A conjugate gradient-based algorithm for large-scale quadratic programming problem with one quadratic constraint

A. Taati · M. Salahi

Abstract

In this paper, we consider the nonconvex quadratically constrained quadratic programming (QCQP) with one quadratic constraint. By employing the conjugate gradient method, an efficient algorithm is proposed to solve QCQP that exploits the sparsity of the involved matrices and solves the problem via solving a sequence of positive definite system of linear equations after identifying suitable generalized eigenvalues. Specifically, we analyze how to recognize hard case (case 2) in a preprocessing step, fixing an error in Sect. 2.2.2 of Pong and Wolkowicz (Comput Optim Appl 58(2):273–322, 2014) which studies the same problem with the two-sided constraint. Some numerical experiments are given to show the effectiveness of the proposed method and to compare it with some recent algorithms in the literature.

Keywords QCQP · Conjugate gradient algorithm · Generalized eigenvalue problem

1 Introduction

We consider the following quadratically constrained quadratic programming (QCQP)

\[
\min_{x} \quad q(x) := x^T Ax + 2a^T x
\]

\[
\text{s.t.} \quad g(x) := x^T Bx + 2b^T x + \beta \leq 0,
\]

where \( A, B \in \mathbb{R}^{n \times n} \) are symmetric matrices with no definiteness assumed, \( a, b \in \mathbb{R}^n \) and \( \beta \in \mathbb{R} \). When \( B = I, b = 0 \) and \( \beta < 0 \), QCQP reduces to the classical trust-region...
subproblem (TRS), which arises in regularization or trust-region methods for unconstrained optimization [5,25]. Despite being nonconvex, numerous efficient algorithms have been developed to solve TRS [1,6,8,9,18]. The existing algorithms for TRS can be classified into two categories; approximate methods and accurate methods. The Steihaug-Toint algorithm is a well-known approximate method that exploits the preconditioned conjugate gradient iterations for unconstrained minimization of \( q(x) \) to obtain an approximate solution to large-scale instances of TRS [21,23]. Precisely, the method follows a piecewise linear path connecting the conjugate gradient iterates for solving the system \( Ax = -a \), which either finds an interior solution of TRS or terminates with a point on the boundary which does not allow the accuracy of the constrained solution to be specified. To overcome the lack of accuracy of Steihaug-Toint algorithm, Gould et al. in [8] proposed the generalized Lanczos trust-region method which minimizes the problem on an expanding sequence of subspaces generated by Lanczos vectors. The classical algorithm [15] by Moré and Sorensen is an accurate method for TRS, which at each iteration makes use of the Cholesky factorization to solve a positive definite linear system and hence is not proper for large-scale instances. A number of algorithms designed for large-scale TRS reformulate the problem as a parameterized eigenvalue problem [6,18,19]. These methods are matrix-free in the sense that only matrix-vector products are required and hence, are appropriate for sparse large-scale applications. In [18], Rendl and Wolkowicz developed a dual simplex-type method to solve TRS via a parametric eigenvalue problem and used the Lanczos algorithm for computing the smallest eigenvalue. Moreover, a primal simplex-type iteration was proposed for the so-called hard case. In [19], the authors presented an approach based on a parameterized eigenvalue problem which finds the optimal solution of TRS from the eigenvectors associated with the two smallest eigenvalues of the parameterized eigenvalue problem corresponding to the optimal parameter. Unlike the method of [18], this algorithm can handle both easy and hard cases in the same basic iteration. More recently, it has been shown that TRS can be solved by one generalized eigenvalue problem [1].

The QCQP is a natural generalization of TRS and is important in scaled trust-region methods for nonlinear optimization, allowing a possible indefinite scaling matrix. It also has applications in double well potential problems [7] and compressed sensing for geological data [11]. In recent years, QCQP has received much attention in the literature and various methods have been developed to solve it [3,13,16,17,22,24]. In [16], Moré gives a characterization of the global minimizer of QCQP and describes an algorithm for the solution of QCQP which extends the one for TRS [15]. A connection between the solution methods for QCQP and semidefinite programming (SDP) is established in [24]. However, the SDP approach is not practical for large-scale problems. Recently, it has been shown that when the quadratic forms are simultaneously diagonalizable, QCQP has a second order cone programming (SOCP) reformulation which is significantly more tractable than a semidefinite problem [3]. In [13], the authors further showed that QCQP with an optimal value bounded from below is SOCP representable, which extends the results in [3]. The QCQP with the two-sided constraint has been studied in [17]. In that paper, the optimality conditions of the underlying problem has been derived under a constraint qualification showing that it can be assumed without loss of generality. In particular, it has been shown that the problem
can be reduced to an equality constrained problem. To solve this equality constrained problem, the authors have extended the method of [18] and transformed the problem to a parameterized generalized eigenvalue problem. Salahi and Taati [22] also derived a diagonalization-based algorithm under the simultaneous diagonalizable condition of the quadratic forms. The method is proper for small and medium-scale instances since it needs matrix factorization. Recently, Adachi and Nakatsukasa developed an algorithm for QCQP that is strictly feasible and there exists $\hat{\lambda} \geq 0$ such that $A + \hat{\lambda}B \succ 0$ [2]. Their algorithm requires $\hat{\lambda}$ and finding just one eigenpair of a $(2n + 1) \times (2n + 1)$ generalized eigenvalue problem, extending the one for TRS [1] to QCQP. The overall complexity of the algorithm is $O(n^3)$. However, due to the requirement of constructing explicit form of the Hessian matrices, when the involved matrices are not highly sparse, the method is not efficient for large-scale problems. Most recently, in [12], the authors have derived a novel convex quadratic reformulation (CQR) for QCQP which is indeed the problem of minimizing a linear objective function over one or two convex quadratic constraints. They also showed that the optimal solution of QCQP can be obtained from the optimal solution of the new reformulation. It is shown that this CQR is equivalent to a minimization of a maximum of two convex quadratic functions. Although the latter minimization problem is nonsmooth, it is convex with good structure and two steepest descent algorithms corresponding to two different line search rules have been presented to solve it. The most expensive operation at each iteration of the two algorithms is several matrix vector products that are cheap when applying to large sparse problems.

In the present paper, we propose a conjugate gradient-based algorithm for QCQP which applies the conjugate gradient method to solve a sequence of positive definite system of linear equations, and hence is efficient for large sparse instances of QCQP. Our approach first verifies hard case (case 2) and if it is the case, then the optimal solution of QCQP is computed via solving a positive definite system of linear equations. Indeed, our discussion fixes an error in Sect. 2.2.2 of [17] to detect hard case 2. Ruling out the hard case 2 instances, QCQP reduces to solving the secular equation $\phi(\lambda) = 0$ where $\phi(\lambda) = x(\lambda)^T B x(\lambda) + 2b^T x(\lambda) + \beta$ and $x(\lambda) = -(A + \lambda B)^{-1}(a + \lambda b)$. Then we propose an efficient algorithm to solve $\phi(\lambda) = 0$ and, specifically, we introduce an innovative way of computing $\phi(\lambda)$ when $A + \lambda B$ is close to being singular in hard case 1 instances. It is worth noting that although the method of [16] and ours follow the same framework to solve the problem in the case where the optimal Lagrangian Hessian matrix is positive definite, there are some differences in our approach. Firstly, in [16], no specific algorithm is proposed to solve the secular equation $\phi(\lambda) = 0$, while here we propose an efficient algorithm to solve it. Secondly, in [16], hard case (case 2) is discussed theoretically only based on sign of function $\phi(\lambda)$ on $(\lambda, \bar{\lambda})$ and looking at $\lim_{\lambda \downarrow \bar{\lambda}} \phi(\lambda)$ and $\lim_{\lambda \uparrow \bar{\lambda}} \phi(\lambda)$ where $A + \lambda B$ is positive semidefinite (singular) at $\lambda$ and $\bar{\lambda}$. In contrast, here we analyze how one can detect hard case 2 without computing $\lim_{\lambda \downarrow \bar{\lambda}} \phi(\lambda)$ and $\lim_{\lambda \uparrow \bar{\lambda}} \phi(\lambda)$.

The rest of the paper is organized as follows. In Sect. 2, we review some known facts related to QCQP. A conjugate gradient-based algorithm is introduced in Sect. 3. Finally, in Sect. 4, we give some numerical results to show the effectiveness of the proposed method comparing with the algorithms from [2,12] and [19].
2 Preliminaries: optimality, easy and hard cases

In this section, we state some results related to the QCQP that are useful in the next section. Throughout this paper, we consider the following assumptions.

**Assumption 1** QCQP has a strictly feasible solution, i.e., there exists \( \hat{x} \) with \( g(\hat{x}) < 0 \).

**Assumption 2** There exists \( \hat{\lambda} \geq 0 \) such that \( A + \hat{\lambda}B \succ 0 \).

When Assumption 1 is violated, QCQP reduces to an unconstrained minimization problem and Assumption 2 ensures that QCQP has an optimal solution [2]. The following theorem gives a set of necessary and sufficient conditions for the global optimal solution of QCQP under Assumption 1.

**Theorem 1** ([16], Theorem 3.4) Consider QCQP. Then \( x^* \) is a global optimal solution if and only if there exists \( \lambda^* \geq 0 \) such that

\[
\begin{align*}
(A + \lambda^* B)x^* &= -(a + \lambda^* b), \\
& (1) \\
g(x^*) &\leq 0, \\
& (2) \\
\lambda^* g(x^*) &= 0, \\
& (3) \\
(A + \lambda^* B) &\succeq 0. \\
& (4)
\end{align*}
\]

An optimal solution of QCQP belongs to one of the following two types: an interior solution with \( g(x^*) < 0 \) or a boundary solution with \( g(x^*) = 0 \). The following lemma states the case where QCQP has no boundary solution.

**Lemma 1** The QCQP has no optimal solution on the boundary of the feasible region if and only if \( A \succ 0 \) and \( g(-A^{-1}a) < 0 \).

**Proof** Let \( x^* \) be the optimal solution of QCQP. If \( A \npreceq 0 \), then from Theorem 1, \( g(x^*) = 0 \) and hence the optimal solution is on the boundary of the feasible region. Now suppose that \( A \) is positive semidefinite (singular) and \( g(x^*) < 0 \). Let \( v \in \text{Null}(A) \) and consider the following quadratic equation of variable \( \alpha \):

\[
g(x^* + \alpha v) = (x^* + \alpha v)^T B(x^* + \alpha v) + 2b^T (x^* + \alpha v) + \beta = 0.
\]

This equation has a root since \( v^T B v > 0 \) and \( g(x^*) < 0 \). To show \( v^T B v > 0 \), recall that, by Assumption 2, there exists \( \hat{\lambda} \geq 0 \) with \( A + \hat{\lambda}B > 0 \). This implies that \( v^T A v + \hat{\lambda} v^T B v > 0 \). Since \( v^T A v = 0 \), then \( v^T B v > 0 \). The discussion above proves
that in the case where \(A\) is positive semidefinite (singular), QCQP has a solution on the boundary. The only case that must be considered is the case where \(A\) is positive definite. In this case, \(x^* = -A^{-1}a\) is the unique unconstrained minimizer of \(q(x)\). Hence, QCQP has no solution with \(g(x^*) = 0\) if and only if \(g(-A^{-1}a) < 0\).

