Optimization Methods and Software
Vol. 19, No. 1, February 2004, pp. 41–67

THE TRUST REGION SUBPROBLEM AND SEMIDEFINITE PROGRAMMING*

CHARLES FORTIN[a,b,†] and HENRY WOLKOWICZ[a,‡]

[a]*Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada;* [b]*Department of Mathematics and Statistics, McGill University, Montréal, Québec H3A 2T5, Canada*

(Received 21 August 2002; In final form 19 March 2003)

Dedicated to Professor Jochem Zowe

The trust region (TR) subproblem (the minimization of a quadratic objective subject to one quadratic constraint and denoted TRS) has many applications in diverse areas, e.g., function minimization, sequential quadratic programming, regularization, ridge regression, and discrete optimization. In particular, it determines the step in TR algorithms for function minimization. Trust region algorithms are popular for their strong convergence properties. However, a drawback has been the inability to exploit sparsity as well as the difficulty in dealing with the so-called hard case. These concerns have been addressed by recent advances in the theory and algorithmic development. In particular, this has allowed Lanczos techniques to replace Cholesky factorizations.

This article provides an in depth study of TRS and its properties as well as a survey of recent advances. We emphasize large scale problems and robustness. This is done using semidefinite programming (SDP) and the modern primal–dual approaches as a unifying framework. The SDP framework arises naturally and solves TRS efficiently. In addition, it shows that TRS is always a well-posed problem, i.e., the optimal value and an optimum can be calculated to a given tolerance. We provide both theoretical and empirical evidence to illustrate the strength of the SDP and duality approach. In particular, this includes new insights and techniques for handling the hard case, as well as numerical results on *large* test problems.

Keywords: Trust regions; Semidefinite programming; Duality; Unconstrained minimization

1 INTRODUCTION

We are concerned with the following quadratic minimization problem:

$$\text{(TRS)}\quad q^* = \min \quad q(x) := x^{\mathrm{T}}Ax - 2a^{\mathrm{T}}x \qquad \text{s.t.} \quad \|x\| \le s.$$

Here, A is an $n \times n$ symmetric (possibly indefinite) matrix, a is an n-vector, s is a positive scalar and x is the n-vector of unknowns. All matrix and vector entries are real. This problem is

* This report is based on a longer version CORR 2002-22, [1]. Both this report and the longer version are available with URL: orion.math.uwaterloo.ca/~hwolkowi/henry/reports/ABSTRACTS.html
† E-mail: fortin@math.mcgill.ca
‡ Corresponding author. Tel.: (519) 888-4567, ext. 5589; Fax: (519) 725-5441; E-mail: hwolkowicz@uwaterloo.ca

ISSN 1055-6788 print; ISSN 1029-4937 online © 2004 Taylor & Francis Ltd
DOI: 10.1080/10556780410001647186

referred as the trust region (TR) subproblem (TRS). This problem has many applications in e.g., forming subproblems for constrained optimization [2], regularization of ill-posed problems [3], and regularization for ill-conditioned linear regression problems (called ridge regression, [4]). In particular, it is important in a class of optimization methods called *trust region methods* for minimization where, at each iteration of the method, the algorithm determines a step by (approximately) finding the minimum of a quadratic function (a local quadratic model of a given function f) restricted to a given ball of radius s. (This is called the spherical TR. We do not discuss scaled, ellipsoidal, box, or other TRs.) The radius s increases or decreases depending on how well the decrease in the quadratic model predicts the true decrease in f. The data, A and a, respectively, represent the Hessian and the gradient of the modeled function. These methods have advantages over, e.g., quasi-Newton methods. Under mild assumptions, they produce a sequence of iterates with an accumulation point that satisfies *both* first and second order necessary optimality conditions [e.g., Ref. 5]. Furthermore, if the accumulation point satisfies the second order sufficient optimality conditions, the method reduces to Newton's method locally and convergence is q-quadratic. [For more details see, e.g., Refs. 2,6.]

However, the popularity of TR methods for unconstrained minimization has lagged behind quasi-Newton methods. Numerical difficulties in standard algorithms for TRS can arise when a is (approximately) perpendicular to the eigenspace of the smallest eigenvalue of A. This is referred to as the (*near*) *hard case* in the literature. In addition, algorithms for TRS were based on the Cholesky factorization of the Hessian of the Lagrangian, thus sparsity could not always be exploited efficiently. On the other hand, algorithms such as *limited memory quasi-Newton methods* proved to be successful, e.g., Refs. [7,8].

Though TRS appears to be a simple problem, there is a long history of elegant theory and algorithms. (The recent books [2,9] contain extensive bibliographies. See also the bibliographical database for Ref. [2] at URL www.fundp.ac.be/˜phtoint/pht/trbook.bib.) In this article, we emphasize the modern primal–dual approaches. In particular, we study three methods that consider the above mentioned concerns, i.e., the dual based algorithm of Moré–Sorensen (MS) [10], the semidefinite programming (SDP) based algorithm of Rendl–Wolkowicz (RW) [11], and the *generalized Lanczos trust region* (GLTR or coincidently GLRT) *method* of Gould *et al.* [12].

