19].

D. *Time-Varying Stepsize $\eta(t)$*

A constant stepsize η in the gradient projection algorithm (10) is convenient if we want to execute the algorithm in real time and rely on the network to automatically compute $y(x(t))$. It also naturally tracks evolving network conditions. The bound on η in Theorems 1 and 3 however is typically too conservative. Alternatively if we execute the algorithm until it converges before applying the result to the network, then using time-varying stepsize $\eta(t)$ in each iteration t can be less conservative.

In this subsection we describe a method (Algorithm 2) to determine the stepsize $\eta(t)$ by a line search along the (negative) direction of the gradient $-\partial_x L$ (or approximate gradient $-\partial_x \hat{L}$ in (21). Algorithm 2 iteratively backs off the stepsize until the cost function L can be well-approximated by its linearization around the current iterate. Three parameters are used in the line search Algorithm 2: $\hat{\alpha}$ (that determines the backoff speed, set to 0.5 in the current implementation), $\hat{\beta}$ (criterion for the linearization of the objective function to be accurate enough, set to 0.5 in the current implementation), and ϵ (criterion for the progress to be too slow, set to 1e-4 in the current implementation).

The next result says that Algorithm 2 is well defined. It is proved in Appendix D.

Theorem 7 (Line serach): Algorithm 2 always terminates.

The "if" condition in Step 6 of Algorithm 2 is to stop the line search when progress is too slow, i.e., when $\|\Delta x\| \le \epsilon$. Otherwise, a large number of iterations will run without updating x significantly. With this condition, it is possible that $L(x'; \mu) > L(x; \mu)$ when Step 6 is exited through the "if" branch. In this case, x' is set to x to ensure that the output x' does not have a higher cost.

An advantage of Algorithm 2 is that it can be implemented in a distributed manner. See [19] for a suite of distributed algorithms for solving OPF(μ) (8) that executes the (approximate) gradient projection algorithm by computing the approximate gradient using (21) and time-varying $\eta(t)$ using line search, all in a distributed manner.

E. Multiphase Networks

So far, we have assumed a single-phase network. All the algorithms we have discussed, however, extend to multiphase networks with mild adjustments; see [19] for details and [18] for a C++ implementation. In this subsection we only highlight the differences from single-phase networks.

We adopt the model and notations of [21] for multiphase networks. The cost function becomes

$$
\begin{aligned}
L(x; \mu) := &\sum_{\phi\in\Phi_0} \left(a \left(p_0^\phi(x) \right)^2 + b p_0^\phi(x) \right) \\
&- \sum_{i=1}^{n} \sum_{\phi\in\Phi_i} \left(\underline{\mu} \ln(v_i^\phi - \underline{v}_i) + \overline{\mu} \ln(\overline{v}_i - v_i^\phi) \right)
\end{aligned}
$$

To compute $\partial_x L$, it suffices to compute $\partial_x p_0^\phi$ and $\partial_x v_i^\phi$ for each $i \in N$ and each $\phi \in \{a, b, c\}$. To simplify computation, we approximate these partial derivatives by $\partial_x \hat{p}_0^\phi$ and $\partial_x \hat{v}_i^\phi$, $i \in N$ and $\phi \in \{a, b, c\}$, obtained from a linearized branch flow model proposed in [21] for multiphase radial networks. Let Diag(v_i) denote a diagonal matrix with v_i as its diagonal entries and diag(M) denote a vector whose components are the diagonal entries M_{ii} of matrix M. The multiphase linearized branch flow model for radial networks is (omitting ˆ on the variables for simplicity):

$$
\begin{aligned}
\sum_{i:i\to j} \Lambda_{ij} &= s_j + \operatorname{diag}\left(\operatorname{Diag}(v_j) \gamma^{\Phi_j} y_j^H \right) + \sum_{k:j\to k} \Lambda_{jk}^{\Phi_j}, \quad j \in N^+ \\
S_{ij} &= \gamma^{\Phi_{ij}} \operatorname{Diag}(\Lambda_{ij}), \quad i \to j \in E \\
v_j &= v_i^{\Phi_{ij}} - \operatorname{diag}(S_{ij} z_{ij}^H + z_{ij} S_{ij}^H), \quad i \to j \in E
\end{aligned}
$$

where

$$
\alpha = e^{-2\pi/3}, \quad \beta = \begin{bmatrix} 1 \\ \alpha \\ \alpha^2 \end{bmatrix}, \quad \gamma = \beta\beta^H.
$$

Write the above equations in scalar form:

$$
\sum_{i:i\to j} \Lambda_{ij}^\phi = s_j^\phi + \sum_{k:j\to k} \Lambda_{jk}^\phi + v_j^\phi \sum_{\varphi\in\Phi_j} \alpha^{\phi-\varphi} \overline{y}_j^{\phi\varphi} \qquad j \in N^+, \phi \in \Phi_j \tag{22a}
$$

$$
v_j^\phi = v_i^\phi - \sum_{\varphi\in\Phi_j} (\Lambda_{ij}^\varphi \alpha^{\phi-\varphi} \overline{z}_{ij}^{\phi\varphi} + \overline{\Lambda}_{ij}^\varphi \alpha^{\varphi-\phi} z_{ij}^{\phi\varphi}) \qquad i \to j, \phi \in \Phi_{ij} \tag{22b}
$$