In view of Lemma 1, unless \(A \succ 0\) and \(g(-A^{-1}a) < 0\), QCQP has a boundary solution. Hence, from now on, we focus on the boundary solutions since an interior solution can be obtained by solving the linear system \(Ax = -a\).

By Assumption 2, \(A\) and \(B\) are simultaneously diagonalizable [14], i.e., there exists an invertible matrix \(Q\) and diagonal matrices \(D = \text{diag}(d_1, \ldots, d_n)\) and \(E = \text{diag}(e_1, \ldots, e_n)\) such that \(Q^TAQ = D\) and \(Q^TBQ = E\). It is easy to see that \(A + \lambda B \succeq 0\) if and only if \(\lambda \leq \lambda \leq \bar{\lambda}\) where

\[
\lambda = \max \left\{ -\frac{d_i}{e_i} | e_i > 0 \right\}, \quad \bar{\lambda} = \min \left\{ -\frac{d_i}{e_i} | e_i < 0 \right\}.
\]

If \(A > 0\), \(\lambda < 0\) else \(\lambda \geq 0\). Let \(I\) be the smallest interval containing all nonnegative \(\lambda\) satisfying the optimality conditions (1) and (4). It follows from Theorem 1 that the interior of \(I\), \(\text{int}(I) = (\max(0, \lambda), \bar{\lambda})\). The interval \(I\) contains \(\bar{\lambda}\) as the right endpoint if \((a + \lambda b) \in \text{Range}(A + \lambda B)\) and it contains \(\lambda\) as the left endpoint if \(\lambda \geq 0\) and \((a + \lambda b) \in \text{Range}(A + \lambda B)\). For any \(\lambda \in (\lambda, \bar{\lambda})\), define

\[
\phi(\lambda) := x(\lambda)^T B x(\lambda) + 2b^T x(\lambda) + \beta,
\]

where \(x(\lambda) = -(A + \lambda B)^{-1}(a + \lambda b)\). The function \(\phi(\lambda)\) has the following properties.

**Lemma 2** ([16], Theorem 5.2) \(\phi(\lambda)\) is either constant or strictly decreasing on \((\lambda, \bar{\lambda})\). \(\phi(\lambda)\) is constant if and only if

\[
\begin{bmatrix}
a \\
b
\end{bmatrix} \in \text{Range} \begin{bmatrix}
A \\
B
\end{bmatrix}.
\]

**Proposition 1** ([2], Proposition 3) If \(B \succeq 0\), then \(\lim_{\lambda \to +\infty} \phi(\lambda) < 0\).

The QCQP similar to TRS is classified into easy case and hard case (case 1 and 2) instances. The characterization of the easy and hard cases of the QCQP with the two-sided constraint is given in [17]. Here we adapt the characterization for the QCQP based on the requirement \(\lambda^* \geq 0\) where \(\lambda^*\) denotes the optimal Lagrange multiplier. We have to consider separately the cases where \(A\) is positive definite and \(A\) is not positive definite as follows:

- \(A\) is positive definite.
  1. The easy case occurs if one of the following holds:
     (i) \(\lambda\) is infinite.
     (ii) \(\lambda\) is finite and \(a + \lambda b \notin \text{Range}(A + \lambda B)\). This implies that \(\lambda^* \in (0, \lambda)\).
  2. The hard case 1 occurs if the following holds:
     (i) \(\lambda\) is finite and \(a + \lambda b \in \text{Range}(A + \lambda B)\) and \(\lambda^* \in (0, \lambda)\).
3. The hard case 2 occurs if the following holds:
   (iii) \( \bar{\lambda} \) is finite and \( a + \lambda \bar{b} \in \text{Range}(A + \bar{\lambda}B) \) and \( \lambda^* = \bar{\lambda} \).

   - \( \lambda^* \) is not positive definite.

   1. The easy case occurs if one of the following three cases holds:
      (i) Both \( \hat{\lambda} \) and \( \bar{\lambda} \) are finite, \( a + \lambda \hat{b} \notin \text{Range}(A + \hat{\lambda}B) \) and \( a + \lambda \hat{b} \notin \text{Range}(A + \bar{\lambda}B) \). This implies that \( \lambda^* \in (\hat{\lambda}, \bar{\lambda}) \).
      (ii) Only \( \hat{\lambda} \) is finite and \( a + \lambda \hat{b} \notin \text{Range}(A + \lambda \hat{b}) \). This implies that \( \lambda^* \in (\lambda, \infty) \).
      (iii) Only \( \bar{\lambda} \) is finite and \( a + \lambda \bar{b} \notin \text{Range}(A + \bar{\lambda}B) \). This implies that \( \lambda^* \in [0, \bar{\lambda}) \).

   2. The hard case 1 occurs if one of the following three cases holds:
      (i) Both \( \hat{\lambda} \) and \( \bar{\lambda} \) are finite and \( a + \lambda \hat{b} \in \text{Range}(A + \hat{\lambda}B) \) or \( a + \bar{\lambda}b \in \text{Range}(A + \bar{\lambda}B) \) and \( \lambda^* \in (\hat{\lambda}, \bar{\lambda}) \).
      (ii) Only \( \hat{\lambda} \) is finite and \( a + \lambda \hat{b} \in \text{Range}(A + \lambda \hat{b}) \) and \( \lambda^* \in (\hat{\lambda}, \infty) \).
      (iii) Only \( \bar{\lambda} \) is finite and \( a + \lambda \bar{b} \in \text{Range}(A + \bar{\lambda}B) \) and \( \lambda^* \in [0, \bar{\lambda}) \).

3. The hard case 2 occurs if one of the following holds:
   (i) Both \( \hat{\lambda} \) and \( \bar{\lambda} \) are finite and \( a + \lambda \hat{b} \in \text{Range}(A + \hat{\lambda}B) \) and \( \lambda^* = \hat{\lambda} \) or \( a + \lambda \bar{b} \in \text{Range}(A + \bar{\lambda}B) \) and \( \lambda^* = \bar{\lambda} \).
   (ii) Only \( \hat{\lambda} \) is finite and \( a + \lambda \hat{b} \in \text{Range}(A + \lambda \hat{b}) \) and \( \lambda^* = \hat{\lambda} \).
   (iii) Only \( \bar{\lambda} \) is finite and \( a + \lambda \bar{b} \in \text{Range}(A + \bar{\lambda}B) \) and \( \lambda^* = \bar{\lambda} \).

From Lemma 2 and Proposition 1, we see that in the easy case and hard case 1, \( \lambda^* \) is the unique solution of equation \( \phi(\lambda) = 0 \) in the interval \( (\max\{0, \hat{\lambda}, \bar{\lambda}\}) \). Hard case 2 corresponds to the case where equation \( \phi(\lambda) = 0 \) has no solution in \( (\max\{0, \hat{\lambda}, \bar{\lambda}\}) \).

## 3 A conjugate gradient-based algorithm

In this section, we assume that a value of \( \hat{\lambda} \) such that \( A + \hat{\lambda}B > 0 \) is known. By Lemma 1, excluding the case where \( A \) is positive definite and \( \phi(\lambda) = 0 \), solving QCQP is equivalent to finding a nonnegative \( \lambda^* \in [\max\{0, \hat{\lambda}, \bar{\lambda}\}] \) such that \( \phi(\lambda^*) = 0 \). Note that except the case where QCQP is a hard case 2 instance, the nonlinear equation \( \phi(\lambda) = 0 \) has a unique root in interval \( (\max\{0, \hat{\lambda}, \bar{\lambda}\}) \). According to this fact, we propose a conjugate gradient-based algorithm for QCQP which first checks for hard case 2 and if this is the case, a global optimal solution of QCQP is obtained via solving a positive definite system of linear equations, otherwise, the global solution of QCQP is computed by finding the root of equation \( \phi(\lambda) = 0 \) in \( (\max\{0, \hat{\lambda}, \bar{\lambda}\}) \). We use the value of \( \phi(\hat{\lambda}) \) to form the algorithm. We consider the following three cases:

1. **Case 1.** \( \phi(\hat{\lambda}) = 0 \). In this case, obviously \( (\hat{\lambda}, x(\hat{\lambda})) \) satisfies the optimality conditions (1)–(4) and hence \( x(\hat{\lambda}) \) is the unique optimal solution of QCQP.

2. **Case 2.** \( \phi(\hat{\lambda}) > 0 \). In this case, by Lemma 2, \( \lambda^* \in (\hat{\lambda}, \bar{\lambda}) \). Now we consider the following two possible subcases:
   (a) \( \hat{\lambda} \) is infinite. In this case, from Proposition 1, equation \( \phi(\lambda) = 0 \) has a unique root in \( (\hat{\lambda}, \bar{\lambda}) \).
(b) \( \tilde{\lambda} \) is finite. In this case, either QCQP is a hard case 2 instance \( (\lambda^* = \tilde{\lambda}) \) or equation \( \phi(\lambda) = 0 \) has a unique root in \( (\tilde{\lambda}, \hat{\lambda}) \).

3. **Case 3.** \( \phi(\hat{\lambda}) < 0 \). In this case, by Lemma 2, \( \lambda^* \in [\max \{0, \tilde{\lambda}\}, \hat{\lambda}] \). Now we consider the following two possible subcases:

(a) If \( \tilde{\lambda} < 0 \), then \( A > 0 \). Set \( x^* = -A^{-1}a \). If \( g(x^*) \leq 0 \), then \( x^* \) with \( \lambda^* = 0 \) is the optimal solution of QCQP. Otherwise, equation \( \phi(\lambda) = 0 \) has a unique root in \( (0, \hat{\lambda}) \).

(b) If \( \tilde{\lambda} \geq 0 \), either QCQP is a hard case 2 instance \( (\lambda^* = \tilde{\lambda}) \) or equation \( \phi(\lambda) = 0 \) has a unique root in \( (\lambda, \hat{\lambda}) \).

To the best of our knowledge, there is no algorithm in the literature to compute \( \lambda \) and \( \bar{\lambda} \) for general \( A \) and \( B \) when the value of \( \hat{\lambda} \) is not known. However, when the value of \( \hat{\lambda} \) is available, \( \lambda \) and \( \bar{\lambda} \) can be efficiently computed via finding some generalized eigenvalues of a matrix pencil, see Sect. 4.3 of [17] and Sect. 2.1 of [12]. Precisely, \( A + \lambda B \succeq 0 \) if and only if \( \lambda \leq \lambda_1 \leq \bar{\lambda} \) where \( \lambda_1 = \lambda_1 + \hat{\lambda}, \bar{\lambda} = \lambda_2 + \hat{\lambda} \).

