First-Order Methods for Fast Feasibility Pursuit of Non-convex QCQPs

Aritra Konar, *Student Member, IEEE*, and Nicholas D. Sidiropoulos, *Fellow, IEEE*

***Abstract*—Quadratically constrained quadratic programming (QCQP) is NP-hard in its general non-convex form, yet it frequently arises in various engineering applications. Several polynomial-time approximation algorithms exist for non-convex QCQP problems (QCQPs), but their success hinges upon the ability to find at least one feasible point—which is also hard for a general problem instance. In this paper, we present a heuristic framework for computing feasible points of general non-convex QCQPs using simple first-order methods. Our approach features low computational and memory requirements, which makes it well suited for application to large-scale problems. While *a priori* it may appear that these benefits come at the expense of technical sophistication, rendering our approach too simple to even merit consideration for a non-convex and NP-hard problem, we provide compelling empirical evidence to the contrary. Experiments on synthetic as well as real-world instances of non-convex QCQPs reveal the surprising effectiveness of first-order methods compared to more established and sophisticated alternatives.**

***Index Terms*—Non-convex optimization, quadratically constrained quadratic programming (QCQP), feasibility pursuit, Nesterov smoothing, first-order methods, power system state estimation (PSSE).**

I. Introduction

NON-CONVEX QCQPs form an important class of optimization problems which find widespread application in various engineering disciplines (see [2] and references therein). Non-convex QCQP is NP–hard in its general form [3], so approximation algorithms are often utilized with the goal of obtaining high quality, albeit sub-optimal solutions in polynomial-time.

The prevailing approach is *Semidefinite Relaxation* (SDR) [4], [5], which uses matrix lifting coupled with rank relaxation to relax the problem into a convex Semidefinite Program (SDP). When the SDP solution is not rank-1 (which is typically the case, except for specially structured QCQPs), a post-processing step is used to convert the SDP solution into a feasible solution for the original non-convex QCQP problem. However, in instances where the constraints of the QCQP problem involve indefinite matrices and/or double sided inequalities, then such post-processing algorithms generally fail to yield a feasible point, thereby limiting the overall effectiveness of SDR – not to mention the potentially very high complexity incurred in solving the relaxed problem in SDP form.

Another popular approach is *Successive Convex Approximation* (SCA) [6]–[9], which approximates the problem via a sequence of convex subproblems initialized from a feasible point. Under certain technical conditions, convergence of the SCA iterates to a stationary point of the QCQP problem can be established. Although this framework is more broadly applicable to non-convex QCQPs compared to SDR, computing an initial feasible point for general non-convex QCQP is NP–hard as well [3].

Hence, one can conclude that determining a feasible point of a non-convex QCQP problem is the critical step for any approximation algorithm to succeed. Recently, an algorithm known as *Feasible Point Pursuit (FPP)-SCA* [10] was proposed specifically for this task. FPP-SCA uses SCA together with auxiliary slack variables to approximate the feasibility problem by a sequence of convex subproblems. The algorithm works with *any* choice of initialization, as the slack variables guarantee that each SCA subproblem is feasible at every step. Empirically, FPP-SCA demonstrates very good performance in attaining feasibility for general non-convex QCQPs. Nevertheless, the algorithm is not without its drawbacks. For one, it is required to iteratively solve a sequence of convex optimization problems via interior-point methods (IPMs), which can be computationally very demanding. In addition, eigen-decomposition of all the quadratic constraint matrices is required, followed by storing the positive and negative definite components in memory.

We point out that a consensus-ADMM (C-ADMM) algorithm for general non-convex QCQPs has been proposed in [11], which can also be used for directly computing a feasible point. The per-iteration complexity of C-ADMM is much lower than that of FPP-SCA, but the drawback is that C-ADMM is very memory intensive, since it uses local copies of the global optimization variable (one for each constraint). Due to their inherently large computational and memory footprint, FPP-SCA and C-ADMM are not well suited for solving problems in large dimensions and/or with a large number of constraints. This motivates the development of low-complexity feasibility pursuit algorithms. Towards this end, we propose to use a modified reformulation of the optimization criterion employed by FPP-SCA, which is well-suited for direct application of *first-order methods* (FOMs). The appeal of using FOMs lies in the fact that they have

Manuscript received February 20, 2017; revised June 2, 2017; accepted July 16, 2017. Date of publication August 7, 2017; date of current version September 19, 2017. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Chandra Murthy. This work was supported in part by the National Science Foundation under Project CIF-1525194 and by the University of Minnesota through a Doctoral Dissertation Fellowship. Conference version of part of this work appeared at the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing [1]. *(Corresponding author: Nicholas D. Sidiropoulos.)*

The authors are with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: konar006@umn.edu; nikos@umn.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/TSP.2017.2736516

minimal memory and computational requirements relative to other optimization schemes, which makes them well-suited for application on large-scale problems. Furthermore, FOMs can exploit data sparsity very effectively, whereas one has to typically expend considerable effort in customizing IPMs to accomplish the same feat (see [12] and references therein). Hence, in this paper, we adopt a first-order based optimization approach for general quadratic feasibility problems, instead of resorting to FPP-SCA or C-ADMM. Our interest in pursuing this approach is partially motivated from recent work which established that FOMs work remarkably well (under certain conditions) for many important non-convex problems arising in low-rank matrix regression and structured matrix factorization [13]–[18], as well as generalized phase retrieval [19]–[21]. In addition, we also attempt to partially alleviate the computational burden associated with FPP-SCA by using FOMs for solving each SCA subproblem.

In prior work [22], we have developed customized FOM-based SCA algorithms for a *special* class of non-convex QCQP, which can be interpreted as solving an alternative formulation of the feasibility problem considered here for non-convex QCQP with two-sided inequalities (see [23]). In contrast, this paper focuses on developing FOMs for the feasibility formulation in [10], which is applicable to *any* instance of non-convex QCQP.

We use judicious experiments to demonstrate the viability of using FOMs as a competitive alternative to pre-existing approaches [5], [10], [11] for determining feasible solutions of general non-convex QCQPs. In several of our experiments, we advocate using empirically chosen steps-sizes for our FOMs, which currently do not feature guaranteed convergence to the set of stationary solutions of our proposed optimization formulation. Such a choice is motivated by the following considerations: i) in several setups, the cost function we employ does not satisfy standard assumptions made in the analysis of FOMs, thereby precluding us from invoking pre-existing convergence results; ii) in other cases where these assumptions are satisfied, the requisite step-sizes are dependent on unknown constants which typically have to be estimated via crude means and ultimately result in very conservative step-sizes that exhibit poor practical performance; and most importantly, iii) a stationary point of our criterion is not guaranteed to be a feasible solution. In order to ensure convergence to a feasible solution (when one exists) using our framework, one has to establish convergence to the globally optimal solution of the non-convex cost function we utilize, which is NP–Hard in general [24]. This implies that the standard metric of ensuring convergence to a stationary point using FOMs is not sufficient to guarantee recovery of a feasible solution in this case. While this may lead one to question the merit of adopting such a FOM based approach, we point out that this should not *a priori* be construed as being a glaring drawback. Indeed, these hardness results ultimately stem from the fact that establishing (in)feasibility of an arbitrary instance of QCQP is NP–hard in general, which implies that *all* possible polynomial-time approximation schemes are doomed to fail on certain instances of the feasibility problem under consideration.

These technical issues notwithstanding, one may also doubt the potency of our direct FOM based approach on the grounds that it is too simplistic for a problem which is non-convex and NP-Hard in its general form. Hence, *a priori*, it may seem a foregone conclusion that the proposed approach is destined to perform poorly compared to sophisticated polynomial-time schemes [5], [10], [11] developed for this problem.

Given these ostensible drawbacks of our approach, the outcome of our experiments comes as a great surprise, as it reveals something entirely unanticipated: on synthetically generated feasible instances of large-scale non-convex QCQP, we provide compelling empirical evidence to demonstrate that the direct FOM based approach works remarkably well and outperforms pre-existing alternatives across all baselines. Additionally, we tested our FOMs on the problem of power system state estimation (PSSE) [26], [27], which is a real world problem arising in power systems engineering that entails solving a system of *non-random* quadratic equations, and is NP–hard in its general form [28]. Our numerical tests on standard power networks demonstrate that the FOMs can achieve very favorable performance (in terms of estimation error) at far lower complexity relative to competing alternatives.

While the authors themselves were initially skeptical about the prowess of the direct FOM based approach, given their startling empirical success, we can reasonably claim that it constitutes a significant advancement in the state-of-art for obtaining feasible solutions for general non-convex QCQP, which, for the better part of almost two decades, has principally been SDR. We hope that our work opens the door and catalyzes future research activity on new applications of non-convex QCQP.

Relative to the conference version [1], which outlines the proposed framework, this journal version brings a new FOM-based SCA algorithm, more detailed technical discussions and results, and real-world PSSE experiments using standard power network topologies.

The rest of the paper is organized as follows. Section II contains the problem statement and our optimization criterion for feasible point pursuit. Section III provides a brief overview of the FOMs we propose to use, while Section IV describes how to apply these FOMs to our formulation. Experimental results are provided in Section V and conclusions are drawn in Section VI. Supplementary results and discussions are relegated to the appendices.

Throughout the paper, we adopt the following notation. Superscript ^{T} is used to denote the transpose of a vector/matrix. Capital boldface is reserved for matrices, while vectors are denoted by small boldface. Scalars are represented in the normal face, while calligraphic font is used to denote sets. The set of natural numbers is indicated by $\mathbb{N}$. We denote the $N-$dimensional real Euclidean space by $\mathbb{R}^N$, while $\mathbb{R}^N_+$ and $\mathbb{R}^N_{++}$ represent the corresponding non-negative and positive orthants respectively. If a function $f(.)$ is differentiable, its gradient and hessian are denoted by $\nabla f(.)$ and $\nabla^2 f(.)$ respectively. For convex, non-differentiable f, the subdifferential set at a point $\mathbf{x}$ is represented by $\partial f(\mathbf{x})$.

II. Feasible Point Pursuit

This paper considers the case of quadratic feasibility problems of the form

$$\underset{\mathbf{x}\in\mathcal{X}}{\text{find}} \quad \mathbf{x} \tag{1a}$$

$$\text{s.t.} \quad \mathbf{x}^T\mathbf{A}_m\mathbf{x} - b_m \leq 0,\ \forall\, m \in \mathcal{M}_{\mathcal{I}} \tag{1b}$$

$$\mathbf{x}^T\mathbf{C}_m\mathbf{x} - d_m = 0,\ \forall\, m \in \mathcal{M}_{\mathcal{E}} \tag{1c}$$

where $\mathcal{X} \subseteq \mathbb{R}^N$ is a *simple*,[1] closed, convex set, while $\mathcal{M}_{\mathcal{I}} := \{1, 2, \cdots, M_I\}$ and $\mathcal{M}_{\mathcal{E}} := \{1, 2, \cdots, M_E\}$ represent the set of inequality and equality constraints respectively. The matrices $\{\mathbf{A}_m\}_{m=1}^{M_I}$ and $\{\mathbf{C}_m\}_{m=1}^{M_E}$ are assumed to be symmetric (without loss of generality), while $\{b_m\}_{m=1}^{M_I}$ and $\{d_m\}_{m=1}^{M_E}$ are real numbers. In the special case where $\mathbf{A}_m \succeq \mathbf{0}, \forall\, m \in \mathcal{M}_{\mathcal{I}}$ and $M_E = 0$ (i.e., the equality constraints are absent), (1) reduces to a convex feasibility problem, for which (in)feasibility can be established in polynomial-time [29]. However, the general case of (1) is a non-convex optimization problem due to the presence of the quadratic equality constraints $\mathcal{M}_{\mathcal{E}}$ and the inequality constraints $\mathcal{M}_{\mathcal{I}}$ involving possibly indefinite/negative semidefinite matrices, and is known to be NP–hard [3].