The classical (MS) algorithm [10] was the first algorithm able to handle the hard case efficiently. (The algorithm of Gay [13] also treats the hard case.) We revisit and modify the RW primal–dual algorithm [11] which is based on a parametric eigenvalue problem using a Lanczos technique, SDP, and duality. It is designed specifically to handle large sparse problems; it also handles the hard case efficiently. The GLTR is the last algorithm we look at, see Ref. [12]. This algorithm uses the Lanczos procedure to obtain a restricted TRS problem with a tridiagonal matrix. This subproblem can be solved quickly using the MS algorithm.

Several other recent approaches deserve mention. The method by Sorensen [14] is similar to the RW algorithm in that it uses a parametric eigenvalue approach. The difference of convex functions (DC) method of Tao and An [15] and the method of Hager [16] are both designed to exploit sparsity. The method in Ref. [16] is similar in spirit to GLTR, i.e., they both solve a sequence of subproblems where TRS is restricted to a special Krylov subspace. The method of Ye [17] exploits a new efficient line search technique.

The main contribution of this article is the use of SDP and the modern primal–dual approach to motivate, view, and modify existing algorithms for TRS. In addition, we present a novel approach to handle the hard case using a shift of the eigenvalues and deflation. We also include numerical comparisons between the algorithms and specific examples that illustrate the performance on the hard case. In particular, we try to answer questions posed in Ref. [12] about the desired accuracy in solving the TRS within a TR minimization algorithm. We include numerical tests, completely in a MATLAB framework, on problems with dimensions of order $n = 10^5$ for the TR method, and order $n = 10^6$ for TRS problems.

1.1 Outline

We continue in Section 2 with the optimality conditions and definitions of the easy and hard cases for TRS. In particular, Section 2.1 describes the shift process that yields an equivalent well-posed convex program. That TRS is well-posed also follows using SDP, and is shown in Theorem 5.2 and Remark 5.1. In Section 3, we use duality to motivate and describe the MS algorithm. The GLTR algorithm is described in Section 4. In Section 5, we present several dual programs to TRS exploiting the strong Lagrangian duality for TRS. These provide the unifying framework for the different algorithms outlined in Section 6. The RW algorithm, with our modifications, is presented in detail in Section 7. The numerical tests appear in Section 8. Concluding remarks are given in Section 9.

2 OPTIMALITY CONDITIONS

It is known [see Refs. 18,19] that x^* is a solution to TRS if and only if

$$\begin{array}{rl} \left.\begin{array}{r} (A-\lambda^* I)x^* = a \\ A-\lambda^* I \succeq 0, \lambda^* \leq 0 \end{array}\right\} & \text{dual feasibility} \\ \|x^*\|^2 \leq s^2 & \text{primal feasibility} \\ \lambda^*(s^2 - \|x^*\|) = 0, & \text{complementary slackness} \end{array} \tag{1}$$

for some (Lagrange multiplier) λ^*. These conditions are surprising in two respects. First, the conditions characterize optimality of a possibly nonconvex problem, i.e., they are necessary and sufficient. Second, the usual second order positive semidefinite necessary conditions hold on all of $\mathbb{R}^n$ rather than just the tangent plane at the optimal point.

We have added the descriptive three phrases in Eq. (1), since this coincides with the framework in Ref. [11] and with the modern primal–dual optimization approach, though no dual program appeared in the earlier papers [18,19].

2.1 The Hard Case

If $A - \lambda^* I \succ 0$ in Eq. (1), then x^* is the unique solution to TRS (this is true generically), i.e., $x^* = (A - \lambda^* I)^{-1}a$. In general, we denote

$$x(\lambda) = (A - \lambda I)^{\dagger}a, \tag{2}$$

where $(\cdot)^{\dagger}$ denotes the Moore–Penrose generalized inverse. Singularity (or near singularity) of $A - \lambda I$ can result in difficulties in using $x(\lambda)$ as a solution, see Table I.