It is worth noting that, when \( \hat{\lambda} \) is given, we only need to compute one extreme eigenvalue to determine the initial interval containing the optimal Lagrange multiplier, i.e., only one of \( \lambda \) and \( \bar{\lambda} \).

### 3.1 Verifying hard case 2

In this subsection, we discuss how one can detect hard case 2 if it is the case. Let \( \hat{\lambda} \in I \) and \( \lambda^* \in (\hat{\lambda}, \bar{\lambda}] \). In this case, it is well-known that, QCQP is a hard case 2 instance if and only if \( \lim_{\lambda \uparrow \hat{\lambda}} \phi(\lambda) \geq 0 \). Similarly, if \( \lambda \in I \) and \( \lambda^* \in [\bar{\lambda}, \hat{\lambda}) \), then QCQP is hard case 2 if and only if \( \lim_{\lambda \downarrow \bar{\lambda}} \phi(\lambda) \leq 0 \) [17]. Pong and Wolkowicz [17], proposed an algorithm based on minimum generalized eigenvalue of a parameterized matrix pencil to solve QCQP with the two-sided constraint that first carries out a preprocessing technique to recognize hard case 2, see Sect. 2.2.2 of [17] for more details. They proposed a procedure for computing \( \lim_{\lambda \uparrow \hat{\lambda}} \phi(\lambda) \) and \( \lim_{\lambda \downarrow \bar{\lambda}} \phi(\lambda) \) which indeed computes \( \phi(\hat{\lambda}) \) and \( \phi(\bar{\lambda}) \), respectively. According to their approach, when \( \lambda \in I \) and \( \lambda^* \in [\hat{\lambda}, \bar{\lambda}) \), QCQP is hard case 2 if and only if \( \phi(\lambda) \leq 0 \) where \( x(\lambda) = -(A + \lambda B)\dagger(a + \lambda b) \). Similarly, when \( \lambda \in I \) and \( \lambda^* \in (\hat{\lambda}, \bar{\lambda}] \), QCQP is hard case 2 if and only if \( \phi(\lambda) \geq 0 \) where \( x(\lambda) = -(A + \lambda B)\dagger(a + \lambda b) \). Unfortunately, in general these assertions are not true, QCQP may be hard case 2 but \( g(x(\lambda)) > 0 \) or \( g(x(\lambda)) < 0 \). A problem of this type is given in the following example.
Example 1 Consider a QCQP with

\[ A = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & 0 \\ 0 & -1 \end{bmatrix}, \quad a = \begin{bmatrix} -25 \\ -33/2 \end{bmatrix}, \quad b = \begin{bmatrix} 50 \\ 25 \end{bmatrix}, \quad \beta = 0. \]

and set \( \hat{\lambda} = \frac{3}{4}. \) It is easy to see that \( \underline{\lambda} = \frac{1}{2}, \hat{\lambda} = 1, \phi(\hat{\lambda}) < 0 \) and \( (a + \lambda b) \in \text{Range}(A + \lambda B) \), and so \( \lambda^* \in [\underline{\lambda}, \hat{\lambda}) \). This problem is hard case 2 since \( x^* = \begin{bmatrix} -25 + \sqrt{457} \\ 8 \end{bmatrix}^T \) with \( \lambda^* = \underline{\lambda} \) satisfy the optimality conditions

\[ (A + \lambda B)x^* = -(a + \lambda b), \quad g(x^*) = 0. \]

On the other hand, we have \( x(\underline{\lambda}) = \begin{bmatrix} 0 \\ 8 \end{bmatrix}^T \) and \( g(x(\underline{\lambda})) = 336 > 0. \)

The reason why Pong and Wolkowicz’s approach fails to detect hard case 2 is that generally \( \lim_{\lambda \uparrow \tilde{\lambda}} \phi(\lambda) \) and \( \lim_{\lambda \downarrow \lambda_0} \phi(\lambda) \) are not equal to \( \phi(\tilde{\lambda}) \) and \( \phi(\lambda_0) \), respectively. The following lemma states situations where we have \( \lim_{\lambda \uparrow \tilde{\lambda}} \phi(\lambda) = \phi(\tilde{\lambda}) \) and \( \lim_{\lambda \downarrow \lambda_0} \phi(\lambda) = \phi(\lambda_0) \).

Lemma 3 Consider QCQP and let \( I \) be defined as before. Define \( N = \{1, 2, \ldots, n\} \)

\( T_1 := \{i \in N | d_i + \lambda e_i \neq 0\} \) and \( T_2 := \{i \in N | d_i + \tilde{\lambda} e_i \neq 0\} \) where \( D = \text{diag}(d_1, \ldots, d_n) \) and \( E = \text{diag}(e_1, \ldots, e_n) \) are defined as before.

(i) Let \( \underline{\lambda} \in I \), then \( \lim_{\lambda \downarrow \lambda} \phi(\lambda) = \phi(\underline{\lambda}) \) if and only if \( \sum_{i \in N \setminus T_1} \frac{b_i^2}{e_i^2} = 0. \)

(ii) Let \( \tilde{\lambda} \in I \), then \( \lim_{\lambda \uparrow \tilde{\lambda}} \phi(\lambda) = \phi(\tilde{\lambda}) \) if and only if \( \sum_{i \in N \setminus T_2} \frac{b_i^2}{e_i^2} = 0. \)

Proof (i) First note that by the change of variables \( x := Q^{-1}x, a := QTa \) and \( b := QTb \), QCQP is equivalent to the problem

\[
\min \quad x^T Dx + 2a^Tx + 2b^Tx + \beta \leq 0,
\]

where \( Q \) is defined as before. Since \( \underline{\lambda} \in I \), it follows from the definition of \( x(\underline{\lambda}) \) that \( x(\underline{\lambda}) = [s_1, \ldots, s_n]^T \) where

\[
s_i = \begin{cases} \frac{(a_i + \lambda b_i)}{(d_i + \lambda e_i)}, & \text{if } d_i + \lambda e_i \neq 0, \\ 0, & \text{if } d_i + \lambda e_i = 0, \end{cases}
\]

for \( i = 1, \ldots, n. \) Then

\[
\phi(\underline{\lambda}) = \sum_{i \in T_1} \left( e_i \frac{(a_i + \lambda b_i)^2}{(d_i + \lambda e_i)^2} - 2b_i \frac{(a_i + \lambda b_i)}{(d_i + \lambda e_i)} \right) + \beta.
\]
On the other hand, we have
\[
\lim_{\lambda \downarrow \hat{\lambda}} \phi(\lambda) = \lim_{\lambda \downarrow \hat{\lambda}} \sum_{i \in T_1} \left( e_i (a_i + \lambda b_i)^2 - 2b_i (a_i + \lambda b_i) \right) \\
+ \sum_{i \in N \backslash T_1} \left( e_i (a_i + \lambda b_i)^2 - 2b_i (a_i + \lambda b_i) \right) + \beta \\
= \phi(\hat{\lambda}) - \sum_{i \in N \backslash T_1} \frac{b_i^2}{e_i},
\]
which completes the proof.

(ii) The proof is similar to part (i).

In what follows, we use the following result to show how one can verify hard case 2 in general.

Proposition 2 ([17], Lemma 3.4) If $\bar{\lambda}$ is finite, then $v^T B v < 0$ for all nonzero $v \in \text{Null}(A + \bar{\lambda} B)$. If $\underline{\lambda}$ is finite, then $v^T B v > 0$ for all nonzero $v \in \text{Null}(A + \underline{\lambda} B)$.

Consider the following two cases.

Case 1. $\hat{\lambda} \in I$ and $\lambda^* \in (\hat{\lambda}, \bar{\lambda}]$.
First notice that since $\hat{\lambda} \in I$, then $(a + \hat{\lambda} b) \in \text{Range}(A + \hat{\lambda} B)$. Hence, system
\[
(A + \hat{\lambda} B) x = -(a + \hat{\lambda} b),
\]
is consistent. Assume that the set $\{v_1, v_2, \ldots, v_r\}$ is an orthonormal basis of $\text{Null}(A + \hat{\lambda} B)$. Set $Z = [v_1, v_2, \ldots, v_r]$, then any solution of system (8) has the form $x = x(\hat{\lambda}) + Z y$ where $x(\hat{\lambda}) = -(A + \hat{\lambda} B)^\dagger (a + \hat{\lambda} b)$ and $y \in \mathbb{R}^r$. Now consider the following maximization problem:
\[
p^* := \max_{y \in \mathbb{R}^r} g(x(\hat{\lambda}) + Z y) = y^T Z^T B Z y + 2y^T Z^T (B x(\hat{\lambda}) + b) + g(x(\hat{\lambda})).
\]
By Proposition 2, $Z^T B Z < 0$ and hence the optimal solution of (9), $y^*$, is the unique solution of the positive definite system
\[
-(Z^T B Z) y^* = Z^T (B x(\hat{\lambda}) + b),
\]
and $p^* = g(x(\hat{\lambda}) + Z y^*)$. In the sequel, we show that $\lambda^* = \hat{\lambda}$ if and only if $p^* \geq 0$. To see this, suppose that $p^* \geq 0$. Set $x^* = x(\hat{\lambda}) + Z y^*$. Next consider the following quadratic equation of variable $\alpha$:
\[
v^T B v \alpha^2 + 2\alpha v^T (B x^* + b) + p^* = 0,
\]
where $v \in \text{Null}(A + \bar{\lambda} B)$. Due to the facts that $v^T B v < 0$ and $p^* \geq 0$, the above equation has a root $\alpha$. Now it is easy to see that $x^* := x^* + \alpha v$ with $\lambda^* = \hat{\lambda}$ satisfies the optimality conditions of QCQP:
\[(A + \tilde{\lambda} B)x^* = -(a + \tilde{\lambda} b),\]
\[g(x^*) = 0.\]

Next suppose that \(\lambda^* = \tilde{\lambda}\) and \(x^*\) is a global optimal solution of QCQP. Since \(\tilde{\lambda} > 0\), it follows from optimality conditions that \(g(x^*) = 0\) and thus \(p^* \geq 0\).