In order to establish the (in)feasibility of a given instance of (1), one may consider the following optimization problem.

$$\min_{\substack{\mathbf{x}\in\mathcal{X},\, \mathbf{s}_{\mathcal{I}}\in\mathbb{R}^{M_I},\\ \mathbf{s}_{\mathcal{E}}\in\mathbb{R}^{M_E}}} \quad \sum_{m=1}^{M_I} s_{\mathcal{I}}(m) + \sum_{m=1}^{M_E} s_{\mathcal{E}}(m) \tag{2a}$$

$$\text{s.t.} \quad \mathbf{x}^T\mathbf{A}_m\mathbf{x} - b_m \leq s_{\mathcal{I}}(m), \tag{2b}$$

$$s_{\mathcal{I}}(m) \geq 0,\ \forall\, m \in \mathcal{M}_{\mathcal{I}} \tag{2c}$$

$$-s_{\mathcal{E}}(m) \leq \mathbf{x}^T\mathbf{C}_m\mathbf{x} - d_m \leq s_{\mathcal{E}}(m), \tag{2d}$$

$$\forall\, m \in \mathcal{M}_{\mathcal{E}} \tag{2e}$$

where we have defined $\mathbf{s}_{\mathcal{I}} := [s_{\mathcal{I}}(1), \cdots, s_{\mathcal{I}}(M_I)]^T$ and $\mathbf{s}_{\mathcal{E}} := [s_{\mathcal{E}}(1), \cdots, s_{\mathcal{E}}(M_E)]^T$ as vectors of slack variables corresponding to the inequality and equality constraints respectively, with one slack variable being added to each constraint in order to ensure the feasibility of the overall problem. Note that the value of each slack variable corresponds to the degree of violation of the constraint with which it is associated. We impose an ℓ_1-penalty on the slack variables in order to promote sparsity of the constraint violations. If an optimal solution $(\mathbf{x}^*, \mathbf{s}_{\mathcal{I}}^*, \mathbf{s}_{\mathcal{E}}^*)$ of (2) can be obtained for which $\mathbf{s}_{\mathcal{I}}^* = \mathbf{0}, \mathbf{s}_{\mathcal{E}}^* = \mathbf{0}$, then $\mathbf{x}^*$ is feasible for (1). Otherwise, (1) is infeasible and from the sparsity pattern of $\mathbf{s}_{\mathcal{I}}^*$ and $\mathbf{s}_{\mathcal{E}}^*$, we can determine the constraints which cause infeasibility. Nonetheless, computing an optimal solution of (2) remains a challenging proposition since it is non-convex and NP–hard in general.

In [10], the SCA technique was used to approximate (2) via a sequence of convex subproblems. Starting from a random initialization point $\mathbf{x}^{(0)} \in \mathbb{R}^N$, at each SCA iteration $k \in \mathbb{N}$, a convex subproblem is obtained by constructing a convex *restriction* of the non-convex constraint set about the current iterate $\mathbf{x}^{(k)}$. This is accomplished by expressing each non-convex quadratic term as a difference of convex functions via eigen-decomposition of its associated matrix, followed by linearization of the non-convex term about $\mathbf{x}^{(k)}$. The resulting convex set can be expressed as

$$\mathcal{P}_r^{(k)} := \begin{cases} \mathbf{x}^T\mathbf{A}_m^{(+)}\mathbf{x} + 2\mathbf{x}^{(k)T}\mathbf{A}_m^{(-)}\mathbf{x} - \mathbf{x}^{(k)T}\mathbf{A}_m^{(-)}\mathbf{x}^{(k)} \\ \qquad\qquad -b_m \leq s_{\mathcal{I}}(m), \forall\, m \in \mathcal{M}_{\mathcal{I}} \\ \mathbf{x}^T\mathbf{C}_m^{(+)}\mathbf{x} + 2\mathbf{x}^{(k)T}\mathbf{C}_m^{(-)}\mathbf{x} - \mathbf{x}^{(k)T}\mathbf{C}_m^{(-)}\mathbf{x}^{(k)} \\ \qquad\qquad -d_m \leq s_{\mathcal{E}}(m), \forall\, m \in \mathcal{M}_{\mathcal{E}} \\ \mathbf{x}^T\mathbf{C}_m^{(-)}\mathbf{x} + 2\mathbf{x}^{(k)T}\mathbf{C}_m^{(+)}\mathbf{x} - \mathbf{x}^{(k)T}\mathbf{C}_m^{(+)}\mathbf{x}^{(k)} \\ \qquad\qquad -d_m \geq -s_{\mathcal{E}}(m), \forall\, m \in \mathcal{M}_{\mathcal{E}} \end{cases} \tag{3}$$

where $\mathbf{A}_m := \mathbf{A}_m^{(+)} + \mathbf{A}_m^{(-)}$, $\mathbf{A}_m^{(+)} \succeq \mathbf{0}$ and $\mathbf{A}_m^{(-)} \prec \mathbf{0}, \forall\, m \in \mathcal{M}_{\mathcal{I}}$ and similarly, $\mathbf{C}_m := \mathbf{C}_m^{(+)} + \mathbf{C}_m^{(-)}$, $\mathbf{C}_m^{(+)} \succeq \mathbf{0}$ and $\mathbf{C}_m^{(-)} \prec \mathbf{0}, \forall\, m \in \mathcal{M}_{\mathcal{E}}$. Thus, at SCA iteration k, we obtain a convex optimization subproblem of the form

$$\mathbf{x}^{(k+1)} \in \underset{\substack{\mathbf{x}\in\mathcal{X}\cap\mathcal{P}_r^{(k)},\\ \mathbf{s}_{\mathcal{I}}\in\mathbb{R}_+^{M_I},\, \mathbf{s}_{\mathcal{E}}\in\mathbb{R}^{M_E}}}{\arg\min} \sum_{m=1}^{M_I} s_{\mathcal{I}}(m) + \sum_{m=1}^{M_E} s_{\mathcal{E}}(m) \tag{4}$$

with the solution of the resulting problem being used for linearization in the next iteration. The overall algorithm has been termed as *Feasible Point Pursuit (FPP)-SCA*. Utilizing the theoretical results developed in [30],[2] we can provide the following characterization of the sequence of generated iterates.

1) The iterate sequence has non-increasing cost.
2) Assuming there exists a convergent subsequence, then provided that Slater's Condition [30, Sec. 2.1.2] is satisfied at the limit of this subsequence, the limit point satisfies the KKT conditions of (2).

We point out that these theoretical results do not imply that FPP-SCA is guaranteed to converge to a feasible point of (1); only convergence to a KKT point of the feasibility problem (2) can be established. Nonetheless, FPP-SCA was empirically demonstrated to be highly successful in converging to a zero-slack solution (i.e., attain feasibility) in a finite number of iterations for various instances of (1). However, this comes at the expense of overall problem complexity. Each SCA subproblem (4) can be recast in Second-order Cone Programming (SoCP) form and solved via general purpose conic programming solvers (which use IPMs) at a worst case complexity of $\mathcal{O}(N + M_I + M_E)^{3.5}$ [31], which is dependent on both N (the number of variables) and $M := M_I + M_E$ (the total number of constraints). It is evident that iteratively solving a sequence of SCA subproblems via such means can prove to be very computationally expensive for large N and/or M. Furthermore, it is required to compute the eigen-decomposition of the matrices $\{\mathbf{A}_m\}_{m=1}^{M_I}$ and $\{\mathbf{C}_m\}_{m=1}^{M_E}$ in order to separate each matrix into its positive and negative definite components, which are then stored in memory. Thus, the overhead in terms of memory can also prove to be quite substantial for large-scale problems.

With the aim of improving scalability and alleviating the aforementioned issues with FPP-SCA, we consider the possibility of using FOMs for feasible point pursuit. As a first step in this direction, we consider the following reformulation of the feasibility problem (2)

$$\min_{\mathbf{x}\in\mathcal{X}} \left\{ F(\mathbf{x}) := \sum_{m=1}^{M_I} (\mathbf{x}^T\mathbf{A}_m\mathbf{x} - b_m)^+ + \sum_{m=1}^{M_E} |\mathbf{x}^T\mathbf{C}_m\mathbf{x} - d_m| \right\} \tag{5}$$

where $(x)^+ := \max\{x, 0\}$ and $|x|$ denotes the absolute value of x. Note formulations (2) and (5) are equivalent at the globally optimal solutions of these two problems. The reformulation results in a problem where all the non-convex constraints of (2) have been incorporated into the cost function which is composed of the sum of M non-convex, non-smooth functions; each of which measures the degree of violation of its corresponding constraint

[1] By simple, we mean that Euclidean projections onto $\mathcal{X}$ can be computed in closed form.

[2] To be precise, we verify that the constraint approximation functions defined in (3) satisfy the conditions laid out in [30, Assumption 1], followed by invoking [30, Th. 1].

via a loss function (quadratic hinge-loss for the inequality constraints and absolute value for the equality constraints). In the literature, such a formulation is also known as an *exact penalty formulation* [32]. We note that (5) remains non-convex and is NP–hard in general. At this point, we can choose to proceed in one of the following two ways.

1) We can apply SCA on the exact penalty formulation (5) to obtain a convex optimization subproblem at each iteration, and then utilize FOMs to solve each subproblem at a reduced computational complexity relative to interior-point methods.
2) Alternatively, we can go one step further by eliminating the SCA procedure from consideration altogether and focus on tackling (5) *directly* via FOMs.

In subsequent sections, we describe in detail how to implement these approaches, followed by experimental evaluations to compare and contrast their performance.

III. Overview of First-Order Methods

In this section, we provide a brief overview of the various FOMs which we propose to use for both direct (non-convex) and SCA (convex) approaches.

A. Direct Approach

Consider the following optimization problem of minimizing averages of finite sums

$$\min_{\mathbf{x}\in\mathcal{X}}\left\{F(\mathbf{x}) := \frac{1}{M}\sum_{m=1}^{M} f_m(\mathbf{x})\right\} \tag{6}$$

where $\mathcal{X}\subset\mathbb{R}^N$ is a convex, compact set and each $f_m : \mathbb{R}^N \to \mathbb{R}$ is a twice differentiable, non-convex function with $L-$Lipschitz continuous gradients[3]; i.e, $\exists\, L \in \mathbb{R}_{++}$ for which

$$\|\nabla f_m(\mathbf{x}) - \nabla f_m(\mathbf{y})\|_2 \leq L\|\mathbf{x}-\mathbf{y}\|_2, \forall\, \mathbf{x},\mathbf{y}\in\mathcal{X} \tag{7}$$

When F is bounded below over $\mathcal{X}$, we can attempt to determine an approximate solution for (6) using the classical *gradient descent* (GD) algorithm which has the following update rule.

$$\mathbf{y}^{(k)} = \mathbf{x}^{(k-1)} - \frac{\alpha_k}{M}\sum_{m=1}^{M}\nabla f_m(\mathbf{x}^{(k-1)}) \tag{8a}$$

$$\mathbf{x}^{(k)} = \Pi_{\mathcal{X}}(\mathbf{y}^{(k)}), \forall\, k\in\mathbb{N} \tag{8b}$$

where $\Pi_{\mathcal{X}}(.)$ denotes the Euclidean projection operator onto $\mathcal{X}$ and $\alpha_k \in \mathbb{R}_{++}$ is the step-size in the kth iteration.

Note that each step requires the computation of M gradients, and hence can be fairly expensive for large M. As a low complexity alternative, we can consider using *stochastic gradient descent* (SGD). The algorithm is iterative in nature, where at each iteration k we randomly draw an index m_k from a uniform distribution defined on the index set $\mathcal{M} = \{1,\cdots,M\}$ and then apply the following update rule

$$\mathbf{y}^{(k)} = \mathbf{x}^{(k-1)} - \alpha_k\nabla f_{m_k}(\mathbf{x}^{(k-1)}) \tag{9a}$$

$$\mathbf{x}^{(k)} = \Pi_{\mathcal{X}}(\mathbf{y}^{(k)}), \forall\, k\in\mathbb{N} \tag{9b}$$

[3]This in turn implies $F(\mathbf{x})$ is $L-$Lipschitz smooth since smoothness is preserved under convex combinations.

Note that the expectation $\mathbb{E}(\mathbf{y}^{(k)}|\,\mathbf{x}^{(k-1)})$ equals (8a) (where the expectation is taken with respect to the random variable m_k). Hence, the SGD updates (9) are equivalent to standard GD updates in expectation. The advantage of SGD is that the updates are $\mathcal{O}(M)$ cheaper compared to GD since at each iteration, we only need to compute the gradient of a single component function.

A third alternative, which has emerged recently, is *Stochastic Variance Reduced Gradient* (SVRG) [33], [34]. The SVRG algorithm can be viewed as a hybrid between SGD and GD, and proceeds in multiple stages. In each stage s, SVRG defines a "centering" variable $\mathbf{y}_s$ from the output of the previous stage and computes its full gradient $\nabla F(\mathbf{y}_s)$. Next, a fixed number (say K) of modified inner SGD iterations are executed, where in each iteration $k\in\{1,\cdots,K\}$, an index m_k is drawn uniformly at random from $\mathcal{M}$ and the following update rule is used

$$\mathbf{x}_s^{(0)} = \mathbf{y}_s \tag{10a}$$

$$\mathbf{v}_s^{(k)} = \mathbf{x}_s^{(k-1)} - \alpha_s^{(k)}(\nabla f_{m_k}(\mathbf{x}_s^{(k-1)}) - \nabla f_{m_k}(\mathbf{y}_s) + \nabla F(\mathbf{y}_s)) \tag{10b}$$

$$\mathbf{x}_s^{(k)} = \Pi_{\mathcal{X}}(\mathbf{v}_s^{(k)}), \forall\, k\in\{1,\cdots,K\} \tag{10c}$$

where the superscript k denotes the inner SGD iteration counter for stage s. Again, the expectation $\mathbb{E}(\mathbf{v}_s^{(k)}|\,\mathbf{x}_s^{(k-1)})$ equals (8a). Hence, in expectation, the SVRG updates are also the same as the GD updates. However, compared to SGD, SVRG uses a different unbiased gradient estimator which corrects the currently sampled gradient $\nabla f_{m_k}(\mathbf{x}_s^{(k-1)})$ by subtracting a bias term. The overall algorithm is given by

Algorithm 1: SVRG.

