# FAST FEASIBILITY PURSUIT FOR NON-CONVEX QCQPS VIA FIRST-ORDER METHODS

*Aritra Konar and Nicholas D. Sidiropoulos*

Dept. of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN

## ABSTRACT

Quadratically Constrained Quadratic Programming (QCQP) is NP–hard in its general non-convex form, but it frequently arises in engineering design and applications ranging from state estimation to beamforming and clustering. 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 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 on large-scale problems. Experiments indicate the empirical effectiveness of our approach, despite currently lacking theoretical guarantees.

## 1. INTRODUCTION

Non-convex QCQPs form an important class of optimization problems which find widespread application in various engineering disciplines [1]. As a result of the problem being NP–hard in its general form [2], approximation algorithms are often employed to obtain high quality, sub-optimal solutions in polynomial-time.

Amongst the most popular of these approximation strategies is *Semidefinite Relaxation* (SDR) [3], which uses matrix lifting coupled with rank relaxation to express the problem as a convex Semidefinite Program (SDP). When the SDP solution is not rank-1 (typical, except for specially structured QCQPs), then a post-processing step is used to obtain a feasible solution for the original non-convex QCQP. However, when the constraints involve indefinite matrices and/or double sided inequalities, then such post-processing algorithms typically fail to yield a feasible point, thereby limiting the overall effectiveness of SDR – not to mention the potentially very high complexity of solving the relaxed problem in SDP form.

Another approach is *Successive Convex Approximation* (SCA) [4–7], which approximates the problem by a sequence of convex subproblems initialized from a feasible point. Under certain conditions, convergence of the SCA iterates to a stationary point can be established. Although this framework is more generally applicable to non-convex QCQPs compared to SDR, we point out that computing an initial feasible point for general non-convex QCQP is NP–Hard [2].

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* [8] was proposed specifically for this task. FPP-SCA uses SCA in conjunction with auxiliary slack variables to approximate the feasibility problem by a sequence of convex subproblems. The algorithm works even with random initialization, as the slack variables ensure that each SCA subproblem is feasible at every step. Empirically, FPP-SCA demonstrates highly effective performance in attaining feasibility for general non-convex QCQPs. Nonetheless, the algorithm is not without its drawbacks. For one, it is required to iteratively solve a sequence of convex optimization problems. Using general purpose conic-programming solvers, this can be very computationally demanding. In addition, an eigen-decomposition of all the matrices in the quadratic constraints has to be performed, followed by storing the positive and negative definite components in memory.

Thus, due to its inherently large computational and memory footprint, FPP-SCA is not well suited for solving problems in large dimensions and/or with a large number of constraints, which motivates the development of low-complexity algorithms for feasible point pursuit. Towards this end, we propose a reformulation of the penalized feasibility formulation 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 minimal memory and computational requirements relative to other optimization schemes, which makes them well-suited for solving large-scale problems. Hence, in this paper we will adopt a first-order based optimization approach for the feasibility problem, instead of resorting to SCA. Our interest in pursuing this approach is partially motivated from recent work which established that FOMs work remarkably well for many important non-convex problems arising in matrix regression and structured matrix factorization [9–13], as well as generalized phase retrieval [14–16].

We note that a consensus-ADMM (C-ADMM) algorithm for general non-convex QCQPs has been proposed in [17], which can also be used for directly computing a feasible point. The major drawback of this method relative to our proposed approach is that C-ADMM is very memory intensive, since one local copy of the global optimization variable is created for every constraint. Furthermore, we do not need to compute an eigen-decomposition of the constraint matrices, which is a requirement common to both FPP-SCA and C-ADMM. We use experimental evaluations to demonstrate the viability of using FOMs as a competitive alternative to the SCA approach for determining feasible solutions of general non-convex QCQPs via feasibility pursuit. At present, we do not possess any theoretical convergence results for our methods. In spite of this, simulations indicate the very favorable performance of our approach.