**Case 2.** \(\tilde{\lambda} \in I\) and \(\lambda^* \in [\underline{\lambda}, \hat{\lambda})\).

This is similar to Case 1. First notice that since \(\tilde{\lambda} \in I\), then \((a + \tilde{\lambda} b) \in \text{Range}(A + \tilde{\lambda} B)\). Hence, system

\[(A + \tilde{\lambda} B)x = -(a + \tilde{\lambda} b),\]

is consistent. Assume that the set \(\{v_1, v_2, \ldots, v_r\}\) is an orthonormal basis of Null\((A + \tilde{\lambda} B)\). Set \(Z = [v_1, v_2, \ldots, v_r]\), then any solution of system (11) has the form \(x = x(\tilde{\lambda}) + Zy\) where \(x(\tilde{\lambda}) = -(A + \tilde{\lambda} B)^\dagger(a + \tilde{\lambda} b)\) and \(y \in \mathbb{R}^r\). Now consider the following minimization problem:

\[p^* := \min_{y \in \mathbb{R}^r} g(x(\tilde{\lambda}) + Zy) = y^T Z^T BZy + 2y^T Z^T (Bx(\tilde{\lambda}) + b) + g(x(\tilde{\lambda})).\]  

(12)

By Proposition 2, \(Z^T BZ \succ 0\) and hence the optimal solution of (12), \(y^*\), is the unique solution of the positive definite system

\[(Z^T BZ)y^* = -Z^T (Bx(\tilde{\lambda}) + b),\]  

and \(p^* = g(x(\tilde{\lambda}) + Zy^*)\). In the sequel, we show that \(\lambda^* = \tilde{\lambda}\) if and only if \(p^* \leq 0\).

To see this, suppose that \(p^* \leq 0\). Set \(x^* = x(\tilde{\lambda}) + Zy^*\). Next consider the following quadratic equation of variable \(\alpha\):

\[v^T Bv\alpha^2 + 2\alpha v^T (Bx^* + b) + p^* = 0,\]

where \(v \in \text{Null}(A + \tilde{\lambda} B)\). Due to the facts that \(v^T Bv > 0\) and \(p^* \leq 0\), the above equation has a root \(\alpha\). Now it is easy to see that \(x^* := x^* + \alpha v\) with \(\lambda^* = \tilde{\lambda}\) satisfies the optimality conditions of QCQP:

\[(A + \tilde{\lambda} B)x^* = -(a + \tilde{\lambda} b),\]
\[g(x^*) = 0.\]

Next suppose that \(\lambda^* = \tilde{\lambda}\) and \(x^*\) is a global optimal solution of QCQP. Since \(\tilde{\lambda} > 0\), it follows from optimality conditions that \(g(x^*) = 0\) and thus \(p^* \leq 0\).

**Example 2** Consider the same problem in Example 1. We have \(\tilde{\lambda} \in I\) and \(\lambda^* \in [\underline{\lambda}, \hat{\lambda})\).

Moreover, \(Z = [1, 0]^T\) is an orthonormal basis of Null\((A + \tilde{\lambda} B)\) and \(x(\tilde{\lambda}) = [0, 8]^T\). It is easy to see that the optimal solution of problem (12), \(y^* = -25\) and so \(p^* = -914 < 0\). Therefore, QCQP is a hard case 2 instance and \(x^* = [-25 + \sqrt{457}, 8]^T\) with \(\lambda^* = \tilde{\lambda}\) satisfy the optimality conditions.
3.2 Solving the nonlinear equation $\phi(\lambda) = 0$

When QCQP is not hard case 2, $\lambda^*$ is the unique solution of the equation $\phi(\lambda) = 0$ on the underlying interval containing the optimal Lagrange multiplier. If $B$ is positive semidefinite, it can be shown that the function $\phi(\lambda)$ is convex on $I$ and thus a safeguarded version of Newton’s method is a reasonable choice for the solution of this equation, see Sect. 5 of [16]. This is the approach used by Moré and Sorenson [15] for solving the TRS which at each iteration makes use of the Cholesky decomposition and hence is not proper for large-scale problems. If $B$ is indefinite, then $\phi(\lambda)$ may not be convex or concave (see the example in Sect. 5 of [16]) and hence there is no guarantee that Newton’s method will be convergent [16]. Here, we propose an algorithm for the solution of $\phi(\lambda) = 0$ which is indeed the bisection method occupied with two techniques to accelerate its convergence [17,22].

Algorithm for solving equation $\phi(\lambda) = 0$

Iterate until a termination criterion is met:

1. Set $\lambda_{new}$ to the midpoint of the interval containing $\lambda^*$.
2. If the points at which $\phi(\lambda)$ has positive and negative values are available, do inverse linear interpolation, update $\lambda_{new}$ if inverse linear interpolation is successful.
3. At $\lambda_{new}$, take a step to the boundary if points at which $\phi(\lambda)$ has positive and negative values are known.

End loop.

In what follows, we explain two techniques used in the algorithm.

Inverse interpolation

Let $[\lambda_b, \lambda_g]$ be the current interval containing $\lambda^*$. Moreover, suppose the relation (5) does not hold. Then, by Lemma 2, $\phi(\lambda)$ is strictly decreasing and hence, we can consider its inverse function. We approximate the inverse function $\lambda(\phi)$ by a linear function. Then we set $\lambda_{new} = \lambda(0)$ if $\lambda(0) \in [\lambda_b, \lambda_g]$.

Primal step to the boundary

Let $x^*$ denote the optimal solution of QCQP. It can be shown that as the algorithm proceeds, $\lambda_{new}$ converges to the $\lambda^*$, the optimal Lagrange multiplier and hence, the sequence $x(\lambda)$ also will be convergent to $x^*$. Now, suppose that there exist values $\lambda_b$ and $\lambda_g$ with $\phi(\lambda_b) > 0$ and $\phi(\lambda_g) < 0$. Then, we can take an inexpensive primal step to the point $\alpha x(\lambda_b) + (1 - \alpha) x(\lambda_g)$ on the boundary by choosing a suitable step length $\alpha$. This likely improves the objective value. We note that the resulting sequence is also convergent to $x^*$.

Computing $\phi(\lambda)$

At each iteration of the algorithm, $\phi(\lambda)$ is computed by applying the conjugate gradient algorithm to the positive definite system of linear equations.
The conjugate-gradient (CG) method is one of the most widely used iterative methods for solving symmetric positive definite linear equations. CG algorithm is matrix-free and hence it is efficient for large-scale applications. In hard case 1 when \( \lambda^* \) is near to \( \hat{\lambda} \) or \( \tilde{\lambda} \), system (14) may become ill-conditioned. In the following two theorems (Theorems 2 and 3), we show that, to overcome near singularity, one can solve an alternative well-conditioned positive definite system that has the same solution as (14). Before stating that, we need the following proposition and lemma.

**Proposition 3** Suppose that Assumption 2 holds. Then \((\lambda, v)\) is an eigenpair of the pencil \((A, -B)\) if and only if \((-\frac{1}{\lambda-\lambda^*}, (A + \hat{\lambda}B)^{-\frac{1}{2}}v)\) is an eigenpair of matrix \((A + \hat{\lambda}B)^{-\frac{1}{2}}B(A + \hat{\lambda}B)^{-\frac{1}{2}}\).

**Proof** \((\lambda, v)\) is an eigenpair of pencil \((A, -B)\) if and only if \((A + \lambda B)v = 0\), implying that \((A + \hat{\lambda}B + (\lambda - \hat{\lambda})B)v = 0\). Since \((A + \hat{\lambda}B) \succ 0\), it follows that \((I + (\lambda - \hat{\lambda})(A + \hat{\lambda}B)^{-\frac{1}{2}}B(A + \hat{\lambda}B)^{-\frac{1}{2}})(A + \hat{\lambda}B)^{\frac{1}{2}}v = 0\). Note that \(\lambda \neq \hat{\lambda}\) because \(\det(A + \hat{\lambda}B) \neq 0\). This implies that \((A + \hat{\lambda}B)^{-\frac{1}{2}}B(A + \hat{\lambda}B)^{-\frac{1}{2}} + \frac{1}{\lambda - \hat{\lambda}}I)(A + \hat{\lambda}B)^{\frac{1}{2}}v = 0\) which completes the proof. \(\blacksquare\)

**Lemma 4** Suppose that \(\lambda \neq \hat{\lambda}\). Then \((a + \lambda b) \in \text{Range}(A + \lambda B)\) if and only if \((-b + B(A + \hat{\lambda}B)^{-1}(a + \lambda b)) \in \text{Range}(A + \lambda B)\).

**Proof** \((a + \lambda b) \in \text{Range}(A + \lambda B)\) if and only if there exists \(\bar{x}\) such that \((A + \lambda B)\bar{x} = -(a + \lambda b)\). This implies that

\[
[(A + \hat{\lambda}B) + (\lambda - \hat{\lambda})B]\bar{x} = -[(a + \lambda b) + (\lambda - \hat{\lambda})b].
\]

Next, the change of variable \(\bar{x} = \bar{y} - (A + \hat{\lambda}B)^{-1}(a + \lambda b)\) gives

\[
(A + \lambda B)\bar{y} = (\lambda - \hat{\lambda})(-b + B(A + \hat{\lambda}B)^{-1}(a + \lambda b)),
\]

implying that \((-b + B(A + \hat{\lambda}B)^{-1}(a + \lambda b)) \in \text{Range}(A + \lambda B)\). \(\blacksquare\)

**Theorem 2** Suppose that \((a + \lambda b) \in \text{Range}(A + \lambda B)\). Moreover, assume that the set \(\{v_1, \ldots, v_r\}\) is a basis of \(\text{Null}(A + \lambda B)\) that is \((A + \hat{\lambda}B)\)-orthonormal, i.e., \(v_i^T(A + \hat{\lambda}B)v_j = 0\) for all \(i \neq j\). Let \(\hat{\lambda} \neq \lambda \in (\Lambda, \tilde{\Lambda})\), then \(x^*\) is the solution of system

\[
(A + \lambda B)x = -(a + \lambda b),
\]

if and only if \(y^* = x^* + z\) is the solution of

\[
\begin{bmatrix}
A + \lambda B + \alpha \sum_{i=1}^r (A + \hat{\lambda}B)v_i v_i^T (A + \hat{\lambda}B)
\end{bmatrix} y = (\lambda - \hat{\lambda})(Bz - b).
\]
where \( z = (A + \hat{\lambda}B)^{-1}(a + \hat{\lambda}b) \) and \( \alpha \) is an arbitrary positive constant. The same assertion holds when \( \lambda \) is replaced by \( \bar{\lambda} \).