Now we estimate $\partial_x(\Lambda_{ij}^\phi, v_i^\phi)$ in two rounds as follows. In the first round, we ignore the last term in (22a) involving v_j^ϕ to obtain:

$$
\begin{aligned}
\partial_{p_i^\varphi} \Lambda_{kl}^\phi &= \mathbb{1}(\phi = \varphi)\mathbb{1}(i \in \text{Down}(l)) \\
\partial_{q_i^\varphi} \Lambda_{kl}^\phi &= \mathbf{i}\mathbb{1}(\phi = \varphi)\mathbb{1}(i \in \text{Down}(l))
\end{aligned}
$$

We use this expression to obtain the estimates of $\partial_x v_k^\phi$ as

$$
\begin{aligned}
\partial_x v_k^\phi &= \partial_x v_j^\phi - \sum_{\varphi\in\Phi_k} (\partial_x \Lambda_{jk}^\varphi \alpha^{\phi-\varphi} \overline{z}_{jk}^{\phi\varphi} + \partial_x \overline{\Lambda}_{jk}^\varphi \alpha^{\varphi-\phi} z_{jk}^{\phi\varphi}) \\
&= - \sum_{(i,j)\in\mathbb{P}_k} \sum_{\varphi\in\Phi_j} (\partial_x \Lambda_{ij}^\varphi \alpha^{\phi-\varphi} \overline{z}_{ij}^{\phi\varphi} + \partial_x \overline{\Lambda}_{ij}^\varphi \alpha^{\varphi-\phi} z_{ij}^{\phi\varphi})
\end{aligned}
$$

which simplifies to

$$
\partial_{p_i^\varphi} v_k^\phi = -2 \sum_{(s,t)\in\mathbb{P}_{k\wedge i}} \operatorname{Re}\left(\alpha^{\phi-\varphi} \overline{z}_{st}^{\phi\varphi} \right) \tag{23a}
$$

$$
\partial_{q_i^\varphi} v_k^\phi = 2 \sum_{(s,t)\in\mathbb{P}_{k\wedge i}} \operatorname{Im}\left(\alpha^{\phi-\varphi} \overline{z}_{st}^{\phi\varphi} \right) \tag{23b}
$$

for $k \in N$, $\phi \in \Phi_k$, and $x = p_i^\varphi$ or q_i^φ for $i \in N$ and $\varphi \in \Phi_i$. At last, we estimate $\partial_x \Lambda_{kl}^\phi$ as: for $k \to l$, $\phi \in \Phi_l$

$$
\begin{aligned}
\partial_x \Lambda_{kl}^\phi &= \partial_x s_l^\phi + \sum_{m:l\to m} \partial_x \Lambda_{lm}^\phi + \partial_x v_l^\phi \sum_{\varphi\in\Phi_l} \alpha^{\phi-\varphi} \overline{y}_l^{\phi\varphi} \\
&= \sum_{t\in\text{Down}(l)} \left[\partial_x s_t^\phi + \partial_x v_t^\phi \sum_{\xi\in\Phi_t} \alpha^{\phi-\xi} \overline{y}_t^{\phi\xi} \right]
\end{aligned} \tag{24}
$$

This can be used to obtain

$$
\begin{aligned}
\partial_{p_i^\varphi} p_0^\phi &= \mathbb{1}(\phi = \varphi) + \sum_{t=1}^{n} \partial_{p_i^\varphi} v_t^\phi \operatorname{Re}\left(\sum_{\xi\in\Phi_t} \alpha^{\phi-\xi} \overline{y}_t^{\phi\xi} \right) \\
\partial_{q_i^\varphi} p_0^\phi &= \sum_{t=1}^{n} \partial_{q_i^\varphi} v_t^\phi \operatorname{Re}\left(\sum_{\xi\in\Phi_t} \alpha^{\phi-\xi} \overline{y}_t^{\phi\xi} \right)
\end{aligned}
$$

for $\phi \in \Phi_0$, $i \in N$, and $\varphi \in \Phi_i$.

To estimate the gradient $\partial_x L$, define

$$
g_k^\phi := (2ap_0^\phi + b)\operatorname{Re}\left(\sum_{\xi\in\Phi_k} \alpha^{\phi-\xi} \overline{y}_k^{\phi\xi} \right) - \frac{\underline{\mu}}{v_k^\phi - \underline{v}_k} + \frac{\overline{\mu}}{\overline{v}_k - v_k^\phi}
$$

for $k \in N$ and $\phi \in \Phi_k$. Then it is derived in [19] that the approximate gradient is:

$$
\begin{aligned}
\partial_{p_i^\varphi} \hat{L} &= 2ap_0^\varphi + b - 2 \sum_{(s,t)\in\mathbb{P}_i} \sum_{\phi\in\Phi_t} \operatorname{Re}\left(\alpha^{\phi-\varphi} \overline{z}_{st}^{\phi\varphi} \right) \sum_{k\in\text{Down}(t)} g_k^\phi \\
\partial_{q_i^\varphi} \hat{L} &= 2 \sum_{(s,t)\in\mathbb{P}_i} \sum_{\phi\in\Phi_t} \operatorname{Im}\left(\alpha^{\phi-\varphi} \overline{z}_{st}^{\phi\varphi} \right) \sum_{k\in\text{Down}(t)} g_k^\phi
\end{aligned}
$$

for $i \in N$ and $\varphi \in \Phi_i$.