2.1.1 Shift, Deflation, and Robustness

First, we note that $\lambda_1(A) > 0$ implies that $\lambda^* \leq 0 < \lambda_1(A)$, i.e., the hard case (case 2) cannot hold. Second, if $\lambda^* = 0$, then $A \succeq 0$ and $\|x^*\| = \|A^{\dagger}a\| \leq s$. These two situations can be handled by our algorithm in a standard way. The following deflation technique forms the basis for our approach to handle the hard case. It shows that we can *deflate* and/or *shift* eigenspaces that are orthogonal to the linear term a and, thus, avoid the hard case. For example, we could first use Lemma 2.1 Part 3 to ensure that $A \succeq 0$. Then, if the possible hard case is detected,

TABLE I The three different cases for the trust region subproblem. We include two subcases (i) and (ii) for the hard case (case 2)

1. Easy case	*2. (a) Hard case (case 1)*	*2. (b) Hard case (case 2)*
$a \not\perp \mathcal{N}(A - \lambda_1(A)I)$ (implies $\lambda^* < \lambda_1(A)$)	$a \perp \mathcal{N}(A - \lambda_1(A)I)$ and $\lambda^* < \lambda_1(A)$	$a \perp \mathcal{N}(A - \lambda_1(A)I)$ and $\lambda^* = \lambda_1(A)$ (i) $\|(A - \lambda^* I)^\dagger a\| = s$ or $\lambda^* = 0$ (ii) $\|(A - \lambda^* I)^\dagger a\| < s, \lambda^* < 0$

we can use the shift in Lemma 2.1 (Part 2). Lemma 2.1 (Part 1) shows that performing the shift when the hard case did not exist does not cause harm. In many of our numerical tests, our heuristics detected an unconstrained problem after the shifts. This latter problem was solved using preconditioned conjugate gradients (PCG).

LEMMA 2.1 *Let $A = \sum_{i=1}^n \lambda_i(A) v_i v_i^{\mathrm{T}} = P \Lambda P^{\mathrm{T}}$ be the spectral decomposition of A, with v_i orthonormal eigenvectors and $P = [v_1\ v_2 \cdots v_n]$ an orthogonal matrix. Set the vector $\bar{a} := P^{\mathrm{T}} a$ and the sets*

$$
\begin{aligned}
S_1 &= \{i : \bar{a}_i \neq 0, \lambda_i(A) > \lambda_1(A)\}\\
S_2 &= \{i : \bar{a}_i = 0, \lambda_i(A) > \lambda_1(A)\}\\
S_3 &= \{i : \bar{a}_i \neq 0, \lambda_i(A) = \lambda_1(A)\}\\
S_4 &= \{i : \bar{a}_i = 0,\ \lambda_i(A) = \lambda_1(A)\}.
\end{aligned}
$$

For $k = 1, 2, 3, 4$: the matrices $A_k := \sum_{i \in S_k} \lambda_i(A) v_i v_i^{\mathrm{T}}$; and the ($A$-invariant subspace) orthogonal projections $P_k := \sum_{i \in S_k} v_i v_i^{\mathrm{T}}$, where $A_k = P_k = 0$, if $S_k = \emptyset$. Then the following holds.

1. *Suppose $S_3 \neq \emptyset$ (easy case), $\alpha > 0$, and $i \in S_2 \cup S_4$. Then*

$$(x^*, \lambda^*) \textit{ solves TRS}$$

iff

$$(x^*, \lambda^*) \textit{ solves TRS when } A \textit{ is replaced by } A + \alpha v_i v_i^{\mathrm{T}}.$$

2. *Let $u^* = (A - \lambda^* I)^\dagger a$ with $\|u^*\| < s$ and suppose that $i \in S_2 \cup S_4$ and $\alpha > 0$. Then*

$$(x^* = u^* + z, \lambda^*), z \in \mathcal{N}(A - \lambda^* I) \textit{ solves TRS}$$

iff

$$(x^* = u^* + z, \lambda^*), z \in \mathcal{N}(A + \alpha v_i v_i^{\mathrm{T}} - \lambda^* I) \textit{ solves TRS when } A \textit{ is replaced by } A + \alpha v_i v_i^{\mathrm{T}}.$$

3. *Let $u^* = (A - \lambda^* I)^\dagger a$. Then*

$$\textit{there exists } z \in \mathcal{N}(A - \lambda^* I) \textit{ such that } (x^*, \lambda^*),\ \textit{with } x^* = u^* + z,\ \textit{solves TRS}$$

iff

$$(u^*, \lambda^* - \lambda_1(A)) \textit{ solves TRS when } A \textit{ is replaced by } A - \lambda_1(A) I.$$

4.

$$x^* \text{ solves TRS and } v_i^{\mathrm{T}} x^* \neq 0, \text{ for some } i \in S_4$$

iff

$$\text{the hard case (case 2(ii)) holds.}$$

Proof The set S_3 can be used to define the hard case, i.e., $S_3 \neq \emptyset$ if and only if a is *not* orthogonal to $\mathcal{N}(A - \lambda_1(A)I)$ if and only if the easy case holds. Note that $S_3 = \emptyset \Rightarrow S_4 \neq \emptyset$.