**Proof** It is easy to see that \( x^* \) is the unique solution of system (16) if and only if \( y^* = x^* + z \) is the solution of

\[
(A + \lambda B)y = (\lambda - \hat{\lambda})(Bz - b).
\]  (18)

To prove the theorem, it is sufficient to show that systems (18) and (17) are equivalent. System (17) can be rewritten as follows:

\[
\left[ (A + \hat{\lambda}B) + (\lambda - \hat{\lambda})B + \alpha \sum_{i=1}^{r}(A + \hat{\lambda}B)v_i v_i^T (A + \hat{\lambda}B) \right] y = (\lambda - \hat{\lambda})(Bz - b).
\]

Since \( (A + \hat{\lambda}B) > 0 \), we have

\[
\left[ I + (\lambda - \hat{\lambda})D + \alpha \sum_{i=1}^{r}(A + \hat{\lambda}B)^{1/2} v_i v_i^T (A + \hat{\lambda}B)^{1/2} \right] (A + \hat{\lambda}B)^{1/2} y
\]

\[
= (\lambda - \hat{\lambda}) (A + \hat{\lambda}B)^{-1/2} (Bz - b).
\]  (19)

Set \( M = (A + \hat{\lambda}B)^{-1/2} B (A + \hat{\lambda}B)^{-1/2} \), then by Proposition 3, \( q_i = (A + \hat{\lambda}B)^{1/2} v_i, i = 1, \ldots, r \), are the eigenvectors of \( M \) corresponding to the eigenvalue \(-\frac{1}{\hat{\lambda} - \lambda}\). Note that \( q_i^T q_i = 1 \) for \( i = 1, \ldots, r \) and \( q_i^T q_j = 0 \) for all \( i \neq j \in \{1, \ldots, r\} \). Without loss of generality, let \( M = QDQ^T \) be the eigenvalue decomposition of \( M \) in which \( Q \) contains \( q_i = 1, \ldots, r \) as its \( r \) first columns. From (19) we get

\[
\left[ I + (\lambda - \hat{\lambda})D + \alpha \sum_{i=1}^{r} Q^T (A + \hat{\lambda}B)^{1/2} v_i v_i^T (A + \hat{\lambda}B)^{1/2} Q \right] Q^T (A + \hat{\lambda}B)^{1/2} y
\]

\[
= (\lambda - \hat{\lambda}) Q^T (A + \hat{\lambda}B)^{-1/2} (Bz - b).
\]  (20)

Furthermore, we have

\[
Q^T (A + \hat{\lambda}B)^{1/2} v_i v_i^T (A + \hat{\lambda}B)^{1/2} Q = e_i e_i^T, \quad i = 1, \ldots, r,
\]  (21)

due to the fact that

\[
q_k^T (A + \hat{\lambda}B)^{1/2} v_i v_i^T (A + \hat{\lambda}B)^{1/2} q_j = \begin{cases} 1 & \text{if } k = j = i \in \{1, \ldots, r\}, \\ 0 & \text{otherwise}, \end{cases}
\]
where \( e_i \) is the unit vector and \( q_i, i = 1, \ldots, n \), denote \( i \)’th column of \( Q \). Therefore, it follows from (20) and (21) that

\[
\left( I + (\lambda - \hat{\lambda})D + \alpha \sum_{i=1}^{r} e_i e_i^T \right) Q^T (A + \hat{\lambda}B)^{\frac{1}{2}} y = (\lambda - \hat{\lambda}) Q^T (A + \hat{\lambda}B)^{-\frac{1}{2}} (Bz - b).
\]

(22)

Now since \((a + \hat{\lambda}b) \in \text{Range}(A + \lambda B)\), by Lemma 4, \((Bz - b) \in \text{Range}(A + \lambda B)\) and consequently \(v_i^T (Bz - b) = 0 \) for \( i = 1, \ldots, r \). This implies that

\[
q_i^T (A + \hat{\lambda}B)^{-\frac{1}{2}} (Bz - b) = 0, \quad i = 1, \ldots, r.
\]

Therefore, the \( r \) first components of the right hand side vector in system (22) are zero. Further let \( c_i, i = 1, \ldots, r \), denote the \( r \) first diagonal elements of matrix \((I + (\lambda - \hat{\lambda})D)\). By Proposition 3, it is easy to see that \( c_i = \frac{\lambda - \lambda_j}{\lambda - \hat{\lambda}} > 0 \), \( i = 1, \ldots, r \). Since \( \alpha > 0 \), the above discussion proves that system (22) is equivalent to the following system:

\[
(I + (\lambda - \hat{\lambda})D) Q^T (A + \hat{\lambda}B)^{\frac{1}{2}} y = (\lambda - \hat{\lambda}) Q^T (A + \hat{\lambda}B)^{-\frac{1}{2}} (Bz - b).
\]

This is also equivalent to

\[
(A + \lambda B) y = (\lambda - \hat{\lambda})(Bz - b),
\]

which completes the proof. When \( \lambda \) is replaced by \( \hat{\lambda} \), the assertion can be proved in a similar manner.

\[\square\]

**Remark 1** It is worth to mention that Theorem 2 is completely different from Theorem 8 of [2] since in Theorem 2 we introduce an alternative positive definite system of linear equations to the system \((A + \hat{\lambda}B)x = -(a + \hat{\lambda}b)\) for all \( \lambda \neq \lambda \in (\hat{\lambda}, \lambda) \) when hard case 1 holds. In contrast, in Theorem 8 of [2], an alternative positive definite system of linear equations is introduced to the system \((A + \lambda^* B)x = -(a + \lambda^* b)\) where \( \lambda^* \) is the optimal Lagrange multiplier and QCQP is hard case 2.

**Theorem 3** Suppose that \((a + \hat{\lambda}b) \in \text{Range}(A + \lambda B)\). Moreover, assume that the set \( \{v_1, \ldots, v_r\} \) is a basis of \( \text{Null}(A + \hat{\lambda}B) \) that is \((A + \hat{\lambda}B)\)-orthogonal, i.e., \( v_i^T (A + \hat{\lambda}B)v_j = 1 \) and \( v_i^T (A + \hat{\lambda}B)v_j = 0 \) for all \( i \neq j \). Then matrix \( \tilde{A} = A + \hat{\lambda}B + \alpha \sum_{i=1}^{r} (A + \hat{\lambda}B)v_i v_i^T (A + \hat{\lambda}B) \) is positive definite. The same assertion holds when \( \hat{\lambda} \) is replaced by \( \lambda \).

**Proof** Since \( A + \hat{\lambda}B \succeq 0 \) and \( \sum_{i=1}^{r} (A + \hat{\lambda}B)v_i v_i^T (A + \hat{\lambda}B) \succeq 0 \), then \( \tilde{A} \succeq 0 \). To show that \( \tilde{A} \succ 0 \) it is sufficient to prove that if \( x^T \tilde{A}x = 0 \) for \( x \in \mathbb{R}^n \), then \( x = 0 \). For any \( x \in \mathbb{R}^n \), there exist unique \( x_1 \in \text{Null}(A + \hat{\lambda}B) \) and \( x_2 \in \text{Range}(A + \hat{\lambda}B) \) such that \( x = x_1 + x_2 \). We have
\[ x^T (\tilde{A}) x = x_2^T (A + \hat{\lambda} B) x_2 + \alpha \sum_{i=1}^{r} (v_i^T (A + \hat{\lambda} B) x)^2 = 0. \]

This implies that
\[ x_2^T (A + \hat{\lambda} B) x_2 = 0, \quad v_i^T (A + \hat{\lambda} B) x = 0, \quad i = 1, \ldots, r. \quad (23) \]

Since \((A + \lambda B) \succeq 0\), we obtain \((A + \lambda B) x_2 = 0\), and hence \(x_2 \in \text{Null}(A + \lambda B)\). Together with the assumption that \(x_2 \in \text{Range}(A + \lambda B)\) we obtain \(x_2 = 0\). Thus \(x = x_1\). On the other hand \(x_1\) can be written as \(x_1 = \sum_{j=1}^{r} c_j v_j\) where \(c_j \in \mathbb{R}\) for \(j = 1, \ldots, r\). So it follows from (23) that \(\sum_{j=1}^{r} c_j v_i^T (A + \hat{\lambda} B) v_j = 0\) for \(i = 1, \ldots, r\), resulting in \(c_j = 0\) for \(j = 1, \ldots, r\). Therefore \(x_1 = 0\) and consequently \(x = 0\), which completes the proof. When \(\lambda\) is replaced by \(\hat{\lambda}\), the assertion can be proved in a similar manner. \qed

### 4 Numerical experiments

In this section, on several randomly generated problems of various dimensions, we compare the proposed conjugate gradient-based algorithm (CGB) with Algorithms 1 and 2 from [12], Algorithm 3.2 from [2] and LSTRS algorithm from [19]. The comparison with Algorithm 3.2 of [2] is done for dimension up to 5000 as it needs longer time to solve larger dimensions. All computations are performed in MATLAB 8.5.0.197613 on a 1.70 GHz laptop with 4 GB of RAM. Throughout the paper, we have assumed that there exists \(\hat{\lambda}\) such that \(A + \hat{\lambda} B \succ 0\). In practice \(\hat{\lambda}\) is usually unknown in advance, but since all four algorithms require it for initialization, we assume that \(\hat{\lambda}\) is known and skip its computation. However, the algorithms in [4,10,16] can be used to find \(\hat{\lambda}\). We use the stopping criterion

\[
\max \{ |\phi(\lambda)|, \| (A + \lambda B) x + (a + \lambda b) \| \} < 10^{-8},
\]

or

\[
\frac{|\text{high} - \text{low}|}{|\text{high}| + |\text{low}|} < 10^{-11}
\]

for solving the equation \(\phi(\lambda) = 0\) where \(x\) is the best feasible solution of QCQP obtained up to the current iteration and [low, high] is the interval containing \(\lambda^*\) at the current iteration. We set \(\epsilon_1 = 10^{-10}\), \(\epsilon_2 = 10^{-13}\) and \(\epsilon_3 = 10^{-10}\) in Algorithms 1 and 2 from [12]. As in [12], to compute \(\hat{\lambda}\) and \(\lambda\), the generalized eigenvalue problem is solved by eigifp.\footnote{eigifp is a MATLAB program for computing a few algebraically smallest or largest eigenvalues and their corresponding eigenvectors of the generalized eigenvalue problem, available from http://www.ms.uky.edu/~qye/software.html.} To verify hard case 2, as described in Sect. 3.1, we need to compute an orthonormal basis of \(\text{Null}(A + \lambda B)\) or \(\text{Null}(A + \hat{\lambda} B)\). Recall that \(A + \lambda B\)
and $A + \tilde{\lambda}B$ are singular and positive semidefinite. Thus, a nullspace vector of $A + \lambda B$ and $A + \tilde{\lambda}B$ can be found by finding an eigenvector corresponding to the smallest eigenvalue, i.e., 0, which in our numerical tests, is done by `eigfzp`. When QCQP is not hard case 2 but it is detected to be hard case 1, to apply Theorem 2, Gram-Schmidt process [20] can be used to compute an $(A + \tilde{\lambda}B)$-orthonormal basis for $\text{Null}(A + \lambda B)$ or $\text{Null}(A + \tilde{\lambda}B)$ that is required in system (17). We apply the Newton refinement process in Sect. 4.1.2 of [2] to all four algorithms to improve the accuracy of the solution. For simplicity, we consider QCQP with nonsingular $B$ including positive definite and indefinite $B$. Hence, we can assume without loss of generality that $b = 0$. Our test problems include both easy and hard case (case 1 and case 2) instances.

4.1 First class of test problems

In this class of test problems, we consider QCQP with positive definite $A$ and indefinite $B$. Thus, we set $\hat{\lambda} = 0$. We follow Procedure 1, Procedure 2 and Procedure 3 to generate an easy case, hard case 1 and hard case 2 instance, respectively.