## 2. PROBLEM STATEMENT

In this paper, we consider quadratic feasibility problems of the form

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

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

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

Contact: (konar006,nikos)@umn.edu. Supported in part by NSF CIF-1525194, and the Univ. of Minnesota through a Doctoral Dissertation Fellowship.

where $\mathcal{X} \subset \mathbb{R}^N$ is a *simple*[1], compact, 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 constraint matrices $\{\mathbf{A}_m\}_{m=1}^{M_I}$ and $\{\mathbf{C}_m\}_{m=1}^{M_E}$ are assumed to be symmetric, while $\{b_m\}_{m=1}^{M_I}$ and $\{d_m\}_{m=1}^{M_E}$ are real numbers. In the special case where $M_E = 0$ (i.e., the equality constraints are absent), and $\mathbf{A}_m \succeq \mathbf{0}$, $\forall\, m \in \mathcal{M}_\mathcal{I}$, (1) reduces to a convex feasibility problem, for which (in)feasibility can be established in polynomial-time [18]. 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 matrices, and is known to be NP–Hard.

One approach for establishing the (in)feasibility of a given instance of (1) is to consider the following optimization problem.

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

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

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

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

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

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 non-negative 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. The $\ell_1$-norm cost function is used 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 the sparsity pattern of $\mathbf{s}_\mathcal{I}^*$ and $\mathbf{s}_\mathcal{E}^*$ reveals the constraints which cause infeasibility. Nonetheless, computing an optimal solution of (2) remains a challenging proposition since it is non-convex and is NP–Hard in general.

In [8], the technique of SCA was used to approximate (2). In this paper, we propose to eliminate the SCA procedure altogether and instead focus on tackling (2) *directly* via FOMs. As a first step in this direction, we equivalently reformulate (2) as

$$\min_{\mathbf{x}\in\mathcal{X}} \left\{ F(\mathbf{x}) := \sum_{m=1}^{M_I} (\mathbf{x}^T\mathbf{A}_m\mathbf{x} - b_m)^+ + \sum_{m=1}^{M_E} |\mathbf{x}^T\mathbf{C}_m\mathbf{x} - d_m| \right\} \tag{3}$$

where $(x)^+ := \max\{x, 0\}$ and $|x|$ denotes the absolute value of $x$. Note that (2) can be obtained via the epigraph transformation of the non-convex cost function of (3), thus establishing equivalence. 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 $M := M_I + M_E$ non-convex, non-smooth functions; each of which measures the degree of violation of its corresponding constraint via a loss function (quadratic hinge-loss for the inequality constraints and absolute value for the equality constraints). The non-differentiability of (3) prevents us from applying FOMs (which are suited for minimizing smooth functions) directly on this formulation. Consequently, we propose to make the following modifications to (3).

First, we 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

$$f_m(\mathbf{x}) = \max_{0\leq y\leq 1} \{y(\mathbf{x}^T\mathbf{A}_m\mathbf{x} - b_m)\}, \forall\, m \in \mathcal{M}_\mathcal{I} \tag{4}$$

In order to construct a smooth surrogate of $f_m(\mathbf{x})$, consider the following modified version of (4)

$$f_m^{(\mu)}(\mathbf{x}) = \max_{0\leq y\leq 1} \{y(\mathbf{x}^T\mathbf{A}_m\mathbf{x} - b_m) - \mu\frac{y^2}{2}\}, \forall\, m \in \mathcal{M}_\mathcal{I} \tag{5}$$

where $\mu \in \mathbb{R} > 0$ is a smoothing parameter. The maximization problem (5) can be solved in closed form (due to strong concavity) to obtain the following equivalent smooth representation

$$f_m^{(\mu)}(\mathbf{x}) = \begin{cases} 0, & \text{if } \mathbf{x}^T\mathbf{A}_m\mathbf{x} \leq b_m \\ \frac{(\mathbf{x}^T\mathbf{A}_m\mathbf{x} - b_m)^2}{2\mu}, & \text{if } b_m < \mathbf{x}^T\mathbf{A}_m\mathbf{x} \leq b_m + \mu \\ \mathbf{x}^T\mathbf{A}_m\mathbf{x} - b_m - \frac{\mu}{2}, & \text{if } \mathbf{x}^T\mathbf{A}_m\mathbf{x} > b_m + \mu \end{cases} \tag{6}$$