Consider the equivalent problem to TRS obtained after the rotation by P^{T} and diagonalization of A:

$$\begin{aligned} (\mathrm{TRS}_P) \quad q^* = \min \quad & (P^{\mathrm{T}}x)^{\mathrm{T}}\Lambda(P^{\mathrm{T}}x) - 2\bar{a}^{\mathrm{T}}(P^{\mathrm{T}}x) = w^{\mathrm{T}}\Lambda w - 2\bar{a}^{\mathrm{T}}w \\ \text{s.t.} \quad & \|w\| \leq s, \quad w = P^{\mathrm{T}}x. \end{aligned} \tag{3}$$

Note that the P_k form a resolution of the identity, $I = \sum_{k=1}^{4} P_k$. Moreover, x^* solves TRS if and only if $w^* = P^{\mathrm{T}}x^*$ solves (TRS_P), Eq. (3). We set $w^* = P^{\mathrm{T}}x^*$ and, for $k = 1, 2, 3, 4$, $E_k = \sum_{i \in S_k} e_i e_i^{\mathrm{T}}$, $\Lambda_k = \sum_{i \in S_k} \lambda_i e_i e_i^{\mathrm{T}}$, $x_k^* := P_k x^*$, $w_k^* := E_k w^*$, where the e_i are unit vectors, and $E_k = \Lambda_k = 0$, $x_k = w_k = 0$, if $S_k = \emptyset$. In addition,

$$\Lambda = \sum_{i=1}^{4} \Lambda_k, \quad w^* = \sum_{i=1}^{4} w_k, \quad I = \sum_{i=1}^{4} E_k, \quad E_k = P^{\mathrm{T}} P_k P, \quad w_k^* = P^{\mathrm{T}} x_k^*.$$

For simplification, we prove the results for this diagonalized equivalent program.

1. Since $S_3 \neq \emptyset$, the definitions imply that the easy case holds and

   $$w^* = (\Lambda - \lambda^* I)^{-1}\bar{a} = (\Lambda + \alpha e_i e_i^{\mathrm{T}} - \lambda^* I)^{-1}\bar{a},$$

   i.e., the optimality conditions are unchanged after the replacement. This completes the proof of Item 1.
2. As in the above proof, the definitions imply

   $$P^{\mathrm{T}}u^* = (\Lambda - \lambda^* I)^{\dagger}\bar{a} = (\Lambda + \alpha e_i e_i^{\mathrm{T}} - \lambda^* I)^{\dagger}\bar{a},$$

   In this case, the optimality conditions are unchanged except for the change in the conditions for z. This completes the proof of Item 2.
3. Note that $u^* \in \mathcal{R}(A - \lambda^* I) \perp \mathcal{N}(A - \lambda^* I)$.
   Necessity: Assume that (x^*, λ^*) with $x^* = u^* + z, z \in \mathcal{N}(A - \lambda^* I)$ solves TRS. Then $w^* = (\Lambda - \lambda^* I)^{\dagger}\bar{a} + P^{\mathrm{T}}z$, $P^{\mathrm{T}}z \in \mathcal{N}(\Lambda - \lambda^* I)$. It follows from the optimality conditions that

   $$\begin{gathered} w^* = (\Lambda - \lambda^* I)^{\dagger}\bar{a} + P^{\mathrm{T}}z = P^{\mathrm{T}}u^* + P^{\mathrm{T}}z, \\ \lambda^*(\|P^{\mathrm{T}}u^*\|^2 + \|P^{\mathrm{T}}z\|^2 - s^2) = 0, \quad (\Lambda - \lambda^* I) \succeq 0. \end{gathered} \tag{4}$$

   By adding and subtracting $\lambda_1(A)$, we see that $(P^{\mathrm{T}}u^*, \lambda^* - \lambda_1(A))$ is optimal for TRS_P if we replace Λ by $\Lambda - \lambda_1(A)I$.

   Conversely, suppose that $(u^*, \lambda^* - \lambda_1(A))$ solves TRS when A is replaced by $A - \lambda_1(A)I$. Then

   $$P^{\mathrm{T}}u^* = (\Lambda - \lambda^* I)^{\dagger}\bar{a}, \quad \|u^*\| \leq s, \quad (\Lambda - \lambda_1(A)I) \succeq 0. \tag{5}$$

We can find an appropriate $z \in \mathcal{N}(A - \lambda_1(A)I)$ if needed so that $\|u^*\|^2 + \|z\|^2 = s^2$, i.e., $(x^* = u^* + z, \lambda^*)$ solves TRS. This completes the proof of Item 3.

4. Assume that $x^* = Pw^*$ solves TRS and$v_i^{\mathrm{T}} x^* \neq 0$, for some $i \in S_4$. Equivalently, $e_i^{\mathrm{T}} w^* \neq 0$, for some $i \in S_4$. From the definitions, $\bar{a}_i = 0, \forall i \in S_2 \cup S_4$. Therefore $w^* = (\Lambda - \lambda^* I)^{\dagger}\bar{a} + E_4 v$, for some $v \in \mathcal{R}^n$. The assumption implies that $E_4 v \neq 0$.