**Procedure 1** (Generating an easy case instance)

We first generate randomly a sparse positive definite matrix $A$ and a sparse indefinite matrix $B$ via $A = \text{sprandsym}(n, \text{density}, 1/\text{cond}, 2)$ and $B = \text{sprandsym}(n, \text{density})$ where `cond` refers to condition number. After computing $\tilde{\lambda}$, we set $a = -(A + \lambda B)x_0$, where $x_0$ is computed via $x_0 = \text{randn}(n, 1)/10$ and $\lambda$ is chosen uniformly from $(0, \tilde{\lambda})$. Next, to avoid generating a QCQP having an interior solution, we chose $\beta$ randomly from $(-s, -\ell)$ where $s = x_c^T B x_c$ and $\ell = x_0^T B x_0$ and $x_c$ is the solution of linear system $Ax = -a$. From the optimality conditions we see that the above construction likely gives an easy case instance.

**Procedure 2** (Generating a hard case 1 instance)

We generate $A$, $B$, $x_0$ and $\beta$ as the above procedure but we set $a = -(A + \tilde{\lambda}B)x_0$. From the optimality conditions we see that this construction gives a hard case 1 instance.

**Procedure 3** (Generating a hard case 2 instance)

We generate $A$, $B$ and $a$ as Procedure 2 but we set $\beta = -\ell$.

We set $\text{cond} = 10, 100, 1000$ and for each dimension and each condition number, we generated 10 instances, and the corresponding numerical results are adjusted in Tables 1 and 2, where we report the average runtime in second (Time) and accuracy (Accuracy). To measure the accuracy, we have computed the relative objective function difference as follows:

$$\frac{q(x^*) - q(x_{\text{best}})}{|q(x_{\text{best}})|},$$

where $x^*$ is the computed solution by each method and $x_{\text{best}}$ is the solution with the smallest objective value among the four algorithms. We use "Alg1" and "Alg2" to denote Algorithms 1 and 2 in [12], respectively, and "Alg3" to denote Algorithm 3.2 in [2].
Table 1: Computational results of the first class of test problems with density=1e−2

| n   | Time(s) | Accuracy |
|-----|---------|----------|
|     | CGB     | Alg1     | Alg2     | Alg3     |
|     | cond = 10 |          |          |          |
| Easy case |          |          |          |          |
| 1000 | 0.24    | 0.32     | 0.27     | 0.60     | 0.0000e+00 | 1.3534e−14 | 2.9908e−14 | 2.703592e−16 |
| 2000 | 0.36    | 0.43     | 0.36     | 3.85     | 9.5123e−17 | 2.1722e−15 | 2.7597e−15 | 4.640596e−17 |
| 3000 | 0.50    | 0.79     | 0.52     | 11.56    | 4.4606e−17 | 5.4207e−16 | 3.9725e−16 | 7.106022e−17 |
| 4000 | 0.92    | 1.40     | 1.24     | 30.17    | 2.4456e−16 | 1.7972e−15 | 2.2517e−15 | 1.711915e−16 |
| 5000 | 1.14    | 1.66     | 1.25     | 417.53   | 4.0827e−16 | 4.2164e−16 | 4.5748e−16 | 2.455677e−16 |
| cond = 100 |          |          |          |          |
| Easy case |          |          |          |          |
| 1000 | 0.56    | 0.74     | 0.58     | 0.54     | 2.2119e−16 | 7.7736e−13 | 1.3902e−11 | 2.089449e−16 |
| 2000 | 0.87    | 1.16     | 0.92     | 3.61     | 6.9482e−17 | 1.2703e−13 | 5.0994e−13 | 1.123734e−16 |
| 3000 | 1.98    | 2.84     | 2.32     | 12.70    | 1.2295e−16 | 6.3770e−14 | 2.6147e−12 | 4.2882e−17   |
| 4000 | 1.98    | 2.84     | 2.32     | 50.70    | 1.6666e−16 | 4.2965e−14 | 4.6357e−13 | 5.4903e−17   |
| 5000 | 4.18    | 6.97     | 5.16     | 673.36   | 2.5057e−16 | 1.8041e−14 | 5.3498e−14 | 1.4510e−17   |
| cond = 1000 |          |          |          |          |
| Easy case |          |          |          |          |
| 1000 | 3.75    | 5.33     | 4.13     | 0.56     | 1.3319e−16 | 1.2508e−11 | 1.5758e−09 | 3.415593e−16 |
| 2000 | 4.29    | 6.47     | 4.45     | 3.84     | 1.5526e−16 | 3.3995e−12 | 2.8861e−10 | 1.421543e−16 |
| 3000 | 8.59    | 17.89    | 9.56     | 12.17    | 4.3130e−15 | 1.2760e−12 | 8.4678e−11 | 2.068219e−16 |
| 4000 | 10.56   | 18.10    | 13.71    | 41.88    | 1.2220e−15 | 5.5054e−13 | 1.1153e−11 | 6.020699e−17 |
| 5000 | 16.36   | 30.95    | 21.69    | 371.55   | 1.7654e−16 | 4.5521e−13 | 3.9881e−11 | 1.804393e−16 |
| n   | Time(s) | Accuracy |
|-----|---------|----------|
|     | CGB     | Alg1     | Alg2 | Alg3 | CGB     | Alg1     | Alg2 | Alg3 |
| cond = 10 |
| Hard case 1 |
| 1000 | 0.29    | 0.30  | **0.23** | 0.68 | 2.5820e−16 | 4.6885e−14 | 4.7315e−13 | 2.141403e−16 |
| 2000 | **0.49** | 1.53  | 0.72 | 5.33 | 8.9252e−17 | 2.9432e−13 | 3.1029e−11 | 9.468005e−17 |
| 3000 | 1.60    | 1.87  | **1.56** | 18.63 | 8.8924e−17 | 1.0251e−14 | 8.0005e−14 | 8.529213e−17 |
| 4000 | **1.34** | 1.63  | 1.50 | 48.54 | 7.6730e−17 | 4.5916e−15 | 5.3948e−15 | 1.863880e−16 |
| 5000 | **6.09** | 9.09  | 6.59 | 307.54 | 1.7215e−16 | 4.9540e−15 | 2.8191e−15 | 4.814561e−16 |
| cond = 100 |
| Hard case 1 |
| 1000 | 0.37    | 0.72  | **0.33** | 0.61 | 3.4527e−16 | 2.1326e−12 | 1.1098e−10 | 1.517419e−16 |
| 2000 | **0.70** | 2.26  | 1.07 | 4.40 | 1.4639e−16 | 1.8446e−12 | 2.0338e−11 | 1.263138e−16 |
| 3000 | 3.46    | 4.41  | **3.29** | 12.75 | 4.1817e−16 | 1.5004e−13 | 1.8512e−12 | 5.416258e−17 |
| 4000 | **5.36** | 8.24  | 6.14 | 43.24 | 2.8051e−16 | 2.1693e−13 | 6.9215e−12 | 1.061026e−16 |
| 5000 | **9.04** | 13.48 | 10.05 | 310.26 | 8.2448e−17 | 6.1508e−14 | 1.5047e−13 | 4.103097e−17 |
| cond = 1000 |
| Hard case 1 |
| 1000 | 1.41    | 6.77  | 1.38 | **0.70** | 1.7472e−16 | 1.0359e−10 | 6.6120e−09 | 1.024406e−16 |
| 2000 | 4.52    | 7.23  | 4.77 | **4.04** | 3.6154e−16 | 3.4160e−12 | 2.2689e−10 | 1.746964e−16 |
| 3000 | **11.11** | 29.03 | 13.83 | 20.93 | 1.5364e−16 | 9.3206e−11 | 2.6199e−09 | 1.716415e−16 |
| 4000 | **23.37** | 47.93 | 27.75 | 52.19 | 1.3664e−16 | 1.7287e−12 | 1.0004e−10 | 1.511822e−16 |
| 5000 | **21.09** | 46.43 | 28.84 | 402.58 | 6.0663e−16 | 1.5179e−12 | 2.0300e−10 | 2.686279e−16 |
Table 1 continued

| n  | Time(s) |       |       |       | Accuracy |       |       |       |
|----|---------|-------|-------|-------|----------|-------|-------|-------|
|    |         | CGB   | Alg1  | Alg2  | Alg3     | CGB   | Alg1  | Alg2  |
|    |         |       |       |       |          |       |       |       |
|    |         |       |       |       |          |       |       |       |
| cond = 10                        |         |       |       |       |          |       |       |       |
| Hard case 2                      |         |       |       |       |          |       |       |       |
| 1000 | 0.26   | 0.88  | 0.36  | 3.00  | 7.4723e−17 | 4.9359e−13 | 4.8652e−11 | 5.471231e−16 |
| 2000 | 0.36   | 1.58  | 0.73  | 16.70 | 0.0000e+00 | 1.6001e−13 | 3.8674e−13 | 3.896115e−16 |
| 3000 | 0.77   | 7.40  | 2.45  | 173.19| 3.4482e−16 | 2.4531e−13 | 7.8146e−11 | 3.030414e−16 |
| 4000 | 1.44   | 12.18 | 3.98  | 852.45| 4.5639e−17 | 8.9613e−14 | 1.3460e−12 | 2.422850e−16 |
| 5000 | 2.03   | 30.51 | 5.31  | 1463.27| 3.0985e−16 | 2.0535e−13 | 3.5345e−12 | 1.831296e−16 |
| cond = 100                       |         |       |       |       |          |       |       |       |
| Hard case 2                      |         |       |       |       |          |       |       |       |
| 1000 | 0.62   | 7.21  | 2.12  | 4.23  | 2.1081e−16 | 2.7349e−11 | 4.0860e−09 | 6.574916e−16 |
| 2000 | 1.01   | 7.94  | 3.01  | 15.10 | 1.3175e−16 | 3.5787e−12 | 5.1980e−10 | 8.660221e−17 |
| 3000 | 2.71   | 46.79 | 9.41  | 236.70| 1.0358e−16 | 1.3325e−11 | 2.5842e−09 | 9.794199e−17 |
| 4000 | 4.29   | 118.69| 20.47 | 495.63| 6.4502e−17 | 7.7936e−12 | 3.0469e−09 | 6.557389e−17 |
| 5000 | 8.22   | 209.70| 52.54 | 697.56| 1.2621e−16 | 6.3835e−12 | 7.1765e−10 | 6.557389e−17 |
| cond = 1000                      |         |       |       |       |          |       |       |       |
| Hard case 2                      |         |       |       |       |          |       |       |       |
| 1000 | 2.16   | 14.14 | 4.75  | 3.24  | 3.6635e−16 | 3.9804e−09 | 5.2611e−08 | 1.920469e−16 |
| 2000 | 3.49   | 31.50 | 7.13  | 29.45 | 4.3374e−17 | 6.5600e−09 | 1.9302e−08 | 1.173405e−16 |
| 3000 | 9.67   | 122.63| 27.69 | 79.25 | 4.2925e−17 | 3.2570e−09 | 3.9270e−08 | 2.720614e−16 |
| 4000 | 12.81  | 167.30| 43.43 | 599.55| 1.2716e−16 | 1.9239e−08 | 1.0997e−08 | 1.315645e−16 |
| 5000 | 17.62  | 245.88| 83.07 | 1151.76| 2.1056e−16 | 2.6294e−08 | 3.0084e−08 | 1.324996e−16 |