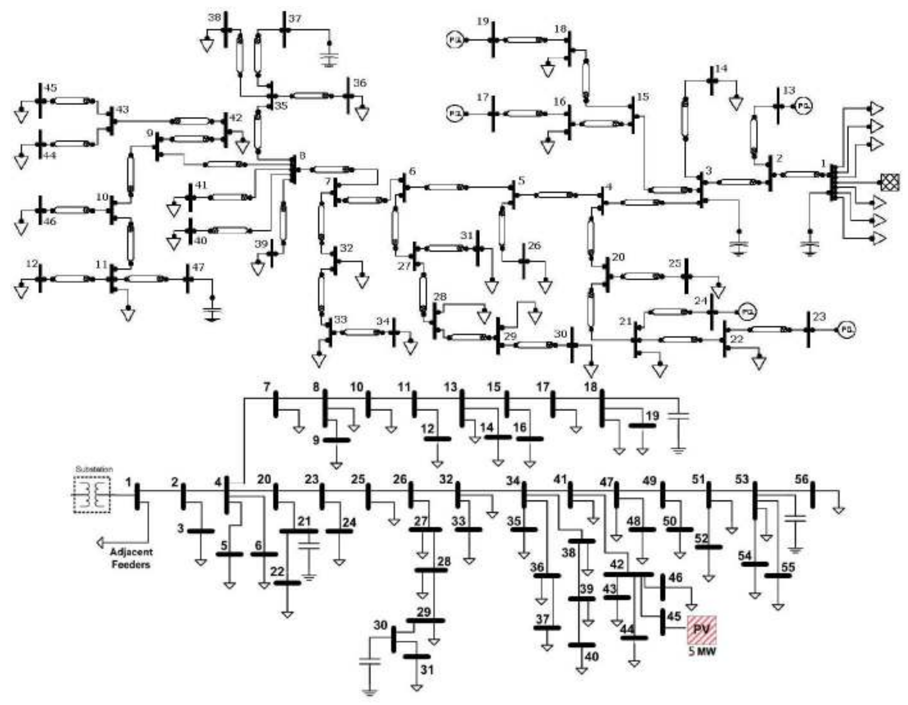


Fig. 3. Topologies of the SCE 47-bus and 56-bus networks from [13], [15].

VII. Numerical Results

In this section we evaluate the accuracy and efficiency of our gradient project algorithm on a number of single-phase radial networks. We compare the results of our algorithm against the global optimal value of (3) obtained from solving its semidefinite relaxation using techniques in [13], [14], [20], [25], [26]. The convex relaxation is solved by CVX [22], [34] in Matlab, and its execution time and objective value are used as a benchmark for our algorithm. All simulations use Matlab 7.9.0.529 (64-bit) with toolbox cvx 1.21 on Mac OS X 10.7.5 with 2.66GHz Intel Core 2 Due CPU and 4GB 1067MHz DDR3 memory.

A. Test Networks

The test networks include a 42-bus network adapted from [13], a 56-bus network from [15], and several subnetworks of a 2065-bus network. These three networks are all models of distribution systems in the service territory of the utility company Southern California Edison (SCE). Topologies of the 47-bus network and the 56-bus network are shown in Figure 3.

B. OPF Setup

The following OPF setup is used throughout this section.

1) The objective is to minimize power loss in the network.
2) The power injection constraints are as follows. For each bus $i \in N$, there may be multiple devices including loads, capacitors, and PV panels. Assume that there is a total of D_i such devices and label them by $1, 2, \dots, D_i$. Let $s_{i,d} = p_{i,d} + \mathbf{i} q_{i,d}$ denote the power injection of device d for $d = 1, 2, \dots, D_i$. If device d is a load with given real and reactive power consumptions p and q, then we impose
$$s_{i,d} = -p - \mathbf{i} q. \tag{25}$$
If device d is a load with given peak apparent power s_{peak}, then we impose
$$s_{i,d} = -s_{\text{peak}} \exp(j\theta) \tag{26}$$
where $\theta = \cos^{-1}(0.9)$, i.e, power injection $s_{i,d}$ is a constant, obtained by assuming a power factor of 0.9 at peak apparent power s_{peak}. If device d is a capacitor with nameplate $\overline{q}$, then we impose
$$\text{Re}(s_{i,d}) = 0 \text{ and } 0 \le \text{Im}(s_{i,d}) \le \overline{q}. \tag{27}$$
If device d is a PV panel with nameplate $\overline{s}$ and real power generation p_i, then we impose
$$\text{Re}(s_{i,d}) = p_i \text{ and } |s_{i,d}| \le \overline{s}. \tag{28}$$
The power injection at bus i is
$$s_i = \sum_{d=1}^{D_i} s_{i,d}$$
where $s_{i,d}$ satisfies one of (25)–(28).
3) The voltage constraint is $0.95^2 \le v_i \le 1.05^2$ for $i \in N$.

C. Results

Numerical results are summarized in Table I. The first column lists the 22 test networks we have used with their numbers of buses. The next two columns labeled "CVX" report the globally optimal objective value and computation time obtained