   Conversely, suppose that the hard case (case 2(ii)) holds. Then $w^* = (\Lambda - \lambda^* I)^{\dagger}\bar{a} + E_4 v$, for some $v \in \mathcal{R}^n$ with $E_4 v \neq 0$. This completes the proof of Item 4. ■

Numerically, we cannot distinguish between the hard case and the near hard case. This is handled using the following.

LEMMA 2.2 *Suppose that x^* solves TRS and $\|x^*\| = s$. Let $\varepsilon > 0$ and $v \in \mathbb{R}^n$ with $\|v\| = 1$. Let $\mu^*(\varepsilon)$ be the optimal value of TRS when a is perturbed to $a + \varepsilon v$. Then*

$$-2s\varepsilon \leq \mu^* - \mu^*(\varepsilon) \leq 2s\varepsilon.$$

Proof

$$\begin{aligned}
\mu^*(\varepsilon) &= \min_{\|x\|=s} q(x) - 2\varepsilon v^{\mathrm{T}} x \\
&\geq \min_{\|x\|=s} q(x) + \min_{\|x\|=s} -2\varepsilon v^{\mathrm{T}} x \\
&= \mu^* - 2\varepsilon s.
\end{aligned}$$

This proves the right-hand-side inequality.

Since x^* is optimal for TRS and on the boundary of the ball, we get

$$\begin{aligned}
\mu^*(\varepsilon) &\leq \mu^* - 2\varepsilon v^{\mathrm{T}} x^* \\
&\leq \mu^* + 2\varepsilon s.
\end{aligned}$$

■

The literature often labels the hard case (case 2) as an ill-posed or degenerate problem, e.g., Refs. [12,16]. Adding a norm constraint to an ill-posed problem is a well-known regularization technique, e.g., Ref. [3]. Thus, it would appear to be contradictory for TRS to be an ill-posed problem. In fact, we can orthogonally diagonalize the quadratic form; and, we note that the symmetric eigenvalue problem has a condition number of 1 [20]. Then, TRS can be shown to be equivalent to a linearly constrained convex programming problem, see Ref. [21], a problem that can be solved efficiently and robustly.

The following lemma and example illustrate that TRS is always a stable convex program. This also follows from the equivalent SDP dual pair in Theorem 5.2.

LEMMA 2.3 *Suppose that the hard case* (*case* 2 (ii)) *holds for TRS. Let $u^* = (A - \lambda^* I)^{\dagger} a$, and $x^* = u^* + z$ be a solution of TRS with $z \in \mathcal{N}(A - \lambda^* I)$. Then $z \neq 0, \lambda_1(A) \leq 0$ and TRS is equivalent to the following stable convex program*

$$\begin{array}{llll}
(\mathrm{TRS}_s) & q_s^* := & \min & q_s(x) := x^{\mathrm{T}}(A - \lambda_1(A)I)x - 2a^{\mathrm{T}}x + s^2\lambda_1(A) \\
& & \text{s.t.} & \|x\| \leq s.
\end{array} \tag{6}$$

The equivalence is in the sense that the optimal values satisfy $q^* = q_s^*$; and (x^*, λ^*) solves TRS if and only if $(x^*, 0)$ solves TRS$_s$.

Proof Since the hard case (case 2 (ii)) holds, we get $z \neq 0$ and $\lambda_1(A) = \lambda^* \leq 0$. From Lemma 2.1, Item 3, we get $(u^*, 0)$ solves TRS$_s$. We can then add $z \in \mathcal{N}(A - \lambda^* I)$ to get $\|u^* + z\| = s$. ■

Convex programs for which Slater's constraint qualification holds are called *stable*, e.g., Refs. [22,23]. They are equivalent to convex programs for which the optimal dual solutions form a convex compact set which further implies that the perturbation function (optimal value function subject to linear perturbations in the data) is convex and Lipschitz continuous. In our case we have the additional strong linear independence CQ, which implies that the optimal dual solution is unique and the perturbation function is differentiable. We also have a compact convex feasible set [see e.g., Refs. 22,23].

Example 2.1 (Hard Case, Case 2(ii)) Let

$$A = \begin{bmatrix} 1+\gamma & 0 \\ 0 & -1+\delta \end{bmatrix}, \quad a = \begin{bmatrix} 2+\alpha \\ \beta \end{bmatrix}, \quad s = \sqrt{2},$$

where $\alpha, \beta, \gamma, \delta$ are perturbations in the data. First suppose that the perturbations are all 0. Then the hard case (case 2(ii)) holds; the optimal Lagrange multiplier is $\lambda^* = \lambda_1(A) = -1$; and the best least squares solution is

$$\bar{x} = (A - \lambda^* I)^{\dagger} a = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

with $\|\bar{x}\| = 1 < s$. The optimal solution is obtained from

$$x^* = x^*(0) = \bar{x} + \begin{bmatrix} 0 \\ \pm 1 \end{bmatrix} = \begin{bmatrix} 1 \\ \pm 1 \end{bmatrix}. \tag{7}$$