We point out that each $f_m^{(\mu)}(\mathbf{x})$ has continuous derivatives given by

$$\nabla f_m^{(\mu)}(\mathbf{x}) = \begin{cases} 0, & \text{if } \mathbf{x}^T\mathbf{A}_m\mathbf{x} \leq b_m \\ \frac{2(\mathbf{x}^T\mathbf{A}_m\mathbf{x} - b_m)}{\mu}\mathbf{A}_m\mathbf{x}, & \text{if } b_m < \mathbf{x}^T\mathbf{A}_m\mathbf{x} \leq b_m + \mu \\ 2\mathbf{A}_m\mathbf{x}, & \text{if } \mathbf{x}^T\mathbf{A}_m\mathbf{x} > b_m + \mu \end{cases} \tag{7}$$

Hence, $f_m^{(\mu)}(\mathbf{x})$ is a smooth surrogate of $f_m(\mathbf{x}), \forall\, m \in \mathcal{M}_\mathcal{I}$. It can also be shown that the following approximation bounds hold [2]

$$f_m^{(\mu)}(\mathbf{x}) \leq f_m(\mathbf{x}) \leq f_m^{(\mu)}(\mathbf{x}) + \frac{\mu}{2}, \forall\, \mathbf{x}, \forall\, m \in \mathcal{M}_\mathcal{I} \tag{8}$$

The smoothing technique we have employed is reminiscent of *Nesterov smoothing* [19], although here it is applied on non-convex functions.

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 (3) corresponding to the equality constraints, we propose to replace them with quadratic penalty functions of the form

$$g_m^{(q)}(\mathbf{x}) := (\mathbf{x}^T\mathbf{C}_m\mathbf{x} - d_m)^2, \forall\, m \in \mathcal{M}_\mathcal{E} \tag{9}$$

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

$$\min_{\mathbf{x}\in\mathcal{X}} \left\{ F^{(s)}(\mathbf{x}) := \frac{1}{M}\left( \sum_{m=1}^{M_I} f_m^{(\mu)}(\mathbf{x}) + \sum_{m=1}^{M_E} g_m^{(q)}(\mathbf{x}) \right) \right\} \tag{10}$$

In the following section, we present a brief overview of the FOMs we will apply on (10).

## 3. OVERVIEW OF FIRST ORDER METHODS

Consider the following optimization problem

$$\min_{\mathbf{x}\in\mathcal{X}} \left\{ F(\mathbf{x}) := \frac{1}{M}\sum_{m=1}^{M} f_m(\mathbf{x}) \right\} \tag{11}$$

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. When $F$ is bounded below over $\mathcal{X}$, we can attempt to determine an approximate solution for (11) using the classical *gradient descent* (GD) algorithm which

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

[2] Due to space constraints, we omit this proof. Nonetheless, it is straightforward to show this result via a replication of the arguments in [19].

has the following update rule.

$$\mathbf{y}^{(k)} = \mathbf{x}^{(k-1)} - \frac{\alpha_k}{M} \sum_{m=1}^{M} \nabla f_m(\mathbf{x}^{(k-1)}) \tag{12a}$$

$$\mathbf{x}^{(k)} = \Pi_{\mathcal{X}}(\mathbf{y}^{(k)}), \forall\, k \in \mathbb{N} \tag{12b}$$

where $\Pi_{\mathcal{X}}(.)$ denotes the Euclidean projection operator onto $\mathcal{X}$ and $\alpha_k \in \mathbb{R} > 0$ is the step-size in the $k^{th}$ iteration.