Initialization: Select number of stages S, update frequency K and step-size sequence. Randomly generate a starting point $\mathbf{z}_0\in\mathcal{X}$.

Iterate: for $s = 1, 2, \cdots, S$

- Set $\mathbf{y}_s = \mathbf{z}_{s-1}$
- Compute $\mathbf{g}_s := \nabla F(\mathbf{y}_s)$
- Set $\mathbf{x}_s^{(0)} = \mathbf{y}_s$
- **Iterate:** for $k = 1,\cdots,K$
  Randomly pick $m_k\in\{1,\cdots,M\}$ and update
  $\mathbf{v}_s^{(k)} = \mathbf{x}_s^{(k-1)} - \alpha_k^{(s)}(\nabla f_{m_k}(\mathbf{x}_s^{(k-1)}) - \nabla f_{m_k}(\mathbf{y}_s) + \mathbf{g}_s)$,
  $\mathbf{x}_s^{(k)} = \Pi_{\mathcal{X}}(\mathbf{v}_s^{(k)})$
- **End**
- Set $\mathbf{z}_s = \mathbf{x}_s^{(K)}$

End

Return: $\mathbf{z}_S$

B. SCA Approach

Consider the following optimization problem

$$\min_{\mathbf{x}\in\mathcal{X}} H(\mathbf{x}) \tag{11}$$

where $\mathcal{X}\subset\mathbb{R}^N$ is a compact, convex set and $H:\mathbb{R}^N\to\mathbb{R}$ is a non-smooth, Lipschitz continuous convex function. A standard method for solving such problems is *subgradient descent* (SD),

which has the following update rule.

$$\mathbf{x}^{(k)} = \Pi_{\mathcal{X}}(\mathbf{x}^{(k-1)} - \alpha_k \mathbf{g}^{(k-1)}), \forall\, k \in \mathbb{N} \tag{12}$$

where $\mathbf{g}^{(k-1)} \in \partial H(\mathbf{x}^{(k-1)})$ is a subgradient drawn from the subdifferential set of the function H at the point $\mathbf{x}^{(k-1)}$.

In the special case $H(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} h_m(\mathbf{x})$, where each $h_m(\mathbf{x})$ is a non-smooth, Lipschitz continuous function, then we can alternatively use the technique of *stochastic subgradient descent* (SSD), which proceeds according to the following update rule

$$\mathbf{x}^{(k)} = \Pi_{\mathcal{X}}(\mathbf{x}^{(k-1)} - \alpha_k \mathbf{g}_{m_k}^{(k-1)}), \forall\, k \in \mathbb{N} \tag{13}$$

where the index m_k is drawn uniformly at random from the set $\mathcal{M}$ and $\mathbf{g}_{m_k}^{(k-1)}$ is a subgradient drawn from the subdifferential set of the function f_{m_k} at the point $\mathbf{x}^{(k-1)}$. Then, we have $\mathbb{E}(\mathbf{g}_{m_k}^{(k-1)}|\mathbf{x}^{(k-1)}) \in \partial F(\mathbf{x}^{(k-1)})$, which implies that the SSD iterations are equivalent to SD in expectation.

This concludes our bare-bones overview on FOMs. A complete discussion of the selection of step-sizes and other problem parameters and their impact on the convergence of the above algorithms is deferred to Appendix A. In the following section, we describe how to apply these methods to (5).

IV. Proposed Algorithms

A. Direct Approach

We refrain from using SD/SSD on (5), since the subdifferential set of a non-smooth, non-convex function is not guaranteed to be non-empty at all points in its domain. This leaves us with GD, SGD and SVRG at our disposal. However, these methods are applicable to differentiable cost functions, whereas each component function of (5) is non-differentiable. Consequently, we propose to make the following modifications to (5).

First, consider the hinge-loss functions corresponding to the quadratic inequality constraints. Define $f_m(\mathbf{x}) := (\mathbf{x}^T\mathbf{A}_m\mathbf{x} - b_m)^+, \forall\, m \in \mathcal{M}_{\mathcal{I}}$. We now describe a procedure for constructing a smooth surrogate for each $f_m(\mathbf{x})$. Note that each $f_m(\mathbf{x})$ can be equivalently expressed as

$$f_m(\mathbf{x}) = \max_{0\le y\le 1}\{y(\mathbf{x}^T\mathbf{A}_m\mathbf{x} - b_m)\}, \forall\, m \in \mathcal{M}_{\mathcal{I}} \tag{14}$$

In order to construct a smooth surrogate of $f_m(\mathbf{x})$, consider the following modified version of (14)

$$f_m^{(\mu)}(\mathbf{x}) = \max_{0\le y\le 1}\left\{y(\mathbf{x}^T\mathbf{A}_m\mathbf{x} - b_m) - \mu\frac{y^2}{2}\right\}, \forall\, m \in \mathcal{M}_{\mathcal{I}} \tag{15}$$

where $\mu \in \mathbb{R}_{++}$ is a smoothing parameter. The maximization problem (15) can be solved in closed form to obtain the following equivalent smooth representation

$$f_m^{(\mu)}(\mathbf{x}) = \begin{cases} 0, & \text{if } \mathbf{x}^T\mathbf{A}_m\mathbf{x} \le b_m \\ \frac{(\mathbf{x}^T\mathbf{A}_m\mathbf{x} - b_m)^2}{2\mu}, & \text{if } b_m < \mathbf{x}^T\mathbf{A}_m\mathbf{x} \le b_m + \mu \\ \mathbf{x}^T\mathbf{A}_m\mathbf{x} - b_m - \frac{\mu}{2}, & \text{if } \mathbf{x}^T\mathbf{A}_m\mathbf{x} > b_m + \mu \end{cases} \tag{16}$$

The derivation is relegated to Appendix B. Note that each $f_m^{(\mu)}(\mathbf{x})$ has continuous derivatives given by

$$\nabla f_m^{(\mu)}(\mathbf{x}) = \begin{cases} 0, & \text{if } \mathbf{x}^T\mathbf{A}_m\mathbf{x} \le b_m \\ \frac{2(\mathbf{x}^T\mathbf{A}_m\mathbf{x} - b_m)}{\mu}\mathbf{A}_m\mathbf{x}, & \text{if } b_m < \mathbf{x}^T\mathbf{A}_m\mathbf{x} \le b_m + \mu \\ 2\mathbf{A}_m\mathbf{x}, & \text{if } \mathbf{x}^T\mathbf{A}_m\mathbf{x} > b_m + \mu \end{cases} \tag{17}$$

Hence, $f_m^{(\mu)}(\mathbf{x})$ is a smooth surrogate of $f_m(\mathbf{x}), \forall\, m \in \mathcal{M}_{\mathcal{I}}$. Furthermore, it can be shown that the following approximation bounds hold (see Appendix C).

$$f_m^{(\mu)}(\mathbf{x}) \le f_m(\mathbf{x}) \le f_m^{(\mu)}(\mathbf{x}) + \frac{\mu}{2}, \forall\, \mathbf{x} \in \mathbb{R}^N, \forall\, m \in \mathcal{M}_{\mathcal{I}} \tag{18}$$

The smoothing technique employed in (15) can be viewed as an extension of *Nesterov smoothing* [35] to the non-convex case – albeit in the non-convex setting the representation of $f_m^{(\mu)}(\mathbf{x})$ in (15) does not correspond to the Fenchel conjugate of a strongly-convex function.

As for the absolute value penalty functions $g_m(\mathbf{x}) := |\mathbf{x}^T\mathbf{C}_m\mathbf{x} - d_m|, \forall\, m \in \mathcal{M}_{\mathcal{E}}$ in (5) corresponding to the equality constraints, we propose to replace them with quadratic penalty functions of the form

$$g_m^{(q)}(\mathbf{x}) := (\mathbf{x}^T\mathbf{C}_m\mathbf{x} - d_m)^2, \forall\, m \in \mathcal{M}_{\mathcal{E}} \tag{19}$$

Following these steps, we obtain a non-convex, differentiable penalty formulation given by

$$\min_{\mathbf{x}\in\mathcal{X}}\left\{F^{(s)}(\mathbf{x}) := \frac{1}{M}\left(\sum_{m=1}^{M_I} f_m^{(\mu)}(\mathbf{x}) + \sum_{m=1}^{M_E} g_m^{(q)}(\mathbf{x})\right)\right\} \tag{20}$$

which is now in a form suitable for application of GD, SGD and SVRG. The convergence behavior of these algorithms is determined by the choice of the step-size sequence (and the additional parameters in the case of SVRG). While several theoretical results have been developed recently which establish non-asymptotic rates of convergence of these algorithms to a stationary point of non-convex problems of the form (6) for appropriate choices of parameters (see Appendix A for a full comparison), adapting these results to the problem under consideration is hampered by the following technical issues.

First, we point out that the cost function of (20) is a quartic polynomial which is neither globally Lipschitz continuous, nor does it possess globally Lipschitz continuous derivatives. Hence, in the unconstrained case (i.e., $\mathcal{X} = \mathbb{R}^N$), the existing non-asymptotic convergence results for GD, SGD and SVRG cannot be applied. When these assumptions do not hold, even establishing meaningful asymptotic convergence guarantees is a challenging proposition in general. In Appendix D, we show that when $M_I = 0$ in (20) (i.e., solving a general system of quadratic equations), it is indeed possible to establish such a meaningful, asymptotic convergence result for GD with backtracking line search.

Next, we consider the case $\mathcal{X} \subset \mathbb{R}^N$. If we make an additional assumption that $\mathcal{X}$ is compact, then $F^{(s)}(\mathbf{x})$ and its gradients are (locally) Lipschitz continuous on $\mathcal{X}$. It can be shown that the Lipschitz constant of $\nabla F^{(s)}(\mathbf{x})$ exhibits a $\mathcal{O}(\frac{1}{\mu})$ dependence on

μ, which is due to the fact that μ is a parameter which controls the level of smoothing applied to the non-differentiable function $F(\mathbf{x})$; i.e., a smaller value of μ allows a tighter degree of approximation, but results in $F^{(s)}(\mathbf{x})$ being less smooth. Typically, we would prefer to choose a small value for μ in order to ensure tight approximation to $F(\mathbf{x})$. Meanwhile, the constants appearing in the $\mathcal{O}(\frac{1}{\mu})$ expression (which depend on the spectral characteristics of the matrices in the constraints) are typically unknown and have to be estimated via some means. In general, such procedures generate overestimates of the constants, which adversely affect convergence speed. Taken together, this ultimately results in the step-size dictated by theory for convergence of GD and SGD being too conservative for the iterates to make any reasonable progress over a prescribed number of iterations.

Finally, we point out that we are ultimately interested in determining a feasible point (i.e., a point $\mathbf{x} \in \mathcal{X}$ for which the globally optimal cost $F^{(s)}(\mathbf{x}) = 0$ is attained) using our FOMs. Hence, it is evident that convergence to a stationary point (which only satisfies the necessary conditions for optimality) is not sufficient to guarantee feasibility. In this case, ensuring recovery of a feasible solution (when one exists) requires establishing convergence to the globally optimal cost 0, which, given the fact that (20) is NP–hard in its general form, is considerably more difficult to establish relative to showing convergence to a stationary point. Hence, in several of our experiments, we resorted to empirical step-size selection strategies for our FOMs. Although we cannot make any theoretical convergence claims for such step-sizes, our experiments indicate that these methods can still perform very favorably with these choices.

B. SCA Approach

The SCA approach is based on approximating (5) via a sequence of convex problems. Since the non-convexity in (5) is restricted to the cost function, this entails approximating the cost function of (5) via a sequence of convex majorization functions. We now describe the procedure for constructing such a majorization function at each iteration.