For (small) nonzero perturbations, the optimal Lagrange multiplier λ^* is still unique and $-1 + \delta$ is the smallest eigenvalue. If $\beta = 0$, then the hard case still holds; the optimum is obtained from $\lambda^* = -1 + \delta$; and

$$x^* = x^*(\alpha, \gamma, \delta) = \begin{bmatrix} \dfrac{2+\alpha}{1+\gamma-\lambda^*} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \varepsilon \pm 1 \end{bmatrix},$$

where ε is chosen to obtain $\|x^*\| = s$, e.g., with $+1$

$$(1+\varepsilon)^2 + \left(\frac{2+\alpha}{2+\gamma-\delta}\right)^2 = s^2 = 2.$$

Depending on the choice of sign, these solutions converge to a solution in Eq. (7), as the perturbations converge to 0. Moreover, a Taylor series expansion shows that $\|x^*(0) - x^*(\alpha, \gamma, \delta)\| \leq 2(|\alpha| + |\gamma| + |\delta|)$ for small perturbations, i.e., we have a bounded condition number.

If $\beta \neq 0$, then we have the easy case. The unique optimal Lagrange multiplier $\lambda^* < -1 + \delta$ and the unique optimum is obtained from

$$x^* = \begin{bmatrix} \dfrac{2+\alpha}{1+\gamma-\lambda^*} \\ \dfrac{\beta}{-1+\delta-\lambda^*} \end{bmatrix},$$

where λ^* satisfies the positive definiteness condition as well as $\|x^*\| = s$. This implies that

$$\left(\frac{2+\alpha}{1+\gamma-\lambda^*}\right)^2 + \left(\frac{\beta}{-1+\delta-\lambda^*}\right)^2 = 2.$$

Since $\lambda^* \to -1$ and $(2+\alpha)/(1+\gamma-\lambda^*) \to 1$, as the perturbations go to 0, we see that the optimal solutions converge appropriately again, i.e., $x^*(\alpha, \beta, \gamma, \delta) \to x^*(0, 0, 0, 0)$.

3 A DUAL ALGORITHM: THE MORÉ–SORENSEN ALGORITHM

We motivate the MS algorithm using duality and illustrate how SDP arises naturally from this setting. The *Lagrangian dual* of TRS is

$$\begin{aligned} q^* = \nu^* &:= \max_{\lambda \le 0} \min_x x^{\mathrm{T}}(A - \lambda I)x - 2a^{\mathrm{T}}x + \lambda s^2 \\ &= \max_{\lambda \le 0} h(\lambda) \end{aligned}$$

where the *Lagrangian* is $L(x, \lambda) := x^{\mathrm{T}}(A - \lambda I)x - 2a^{\mathrm{T}}x + \lambda s^2$ and the *dual functional* is $h(\lambda) := \min_x L(x, \lambda)$. Strong duality ($q^* = \nu^*$ and dual attainment [24]) holds. Since the inner minimization is unconstrained, we have a *hidden semidefinite constraint* that the Hessian, $\nabla^2 L(x, \lambda) \succeq 0$, is positive definite. The dual functional h is concave with domain within the semidefinite constraint.

Therefore, we can replace TRS with the simpler root finding problem

$$h'(\lambda) = 0 \tag{8}$$

(if h is differentiable), with the restrictions:

$$\lambda \le 0, \quad \nabla^2 L(x, \lambda) = A - \lambda I \succeq 0. \tag{9}$$

We see that semidefiniteness (convexity of the Lagrangian) arises naturally from the duality setting. Note that $h'(\lambda) = s^2 - x(\lambda)^{\mathrm{T}}x(\lambda)$, when it exists. If we use the optimality conditions in Eq. (1) with the descriptive phrases, then the MS algorithm maintains dual feasibility and complementary slackness, while trying to attain primal feasibility (cf. the dual simplex method for linear programming).

Though Newton's method has asymptotic q-quadratic convergence, the Newton step does not take into account the semidefinite restrictions, which can result in many backtracking steps. This illustrates the weakness of a dual method compared to a primal–dual method.

The main work in the iterations is a Cholesky factorization used in the evaluation of the derivatives and the Newton step for λ, as well as in the safeguarding and updating scheme that produces either a point λ from which quadratic convergence ensues, or reduces the interval of uncertainty for the optimal λ. If the possible hard case is detected, optimality is reached by taking primal steps to the boundary of the ball. In both cases, given two parameters σ_1 and σ_2 in (0, 1), the algorithm terminates in a finite number of iterations with an approximate solution $\bar{x}$ which satisfies

$$q(\bar{x}) - q^* \leq \sigma_1(2-\sigma_1)\max\{\|q^*\|, \sigma_2\}, \quad \|\bar{x}\| \leq (1+\sigma_1)\Delta. \tag{10}$$