Each step of GD requires the computation of $M$ gradients, and hence can be fairly expensive when $M$ is large. As a low complexity alternative, we can consider using *stochastic gradient descent* (SGD). At each iteration $k$ of SGD, 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

$$\mathbf{y}^{(k)} = \mathbf{x}^{(k-1)} - \alpha_k \nabla f_{m_k}(\mathbf{x}^{(k-1)}) \tag{13a}$$

$$\mathbf{x}^{(k)} = \Pi_{\mathcal{X}}(\mathbf{y}^{(k)}), \forall\, k \in \mathbb{N} \tag{13b}$$

Note that the expectation $\mathbb{E}(\mathbf{y}^{(k)} | \, \mathbf{x}^{(k-1)})$ equals (12a) (where the expectation is taken with respect to the random variable $m_k$). Hence, the SGD updates (13) 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) [20, 21]. 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

$$\mathbf{x}_s^{(0)} = \mathbf{y}_s \tag{14a}$$

$$\mathbf{v}_s^{(k)} = \mathbf{x}_s^{(k-1)} - \alpha_s^{(k)}(\nabla f_{m_k}(\mathbf{x}_s^{(k-1)}) - \nabla f_{m_k}(\mathbf{y}_s) + \nabla F(\mathbf{y}_s)) \tag{14b}$$

$$\mathbf{x}_s^{(k)} = \Pi_{\mathcal{X}}(\mathbf{v}_s^{(k)}), \forall\, k \in \{1, \cdots, K\} \tag{14c}$$

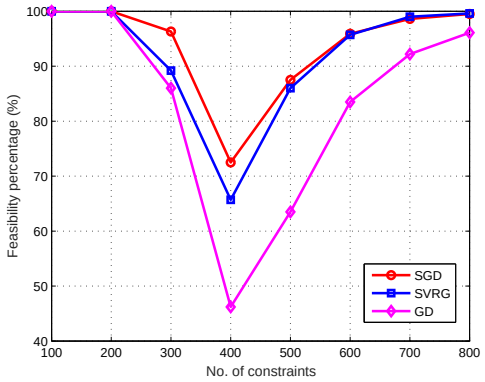
where the superscript $k$ denotes the inner SGD iteration counter for stage $s$. Again, the expectation $\mathbb{E}(\mathbf{v}_s^{(k)} | \, \mathbf{x}_s^{(k-1)})$ equals (12a). Hence, in expectation, the SVRG updates are also the same as GD updates. The overall algorithm is given by

**Algorithm 1** : SVRG

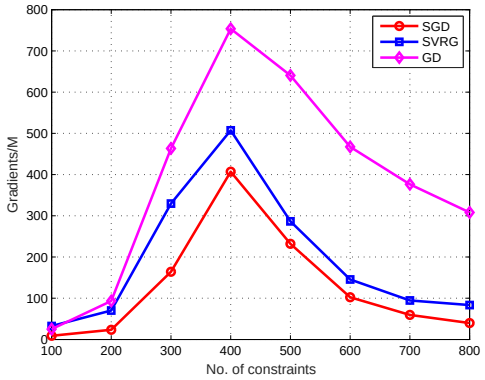
**Initialization:** Select number of stages $S$, update frequency $K$ and step-size sequence. Randomly generate a starting point $\mathbf{z}_0 \in \mathcal{X}$.
**Iterate:** for $s = 1, 2, \cdots, S$

- 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:** $\mathbf{z}_S$




(a) Feasibility percentage vs $M$




(b) Gradient evaluations (normalized) vs $M$

**Fig. 1**: Illustrative example for $N = 100$

The convergence behavior of these algorithms is determined by the choice of the step-size sequence. We point out that several results [22–28] have appeared recently which, under certain assumptions, establish the convergence (both asymptotic and non-asymptotic) of GD, SGD and SVRG to a stationary point of non-convex problems of the form (11) for specific step-sizes. Due to space constraints, we are unable to discuss at length the particulars of each such result. Nonetheless, it suffices to say that while the assumptions used in some of these results do not hold in our case, in others, the dictated choice of step-size is too small to be of practical use[3]. Hence, we used empirically-tuned step-sizes in our experiments. Although we cannot make any theoretical claims about the convergence of our methods with these step-size rules, as we demonstrate in the following section, they exhibit very effective performance in attaining feasibility.[4]