First, consider the hinge-loss functions $f_m(\mathbf{x}), \forall\, m \in \mathcal{M}_{\mathcal{I}}$. Again, we utilize eigen-decomposition to decompose each matrix $\mathbf{A}_m$ into its constituent positive and negative semidefinite components and then express the associated quadratic term as a difference of quadratic convex functions. After linearizing the concave term $\mathbf{x}^T \mathbf{A}_m^{(-)} \mathbf{x}$ about the current iterate $\mathbf{x} = \mathbf{x}^{(k)}$, we obtain the following function

$$u_m(\mathbf{x}, \mathbf{x}^{(k)}) := (\mathbf{x}^T \mathbf{A}_m^{(+)} \mathbf{x} + (2\mathbf{A}_m^{(-)} \mathbf{x}^{(k)})^T \mathbf{x} - \mathbf{x}^{(k)T} \mathbf{A}_m^{(-)} \mathbf{x}^{(k)} - b_m)^+, \forall\, m \in \mathcal{M}_{\mathcal{I}} \quad (21)$$

It can be readily verified that $\forall\, m \in \mathcal{M}_{\mathcal{I}}$, $u_m(\mathbf{x}, \mathbf{x}^{(k)})$ is a convex, non-differentiable majorizer of $f_m(\mathbf{x})$ which is tight at $\mathbf{x} = \mathbf{x}^{(k)}$. Next, we equivalently express each absolute penalty function $g_m(\mathbf{x})$ as

$$g_m(\mathbf{x}) = |\mathbf{x}^T \mathbf{C}_m \mathbf{x} - d_m| = \max\{\mathbf{x}^T \mathbf{C}_m \mathbf{x} - d_m, -\mathbf{x}^T \mathbf{C}_m \mathbf{x} + d_m\}, \forall\, m \in \mathcal{M}_{\mathcal{E}} \quad (22)$$

In order to majorize each such $g_m(\mathbf{x})$, we resort to the eigen-decomposition technique to express each of the quadratic terms inside the point-wise maximization operator as the difference of convex quadratics. By linearizing the appropriate non-convex term about $\mathbf{x} = \mathbf{x}^{(k)}$, we obtain the pair of convex functions

$$v_m^{(+)}(\mathbf{x}, \mathbf{x}^{(k)}) := \mathbf{x}^T \mathbf{C}_m^{(+)} \mathbf{x} + (2\mathbf{C}_m^{(-)} \mathbf{x}^{(k)})^T \mathbf{x} - \mathbf{x}^{(k)T} \mathbf{C}_m^{(-)} \mathbf{x}^{(k)} - d_m, \forall\, m \in \mathcal{M}_{\mathcal{E}} \quad (23a)$$

$$v_m^{(-)}(\mathbf{x}, \mathbf{x}^{(k)}) := -\mathbf{x}^T \mathbf{C}_m^{(-)} \mathbf{x} - (2\mathbf{C}_m^{(+)} \mathbf{x}^{(k)})^T \mathbf{x} + \mathbf{x}^{(k)T} \mathbf{C}_m^{(+)} \mathbf{x}^{(k)} + d_m, \forall\, m \in \mathcal{M}_{\mathcal{E}} \quad (23b)$$

On defining

$$\omega_m(\mathbf{x}, \mathbf{x}^{(k)}) := \max\{v_m^{(+)}(\mathbf{x}, \mathbf{x}^{(k)}), v_m^{(-)}(\mathbf{x}, \mathbf{x}^{(k)})\}, \forall\, m \in \mathcal{M}_{\mathcal{E}} \quad (24)$$

we obtain a convex majorization function for each $g_m(\mathbf{x}), \forall\, m \in \mathcal{M}_{\mathcal{E}}$. Hence, at each SCA iteration, we obtain a non-smooth, convex optimization problem of the following form

$$\mathbf{x}^{(k+1)} \in \arg\min_{\mathbf{x} \in \mathcal{X}} \sum_{m=1}^{M_I} u_m(\mathbf{x}, \mathbf{x}^{(k)}) + \sum_{m=1}^{M_E} \omega_m(\mathbf{x}, \mathbf{x}^{(k)}) \quad (25)$$

We now point out that problem (25) is actually equivalent to (4), since (4) can be obtained via the epigraph transformation of (25). Hence, the resulting SCA algorithm inherits the same convergence properties as FPP-SCA.[4] From a computational standpoint, formulation (25) possesses the advantage of being in a form suitable for the application of low-complexity subgradient methods. While subgradient descent can be applied to solve each SCA subproblem of the form (25) at a rate independent of the problem dimension N, there is still an implicit dependence on the total number of constraints M (see Appendix A). In order to remove the dependence on M, we can solve (25) using the SSGD algorithm, which has the benefit of possessing a convergence rate independent of N and M. However, the drawback of using SSGD is that it only converges in expectation, thus implying that the SCA iterates obtained via this method are not even guaranteed to exhibit monotonic decrease of the cost function in this case. Nevertheless, it offers a substantially low-complexity alternative for decreasing the cost function of (25) initially, with possible "last mile" refinement at a later stage via a more sophisticated algorithm.

For a given SCA subproblem (25), at each iteration of SSGD, we are only required to sample an index m from the set $\{1, \cdots, M\}$ uniformly at random and then compute a subgradient for the associated function indexed by m in order to compute the update (13). If the indexed function is of the form $f(\mathbf{x}) = (\mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{b}^T \mathbf{x} + c)^+$, (where $\mathbf{A} \succeq \mathbf{0}$) we can compute a subgradient $\mathbf{g} \in \partial f(\mathbf{x})$ at the point $\mathbf{x}$ according to the following equation

$$\mathbf{g} = \begin{cases} 2\mathbf{A}\mathbf{x} + \mathbf{b}, & \text{if } \mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{b}^T \mathbf{x} + c > 0 \\ \mathbf{0}, & \text{otherwise} \end{cases} \quad (26)$$

whereas for $f(\mathbf{x}) = \max\{h_1(\mathbf{x}), h_2(\mathbf{x})\}$, where $h_i(\mathbf{x}) = \mathbf{x}^T \mathbf{A}_i \mathbf{x} + \mathbf{b}_i^T \mathbf{x} + c_i, \forall\, i = \{1, 2\}$ are convex quadratics, a subgradient $\mathbf{g} \in \partial f(\mathbf{x})$ at the point $\mathbf{x}$ can be obtained by simply

[4] i.e., convergence to a KKT point of the *smooth* feasibility problem (2).

selecting any one of the two functions $h_1(\mathbf{x}), h_2(\mathbf{x})$ which attains the maximum and then taking its gradient. Additionally, in each SCA iteration, we warm start the SSGD algorithm from the current iterate $\mathbf{x}^{(k)}$ in order to obtain further savings in computation.

V. Experimental Results

In this section, we evaluate and compare the performance of our methods on synthetically generated experiments as well as on real engineering problems. First, we provide a few details regarding the implementation of the methods.

A. Implementation

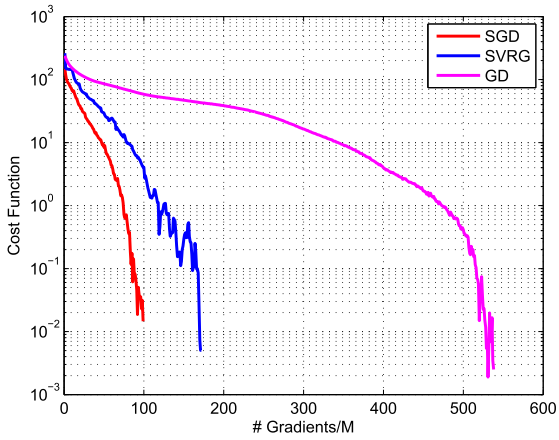
All our methods were implemented in MATLAB on a Linux desktop with 4 Intel i7 cores and 16 GB of RAM. We tested the performance of the direct methods (i.e., GD, SGD, and SVRG) with the following parameter settings.

1) The smoothing parameter for inequality constraints was set to $\mu = 10^{-4}$.
2) For selecting the step-size, the following rules were used
   a) *Diminishing:* $\alpha_k = \frac{c_1}{k^\gamma}$
   b) *Polynomial:* $\alpha_k = \frac{c_2}{(1+c_3 k/M)^\gamma}$
   c) *Norm regularized:* $\alpha_k = c_4/\|\mathbf{x}^{(k)}\|_2^2$
   where $k \in \mathbb{N}$ denotes the iteration index, $c_1, c_2, c_3, c_4 \in \mathbb{R}_{++}$ and $\gamma \in (0, 1]$. The polynomial step-size rule can be viewed as a generalization of the popular inverse-t schedule (corresponding to $\gamma = 1$) while the norm regularized step-size rule can be motivated via arguments made in [14, Proposition 1] regarding worst-case convergence results of FOMs applied to minimize quartic functions. The parameters were empirically tuned to yield the best performance.
3) For SVRG, the length of each stage was set to $S = 4M$.
4) Since each method requires a different number of gradient evaluations per iteration, for fair comparison, we allocated a fixed number of total gradient evaluations to each method and evaluated the cost function after every M gradient evaluations. Of course, this implies that the maximum number of iterations for each method is different, depending on the number of gradients evaluated per iteration.

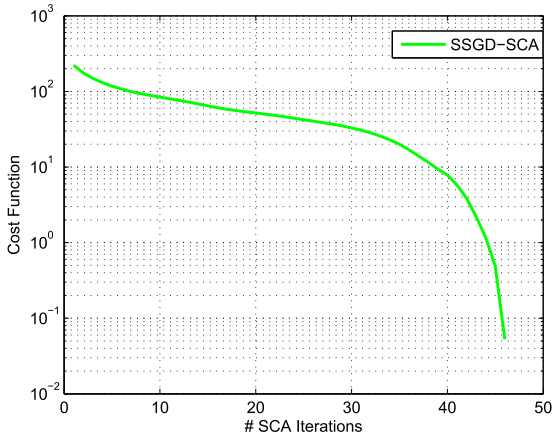
Regarding SSGD-SCA, we used a maximum of 50 SCA iterations while each inner convex subproblem was solved using 50×10^3 SSGD iterations with a step-size of $\mathcal{O}(\frac{1}{\sqrt{k}})$ and iterate averaging.

B. Experiments on Synthetic Data

First, we present an illustrative experiment on a synthetically generated instance of a non-convex QCQP problem with $N = 200$ variables and $M = 1000$ constraints. Here, we set $M_E = 0$ (i.e., no equalities) and randomly generated the inequality constraint matrices $\{\mathbf{A}_m\}_{m=1}^M$ from a zero mean, i.i.d. Gaussian distribution with unit variance (followed by symmetrization). In order to ensure that the problem is feasible, we randomly generated a unit norm vector $\mathbf{p}$ and drew each of the right-hand sides $\{b_m\}_{m=1}^M$ from a Gaussian distribution



(a) Evolution of penalty function for direct FOMs



(b) Evolution of penalty function for SSGD-SCA

Fig. 1. Single instance of feasibility problem with $N = 200$ variables and $M = 1000$ non-convex quadratic inequalities.

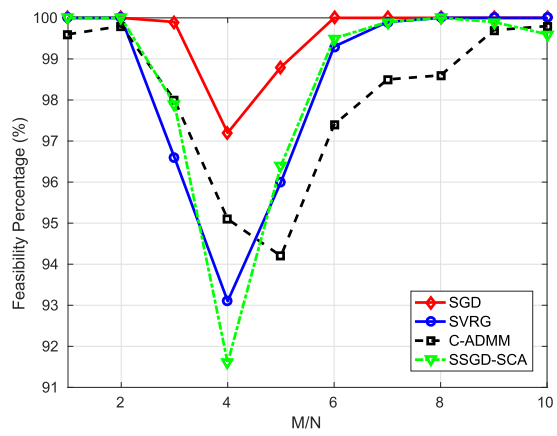
$b_m \sim \mathcal{N}(\mathbf{p}^T \mathbf{A}_m \mathbf{p}, 1)$. In the event $\mathbf{p}^T \mathbf{A}_m \mathbf{p} > b_m$, we multiplied both sides of the inequality by -1 to get $\leq$ inequalities. In short, we randomly generated a quadratic feasibility problem with indefinite matrices which possesses a unit-norm feasible solution. We exploit this prior knowledge in our setup by setting $\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^N | \|\mathbf{x}\|_2 \leq 1\}$. A randomly generated unit-norm vector was used to initialize GD, SGD, SVRG and SSGD-SCA. We used a maximum budget of $1000M$ gradient evaluations for each of the direct methods. For SGD, we use the diminishing step-size rule with $c_1 = 0.1$ and $\gamma = 0.5$, while for SVRG and GD, we used the polynomial averaging step-size rule with $c_3 = 1$ and $c_2 = 0.1, \gamma = 1$ for GD and $c_2 = 0.01, \gamma = 0.5$ for SVRG. We declare success in finding a feasible point if the value of the cost function in the exact penalty formulation (5) is smaller than a tolerance value of 10^{-6}. The results are depicted in Fig. 1, where we plot the evolution of the constraint violation (as measured by the quadratic hinge-loss function in (5)) for the various methods.

In this case, all methods were successful in achieving feasibility; i.e., attaining the globally optimal cost 0. Since the evolution of the penalty function in Fig. 1 is represented on a

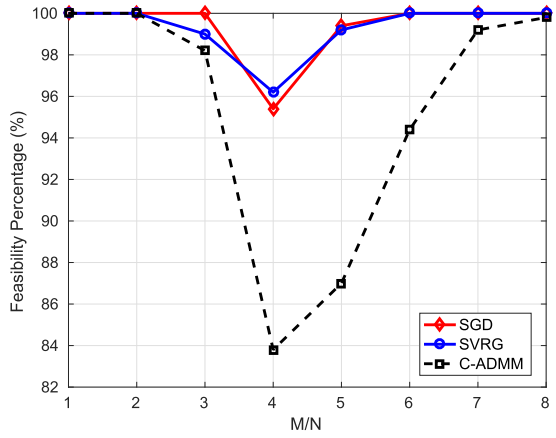
logarithmic scale, it depicts the cost of the iterates before the value of 0 was attained. For the direct FOMs, the potential benefits accrued in opting for an aggressive, empirically chosen step-size strategy is clear. Meanwhile, it is obvious that, given the scale of the problem, we should refrain from using IPMs to solve each SCA sub-problem. As an alternative, SSGD-SCA performs admirably, and even exhibits monotonic decrease of the cost function in this case. Regarding timing, SGD performed the best by attaining feasibility in 17 secs, while SVRG, GD and SSGD-SCA required 27, 85 and 233 secs respectively. Note that although we adopt a FOM based approach to SCA here, it still incurs substantially more complexity compared to the direct FOM approach. Overall, these preliminary results indicate that the direct FOMs have a distinct advantage over SSGD-SCA. We now seek to corroborate these findings via the following set of exhaustive simulations.