One additional innovation makes the MS algorithm fast. Assume $A - \lambda I \succ 0$. There are disadvantages in applying Newton's method to find a root of the function h or equivalently of $\psi(\lambda) := \|x(\lambda)\| - s$. For $\lambda < \lambda_1(A)$ and close to $\lambda_1(A)$, the orthogonal diagonalization of A shows that

$$\psi(\lambda) = \|Q(\Lambda - \lambda I)^{-1}Q^{\mathrm{T}}a\| - s \approx \frac{c_1}{\lambda_1(A) - \lambda} + d,$$

for some constants $c_1 > 0, d$. This function is highly nonlinear for values of λ near $\lambda_1(A)$, which equates to slow convergence for Newton's method. Moré–Sorensen solve the equivalent so-called secular equation

$$\phi(\lambda) := \frac{1}{s} - \frac{1}{\|x(\lambda)\|} = 0. \tag{11}$$

[See e.g., Refs. 25–28]. The rational structure of $\|x(\lambda)\|^2$, shows that this function is less nonlinear, i.e.,

$$\phi(\lambda) \approx \frac{1}{s} - \frac{\lambda_1(A) - \lambda}{c_2},$$

for some $c_2 > 0$. Therefore, Newton's method applied to this function will be more efficient. One can also show $\phi(\lambda)$ is a convex function strictly increasing on $(-\infty, \lambda_1(A))$ [see Ref. 10, p. 562 for the details].

3.1 Handling the Hard Case in Moré–Sorensen Algorithm

From Fig. 1, we get the following indicator of the easy case for TRS.

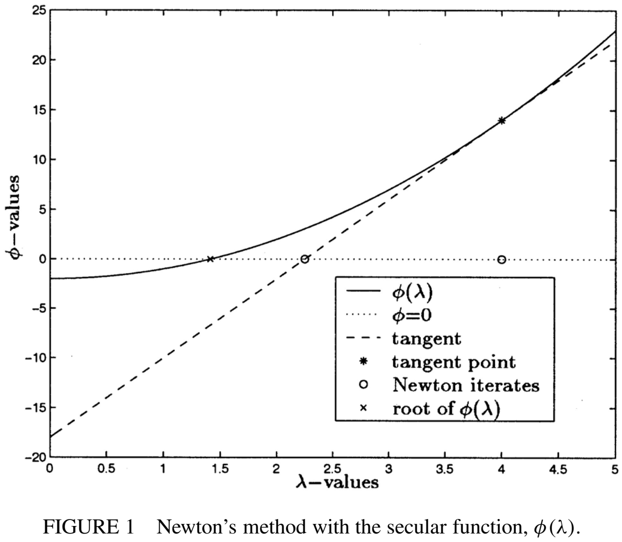

FIGURE 1 Newton's method with the secular function, $\phi(\lambda)$.

LEMMA 3.1 *Suppose that* $\lambda_0 \leq \min\{0, \lambda_1(A)\}$ *and* $\phi(\lambda_0) > 0$ (*equivalently* $\|x(\lambda_0)\| > s$). *Then the hard case* (*case* 2) *cannot occur for TRS*, i.e. $\lambda^* < \lambda_1(A)$.

However, Fig. 1 also shows that Newton's method can provide a poor prediction for λ^* if $\phi(\lambda) < 0$ and λ^* is close to 0 and/or $\lambda_1(A)$. This would result in many backtracking steps to find the above λ_0 (each of which involves an attempted Cholesky factorization) and would make the algorithm inefficient. As discussed above in Section 2.1, in the hard case (case 2), a solution to TRS can be obtained by first finding a solution $x(\lambda_1(A))$ to the system

$$(A - \lambda_1(A)I)x = a \tag{12}$$

with $\|x\| \leq s$. If strict inequality holds, $\|x\| < s$, then we need an eigenvector $z \in \mathcal{N}(A - \lambda_1(A)I)$, and $\tau \in \mathbb{R}$, such that $\|x(\lambda_1(A)) + \tau z\| = s$, i.e.,

$$x^* = x(\lambda_1(A)) + \tau z \tag{13}$$

satisfies the optimality conditions. The following lemma by MS [25] is the key to implementing this idea numerically.

LEMMA 3.2 (Primal step to the boundary) *Let* $0 < \sigma < 1$ *be given and suppose that*

$$A - \lambda I = R^{\mathrm{T}}R, \quad (A - \lambda I)x = a, \quad \lambda \leq 0. \tag{14}$$

Let $z \in \mathbb{R}^n$ *satisfy*

$$\|x + z\|^2 = s^2 \tag{15}$$

and

$$\|Rz\|^2 \leq \sigma(\|Rx\|^2 - \lambda s^2). \tag{16}$$

Then

$$|q(x + z) - q(x^*)| \leq \sigma|q(x^*)| \tag{17}$$

where x^* *is optimal for TRS.*

We will get back to this lemma in Section 6.1, where we show that this lemma is measuring a duality gap in a primal–dual pair of SDPs. Note that Eqs. (14) and (15) guarantee the dual and primal feasibility constraints in the optimality conditions (1). Comparisons with the RW algorithm are included in Section 7.