## 4. NUMERICAL RESULTS

We carried out our experiments in MATLAB on a Linux desktop with 4 Intel i7 cores and 16GB of RAM. First, we present an experiment on synthetic data, where we fixed the number of variables

[3]Most of these assumptions pertain to the (global or local) Lipschitz constants of the cost function of (10) and its gradient. When such constants exist, they are typically estimated via crude means which ultimately results in very conservative step-sizes.

[4]We note that when achieving feasibility is the ultimate goal, convergence to a stationary point of the feasibility problem may not be very interesting.

to $N = 100$ and randomly drew QCQPs with varying number of constraints. To be more precise, here we set $M_E = 0$ (i.e., no equalities) and varied $M = M_I$ from 100 to 800 in increments of 100. In each instance, the constraint matrices $\{\mathbf{A}_m\}_{m=1}^M$ were randomly generated from a zero mean, i.i.d. Gaussian distribution with unit variance and then symmetrized. In order to ensure that the problem has a feasible solution, 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 $b_m \sim \mathcal{N}(\mathbf{p}^T\mathbf{A}_m\mathbf{p}, 1)$. 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 other words, 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 $\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^N | \|\mathbf{x}\|_2 \leq 1\}$. For each value of $M$, we generated 1000 such instances. In each instance, we randomly generated a unit-norm vector which was used for initializing GD, SGD and SVRG. We note that each method requires a different number of gradient evaluations per iteration. Hence, for fair comparison, we allocated a maximum budget of $1000M$ gradient evaluations for each method. Of course, this implies that the maximum number of iterations for each method is different, depending on the number of gradients evaluated per iteration. In this case, the step-sizes used were $\alpha_k = 0.1/\sqrt{(1+k/M)}$ for GD, $\alpha_k = 0.1/\sqrt{k}$ for SGD, and $\alpha_k^{(s)} = 0.01/\sqrt{1+((s-1)*K+k)/M}$ for SVRG while the value of the smoothing parameter $\mu$ for the inequality constraints was set to be $10^{-4}$. Furthermore, we set the update frequency parameter of SVRG to be $K = 4M$.

We plot the feasibility percentages averaged out over 1000 instances in Figure 1a as a function of $M$ while the average number of (normalized) gradient evaluations required is depicted in 1b. Here, we declared convergence to a feasible point once the cost function (10) attained a value of $\epsilon = 10^{-6}$. It is evident that SGD demonstrates the best overall performance in terms of feasibility percentage and computational complexity, with SVRG close behind. Interestingly, we observe that the feasibility percentages of all methods decrease as $M$ is increased from $2N$ to $4N$, but as $M$ is increased further, the feasibility percentages increase again. The complexity curve also depicts a similar trend, i.e., the average number of iterations for attaining feasibility increases with $M$ initially, but then decreases again. We have observed a similar phenomenon when this experiment is repeated for different values of $N$. This observation warrants further investigation, but as of this moment, we do not have a complete explanation of this phenomenon.

Although we do not present a complete report of the wall time results, the worst-case average time was 30 secs, 17 secs, and 20 secs for GD, SGD and SVRG respectively (corresponding to $M = 4N$). In contrast, FPP-SCA, even when applied on an instance with $M = N$, required 15 minutes on average for a *single iteration*. [5] Given this fact, in this case, we are well justified in omitting FPP-SCA from comparison. We also note that using C-ADMM as an alternative to our approach would prove to be extremely memory intensive. Overall, this showcases the effectiveness of our FOM-based approach in terms of performance and scalability.