The best values among all algorithms are marked with bold font
| n     | Time(s) | Accuracy |
|-------|---------|----------|
|       | CGB     | Alg1     | Alg2     | CGB     | Alg1     | Alg2     |
|       |         |          |          |         |          |          |
| cond = 10 |
| Easy case |         |          |          |         |          |          |
| 10,000 | 0.47    | 0.62     | **0.43** | 1.1725e−15 | 2.9305e−15 | 1.9469e−15 |
| 20,000 | 1.10    | 1.65     | 1.10     | 1.6881e−15 | 7.8945e−16  | 6.9847e−16  |
| 30,000 | 1.43    | 2.29     | **1.34** | 8.5874e−16  | 5.0097e−16  | 4.8177e−16  |
| 40,000 | **2.78** | 4.07     | 2.82     | 3.9322e−16  | 8.6336e−16  | 6.7728e−16  |
| 50,000 | **3.53** | 5.74     | 3.82     | 2.1866e−16  | 1.5472e−15  | 1.6432e−14  |
| cond = 100 |
| Easy case |         |          |          |         |          |          |
| 10,000 | **2.79** | 3.69     | 2.84     | 0.0000e+00 | 5.1216e−13  | 9.4987e−12  |
| 20,000 | **4.23** | 6.40     | 4.73     | 0.0000e+00 | 5.3774e−14  | 1.7939e−13  |
| 30,000 | **4.43** | 6.05     | 4.52     | 0.0000e+00 | 2.2114e−14  | 7.4127e−14  |
| 40,000 | **8.26** | 12.96    | 8.76     | 1.6904e−16  | 2.7534e−14  | 9.8461e−14  |
| 50,000 | **8.70** | 11.88    | 9.25     | 2.9568e−16  | 9.3602e−15  | 2.6464e−15  |
| cond = 1000 |
| Easy case |         |          |          |         |          |          |
| 10,000 | **8.48** | 13.33    | 9.09     | 0.0000e+00 | 6.7834e−12  | 7.8038e−10  |
| 20,000 | **14.55** | 23.77   | 16.44    | 0.0000e+00 | 7.7513e−13  | 2.9847e−11  |
| 30,000 | **23.90** | 40.75   | 27.00    | 0.0000e+00 | 4.3943e−13  | 1.2151e−11  |
| 40,000 | **44.37** | 71.48   | 50.39    | 0.0000e+00 | 3.7856e−13  | 5.2588e−12  |
| 50,000 | **54.38** | 84.49   | 63.40    | 0.0000e+00 | 2.6751e−13  | 4.2168e−12  |
Table 2 continued

| n     | Time(s)                  | Accuracy        |         |         |         |
|-------|--------------------------|-----------------|---------|---------|---------|
|       | CGB | Alg1 | Alg2 | CGB | Alg1 | Alg2 |         |
| cond = 10 |     |      |      |     |      |      |         |
| Hard case 1 |     |      |      |     |      |      |         |
| 10,000 | 0.88 | 0.96 | 0.64 | 5.5224e−17 | 2.2499e−14 | 2.7961e−13 |         |
| 20,000 | 1.60 | 1.58 | 1.08 | 3.6226e−16 | 5.7196e−15 | 1.0666e−14 |         |
| 30,000 | 3.61 | 3.90 | 2.88 | 4.2608e−16 | 1.6371e−14 | 2.1939e−14 |         |
| 40,000 | 4.34 | 3.87 | 2.40 | 1.7872e−15 | 2.1961e−15 | 2.1956e−16 |         |
| 50,000 | 6.73 | 6.72 | 5.20 | 1.1921e−15 | 1.6576e−14 | 6.3980e−15 |         |
| cond = 100 |     |      |      |     |      |      |         |
| Hard case 1 |     |      |      |     |      |      |         |
| 10,000 | 2.91 | 4.10 | 2.65 | 0.0000e+00 | 1.0370e−12 | 3.1089e−11 |         |
| 20,000 | 4.40 | 5.36 | 3.30 | 0.0000e+00 | 1.0955e−13 | 1.4446e−11 |         |
| 30,000 | 12.51 | 14.60 | 7.99 | 0.0000e+00 | 2.4296e−13 | 8.2524e−13 |         |
| 40,000 | 10.66 | 10.68 | 7.37 | 3.2416e−15 | 1.7511e−14 | 1.8957e−14 |         |
| 50,000 | 19.62 | 16.69 | 11.66 | 5.8187e−15 | 9.0692e−15 | 3.5658e−15 |         |
| cond = 1000 |     |      |      |     |      |      |         |
| Hard case 1 |     |      |      |     |      |      |         |
| 10,000 | 19.04 | 20.76 | 14.89 | 0.0000e+00 | 7.8027e−12 | 1.6214e−09 |         |
| 20,000 | 28.85 | 45.77 | 27.56 | 0.0000e+00 | 1.5992e−12 | 1.1197e−10 |         |
| 30,000 | 64.18 | 42.81 | 0.0000e+00 | 2.7761e−11 |         |         |         |
| 40,000 | 106.70 | 61.81 | 0.0000e+00 | 2.2214e−11 |         |         |         |
| 50,000 | 58.34 | 93.55 | 58.01 | 0.0000e+00 | 4.6131e−13 | 4.4259e−12 |         |
| n          | Time(s) | Accuracy |
|------------|---------|----------|
|            | CGB     | Alg1     | Alg2     | CGB     | Alg1     | Alg2     |
| cond = 10  |         |          |          |         |          |          |
| Hard case 2| 10,000  | 0.41     | 4.25     | 1.22    | 0.0000e+00 | 2.3268e−13 | 5.8761e−11 |
|            | 20,000  | 0.76     | 3.50     | 1.04    | 0.0000e+00 | 5.8964e−14 | 7.2553e−14 |
|            | 30,000  | 1.74     | 25.12    | 9.12    | 0.0000e+00 | 6.8237e−13 | 5.6535e−12 |
|            | 40,000  | 2.06     | 14.36    | 6.11    | 0.0000e+00 | 1.3294e−13 | 3.8901e−13 |
|            | 50,000  | 2.78     | 14.73    | 5.20    | 0.0000e+00 | 1.0006e−13 | 7.4351e−14 |
| cond = 100 |         |          |          |         |          |          |
| Hard case 2| 10,000  | 1.78     | 15.16    | 3.42    | 0.0000e+00 | 5.5265e−10 | 5.1118e−09 |
|            | 20,000  | 1.84     | 25.53    | 5.79    | 0.0000e+00 | 6.0442e−12 | 9.0442e−10 |
|            | 30,000  | 2.61     | 30.48    | 7.78    | 0.0000e+00 | 8.1750e−13 | 1.6605e−11 |
|            | 40,000  | 4.39     | 91.63    | 16.24   | 0.0000e+00 | 4.7839e−12 | 1.4334e−10 |
|            | 50,000  | 7.59     | 136.59   | 27.00   | 0.0000e+00 | 4.5553e−11 | 1.3001e−10 |
| cond = 1000|        |          |          |         |          |          |
| Hard case 2| 10,000  | 4.70     | 38.58    | 9.19    | 0.0000e+00 | 1.5298e−08 | 1.2372e−07 |
|            | 20,000  | 7.41     | 92.02    | 20.77   | 0.0000e+00 | 6.0570e−09 | 2.0386e−08 |
|            | 30,000  | 34.75    | 293.60   | 79.92   | 0.0000e+00 | 9.4080e−10 | 9.4997e−10 |
|            | 40,000  | 57.08    | 344.94   | 70.91   | 0.0000e+00 | 1.5930e−10 | 3.6650e−09 |
|            | 50,000  | 53.14    | 631.03   | 138.77  | 0.0000e+00 | 1.0949e−10 | 6.6408e−10 |

The best values among all algorithms are marked with bold font.
4.2 Second class of test problems

We first generate randomly a sparse positive definite matrix $C$ and a sparse indefinite matrix $B$ via $C = \text{sprandsym}(n,\text{density},1/\text{cond},2)$ and $B = \text{sprandsym}(n,\text{density})$. Next, we set $A = C - B$. In this case, obviously, we can set $\hat{\lambda} = 1$. We follow Procedure 1 to generate an easy case instance but we choose $\lambda$ uniformly from $(\underline{\lambda}, \overline{\lambda})$ and if $s > \ell$, we choose $\beta$ randomly from $(-s, -\ell)$ else we choose $\beta$ from $(-\ell, -s)$ where $s = x_c^TBx_c$ and $x_c$ is the solution of linear system $Cx = -a$. Procedures 2 and 3 are followed to generate hard case (case 1 and 2) instances. The performance (speed and accuracy) of all four algorithms on this class of test problems for $\text{cond} = 10, 100, 1000$ are compared for several dimensions. Since the results are similar to the results reported in Tables 1 and 2, we just report the results for $\text{cond} = 1000$ in Figs. 1 and 2.

As we mentioned in Sect. 3.2, when QCQP is hard case 1 and $\lambda^*$ is close to $\underline{\lambda}$ or $\overline{\lambda}$, system (14) may become ill-conditioned and in this case, Theorem 2 can be applied to overcome this difficulty. For the first and second classes of test problems, $\lambda^*$ is not so close to $\underline{\lambda}$ and $\overline{\lambda}$ and hence system (14) is solved efficiently by the $\text{pcg}$ command in MATLAB. Therefore, for hard case 1 instances, we have reported the results of CGB algorithm without applying Theorem 2.

In our tests on the first and second class of test problems, we found out that for 90% of the generated instances, CGB algorithm is faster than Algorithm 3.2 from [2] while having comparable accuracy. The time difference becomes significant when we increase the dimension, since Algorithm 3.2 requires computing an extremal eigenpair of an $(2n + 1) \times (2n + 1)$ generalized eigenvalue problem which is time-consuming when the involved matrices are not highly sparse.

Moreover, we found out that CGB algorithm computes more accurate solutions than Algorithms 1 and 2. Furthermore, except for one dimension, it is always faster than Algorithm 1 and when we increase the dimension, the time difference becomes significant. For easy case problems, CGB Algorithm is faster than Algorithm 2 for about 77% of cases, specifically, CGB Algorithm is more efficient than Algorithm 2 when condition number is large. In hard Case 2, CGB Algorithm is much more efficient than Algorithms 1 and 2 because it first checks for hard case 2, if it is the case, the optimal solution of QCQP is computed via solving a positive definite system of linear equations. In hard case 1, CGB algorithm is slower than Algorithm 2 for about 60% of the cases but still faster than Algorithm 1 because of the extra time it takes for verifying hard case 2.