TABLE I
OBJECTIVE VALUES AND CPU TIMES OF CVX AND IPM

# bus	CVX		IPM		error	speedup
	obj	time(s)	obj	time(s)		
42	10.4585	6.5267	10.4585	0.2679	-0.0e-7	24.36
56	34.8989	7.1077	34.8989	0.3924	+0.2e-7	18.11
111	0.0751	11.3793	0.0751	0.8529	+5.4e-6	13.34
190	0.1394	20.2745	0.1394	1.9968	+3.3e-6	10.15
290	0.2817	23.8817	0.2817	4.3564	+1.1e-7	5.48
390	0.4292	29.8620	0.4292	2.9405	+5.4e-7	10.16
490	0.5526	36.3591	0.5526	3.0072	+2.9e-7	12.09
590	0.7035	43.6932	0.7035	4.4655	+2.4e-7	9.78
690	0.8546	51.9830	0.8546	3.2247	+0.7e-7	16.12
790	0.9975	62.3654	0.9975	2.6228	+0.7e-7	23.78
890	1.1685	67.7256	1.1685	2.0507	+0.8e-7	33.03
990	1.3930	74.8522	1.3930	2.7747	+1.0e-7	26.98
1091	1.5869	83.2236	1.5869	1.0869	+1.2e-7	76.57
1190	1.8123	92.4484	1.8123	1.2121	+1.4e-7	76.27
1290	2.0134	101.0380	2.0134	1.3525	+1.6e-7	74.70
1390	2.2007	111.0839	2.2007	1.4883	+1.7e-7	74.64
1490	2.4523	122.1819	2.4523	1.6372	+1.9e-7	74.83
1590	2.6477	157.8238	2.6477	1.8021	+2.0e-7	87.58
1690	2.8441	147.6862	2.8441	1.9166	+2.1e-7	77.06
1790	3.0495	152.6081	3.0495	2.0603	+2.1e-7	74.07
1890	3.8555	160.4689	3.8555	2.1963	+1.9e-7	73.06
1990	4.1424	171.8137	4.1424	2.3586	+1.9e-7	72.84

from solving semidefinite relaxation of OPF (3) using CVX in Matlab. The two columns labeled "IPM" report the results obtained from solving the approximate OPF problem (8) using our interior-point method, i.e., the gradient projection algorithm discussed in Section IV with the following modifications:

- instead of exact gradient $\partial_x L$, it uses approximate gradient $\partial_x \hat{L}$ in (21) obtained from linearized power flow equations as described in Section VI-C;
- instead of a constant stepsize, it uses time-varying stepsizes $\eta(t)$ obtained from line search as described in Section VI-D
- the networks are multiphase unbalanced.

The next column "error" is the difference between the objective value of CVX and that of IPM. The last column "speedup" is the ratio of the computation time of CVX to that of IPM.

Each row summarizes the result for a test network. For example, the first row of the table summarizes the simulation result for a test network with 42 buses adapted from the 47-bus network in Fig. 3.

From the last two columns, we conclude that while semidefinite relaxation is able to compute globally optimal solutions, it takes a much longer time. In comparison, the algorithm developed in this paper takes a much shorter time and is able to obtain a close-to-optimal solution. Specifically, for all our test networks, the difference in objective values is below 10^{-5} between these two methods but the speedup is over 70x for large networks.

VIII. CONCLUSIONS

We have proposed an online algorithm for solving OPF on radial networks where the controllable devices continuously interact with the network that implicitly computes a power flow solution given a control action. Collectively the controllable devices and the network implement a gradient projection algorithm for the OPF problem in real time. The key feature that enables this approach is that the intermediate iterates always satisfy power flow equations and operational constraints. We have proved the convergence and optimality of the proposed algorithm and have bounded the suboptimality gap of any local minimum.

To greatly reduce the gradient computation in each iteration, we have derive approximate gradient based on linearized power flow equations. We have also outlined how these algorithms can be extended to multiphase unbalanced networks. Finally the evaluation of our algorithm on test networks, ranging from 42-bus to 1990-bus, shows more than 70x speedup over a semidefinite relaxation method with negligible difference in objective values.

It is therefore promising to further develop the algorithms in this paper for real-time applications. A key challenge to overcome is to minimize the measurement and communication requirements so that these algorithms can be implemented in real time by a large network of distributed energy resources, building on the ideas in [19].

APPENDIX A
PROOF OF THEOREM 1: LOCAL OPTIMALITY

Proof: Fix any $\mu > 0$. Since the objective function $L(x;\mu)$ is continuous in x and X is compact, $L(x;\mu)$ is lower bounded. Then, with $0 < \eta < 2/K$, the first assertion (if any subsequence of $x(t)$ converges to an x^* then x^* is locally optimal) follows from standard convergence results of gradient projection algorithms; see e.g. [3, Proposition 3.4, p. 214].

This is equivalent to the second claim that $x(t)$ converges to the set $\mathbb{O}^*$ of local minima, i.e., for any $\epsilon > 0$, we have

$$\min_{x\in\mathbb{O}^*} \|x(t) - x\|_2 < \epsilon \tag{29}$$

for all sufficiently large t. This is because, otherwise, there is an $\epsilon > 0$ such that for each integer $k = 1, 2, \ldots,$ there is an $x(t_k)$ with $\min_{x\in\mathbb{O}^*} \|x(t_k) - x\|_2 \geq \epsilon$. Since the (sub)sequence $(x(t_k), k = 1, 2, \ldots)$, lives in the compact set X, it has a convergent subsequence $(x(t_{k_j}), j = 1, 2, \ldots)$ which converges to a certain point $\hat{x}$. But $\min_{x\in\mathbb{O}^*} \|x(t_{k_j}) - x\|_2 \geq \epsilon$ for all j implies that

$$\min_{x\in\mathbb{O}^*} \|\hat{x} - x\|_2 \geq \epsilon$$

i.e., $\hat{x} \notin \mathbb{O}^*$, contradicting that any limit point of $x(t)$ is a local minimum (first assertion).

For the third assertion, suppose $\mathbb{O}^*$ has only finitely many local minima and let $d > 0$ denote the minimum distance among these local minima. Pick any convergent subsequence $(x(t_k), k = 1, 2, \ldots)$ of $x(t)$ and suppose without loss of generality that it converges to one of the local minima $x^* \in \mathbb{O}^*$:

$$x(t_k) \to x^* \qquad \text{as } k \to \infty \tag{30}$$

We now argue that $x(t)$ itself converges to x^*, in two steps.