In our setup, we fixed the number of variables N and randomly generated instances of quadratic feasibility problems with varying number of inequality constraints M via the procedure describe in the previously. For each value of M, we generated 1000 such instances. We also added the C-ADMM algorithm proposed in [11] for comparison against our FOMs. We set the maximum iteration counter of C-ADMM to 5000 iterations. Meanwhile, for the direct FOMs, we again use a budget of $1000M$ gradient evaluations and a maximum of 50 SCA iterations for SSGD-SSCA. We also remove GD from contention here since our experiments indicate that it is always outperformed by SGD and SVRG at lower complexity. Furthermore, we allow a maximum of 2 restarts for SGD and SVRG in the event that feasibility is not attained within the prescribed number of iterations. In each instance, we initialize all the methods from a randomly generated unit-norm vector. The step-size rules for the direct FOMs and SSGD-SCA are also unchanged from the previous experiment. An alternative approach could be to tune the step-size parameters to achieve the best performance for each M, at the cost of more effort. As we demonstrate, our chosen parameters work well across a wide range of N and M, thereby considerably alleviating the burden of tuning parameters while simultaneously providing further empirical validation of our heuristic step-size sequences. We also used the same termination criterion used in the previous experiment for declaring convergence to a feasible point for all methods.

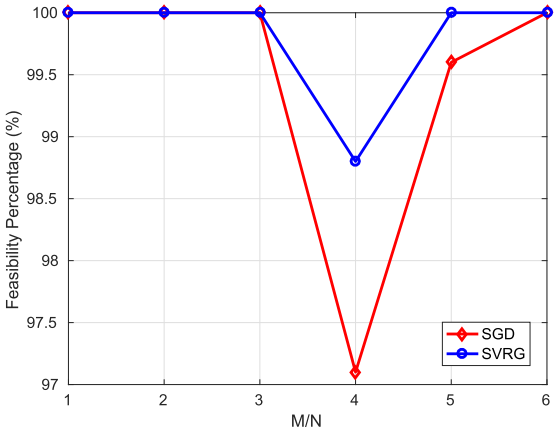
We plot the feasibility percentages averaged out over 1000 instances in Fig. 2 as a function of M/N while the timing results, averaged out over the subset of instances where feasibility was successfully attained by the respective algorithms, are depicted in Fig. 3. From these figures, it can be observed that for all N considered, SGD and SVRG demonstrate the best performance in terms of feasibility and timing (including restarts). For variable dimensions $N = 100$ and larger, the running time of SSGD-SCA becomes too expensive to merit comparison. Similarly, for $N = 200$, we remove C-ADMM as well, since the average running time of C-ADMM is approximately 20 minutes in this case. In contrast, even for $N = 200$, the worst average running-time of SGD and SVRG, with restarts taken into account, is slightly in excess of one minute. It is evident that SVRG and SGD are significantly more scalable compared to the existing state-of-art, while, remarkably, exhibiting near-optimal performance with regard to recovering feasible solutions in all



(a) $N = 50$



(b) $N = 100$

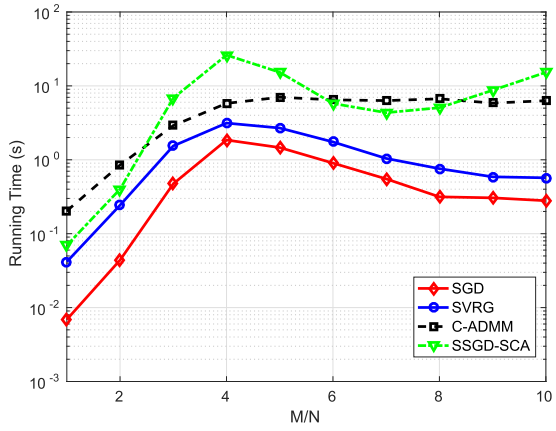


(c) $N = 200$

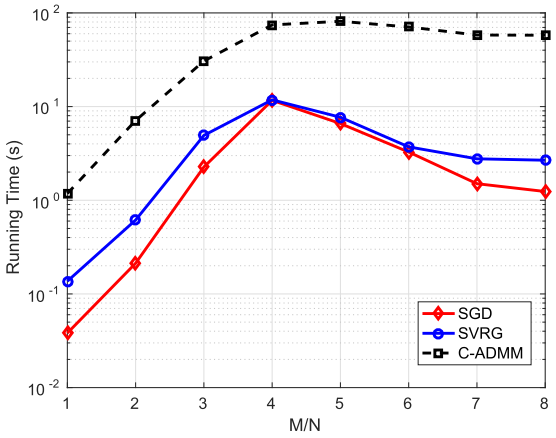
Fig. 2. Average feasibility percentage versus M/N over 1000 instances.

cases. Hence, SGD and SVRG emerge as the algorithms of choice in this case.

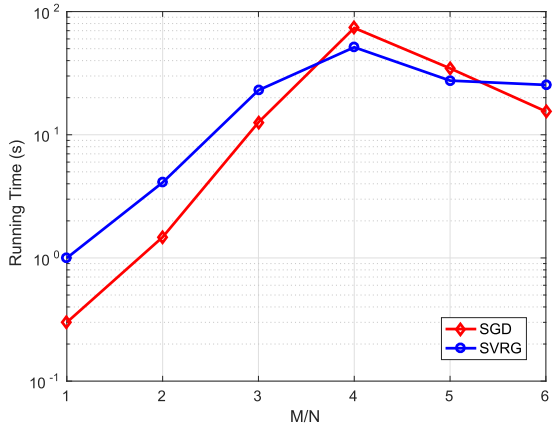
Finally, although the algorithms we have applied on the feasibility problem are quite different from each other, we point out that they exhibit a slight phase transition in terms of feasibility percentages as M/N varies. Note that this effect is least pronounced overall in the case of SGD and SVRG. The presented results (showing enhanced feasibility with more



(a) $N = 50$



(b) $N = 100$



(c) $N = 200$

Fig. 3. Average timing (seconds) versus M/N over 1000 instances.

constraints) hinge upon the method adopted for generating instances of (1), in particular that a unit-norm solution exists. While we do not have a complete explanation of this phenomenon at present, a similar observation has been made and theoretically explained in the special case of solving random systems of quadratic equations (see [18], [25] for details). We conjecture that a similar line of reasoning can also be applied here; however, formally establishing this argument is beyond the scope of this paper.

C. Experiments on Real Data

In this section, we investigate the efficacy of using FOMs on the power system state estimation (PSSE) problem, which entails solving a system of indefinite quadratic equations, thus rendering it NP–hard in its general form. PSSE is a classical problem in power systems engineering [26] where, given a power transmission network comprising $\mathcal{N} := \{1, \cdots, N_b\}$ buses with $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ transmission lines, the goal is to estimate complex voltages across all buses $\mathcal{N}$ from a subset of (perhaps noisy) supervisory control and data acquisition (SCADA) measurements, which include (active/reactive) power injections and flows along with voltage magnitudes. In a AC power flow model, physical current and voltage laws dictate that the SCADA measurements are quadratically related to the state voltage variables to be estimated. Due to lack of space, we omit the exact derivation of these relationships. Nevertheless, it can be readily shown that the problem boils down to solving a system of quadratic equations (see [36] for a complete derivation). We utilize the quadratic penalty formulation (20) for applying our FOMs. Here, we have $M_I = 0, N = 2N_b$ (corresponding to the real and imaginary parts of the voltage variables), $M = M_E$ (the number of SCADA measurements), $\mathcal{X} = \mathbb{R}^N$, $\{d_m\}_{m=1}^M$ is the set of observed SCADA measurements and $\{\mathbf{C}_m\}_{m=1}^M$ are the corresponding symmetrized bus admittance related matrices which are sparse, low-rank and indefinite. In addition, when the SCADA measurements are corrupted by additive white Gaussian noise, then the maximum likelihood estimator of the voltage profile corresponds to the weighted least-squares (WLS) version of (20), where each quadratic penalty term is inversely weighed by the noise variance (assumed independent across measurements). In short, we obtain an optimization problem of the form

$$\min_{\mathbf{x}\in\mathbb{R}^N} \frac{1}{M} \sum_{m=1}^{M} \frac{(\mathbf{x}^T \mathbf{C}_m \mathbf{x} - d_m)^2}{\sigma_m^2} \tag{27}$$

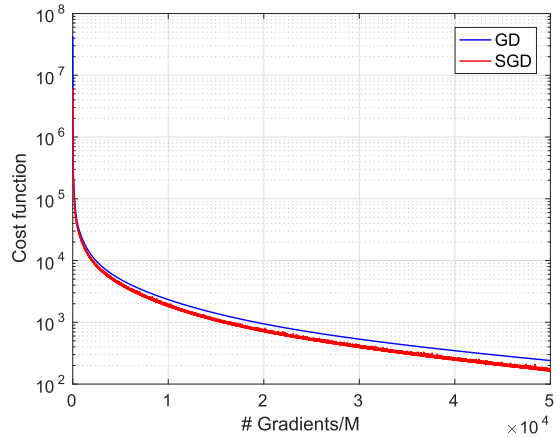
where σ_m^2 is the noise power associated with measurement $m \in \{1, \cdots, M\}$.

The standard workhorse algorithm for this problem is the Gauss-Newton (GN) method, which is well suited for application on non-linear least squares problems. When initialized close to a local minimum, convergence of GN can be established. However, determining such an initialization is non-trivial in general, and the performance of GN is known to be sensitive to the choice of initialization. Here, we implemented a modified version of the GN algorithm described in [37, p. 61], which uses a backtracking line-search procedure for improved performance in practice. Our experiments indicate that this modified GN algorithm exhibits superior performance over standard GN, and should be used instead as the de-facto performance benchmark for this problem. Regarding our FOMs, theoretical convergence of GD with backtracking line-search can be established for this problem (see Appendix D), while as pointed out earlier, establishing convergence for the stochastic gradient methods

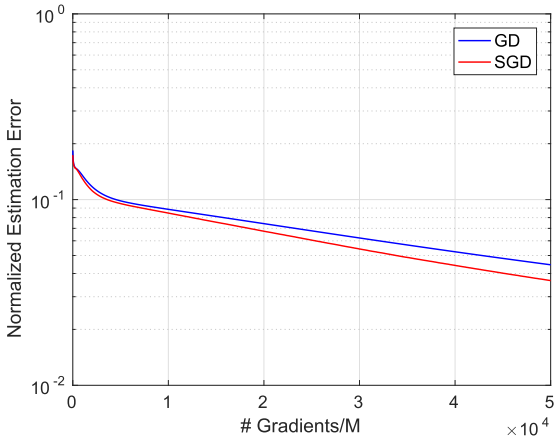
is still an open problem. We also point out that a modified version of FPP-SCA for the WLS formulation has been recently proposed in [36], which essentially uses the weighted ℓ_2-norm square of the slack variables corresponding to the equalities in each SCA subproblem of the form (4). Using the arguments made in Appendix D, a similar convergence claim (to a KKT point) can also be established for FPP-SCA in this case. Due to space constraints, we omit the proof. Thus, GD and FPP-SCA are the only algorithms under consideration here with convergence guarantees for PSSE. We also point out that due to data sparsity, computing gradients for the direct FOMs in this case requires sparse matrix-vector multiplications, which can be accomplished very efficiently. In contrast, exploiting data sparsity in general purpose conic programming solvers (utilized by FPP-SCA) is a more challenging proposition.

In our experiments, we evaluate estimation performance according to the normalized mean square error criterion, defined as $\text{NMSE} := \frac{\|\hat{\mathbf{x}}-\mathbf{x}\|_2}{\|\mathbf{x}\|_2}$, where $\hat{\mathbf{x}}$ is the estimated voltage profile and $\mathbf{x}$ is the true voltage profile. We used MATPOWER [38] for generating the voltage profiles and SCADA measurements. The voltage magnitude at each bus was generated from a uniform distribution over the interval $[0.9, 1.1]$ while the voltage angle was uniformly distributed over $[-0.1\pi, 0.1\pi]$. The phase of the reference bus was set to 0 in order to resolve the phase ambiguity in all experiments. We also added independent, zero-mean Gaussian noise with variances of 10 and 13 dBm to the measurements corresponding to the voltage magnitudes and power flow/injections respectively. Regarding the algorithms, we run (modified) GN for a maximum of 100 iterations, since we observed that GN either always converges or ceases to improve the cost function within this iteration limit. We added a small regularization term to the inexact Hessian at each step in order to ensure that it is well-conditioned and set the line-search parameter $\alpha = 0.1$. For implementing FPP-SCA, we used the modeling language YALMIP [39] along with the general convex programming solver MOSEK [40]. As for the direct FOMs, we fix a maximum gradient budget to ensure a fair comparison between GD and SGD. For GD, we use a crude choice of line-search parameters (see [29, p. 466]) while we implemented SGD with minibatch stochastic gradients of size $\lfloor \frac{M}{10} \rfloor$ and the norm-regularized step-size rule. GN, GD and SGD were always initialized from the flat-voltage profile (i.e., the all-ones vector). The test buses used in our experiments were obtained from the NESTA archive [41].