4.3 Third class of test problems

Here we generate two sets of QCQP instances that are hard case 1 and $\lambda^*$ is near to $\overline{\lambda}$. For the first set of test problems, we generate randomly a sparse positive definite matrix $A$ via $A = \text{spdiags}([3, \text{rand}(1,n-1)]', 0, n, n)$ and a sparse indefinite matrix $B$ via $B = \text{sprandsym}(n,\text{density})$. We set $\hat{\lambda} = 0$ and follow Procedure 2 to produce a hard case 1 instance but we set $s = x_c^T B x_c$ where $x_c$ is the solution of system $(A + (\hat{\lambda} - 10^{-4})B)x = -a$. To generate second set
Fig. 1 Average runtime and accuracy for the second class of test problems with density $= 1e^{-2}$ and cond $= 1000$
Fig. 2  Average runtime and accuracy for the second class of test problems with density $= 1e - 4$ and cond $= 1000$
of test problems, we first generate a sparse positive definite matrix $C$ and a sparse indefinite matrix $B$ via $C = \text{spdiags}([3, \text{rand}(1,n-1)], 0, n, n)$ and $B = \text{sprandsym}(n, \text{density})$. Next, we set $A = C - B$. In this case, obviously, we can set $\hat{\lambda} = 1$. We follow Procedure 2 to produce a hard case 1 instance but we set $s = x_c^T B x_c$ where $x_c$ is the solution of the system $(A + (\hat{\lambda} - 10^{-4}) B)x = -a$. The corresponding results are shown in Figs. 3 and 4 where we use "CGB2" to denote CGB algorithm using Theorem 2. As we see in Figs. 3 and 4, in the hard case 1 when $A + \lambda^* B$ is close to being singular, Theorem 2 improves both time and accuracy of CGB algorithm.

Fig. 3 Average runtime and accuracy for the first set of third class of test problems with density $= 1e^{-4}$

Fig. 4 Average runtime and accuracy for the second set of third class of test problems with density $= 1e^{-4}$
Table 3  Computational results of the forth class of test problems with density=1e−4

| n    | Time(s) | Accuracy |
|------|---------|----------|
|      | CGB     | LSTRS    | CGB     | LSTRS    |
| Easy case |         |          |         |          |
| 10,000 | 0.17    | 0.38     | 1.1786e−09 | 0.0000e+00 |
| 20,000 | 0.27    | 0.64     | 1.8697e−09 | 2.8298e−05 |
| 30,000 | 0.50    | 1.51     | 7.4911e−15 | 3.6290e−05 |
| 40,000 | 0.85    | 3.21     | 5.6924e−14 | 0.0000e+00 |
| 50,000 | 1.29    | 5.42     | 5.2551e−13 | 0.0000e+00 |
| Hard case 1 |         |          |         |          |
| 10,000 | 0.53    | 0.65     | 2.1961e−13 | 3.3070e−05 |
| 20,000 | 0.99    | 1.46     | 3.4968e−11 | 7.9676e−06 |
| 30,000 | 2.00    | 4.47     | 7.1046e−10 | 1.2190e−06 |
| 40,000 | 2.63    | 5.34     | 3.8496e−10 | 1.1510e−07 |
| 50,000 | 4.07    | 9.13     | 2.5284e−10 | 3.1789e−07 |
| Hard case 2 |         |          |         |          |
| 10,000 | 0.20    | 0.62     | 1.0914e−09 | 2.1140e−06 |
| 20,000 | 0.37    | 1.39     | 0.0000e+00 | 5.4198e−06 |
| 30,000 | 0.81    | 3.83     | 0.0000e+00 | 8.3600e−06 |
| 40,000 | 1.21    | 6.19     | 0.0000e+00 | 1.0990e−07 |
| 50,000 | 1.74    | 8.25     | 0.0000e+00 | 3.1789e−07 |

The best values among all algorithms are marked with bold font.

4.4 Fourth class of test problems

In this subsection, we compare CGB algorithm with LSTRS method in [19] on some random sparse large-scale instances of TRS. Our test problems include both easy and hard case (case 1 and case 2) instances. For TRS, we have $\lambda = \infty$ and hence, one can always find $\hat{\lambda} > \max\{0, \lambda\}$ such that $\phi(\hat{\lambda}) < 0$ following Procedure 4. This means that $\lambda^* \in [\max\{0, \lambda\}, \hat{\lambda}]$.

Procedure 4  (Computing $\hat{\lambda} > \max\{0, \lambda\}$ such that $\phi(\hat{\lambda}) < 0$)

Step 1. Choose $k > 0$.
Step 2. While $\phi(\max\{0, \lambda\} + k) > 0$, update $k := k + \epsilon$ for some fixed $\epsilon > 0$ and repeat this step.

To generate an easy case instance, we first generate randomly a sparse indefinite matrix $A$ via $A = \text{sprandsym}(n, \text{density})$. After computing $\lambda$, we set $a = -(A + \lambda I)x_0$, $\beta = -x_0^T x_0$ where $x_0$ is computed via $x_0 = \text{randn}(n, 1) / 10$ and $\lambda = \hat{\lambda} + r$ with $r$ chosen uniformly from $(0, \hat{\lambda} - \lambda)$. From optimality conditions, it follows that this procedure likely produces an easy case instance. To produce a hard case 1 instance, we set $a = -(A + \lambda I)x_0$ and $\beta = -x_0^T x_0 + 2$. Finally, if we set $\beta = -x_0^T x_0$, we obtain a hard case 2 instance. We used the default values for parameters of LSTRS method but provided it with a user eigensolver routine which applies the eigs function in MATLAB to solve the eigenvalue problems in the method. For each dimension, we

---

2 This quantity turns out to be negative for all randomly generated instances.
generated 10 instances, and the corresponding average results are adjusted in Table 3. As we see in Table 3, CGB algorithm is significantly faster than LSTRS method and it computes more accurate solutions than LSTRS algorithm in hard case (case 1 and 2) and for 40% of the generated instances in easy case. It is worth to note that both CGB and LSTRS algorithms are matrix-free while LSTRS at each iteration requires computing two eigenpairs of a \((n + 1) \times (n + 1)\) matrix and CGB algorithm requires solving a sequence of positive definite system of linear equations by conjugate gradient method.

5 Conclusions

In this paper, we have considered the problem of minimizing a general quadratic function subject to one general quadratic constraint. A conjugate gradient-based algorithm is introduced to solve the problem which is based on solving a sequence of positive definite system of linear equations by conjugate gradient method. Our computational experiments on several randomly generated test problems show that the proposed method is efficient for large sparse instances since the most expensive operations at each iteration is several matrix vector products, that are cheap when the matrix is sparse.

Acknowledgements The authors would like to thank both reviewers and the editor for their constructive comments which led to significant improvement of the paper. The authors also would like to thank Iran National Science Foundation (INSF) for supporting this research under Project No. 95843381.

References

1. Adachi, S., Iwata, S., Nakatsukasa, Y., Takeda, A.: Solving the trust region subproblem by a generalized eigenvalue problem. SIAM J. Optim. 27(1), 269–291 (2017)
2. Adachi, S., Nakatsukasa, Y.: Eigenvalue-based algorithm and analysis for nonconvex QCQP with one constraint. Math. Program. (2017). https://doi.org/10.1007/s10107-017-1206-8
3. Ben-Tal, A., den Hertog, D.: Hidden conic quadratic representation of some nonconvex quadratic optimization problems. Math. Program. 143(1–2), 1–29 (2014)
4. Crawford, C.R., Moon, Y.S.: Finding a positive definite linear combination of two Hermitian matrices. Linear Algebra Appl. 51, 37–48 (1983)
5. Conn, A.R., Gould, N.I.M., Toint, P.L.: Trust Region Methods. SIAM, Philadelphia (2000)
6. Fortin, C., Wolkowicz, H.: The trust region subproblem and semidefinite programming. Optim. Methods Softw. 19(1), 41–67 (2004)
7. Feng, J.-M., Lin, G.-X., Heu, R.-L., Xia, Y.: Duality and solutions for quadratic programming over single non-homogeneous quadratic constraint. J. Glob. Optim. 54(2), 275–293 (2012)
8. Gould, N.I., Lucidi, S., Roma, M., Toint, P.L.: Solving the trust-region subproblem using the Lanczos method. SIAM J. Optim. 9(2), 504–525 (1999)
9. Gould, N.I., Robinson, D.P., Thorne, H.S.: On solving trust-region and other regularised subproblems in optimization. Math. Program. Comput. 2(1), 21–57 (2010)
10. Guo, C.-H., Higham, N.J., Tisseur, F.: An improved arc algorithm for detecting definite Hermitian pairs. SIAM J. Matrix Anal. Appl. 31(3), 1131–1151 (2009)
11. Jegelka, S.: Private communication (2015)
12. Jian, R., Li, D.: Novel reformulations and efficient algorithms for the generalized trust region subproblem. arXiv:1707.08706 (2017)
13. Jiang, R., Li, D., Wu, B.: SOCP reformulation for the generalized trust region subproblem via a canonical form of two symmetric matrices. Math. Program. 169(2), 531–563 (2018)
14. Lancaster, P., Rodman, L.: Canonical forms for Hermitian matrix pairs under strict equivalence and congruence. SIAM Rev. 47(3), 407–443 (2005)
15. Moré, J.J., Sorensen, D.C.: Computing a trust region step. SIAM J. Sci. Stat. Comput. 4, 553–572 (1983)
16. Moré, J.J.: Generalizations of the trust region problem. Optim. Methods Softw. 2(3–4), 189–209 (1993)
17. Pong, T.K., Wolkowicz, H.: The generalized trust region subproblem. Comput. Optim. Appl. 58(2), 273–322 (2014)
18. Rendl, F., Wolkowicz, H.: A semidefinite framework for trust region subproblems with applications to large scale minimization. Math. Program. 77(1), 273–299 (1997)
19. Rojas, M., Santos, S.A., Sorensen, D.C.: A new matrix-free algorithm for the large-scale trust-region subproblem. SIAM J. Optim. 11(3), 611–646 (2001)
20. Saad, Y.: Numerical Methods for Large Eigenvalue Problems. Manchester University Press, Manchester (1992)
21. Steihaug, T.: The conjugate gradient method and trust regions in large scale optimization. SIAM J. Numer. Anal. 20(3), 626–637 (1983)
22. Salahi, M., Taati, A.: An efficient algorithm for solving the generalized trust region subproblem. Comput. Appl. Math. 37(1), 395–413 (2018)
23. Toint, P.L.: Towards an efficient sparsity exploiting Newton method for minimization. In: Duff, I.S. (ed.) Sparse Matrices and Their Uses, pp. 57–88. Academic Press, London (1981)
24. Ye, Y., Zhang, S.: New results on quadratic minimization. SIAM J. Optim. 14(1), 245–267 (2003)
25. Yuan, Y.: Recent advances in trust region algorithms. Math. Program. 151(1), 249–281 (2015)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.