First we claim that

$$\|x(t+1) - x(t)\|_2 \to 0 \quad \text{as } t \to \infty \tag{31}$$

To see this, using the Descent Lemma [3, Lemma 2.1], it can be shown that (omitting μ in $L(x;\mu)$ for simplicity)

$$\begin{aligned} L(x(t+1)) &\le L(x(t)) \;+\; (x(t+1)-x(t))^T \partial_x L(x) \\ &\quad +\; \|x(t+1)-x(t)\|_2^2 \\ &\le L(x(t)) - \kappa\,\|x(t+1)-x(t)\|_2^2 \end{aligned}$$

where

$$\kappa := \frac{1}{\eta} - \frac{K}{2} \;>\; 0$$

Since this holds for all $t = 0, 1, \ldots$, sum the above inequality over all t to obtain

$$L(x(t+1)) \le L(x(0) \;+\; \kappa \sum_{t=0}^{\infty} \|x(t+1)-x(t)\|_2^2$$

But $L(x)$ is lower bounded on X and hence (31) holds.

Next we argue that (29), (30) and (31) imply that $x(t)$ converges to x^*. Indeed, fix any $0 < \epsilon < d/3$. Then (30) and (31) imply that there is a certain t such that

$$\|x(t) - x^*\|_2 \;<\; \epsilon, \|x(t+1)-x(t)\|_2 \;<\; \epsilon$$

This implies

$$\|x(t+1)-x^*\|_2 \;<\; 2\epsilon \;<\; \frac{2d}{3}$$

and hence

$$\|x(t+1)-x\|_2 \;>\; \frac{d}{3} \qquad \text{for all } x \in \mathbb{O}^*,\; x \neq x^*$$

But (29) implies that $x(t+1)$ must be less than ϵ away from some point in $\mathbb{O}^*$, and hence we must have

$$\|x(t+1) - x^*\|_2 \;<\; \epsilon$$

This, together with $\|x(t+2)-x(t+1)\|_2 < \epsilon$ (due to (31)), implies by induction that

$$\|x(t+k) - x^*\|_2 \;<\; \epsilon \qquad \text{for all } k = 0, 1, \ldots$$

i.e., $x(t) \to x^*$. ■

Appendix B
Proof of Theorem 4: Convexity of p_0

Proof: Let $\tilde{x} = (\tilde{p}, \tilde{q}) \in C$ and $\hat{x} = (\hat{p}, \hat{q}) \in C$ be distinct. It suffices to show that the point $x = \theta\tilde{x} + (1-\theta)\hat{x}$ satisfies $p_0(x) \le \theta p_0(\tilde{x}) + (1-\theta)p_0(\hat{x})$ for any $\theta \in (0,1)$, i.e., $p_0(x)$ is convex in x.

Let $\tilde{y} := (\tilde{p}_0, \tilde{q}_0, \tilde{v}, \tilde{P}, \tilde{Q}, \tilde{\ell})$ denote the power flow solution given $\tilde{x}$. Then $\tilde{p}_0 = p_0(\tilde{x})$. Let $\hat{y} := (\hat{p}_0, \hat{q}_0, \hat{v}, \hat{P}, \hat{Q}, \hat{\ell})$ denote the power flow solution given $\hat{x}$ and $\hat{p}_0 = p_0(\hat{x})$. Since $p_0(x)$ is the optimal value of SOCP-$p_0(x)$, and the point

$$(p_0, q_0, v, P, Q, \ell,) := \theta\tilde{y} + (1-\theta)\hat{y}$$

is feasible for SOCP-$p_0(x)$, one must have

$$p_0(x) \le p_0 = \theta\tilde{p}_0 + (1-\theta)\hat{p}_0 = \theta p_0(\tilde{x}) + (1-\theta)p_0(\hat{x}).$$

This completes the proof of Theorem 4. ■

Appendix C
Proof of Theorem 6: Suboptimality Bound

Proof: The idea is to create a trajectory $x(\theta)$ of feasible solutions of (6) that approaches x^* as $\theta \to 0$, and make use of the fact that $L(x^*) \le L(x(\theta))$ for sufficiently small θ.

The trajectory $x(\theta)$ is constructed using the implicit function theorem. Let (v', p_0') and (v^*, p_0^*) denote the power flow solutions corresponding to x' and x^* respectively, i.e., $v' = v(x')$, $p_0' = p_0(x')$, $v^* = v(x^*)$, $p_0^* = p_0(x^*)$. Consider the following function

$$f(q,\theta) := (1-\theta)v^* + \theta v' - v\left[(1-\theta)p^* + \theta p', q\right].$$

Note that $f(q^*, 0) = v^* - v(p^*, q^*) = 0$ and that the partial derivative $\partial_q f = -\partial_q v$ is full rank. Hence, there exists $q(\theta)$ near a small neighborhood $(-\omega, \omega)$ where $\omega > 0$ of 0 that satisfies

$$q(0) = q^*, \quad f(q(\theta), \theta) = 0.$$

The equality $f(q(\theta), \theta) = 0$ is equivalent to

$$(1-\theta)v^* + \theta v' = v\left[(1-\theta)p^* + \theta p', q(\theta)\right].$$

Let $v(\theta) := (1-\theta)v^* + \theta v'$ and $p(\theta) := (1-\theta)p^* + \theta p'$ for $\theta \in (-\omega, \omega)$, then $v(\theta) = v(p(\theta), q(\theta))$. Let $p_0(\theta) := p_0(p(\theta), q(\theta))$ for $\theta \in (-\omega, \omega)$. At this point, the trajectory $x(\theta)$ has been constructed.