Next, we consider a specific case of a problem in low-rank matrix regression. Here, the goal is to recover a rank-1 positive semidefinite matrix from random linear measurements, i.e., solve a problem

[5] Here, we implemented FPP-SCA using YALMIP [29] with SeDuMi [30] as the solver of choice.

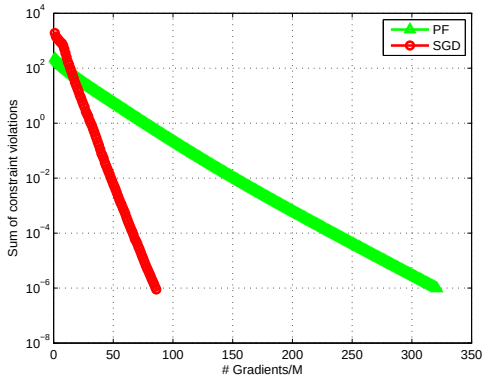



**Fig. 2**: Cost function vs No. of normalized gradients

of the form

$$\underset{\substack{\mathbf{X}\succeq\mathbf{0},\\ \text{rank}(\mathbf{X})=1}}{\text{find}} \quad \mathbf{X} \in \mathbb{R}^{N\times N} \tag{15a}$$

$$\text{s.t.} \quad \text{trace}(\mathbf{C}_m\mathbf{X}) = d_m,\ \forall\, m \in \mathcal{M}_{\mathcal{E}} \tag{15b}$$

which, of course, is equivalent to solving a system of quadratic equalities of the form

$$\underset{\mathbf{x}\in\mathbb{R}^N}{\text{find}} \quad \mathbf{x} \tag{16a}$$

$$\text{s.t.} \quad \mathbf{x}^T\mathbf{C}_m\mathbf{x} = d_m,\ \forall\, m \in \mathcal{M}_{\mathcal{E}} \tag{16b}$$

In [9], it is shown that when the measurement matrices $\{\mathbf{C}_m\}_{m=1}^{M_E}$ satisfy a version of the Restricted Isometry Property (RIP) [9, Definiton 3.1], then GD applied on (10) (here $M_I = 0$ and $\mathcal{X} = \mathbb{R}^N$) with spectral initialization (termed *Procustes Flow* (PF) by the authors) provably converges to an optimal solution of (16) at a linear rate. Assuming Gaussian measurements, $\Omega(N)$ measurements suffice for guaranteeing recovery. We considered an experiment with $N = 50$ and $M = M_E = 200$ (symmetric) measurement matrices drawn from the spiked Gaussian random ensemble [9, Footnote 2] to compare the performance of PF with our empirically tuned SGD method. We plot the least-squares cost function versus the number of gradient evaluations required to achieve $\epsilon = 10^{-6}$ optimality for a single representative realization in Figure 2 . Here, SGD is *randomly initialized* (standard Gaussian) with a step-size rule $\alpha_k = 0.1/\|\mathbf{x}^{(k)}\|_2^2$ while PF is implemented as described in [9] with spectral initialization and a constant step-size. From the figure, the speedup offered by SGD over PF is readily apparent, in spite of the fact that our approach currently lacks theoretical guarantees. We view these results as promising and seek, in our future work, to address this point.

## 5. CONCLUSIONS

In this paper, we developed a framework for computing feasible points of general non-convex QCQPs via simple first-order methods, which exhibit very effective performance on large-scale problems at low complexity relative to competing alternatives. In spite of the lack of any theoretical guarantees in general, we demonstrated a problem where our empirically-tuned SGD method with random initialization outperforms a provably convergent GD method with spectral initialization. We finally note that while feasible points can be computed efficiently using our methods, iteratively refining the solution via other approximation strategies (e.g., SCA) can still be expensive for large scale QCQPs. We are currently working towards extending our approach to cover this more general case.

## 6. REFERENCES


[1] 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.), pp. 117–165, Cambridge University Press, 2010.