In Fig. 4, we present a preliminary simulation result demonstrating the performance of SGD and GD (with backtracking line-search) for a single realization of the voltage profile on the PEGASE-89 bus system with the full set of noisy SCADA measurements (corresponding to $N = 178, M = 687$). Here, we set $c_4 = 4 \times 10^{-5}$ for the norm-regularized step size rule of SGD. Fig. 4(a) depicts the evolution of the WLS cost function of (27), from which it can be seen that SGD attains a solution with lower cost compared to GD within the prescribed gradient budget, which was set to be $5 \times 10^4 M$. As evidenced by (Fig. 4(b)), this also translates into better estimation performance for SGD compared to GD. In terms of timing, SGD was roughly 6 times faster compared to GD on our machine. Hence, on this real world problem, SGD is also capable of exhibiting very favorable performance, even when pitted against a provably convergent FOM.



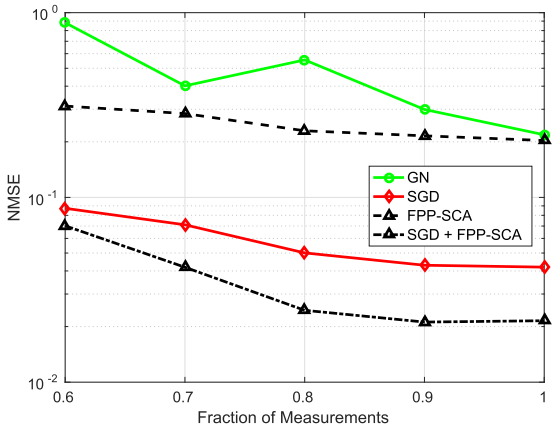
(a) Evolution of cost function



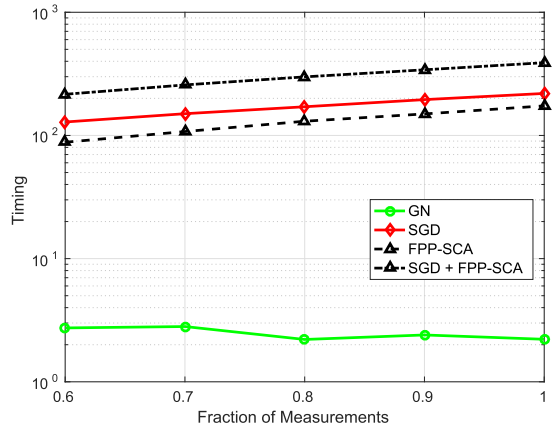
(b) Evolution of NMSE

Fig. 4. Performance comparison on PEGASE 89 bus system with full set of SCADA measurements.

Next, we devised an experiment where we evaluated the estimation performance of SGD, GN, and FPP-SCA (all initialized from the flat-voltage profile) with varying number of noisy measurements. GD is omitted here since it exhibits worse estimation performance compared to SGD at higher complexity on this network. We only run 2 iterations of FPP-SCA due to its high complexity. Additionally, we also refine the final solution returned by SGD using 2 iterations of FPP-SCA. Given a sampling fraction $\gamma \in (0, 1]$, we sample a fraction γ of the total measurements uniformly at random from each measurement type. This implies that for the active power injections, for example, out of a total of $N_b = 89$ available measurements, we subsampled $\lfloor \gamma N_b \rfloor$ measurements uniformly at random, and likewise for the other measurement types. All our results were averaged over 200 Monte-Carlo trials, and are depicted in Fig. 5. It is evident that on this network, GN, while being the fastest method, is also the one which exhibits the worst estimation performance. In contrast, SGD performs significantly better, albeit at (moderately) higher complexity. Owing to its high complexity,

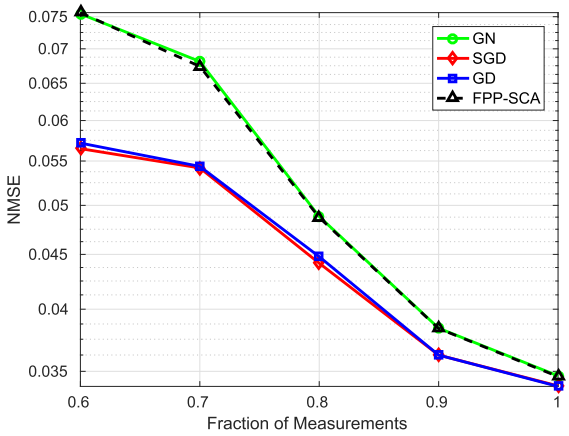


(a) Avg. NMSE vs γ

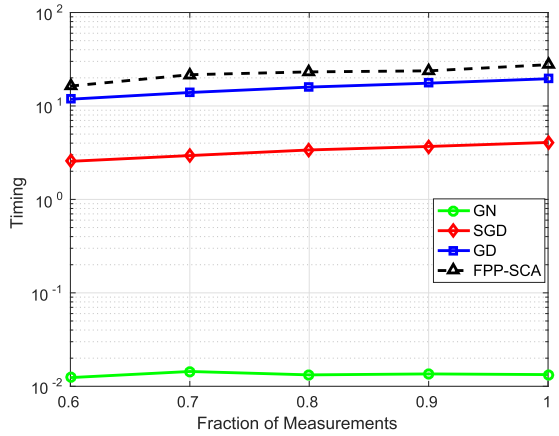


(b) Avg. Wall Time vs γ

Fig. 5. Performance results for PEGASE 89 bus system.



(a) Avg. NMSE vs γ



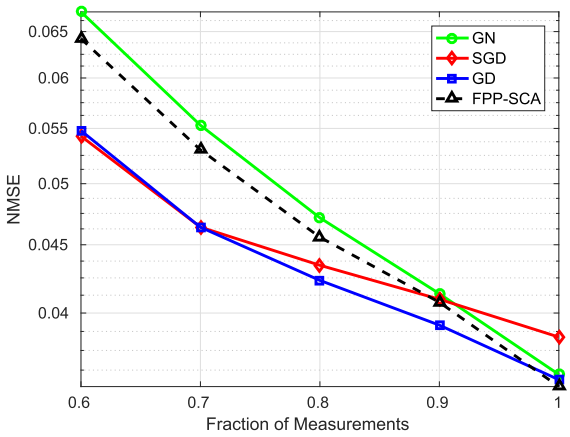
(b) Avg. Wall Time vs γ

Fig. 6. Performance results for IEEE 30 bus system.

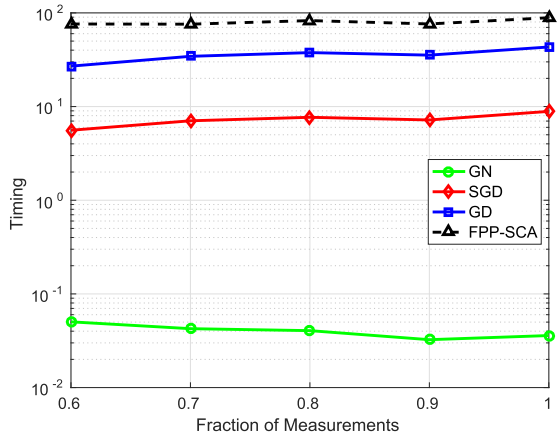
running FPP-SCA from the flat-start for 2 iterations incurs approximately the same running-time as SGD while being significantly worse-off in terms of NMSE. This highlights the ability of SGD to attain a very favorable performance-complexity trade-off compared to other alternatives. We additionally note that the solution determined by SGD can serve as a good initialization for FPP-SCA, which obviates the need to initialize FPP-SCA from the flat voltage profile and thereby incur significantly higher complexity. Clearly, combining SGD with 2 iterations of FPP achieves the lowest NMSE. However, this combined heuristic still exhibits the highest complexity (even with 2 SCA iterations), which further underscores the benefit of SGD initialization.

We also carried out similar experiments on a series of test bus systems. Fig. 6 depicts the results for the IEEE 30 bus network. In this case, we used a gradient budget of $5000M$ for the FOMs and set $c_4 = 0.02$ for SGD. As this is a fairly small network, we initialized FPP-SCA from the flat start and ran it until it attains convergence in the cost function, or a maximum of 20 iterations are executed. It is evident that both FOMs outperform GN and FPP-SCA in terms of estimation error this case, which is remarkable given the fact that they are considerably less sophisticated compared to the aforementioned methods. Furthermore, while it may be tempting to conjecture from the figures that GN attains the best performance-complexity trade-off in this case, this is not so: in our experiments, GN never improves upon its estimates beyond 100 iterations, and thus one cannot obtain better estimation performance for GN by using more iterations. Hence, it is SGD which is overall the best in this case from the perspective of performance-complexity trade-off, which makes it all the more remarkable given that it is the *least* sophisticated technique amongst all the methods under consideration.

In Fig. 7, we show the results obtained on the IEEE 57 bus network, for which we used a maximum gradient budget of $5000M$ and set $c_4 = 0.05$. In this case as well, we initialized FPP-SCA from the flat start until convergence or a maximum of 20 iterations are reached. From the figures, it can be seen that both FOMs perform very admirably in terms of estimation error compared to both GN and FPP-SCA: GN demonstrates class-leading performance only for $\gamma \geq 0.9$ while running FPP-SCA always results in higher complexity relative to the other methods. It is clear that when one has access to a partial set of measurements (i.e., $\gamma \leq 0.8$), the FOMs possess the upper hand in terms of performance.



(a) Avg. NMSE vs γ



(b) Avg. Wall Time vs γ

Fig. 7. Performance results for the IEEE 57 bus system.

VI. Conclusion

We presented a simple heuristic framework for determining feasible points of general non-convex QCQPs using memory efficient and computationally lightweight FOMs, which makes our approach very scalable. While a general theory of provable performance guarantees is elusive at present, we provided a selection of empirical step-sizes for which the FOMs surprisingly exhibit very favorable performance in terms of feasibility attained on synthetic experiments and estimation error for PSSE using real power network data compared to more established and sophisticated alternatives. Given the startling empirical performance of FOMs for this class of hard optimization problems, we can reasonably claim that our direct FOM based approach constitutes a significant advancement in the state-of-art for computing feasible solutions of large-scale non-convex QCQP. In particular, the stochastic gradient methods emerge as the algorithms of choice in our experiments.

Appendix A
Parameter Selection and Algorithm Convergence

In this section, we describe the convergence behavior of the various algorithms in their prescribed settings (i.e., non-convex or convex) as outlined in Section III.

A. Direct Approach

We begin with some results on the convergence of GD for non-convex problems of the form (6). First, consider the case $\mathcal{X} = \mathbb{R}^N$. We define a point $\bar{\mathbf{x}} \in \mathbb{R}^N$ to be $\epsilon-$approximate stationary if $\|\nabla F(\bar{\mathbf{x}})\|_2^2 \leq \epsilon$. Assuming $F(\mathbf{x})$ is L-Lipschitz smooth on $\mathcal{X}$, then, GD with a $1/L$ step-size requires at most $\mathcal{O}(\frac{L}{\epsilon})$ iterations to attain $\epsilon-$approximate stationarity [42, p. 29]. In the more general case of $\mathcal{X} \subset \mathbb{R}^N$, convergence rate guarantees can be established in terms of the *generalized projected gradient*, which is defined as

$$P_{\mathcal{X}}(\mathbf{x}, \alpha) := \frac{1}{\alpha}[\mathbf{x} - \mathbf{x}^+] \tag{28}$$

where $\mathbf{x}^+ := \arg\min_{\mathbf{u}\in\mathcal{X}} \nabla F(\mathbf{x})^T \mathbf{u} + \frac{1}{2\alpha}\|\mathbf{u} - \mathbf{x}\|_2^2$ for a given point $\mathbf{x} \in \mathbb{R}^N$ and step-size α. In [43, Lemma 3], it is shown that as $\|P_{\mathcal{X}}(\mathbf{x}, \alpha)\|_2$ diminishes, $\mathbf{x}^+$ approaches a stationary point of (6). For a constant $1/L$ step-size, it has been established [44, Corollary 1] that the number of iterations required for $\|P_{\mathcal{X}}(\mathbf{x}, \alpha)\|_2^2 \leq \epsilon$ is $\mathcal{O}(\frac{L}{\epsilon})$ in the worst case. Since $F(\mathbf{x})$ is defined to be the average of M component functions, this translates into an iteration complexity bound of $\mathcal{O}(\frac{ML}{\epsilon})$ for attaining ϵ-stationarity. While it is NP–hard in general to establish convergence to a local minimizer of a non-convex cost function [24], in the special case where $F(\mathbf{x})$ possesses the *strict saddle property*[5] and $\mathcal{X} = \mathbb{R}^N$, it has been shown [45] that GD with a constant step-size $< 1/L$ converges almost surely to a local minimizer.