Now we show that $x(\theta) \in \tilde{X}$ for sufficiently small θ. In particular, it suffices to prove $q(\theta) \in [\underline{q}, \overline{q}]$ for sufficiently small θ. It follows from

$$\begin{aligned} 0 &= \partial_q f(q^*, 0) \cdot \partial_\theta q(0) + \partial_\theta f(q^*, 0) \\ &= -\partial_q v(x^*) \cdot \partial_\theta q(0) + v' - v^* - \partial_p v(x^*) \cdot (p' - p^*) \end{aligned}$$

that

$$\begin{aligned} \partial_\theta q(0) &= \left[\partial_q v(x^*)\right]^{-1} \cdot \left[v' - v^* - \partial_p v(x^*) \cdot (p' - p^*)\right] \\ &= q' - q^* + r(\nu) \in \left(\underline{q}, \overline{q}\right) - q^* \end{aligned}$$

for some $\nu \in (0,1)$. Therefore,

$$q(\theta) = q^* + \theta \cdot \partial_\theta q(0) + o(\theta) \;\in\; \left(\underline{q}, \overline{q}\right)$$

for sufficiently small θ.

Finally we make use of the local optimality of $L(x^*)$, which implies $L(x^*) \le L(x(\theta))$ for sufficiently small $\theta > 0$. Substitute

$$\begin{aligned} &L(x(\theta)) \\ &\quad = a_0 p_0^2(\theta) + b_0 p_0(\theta) + \sum_{i=1}^{n} \left(a_i p_i^2(\theta) + b_i p_i(\theta)\right) \\ &\quad\quad - \sum_{i=1}^{n} \left(\underline{\mu}\ln(v_i(\theta) - \underline{v}_i) + \overline{\mu}\ln(\overline{v}_i - v_i(\theta))\right) \end{aligned}$$

$$
\begin{aligned}
&\le a_0 p_0^2(\theta) + b_0 p_0(\theta) + (1-\theta)\left(\sum_{i=1}^n a_i\left(p_i^*\right)^2 + b_i p_i^*\right)\\
&\quad + \theta\left(\sum_{i=1}^n a_i\left[p_i'\right]^2 + b_i p_i'\right)\\
&\quad - (1-\theta)\sum_{i=1}^n\left(\underline{\mu}\ln(v_i^* - \underline{v}_i) + \overline{\mu}\ln(\overline{v}_i - v_i^*)\right)\\
&\quad - \theta\sum_{i=1}^n\left(\underline{\mu}\ln(v_i' - \underline{v}_i) + \overline{\mu}\ln(\overline{v}_i - v_i')\right)\\
&= (1-\theta)L(x^*) + \theta L(x') + a_0 p_0^2(\theta) + b_0 p_0(\theta)\\
&\quad - (1-\theta)\left(a_0\left(p_0^*\right)^2 + b_0 p_0^*\right) - \theta\left(a_0\left(p_0'\right)^2 + b_0 p_0'\right)
\end{aligned}
$$

to obtain

$$
\begin{aligned}
\theta\left[L(x^*) - L(x')\right] &\le a_0 p_0^2(\theta) + b_0 p_0(\theta) - a_0\left(p_0^*\right)^2 - b_0 p_0^*\\
&\quad + \theta\left(a_0\left(p_0^*\right)^2 + b_0 p_0^* - a_0\left(p_0'\right)^2 - b_0 p_0'\right)
\end{aligned}
$$

for sufficiently small $\theta > 0$. Take the gradient with respect to θ at $\theta = 0$ to obtain that

$$
\begin{aligned}
L(x^*) - L(x') \le{}& \left(2a_0 p_0^* + b_0\right)\cdot\left[\partial_p p_0(x^*)\cdot(p' - p^*)\right.\\
&\left. + \partial_q p_0(x^*)\cdot(q' - q^* + r(v))\right]\\
&+ a_0\left(p_0^*\right)^2 + b_0 p_0^* - a_0\left(p_0'\right)^2 - b_0 p_0'.
\end{aligned}
$$

Due to the convexity of $p_0(x)$, one has

$$
p_0' - p_0^* \ge \partial_p p_0(x^*)\cdot(p' - p^*) + \partial_q p_0(x^*)\cdot(q' - q^*)
$$

and therefore

$$
\begin{aligned}
L(x^*) - L(x') &\le \left(2a_0 p_0^* + b_0\right)\cdot\left(p_0' - p_0^* + \partial_q p_0(x^*)\cdot r(v)\right)\\
&\quad + a_0\left(p_0^*\right)^2 + b_0 p_0^* - a_0\left(p_0'\right)^2 - b_0 p_0'\\
&= \left(2a_0 p_0^* + b_0\right)\cdot\partial_q p_0(x^*)\cdot r(v)\\
&\quad - a_0(p_0^* - p_0')^2
\end{aligned}
$$

This completes the proof of Theorem 6. ■

Appendix D
Proof of Theorem 7: Line Search

Proof: Assume that Algorithm 2 fails to produce $(p', q', \text{stopFlag})$ for some instance. Consider this instance and derive a contradiction as follows. Let the superscript (k) denote the round of iteration for $k = 0, 1, 2, \ldots$ where iteration 0 refers to the initial value, e.g., $\Delta p^{(k)} = p^{(k)} - p$ for $k \ge 1$.