[2] A. d'Aspremont, and S. Boyd, "Relaxations and Randomized Methods for Nonconvex QCQPs," *EE392 Lecture Notes, Stanford University*, 2003.

[3] 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.

[4] B. Marks and G. Wright, "A general inner approximation algorithm for nonconvex mathematical programs," *Oper. Res.*, vol. 26, no. 4, pp. 681–683, 1978.

[5] A. Beck, A. Ben-Tal, and L. Tetruashvili, "A sequential parametric convex approximation method with applications to nonconvex truss topology design problems," *J. Global Optim.*, vol. 47, no. 1, pp. 29-51, 2010.

[6] 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.

[7] G. Scutari, F. Facchinei, L. Lampariello, and P. Song, "Parallel and distributed methods for nonconvex optimization-part I: Theory," *arXiv preprint arXiv:1410.4754v2*, 2016.

[8] 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, July 2015.

[9] S. Tu, R. Boczar, M. Soltanolkotabi, and B. Recht, "Low-rank solutions of linear matrix equations via procrustes flow," *arXiv preprint arXiv:1507.03566*, 2015.

[10] C. De Sa, K. Olukotun, and C. Re, "Global convergence of stochastic gradient descent for some non-convex matrix problems," *arXiv preprint arXiv:1411.1134v3*, 2015.

[11] R. Sun, and Z.-Q. Luo, "Guaranteed matrix completion via nonconvex factorization," *Proc. IEEE FOCS*, pp. 270–289, Oct. 17-20, 2015, Berkley, CA.

[12] S. Bhojanapalli, A. Kyrillidis, and S. Sanghavi, "Dropping convexity for faster semi-definite optimization", *arXiv preprint arXiv:1509.03917*, 2015.

[13] Y. Chen, and M. J. Wainwright, "Fast low-rank estimation by projected gradient descent: General statistical and algorithmic guarantees", *arXiv preprint arXiv:1509.03025*, 2015.

[14] 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, 2015.

[15] 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.

[16] J. Sun, Q. Qu, and J. Wright, "A geometric analysis of phase retrieval," *arXiv preprint arXiv:1602.06664*, 2016.

[17] 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, 2016.

[18] S. Boyd, and L. Vandenberghe, "*Convex Optimization*," Cambridge, U.K.: Cambridge Univ. Press, 2004.

[19] Y. Nesterov, "Smooth minimization of non-smooth functions," *Math. Program.*, vol. 103, no. 1, pp 127-152, May 2005.

[20] R. Johnson, and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," *Adv. Neural Info. Process Syst.*, pp. 315–323, 2013.

[21] 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.

[22] 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.

[23] S. Ghadimi, and G. Lan, "Accelerated gradient methods for nonconvex nonlinear and stochastic programming", *Math. Prog.*, vol. 156, no. 1-2, pp. 59-99, 2016.

[24] S. Ghadimi, G. Lan, and H. Zhang, "Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization", *Math. Prog.*, vol. 155, no. 1-2, pp. 267-305, 2016.

[25] 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. Prog.*, vol. 157, no. 2, pp. 515-545, 2016.

[26] S. J. Reddi, A. Hefny, S. Sra, B. Poczos, and A. Smola, "Stochastic variance reduction for nonconvex optimization," *arXiv preprint arXiv:1603.06160v2*, 2016.

[27] Z.-A. Zhu, and E. Hazan, "Variance reduction for faster nonconvex optimization," *arXiv preprint arXiv:1603.05643*, 2016.

[28] S. J. Reddi, S. Sra, B. Poczos, and A. Smola, "Fast stochastic methods for nonsmooth nonconvex optimization," *arXiv preprint arXiv:1605.06900v1*, 2016.

[29] J. Lofberg, "Yalmip: A toolbox for modeling and optimization in MATLAB," in *Proc. CACSD*, Taipei, Sep. 4, 2004.

[30] J. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optimization Methods and Software*, vol. 11, no. 1, pp. 625–653, 1999.