Next, we consider the convergence behavior of SGD. In [46], it is shown that the $\alpha_k = \frac{1}{kL}$ step-size rule can be used to prove almost-sure (asymptotic) convergence of the SGD iterates to a stationary point. When $\mathcal{X} = \mathbb{R}^N$, an explicit iteration complexity upper bound for establishing convergence of SGD to an $\epsilon-$approximate stationary point ($\mathbb{E}[\|\nabla F(\mathbf{x})\|_2^2] \leq \epsilon$)[6] can also be derived, if, in addition to $L-$Lipschitz smoothness, the following assumption is made

$$\max_{m\in\mathcal{M}}\{\|\nabla f_m(\mathbf{x})\|_2\} \leq \sigma, \forall\, \mathbf{x} \in \mathbb{R}^N \tag{29}$$

i.e., all component functions possess uniformly bounded gradients, which is also equivalent to each $f_m(\mathbf{x})$ being σ-Lipschitz continuous. Then, for a specifically chosen constant step-size, SGD requires $\mathcal{O}(\frac{L\sigma^2}{\epsilon^2})$ iterations to obtain an $\epsilon-$approximate stationary point [47], [48, Th. 1]. This choice of step-size requires knowing the total number of iterations beforehand, which may not be practical in all cases. Note that while this bound is independent of M, it depends on the variance of the stochastic gradients. Additionally, if $F(\mathbf{x})$ possesses the strict saddle property, then under certain conditions, convergence of SGD to a local minimizer can be established in (large) polynomial-time [49]. For the case $\mathcal{X} \subset \mathbb{R}^N$, an SGD algorithm with a randomized stopping criterion is described in [44] which achieves $\epsilon-$stationarity (i.e., $\mathbb{E}[\|\nabla P(\mathbf{x}, \alpha)\|_2^2] \leq \epsilon$) in at most $\mathcal{O}(\frac{L\sigma^2}{\epsilon^2})$ iterations with a constant $1/2L$ step-size.

[5] i.e., the Hessian at every local minimizer is positive definite and at all other stationary points possesses *at least* one strictly negative eigenvalue.

[6] For stochastic iterative algorithms which make use of unbiased gradient estimators, the expectation is taken with respect to the stochasticity of the algorithm.

Finally, we focus on the convergence of SVRG for non-convex problems. In the unconstrained setting (i.e., $\mathcal{X} = \mathbb{R}^N$), assuming $F(\mathbf{x})$ is $L-$Lipschitz smooth, the references [48], [50] have established that by properly tuning the algorithm parameters, the SVRG iterates require $\mathcal{O}(\frac{M^{2/3}L}{\epsilon})$ iterations to converge to an ϵ-approximate stationary point of (6) (in expectation). The reference [51] considers the constrained case, and shows that SVRG requires $\mathcal{O}(\frac{ML}{\epsilon})$ iterations to attain ϵ-approximate stationarity (with respect to $\mathbb{E}(\|P_{\mathcal{X}}(\mathbf{x}, \alpha)\|_2^2)$). With minibatching, this rate can be improved to $\mathcal{O}(M + \frac{(M^{2/3})L}{\epsilon})$. However, the proof requires $F(\mathbf{x})$ to be *globally* L-smooth rather than being locally smooth over $\mathcal{X}$. We note that in spite of using randomly sampled gradients, the complexity bounds for SVRG are independent of the variance of the stochastic gradients due to the explicit variance reduction technique employed by SVRG.

B. SCA Approach

In this section, we discuss the convergence behavior of the algorithms selected for solving (11). The convergence rates of these algorithms for convex problems are well established. For SD, selecting a step-size sequence $\alpha_k = \mathcal{O}(\frac{1}{\sqrt{k}})$ guarantees a convergence rate of $\mathcal{O}(\frac{1}{\sqrt{k}})$ in terms of the cost function, which is minimax optimal for this class of problems [53]. Using a similar step-size as SD together with iterate averaging, SSGD is also able to attain the same $\mathcal{O}(\frac{1}{\sqrt{k}})$ convergence rate in expectation. However, the advantage of SSD lies in the fact that its individual iterations are $\mathcal{O}(M)$ faster since at every iteration we only need to compute a single subgradient.

Appendix B
Derivation of Smoothed Function

Consider the following maximization problem

$$f_m^{(\mu)}(\mathbf{x}) = \max_{0\leq y\leq 1}\{y(\mathbf{x}^T\mathbf{A}_m\mathbf{x} - b_m) - \mu\frac{y^2}{2}\}, \forall\, m \in \mathcal{M} \tag{30}$$

Note that for every $\mathbf{x} \in \mathbb{R}^N$, the corresponding maximization problem is strongly concave in y, and hence the maximum is always uniquely attained. Define the function $g(y) := y(\mathbf{x}^T\mathbf{A}_m\mathbf{x} - b_m) - \mu\frac{y^2}{2}$. In order to obtain a closed form solution to (30), we consider the following three cases.

1) $\mathbf{x}^T\mathbf{A}_m\mathbf{x} - b_m \leq 0$: In this case, it can be readily seen that the choice of y which maximizes $g(y)$ over the interval $[0, 1]$ is $y = 0$. Thus, we obtain
$$f_m^{(\mu)}(\mathbf{x}) = 0, \text{if } \mathbf{x}^T\mathbf{A}_m\mathbf{x} \leq b_m \tag{31}$$
2) $0 < \mathbf{x}^T\mathbf{A}_m\mathbf{x} - b_m \leq \mu$: The function $g(y)$ attains its maximum at $y = \frac{\mathbf{x}^T\mathbf{A}_m\mathbf{x} - b_m}{\mu}$, which in this case, lies in the interval $(0, 1]$. Substituting this value in (30) yields
$$f_m^{(\mu)}(\mathbf{x}) = \frac{(\mathbf{x}^T\mathbf{A}_m\mathbf{x} - b_m)^2}{2\mu}, \text{if } b_m < \mathbf{x}^T\mathbf{A}_m\mathbf{x} \leq b_m + \mu \tag{32}$$
3) $\mathbf{x}^T\mathbf{A}_m\mathbf{x} - b_m > \mu$: In this case, the function $g(y)$ attains its maximum at a point $y > 1$ which lies outside the interval $[0, 1]$. As the function is monotonically increasing, we choose the value $y = 1$ which maximizes $g(y)$ over $[0, 1]$ to obtain
$$f_m^{(\mu)}(\mathbf{x}) = \mathbf{x}^T\mathbf{A}_m\mathbf{x} - b_m - \frac{\mu}{2}, \text{if } \mathbf{x}^T\mathbf{A}_m\mathbf{x} > b_m + \mu \tag{33}$$

Appendix C
Derivation of Approximation Bounds

The approximation bounds (18) can be derived using the same arguments first used in [35] for establishing approximation bounds for non-smooth, *convex* functions using Nesterov smoothing. Here, we show that these results can also be extended to our non-convex setting.

In order to derive the desired bounds (18), we will equivalently show that the following result holds.

$$f_m(\mathbf{x}) - \frac{\mu}{2} \leq f_m^{(\mu)}(\mathbf{x}) \leq f_m(\mathbf{x}), \forall\, \mathbf{x} \in \mathbb{R}^N, \forall\, m \in \mathcal{M} \tag{34}$$

It is evident that the upper bound holds by inspection of the definitions of $f_m(\mathbf{x})$ (14) and $f_m^{(\mu)}(\mathbf{x})$ (15). To show that the lower bound holds, we first note that $\max_{0\leq y\leq 1}\frac{y^2}{2} = \frac{1}{2}$. Hence, it follows that for any $\mathbf{x} \in \mathbb{R}^N, \mu \in \mathbb{R}_{++}$ and $m \in \mathcal{M}$, we have

$$\begin{aligned} y(\mathbf{x}^T\mathbf{A}_m\mathbf{x} - b_m) - \mu\frac{y^2}{2} &\geq y(\mathbf{x}^T\mathbf{A}_m\mathbf{x} - b_m) - \frac{\mu}{2}, \\ &\qquad \forall\, 0 \leq y \leq 1 \\ \Rightarrow \max_{0\leq y\leq 1}\{y(\mathbf{x}^T\mathbf{A}_m\mathbf{x} - b_m) - \mu\frac{y^2}{2}\} &\\ \geq \max_{0\leq y\leq 1}\{y(\mathbf{x}^T\mathbf{A}_m\mathbf{x} - b_m)\} - \frac{\mu}{2}, &\\ \Rightarrow f_m^{(\mu)}(\mathbf{x}) \geq f_m(\mathbf{x}) - \frac{\mu}{2} & \end{aligned} \tag{35}$$

This concludes the proof.

Appendix D
Convergence of Gradient Descent

In this section, we establish that GD with backtracking line search globally converges to the set of stationary points of the quadratic penalty formulation for solving general systems of quadratic equations (i.e., (20) with $M_I = 0$), without any Lipschitz smoothness assumptions. We emphasize that this is not a new result, but rather a refinement of the following pre-existing result.

Lemma 1 [54, Proposition 1.2.1]: Let $\{\mathbf{x}^{(k)}\}$ be a sequence of iterates generated by GD with backtracking line search for step-size selection. Then, every limit point of $\{\mathbf{x}^{(k)}\}$ is a stationary point.

This result, while not requiring any Lipschitz smoothness assumptions, is still contingent on a general condition for gradient based methods being satisfied, which requires that the descent direction does not become asymptotically orthogonal to the gradient, unless the gradient vanishes (see [54, p. 35]). We point out that GD naturally satisfies this condition, thus implying that the convergence claim stated above holds. However, this result, on its own, is rather weak. It states that *if* the sequence $\{\mathbf{x}^{(k)}\}$ possesses a convergent subsequence, then the limit of the *subsequence* is a stationary point. In order to further improve upon this claim, we proceed as follows:

1) First, we show that irrespective of initialization, the sequence of iterates $\{\mathbf{x}^{(k)}\}$ generated by GD with backtracking line search always possesses a convergent subsequence.
2) Using the above result in conjunction with Lemma 1, we will show that the *entire sequence* $\{\mathbf{x}^{(k)}\}$ converges to a stationary point.

The first condition can be established by exploiting the fact the cost function is a quartic polynomial which is *coercive*.[7] A useful attribute of coercive functions is that they satisfy the following property.

Lemma 2 [55, Lemma 8.3]: Let f be a continuous, coercive function. Then, every sub-level set $\mathcal{X}_\gamma := \{\mathbf{x} \in \mathbb{R}^N \mid f(\mathbf{x}) \leq \gamma\}$ of f is compact.

This implies that for any point $\mathbf{x}^{(0)} \in \mathbb{R}^N$ used to initialize GD, the initial sub-level set $\mathcal{X}_{f(\mathbf{x}^{(0)})} := \{\mathbf{x} \in \mathbb{R} \mid f(\mathbf{x}) \leq f(\mathbf{x}^{(0)})\}$ is always compact. Since backtracking line search is used to ensure descent of GD at each iteration, it follows that the entire sequence of iterates $\{\mathbf{x}^{(k)}\}$ generated by GD lie in $\mathcal{X}_{f(\mathbf{x}^{(0)})}$, and hence, are bounded. By appealing to the Weierstrass theorem [54, Proposition A.8], it then follows that $\{\mathbf{x}^{(k)}\}$ possesses a convergent subsequence.

In order to show the final part of our claim, we will utilize the following result, which is extracted from [8, Corollary 1]. For completeness, we include the proof of the statement.

Lemma 3: Assume that $\mathcal{X}_{f(\mathbf{x}^{(0)})}$ is compact. Then, the sequence of iterates $\{\mathbf{x}^{(k)}\}$ generated by GD with backtracking line search satisfy

$$\lim_{k\to\infty} d(\mathbf{x}^{(k)}, \mathcal{X}^*) = 0$$

where $\mathcal{X}^*$ is the set of stationary points.

Proof: We have already shown that $\mathcal{X}_{f(\mathbf{x}^{(0)})}$ is compact. The remainder of the claim is proven by contradiction. We assume that there exists a subsequence $\{\mathbf{x}^{(k_j)}\}$ such that $\lim_{k\to\infty} d(\mathbf{x}^{(k_j)}, \mathcal{X}^*) \geq \delta$ for some $\delta \in \mathbb{R}_{++}$. Since this subsequence lies in the compact set $\mathcal{X}_{f(\mathbf{x}^{(0)})}$, it has a limit point $\bar{\mathbf{x}}$. By further restricting the indices of this subsequence, we obtain

$$d(\bar{\mathbf{x}}, \mathcal{X}^*) = \lim_{k\to\infty} d(\mathbf{x}^{(k_j)}, \mathcal{X}^*) \geq \delta$$

However, this is a contradiction of the fact, that, due to Lemma 1, we have $\bar{\mathbf{x}} \in \mathcal{X}^*$. ■

Hence, it follows that the entire sequence $\{\mathbf{x}^{(k)}\}$ globally converges to the set of stationary points, in an asymptotic sense.