Let $\hat{m} > 0$ denote the minimum positive number among $\{|\partial_{p_i} L|, |\partial_{q_i} L| \ : \ i = 1, 2, \ldots, n\}$. Note that $\partial_{p_i} L\cdot\Delta p_i^{(k)} \le 0$ and $\partial_{q_i} L\cdot\Delta q_i^{(k)} \le 0$ for $k \ge 1$ and $i \in N$. Furthermore, $\partial_{p_i} L = 0$ implies $\Delta p_i^{(k)} = 0$ for $k \ge 1$ and $i \in N$. Hence,

$$
\partial_{p_i} L\cdot\Delta p_i^{(k)} \le -\hat{m}|\Delta p_i^{(k)}|, \quad k \ge 1, i \in N.
$$

It follows that

$$
\begin{aligned}
\partial_p L\cdot\Delta p^{(k)} &= \sum_{i\in N}\partial_{p_i} L\cdot\Delta p_i^{(k)}\\
&\le \sum_{i\in N} -\hat{m}\left|\Delta p_i^{(k)}\right|\\
&= -\hat{m}\left\|\Delta p^{(k)}\right\|_1
\end{aligned}
$$

for $k \ge 1$, where $\|\cdot\|_1$ denotes the ℓ_1 norm of a vector, i.e., $\|x\|_1 = \sum_{i=1}^n |x_i|$ for $x \in \mathbb{R}^n$. Similarly

$$
\partial_q L\cdot\Delta q^{(k)} \le -\hat{m}\|\Delta q^{(k)}\|_1
$$

for $k \ge 1$. It follows that

$$
\begin{aligned}
L(p^{(k)}, q^{(k)}) &= L(p + \Delta p^{(k)}, q + \Delta q^{(k)})\\
&= L(p, q) + \partial_p L\cdot\Delta p^{(k)} + \partial_q L\cdot\Delta q^{(k)}\\
&\quad + o(\Delta p^{(k)}, \Delta q^{(k)})\\
&= L(p, q) + \hat{\beta}\left(\partial_p L\cdot\Delta p^{(k)} + \partial_q L\cdot\Delta q^{(k)}\right)\\
&\quad + (1 - \hat{\beta})\left(\partial_p L\cdot\Delta p^{(k)} + \partial_q L\cdot\Delta q^{(k)}\right)\\
&\quad + o(\Delta p^{(k)}, \Delta q^{(k)})\\
&\le L(p, q) + \hat{\beta}\left(\partial_p L\cdot\Delta p^{(k)} + \partial_q L\cdot\Delta q^{(k)}\right)\\
&\quad - \hat{m}(1 - \hat{\beta})\left(\left\|\Delta p^{(k)}\right\|_1 + \left\|\Delta q^{(k)}\right\|_1\right)\\
&\quad + o(\Delta p^{(k)}, \Delta q^{(k)})
\end{aligned}
$$

for $k \ge 1$. When k is sufficiently big, $\|\Delta p^{(k)}\|_1 + \|\Delta q^{(k)}\|_1$ is sufficiently small such that

$$
o(\Delta p^{(k)}, \Delta q^{(k)}) \le \hat{m}(1 - \hat{\beta})\left(\left\|\Delta p^{(k)}\right\|_1 + \left\|\Delta q^{(k)}\right\|_1\right)
$$

Hence, eventually

$$
L(p^{(k)}, q^{(k)}) \le L(p, q) + \hat{\beta}\left(\partial_p L\cdot\Delta p^{(k)} + \partial_q L\cdot\Delta q^{(k)}\right)
$$

Then, the loop specified by Step 6 is exited and $(p', q', \text{stopFlag})$ is produced. This contradicts with the assumption that Algorithm 2 fails to produce a $(p', q', \text{stopFlag})$ and completes the proof of Theorem 7. ■

References

[1] M. E. Baran and F. F. Wu, "Optimal capacitor placement on radial distribution systems," *IEEE Trans. Power Del.*, vol. 4, no. 1, pp. 725–734, Jan. 1989.

[2] M. E. Baran and F. F. Wu, "Optimal sizing of capacitors placed on a radial distribution system," *IEEE Trans. Power Del.*, vol. 4, no. 1, pp. 735–743, Jan. 1989.

[3] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1989.

[4] S. Bolognani, R. Carli, G. Cavraro, and S. Zampieri, "A distributed feedback control strategy for optimal reactive power flow with voltage constraints," arXiv, 2013.

[5] M. B. Cain, R. P. O'Neill, and A. Castillo, "History of optimal power flow and formulations," *Fed. Energy Regul. Commission*, 2012.

[6] J. Carpentier, "Contribution to the economic dispatch problem," *Bull. Soc. Francoise Electriciens*, vol. 3, no. 8, pp. 431–447, 1962. (in French).

[7] A. Castillo and R. P. O'Neill, "Computational performance of solution techniques applied to the ACOPF (OPF Paper 5)," US FERC, Tech. Rep., Mar. 2013.

[8] A. Castillo and R. P. O'Neill, "Survey of approaches to solving the ACOPF," US FERC, Tech. Rep., 2013.

[9] H.-D. Chiang and M. E. Baran, "On the existence and uniqueness of load flow solution for radial distribution power networks," *IEEE Trans. Circuits Syst.*, vol. 37, no. 3, pp. 410–416, Mar. 1990.

[10] E. Dall'Anese, H. Zhu, and G. B. Giannakis, "Distributed optimal power flow for smart microgrids," *IEEE Trans. Smart Grid*, vol. 4, no. 3, pp. 1464–1475, Sep. 2013.

[11] H. Dommel and W. Tinney, "Optimal power flow solutions," *IEEE Trans. Power App. Syst.*, vol. PAS-87, no. 10, pp. 1866–1876, Oct. 1968.

[12] M. Farivar, L. Chen, and S. H. Low, "Equilibrium and dynamics of local voltage control in distribution systems," in *Proc. IEEE 52nd Annu. Conf. Decision Control (CDC)*, Dec. 2013, pp. 4329–4334.

[13] M. Farivar, C. R. Clarke, S. H. Low, and K. M. Chandy, "Inverter VAR control for distribution systems with renewables," in *Proc. IEEE SmartGridComm*, 2011, pp. 457–462.

[14] M. Farivar and S. H. Low, "Branch flow model: Relaxations and convexification—Part II,"*IEEE Trans. Power Syst.*, vol. 28, no. 3, pp. 2565–2572, Aug. 2013.

[15] M. Farivar, R. Neal, C. Clarke, and S. Low, "Optimal inverter VAR control in distribution systems with high PV penetration," in *Proc. Power Energy Soc. Gen. Meeting*, 2012, pp. 1–7.

[16] S. Frank, I. Steponavice, and S. Rebennack, "Optimal power flow: A bibliographic survey I," *Energy Syst.*, vol. 3, no. 3, pp. 221–258, 2012.

[17] S. Frank, I. Steponavice, and S. Rebennack, "Optimal power flow: A bibliographic survey—Part II: Nondeterministic and hybrid methods," *Energy Syst.*, vol. 3, pp. 259–289, Sep. 2013.

[18] L. Gan. (2014, Aug.). A Distributed Algorithm for Solving the Optimal Load Control Problem in Multiphase Radial Networks [Online]. Available: http://www.its.caltech.edu/~lgan/

[19] L. Gan, "Distributed load control in multiphase radial networks," Dept. Electr. Eng., Caltech, Pasadena, CA, USA, Ph.D. dissertation, Caltech, 2015.

[20] L. Gan, N. Li, U. Topcu, and S. H. Low, "Exact convex relaxation of optimal power flow in radial networks," *IEEE Trans. Automat. Control*, vol. 60, no. 1, pp. 72–87, Jan. 2015.

[21] L. Gan and S. H. Low, "Convex relaxations and linear approximation for optimal power flow in multiphase radial networks," in *Proc. 18th Power Syst. Comput. Conf. (PSCC)*, Wroclaw, Poland, Aug. 2014, pp. 1–9.

[22] M. Grant, S. Boyd, and Y. Ye. (2008). *Cvx: Matlab Software for Disciplined Convex Programming* [Online]. Available: http://cvxr.com/cvx/

[23] M. Huneault and F. D. Galiana, "A survey of the optimal power flow literature," *IEEE Trans. Power Syst.*, vol. 6, no. 2, pp. 762–770, May 1991.

[24] W. H. Kersting, *Distribution Systems Modeling and Analysis*. Boca Raton, FL, USA: CRC Press, 2002.

[25] S. H. Low, "Convex relaxation of optimal power flow—Part I: Formulations and relaxations," *IEEE Trans. Control Netw. Syst.*, vol. 1, no. 1, pp. 15–27, Mar. 2014.

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

[27] J. A. Momoh, *Electric Power System Applications of Optimization*. New York, NY, USA: Marcel Dekker, 2001.

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

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

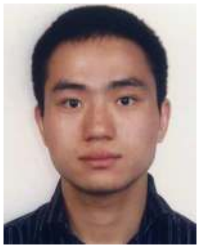
[30] R. P. O'Neill, A. Castillo, and M. B. Cain, "The computational testing of ac optimal power flow using the current voltage formulations (OPF Paper 3)," US FERC, Tech. Rep., Dec. 2012.

[31] R. P. O'Neill, A. Castillo, and M. B. Cain, "The IV formulation and linear approximations of the ac optimal power flow problem (OPF Paper 2)," US FERC, Tech. Rep., Dec. 2012.

[32] K. S. Pandya and S. K. Joshi, "A survey of optimal power flow methods," *J. Theor. Appl. Inf. Technol.*, vol. 4, no. 5, pp. 450–458, 2008.

[33] B. Stott and O. Alsac, "Fast decoupled load flow," *IEEE Trans. Power App. Syst.*, vol. PAS-93, no. 3, pp. 859–869, May/Jun. 1974.

[34] J. F. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optim. Methods Softw.*, vol. 11, nos. 1–4, pp. 625–653, 1999.



Lingwen Gan received the B.E. degree in electronics engineering from Tsinghua University, Beijing, China, in 2010, and the M.S. and Ph.D. degrees from Caltech, Pasadena, CA, USA, in 2012 and 2015, respectively, both in electrical engineering. He is currently a Research Scientist with Facebook. He was the recipient of the Demetriades-Tsafka-Kokkalis Prize for his Ph.D. Thesis from the Engineering and Applied Science Division of Caltech.

Steven H. Low (F'08) received the B.S. degree from Cornell University, Ithaca, NY, USA, and the Ph.D. degree from University of California, Berkeley, Berkeley, USA, both in EE. He is a Professor with the Department of Computing and & Mathematical Sciences and Department of Electrical Engineering, Caltech, Pasadena, CA, USA, and an Adjunct Chaired Professor of Zhejiang University, Hangzhou, China. Before that, he was with AT&T Bell Laboratories, Murray Hill, NJ, USA, and the University of Melbourne, Parkville, Vic., Australia. He was a Senior Editor of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, is a Senior Editor of the IEEE TRANSACTION CONTROL OF NETWORK SYSTEMS and the IEEE TRANSACTION ON NETWORK SCIENCE AND ENGINEERING. He is also on the Editorial Boards of NOW Foundations and Trends in Energy Systems and in Networking.