ACKNOWLEDGMENT

The authors would like to thank Ahmed S. Zamzam for his guidance in designing the PSSE experiments.

REFERENCES

[1] A. Konar and N. D. Sidiropoulos, "Fast feasibility pursuit for non-convex QCQPs via first-order methods," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, New Orleans, LA, USA, Mar. 2017, pp. 4064–4068.
[2] Z.-Q. Luo and T.-H. Chang, "SDP relaxation of homogeneous quadratic optimization: approximation bounds and applications," in *Convex Optimization in Signal Processing and Communications*, D. Palomar and Y. Eldar, Eds. Cambridge, U.K.: Cambridge Univ. Press, 2010, pp. 117–165.
[3] A. d'Aspremont and S. Boyd, "Relaxations and randomized methods for nonconvex QCQPs," EE392 Lecture Notes, Stanford Univ., Stanford, CA, USA, 2003.
[4] H. Wolkowicz, "Relaxations of Q2P," in *Handbook of Semidefinite Programming: Theory, Algorithms, and Applications*, H. Wolkowicz, R. Saigal, and L. Vandenberghe, Eds. Norwell, MA, USA: Kluwer, 2000, ch. 13.4.
[5] Z.-Q. Luo, W.-K. Ma, A.-C. So, Y. Ye, and S. Zhang, "Semidefinite relaxation of quadratic optimization problems," *IEEE Signal Process. Mag.*, vol. 27, no. 3, pp. 20–34, May 2010.
[6] B. Marks and G. Wright, "A general inner approximation algorithm for nonconvex mathematical programs," *Oper. Res.*, vol. 26, no. 4, pp. 681–683, 1978.
[7] A. Beck, A. Ben-Tal, and L. Tetruashvili, "A sequential parametric convex approximation method with applications to non-convex truss topology design problems," *J. Global Optim.*, vol. 47, no. 1, pp. 29–51, 2010.
[8] M. Razaviyayn, M. Hong, and Z.-Q. Luo, "A unified convergence analysis of block successive minimization methods for nonsmooth optimization," *SIAM J. Optim.*, vol. 23, no. 2, pp. 1126–1153, 2013.
[9] G. Scutari, F. Facchinei, L. Lampariello, and P. Song, "Parallel and distributed methods for nonconvex optimization—Part I: Theory," *IEEE Trans. Signal Process.*, vol. 65, no. 8, pp. 1929–1944, Apr. 2017.
[10] O. Mehanna, K. Huang, B. Gopalakrishnan, A. Konar, and N. D. Sidiropoulos, "Feasible point pursuit and successive approximation of non-convex QCQPs," *IEEE Signal Process. Lett.*, vol. 22, no. 7, pp. 804–808, Jul. 2015.
[11] K. Huang and N. D. Sidiropoulos, "Consensus-ADMM for general quadratically constrained quadratic programming," *IEEE Trans. Signal Process.*, vol. 64, no. 20, pp. 5297–5310, Oct. 2016.
[12] M. Andersen, J. Dahl, Z. Liu, and L. Vandenberghe, "Interior-point methods for large-scale cone programming," in *Optimization for Machine Learning*, S. Sra, S. Nowozin, and S. J. Wright, Eds. Cambridge, MA, USA: MIT Press, 2012, pp. 55–84.
[13] S. Tu, R. Boczar, M. Soltanolkotabi, and B. Recht, "Low-rank solutions of linear matrix equations via procrustes flow," arXiv:1507.03566, 2015.
[14] C. De Sa, K. Olukotun, and C. Re, "Global convergence of stochastic gradient descent for some non-convex matrix problems," arXiv:1411.1134v3, 2015.
[15] R. Sun and Z.-Q. Luo, "Guaranteed matrix completion via nonconvex factorization," in *Proc. IEEE 56th Annu. Symp. Found. Comput. Sci.*, Berkeley, CA, USA, Oct. 17–20, 2015, pp. 270–289.
[16] Y. Chen and M. J. Wainwright, "Fast low-rank estimation by projected gradient descent: General statistical and algorithmic guarantees," arXiv:1509.03025, 2015.
[17] S. Bhojanapalli, A. Kyrillidis, and S. Sanghavi, "Dropping convexity for faster semi-definite optimization," arXiv:1509.03917, 2015.
[18] S. Bhojanapalli, B. Neyshabur, and N. Srebro, "Global optimality of local search for low rank matrix recovery," arXiv:1605.07221v2, 2016.
[19] E. J. Candes, X. Li, and M. Soltanolkotabi, "Phase retrieval via Wirtinger flow: Theory and algorithms," *IEEE Trans. Inf. Theory*, vol. 61, no. 4, pp. 1985–2007, Apr. 2015.
[20] Y. Chen and E. J. Candes, "Solving random quadratic systems of equations is nearly as easy as solving linear systems," *Adv. Neural Info. Process Syst.*, pp. 739–747, 2015.
[21] J. Sun, Q. Qu, and J. Wright, "A geometric analysis of phase retrieval," arXiv:1602.06664, 2016.
[22] A. Konar and N. D. Sidiropoulos, "Fast approximation algorithms for a class of nonconvex QCQP problems using first-order methods," *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3494–3509, Jul. 2017.
[23] C. I. Kanatsoulis and N. D. Sidiropoulos, "Max-min feasible point pursuit for non-convex QCQP," in *Proc. 49th Asilomar Conf. Signals, Syst., Comput.*, Pacific Grove, CA, USA, Nov. 2015, pp. 401–405.
[24] K. G. Murty and S. N. Kabadi, "Some NP-complete problems in quadratic and nonlinear programming," *Math. Program.*, vol. 39, no. 2, pp. 117–129, 1987.
[25] Y. Ye, "Data randomness makes optimization problems easier to solve?" Tech. Rep. Dec. 2016. [Online]. Available: http://web.stanford.edu/~yyye/NLPwithrandomdata.pdf
[26] F. C. Schweppe, J. Wildes, and D. Rom, "Power system static state estimation: Parts I, II, and III," *IEEE Trans. Power App. Syst.*, vol. PAS-89, pp. 120–135, Jan. 1970.
[27] G. B. Giannakis, V. Kekatos, N. Gatsis, S.-J. Kim, H. Zhu, and B. Wollenberg, "Monitoring and optimization for power grids: A signal processing perspective," *IEEE Signal Process. Mag.*, vol. 30, no. 5, pp. 107–128, Sep. 2013.
[28] K. Lehmann, A. Grastien, and P. Van Hentenryck, "AC-feasibility on tree networks is NP-hard," *IEEE Trans. Power Syst.*, vol. 31, no. 1, pp. 798–801, Jan. 2016.

[7] A function $f : \mathbb{R}^N \to \mathbb{R}$ is said to be coercive if for every sequence $\{\mathbf{x}^{(k)}\}$ for which $\|\mathbf{x}^{(k)}\| \to \infty$, we have $\lim_{k\to\infty} f(\mathbf{x}^{(k)}) = \infty$ [54, Definition A.4].

[29] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[30] M. Razaviyayn, "Successive convex approximation: Analysis and applications," Ph.D. dissertation, Dept. Elect. Comput. Eng., Univ. Minnesota, Minneapolis, MN, USA, 2014.
[31] Y. Nesterov and A. Nemirovski, *Interior Point Polynomial Methods in Convex Programming: Theory and Applications*. Philadelphia, PA, USA: SIAM, 1994.
[32] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York, NY, USA: Springer-Verlag, 2006.
[33] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Proc. 26th Int. Conf. Neural Inf. Process Syst.*, 2013, pp. 315–323.
[34] L. Xiao and T. Zhang, "A proximal stochastic gradient method with progressive variance reduction," *SIAM J. Optim.*, vol. 24, no. 4, pp. 2057–2075, 2014.
[35] Y. Nesterov, "Smooth minimization of non-smooth functions," *Math. Program.*, vol. 103, no. 1, pp. 127–152, May 2005.
[36] G. Wang, A. S. Zamzam, G. B. Giannakis, and N. D. Sidiropoulos, "Power system state estimation via feasible point pursuit," in *Proc. IEEE Global Conf. Signal Inf. Process.*, Washington, DC, USA, Dec. 2016.
[37] L. Vandenberghe, "EE 133A Lecture Notes," *EE133A Lecture Notes, UCLA*, 2016. [Online]. Available: http://www.seas.ucla.edu/vandenbe/133A/133A-notes.pdf
[38] R. D. Zimmerman, C. E. Murillo-Sanchez, 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, Feb. 2011.
[39] J. Lofberg, "YALMIP: A toolbox for modeling and optimization in MATLAB," in *Proc. IEEE Int. Symp. Comput. Aided Control Syst. Des.*, Taipei, Taiwan, Sep. 4, 2004, pp. 284–289.
[40] APS Mosek, "The MOSEK optimization software," 2010. [Online]. Available: http://www.mosek.com
[41] C. Coffrin, D. Gordon, and P. Scott, "Nesta, the NICTA energy system test case archive," arXiv:1411.0359, 2014.
[42] Y. Nesterov, *Introductory Lectures on Convex Optimization*, vol. 87. Berlin, Germany: Springer, 2004.
[43] S. Ghadimi and G. Lan, "Accelerated gradient methods for nonconvex nonlinear and stochastic programming," *Math. Program.*, vol. 156, nos. 1/2, pp. 59–99, 2016.
[44] S. Ghadimi, G. Lan, and H. Zhang, "Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization," *Math. Program.*, vol. 155, nos. 1/2, pp. 267–305, 2016.
[45] J. D. Lee, M. Simchowitz, M. I. Jordan, and B. Recht, "Gradient descent only converges to minimizers," *Proc. 29th Annu. Conf. Learn. Theory*, 2016, pp. 1246–1257.
[46] M. Razaviyayn, M. Sanjabi, and Z.-Q. Luo, "A stochastic successive minimization method for nonsmooth nonconvex optimization with applications to transceiver design in wireless communication networks," *Math. Program.*, vol. 157, no. 2, pp. 515–545, Jun. 2016.
[47] S. Ghadimi, and G. Lan, "Stochastic first and zeroth-order methods for nonconvex stochastic programming," *SIAM J. Optim.*, vol. 23, no. 4, pp. 2341–2368, 2013.
[48] S. J. Reddi, A. Hefny, S. Sra, B. Poczos, and A. Smola, "Stochastic variance reduction for nonconvex optimization," arXiv:1603.06160v2, 2016.
[49] R. Ge, F. Huang, C. Jin, and Y. Yuan, "Escaping from saddle points–online stochastic gradient for tensor decomposition," *J. Mach. Learn. Res.*, vol. 40, pp. 1–46, Jun. 2015.
[50] Z.-A. Zhu and E. Hazan, "Variance reduction for faster non-convex optimization," arXiv:1603.05643, 2016.
[51] S. J. Reddi, S. Sra, B. Poczos, and A. Smola, "Fast stochastic methods for nonsmooth nonconvex optimization," arXiv:1605.06900v1, 2016.
[52] N. Srebro and A. Tewari, "Stochastic optimization for machine learning," (Tutorial), in *Proc. Int. Conf. Mach. Learn.*, Jun. 2010.
[53] A. Nemirovski, D. Yudin, *Problem Complexity and Method Efficiency in Optimization*. New York, NY, USA: Wiley, 1983.
[54] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont, MA, USA: Athena Scientific, 1999.
[55] G. C. Calafiore and L. El Ghaoui, *Optimization Models*. Cambridge, U.K.: Cambridge Univ. Press, 2014.

Aritra Konar (S'14) received the B.Tech. degree in electronics and communications engineering from West Bengal University of Technology, Kolkata, India, in 2011, and the M.S. degree in electrical engineering from University of Minnesota, Minneapolis, MN, USA, in 2014. Since then, he has been working toward the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Minnesota. His research interests include statistical signal processing, wireless communications, nonlinear optimization, and data analytics.

Nicholas D. Sidiropoulos (F'09) received the Diploma degree in electrical engineering from the Aristotelian University of Thessaloniki, Thessaloniki, Greece, and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland, College Park, MD, USA, in 1988, 1990, and 1992, respectively. He was an Assistant Professor with the University of Virginia, an Associate Professor with the University of Minnesota, Minneapolis, MN, USA, and a Professor with the Technical University of Crete, Greece. Since 2011, he has been with the University of Minnesota, where he currently holds an ADC Chair in digital technology. His research spans topics in signal processing theory and algorithms, optimization, communications, and factor analysis—with a long-term interest in tensor decomposition and its applications. His current focus is primarily on signal and tensor analytics for learning from big data. He received the NSF/CAREER Award in 1998 and the IEEE Signal Processing (SP) Society Best Paper Award in 2001, 2007, and 2011. He was an IEEE SP Society Distinguished Lecturer (2008—2009) and a Chair of the IEEE Signal Processing for Communications and Networking Technical Committee (2007—2008). He received the 2010 IEEE SP Society Meritorious Service Award and the 2013 Distinguished Alumni Award from the Department of Electrical and Computer Engineering, University of Maryland. He has been a Fellow of EURASIP since 2014.