4 THE GENERALIZED LANCZOS TRUST REGION ALGORITHM

As mentioned above, current attempts to solve TRS focus on exploiting sparsity. The Cholesky factorization can be a bottleneck in the MS algorithm for sparse problems without special structure. The GLTR algorithm involves a Lanczos tridiagonalization of the matrix A, which then allows for fast Cholesky factorizations. The method requires only matrix–vector multiplications (MVM) and exploits sparsity in A. The method solves a *sequence* of restricted problems

$$\begin{array}{ll} \min & q(x) \\ \text{s.t.} & \|x\| \le s \\ & x \in S \equiv \mathcal{K}_k, \end{array} \tag{18}$$

where $\mathcal{K}_k := \operatorname{span}\{a, Aa, A^2a, A^3a, \ldots, A^ka\}$ are specially chosen (Krylov) subspaces of $\mathbb{R}^n$. (A similar approach based on Krylov subspaces is presented in Ref. [16].) The way S is chosen is inspired by the Steihaug algorithm of Ref. [29] [see also Refs. 12,30]. The authors use heuristics for the MS algorithm that find a starting value of λ on the correct side of λ^* to ensure asymptotic q-quadratic convergence of the Newton iteration λ^*.

4.1 Handling the Near Hard Case in Generalized Lanczos Trust Region Algorithm

If the near hard case (case 2) occurs for the tridiagonal subproblem, then finding a λ that guarantees the q-quadratic convergence may be difficult and time consuming. (The GLTR algorithm fails if the hard case (case 2) occurs.) In addition, ill-conditioning will slow down both the MS algorithm and therefore the GLTR method [see Ref. 12, p. 515].

Further details are included within the semidefinite framework in Section 6.2.

As mentioned above, the method in Ref. [16] is similar in spirit to GLTR. The main difference is the choice of Krylov subspaces $\mathcal{K}_k$. The vectors include the direction found from a sequential quadratic programming (SQP) model for TRS.

5 DUALITY AND A SEMIDEFINITE FRAMEWORK FOR THE TRUST REGION SUBPROBLEM

Duality plays a central role in designing optimization algorithms, as illustrated in our motivation for the MS algorithm in Section 3. In this section, we focus our attention on different dual programs associated with TRS. We will further see below the role played by duality in both the MS and GLTR algorithms, i.e., they are both *dual based*, and so they exhibit slow convergence and lack of robustness, characteristics of dual algorithms.

For simplicity, we restrict ourselves to the equality TRS, i.e., we minimize over the sphere rather than the ball. (To extend to the standard TRS we would need to add the dual constraint $\lambda \le 0$.) Precisely, consider the slightly different problem

$$\begin{array}{llll} (\text{TRS}_=) & q^* = \min & q(x) \\ & \text{s.t.} & \|x\| = s. \end{array} \tag{19}$$

5.1 Lagrangian Duality and Semi Definite Programming

From Ref. [24], we know that strong Lagrangian duality holds for TRS$_=$, i.e.,

$$q^* = \nu^* := \max_{\lambda} \min_{x} L(x, \lambda). \tag{20}$$

Since $L(x, \lambda)$ is a quadratic, the inner min problem is unbounded below unless the hidden constraints,

$$A - \lambda I \succeq 0, \quad a \in \mathcal{R}(A - \lambda I), \tag{21}$$

hold. (This can be seen by moving in a direction of an eigenvector corresponding to a negative eigenvalue or, if $A - \lambda I \succeq 0$, $a \notin \mathcal{R}(A - \lambda I)$, then moving in a direction $d \in \mathcal{N}(A - \lambda I)$ such that $d^{\mathrm{T}} a > 0$.) This yields the equivalent dual problem

$$q^* = \max_{\substack{A-\lambda I \succeq 0, \\ a \in \mathcal{R}(A-\lambda I)}} \min_x L(x, \lambda).$$

The inner minimum is attained if bounded. We get the following.

THEOREM 5.1 [24] *The Lagrangian dual for TRS$_=$ is*

$$\text{(D)} \quad q^* = \sup_{A-\lambda I \succ 0} h(\lambda), \tag{22}$$

where

$$h(\lambda) := \lambda s^2 - a^{\mathrm{T}}(A - \lambda I)^{\dagger} a,$$

where h is a concave function on the feasible set. In the easy case and hard case (*case* 1), *the* sup *can be replaced by a* max.

COROLLARY 5.1 *The Lagrangian dual for TRS is equivalent to*

$$\text{(D)} \quad q^* = \sup_{\substack{A-\lambda I \succ 0 \\ \lambda \le 0}} h(\lambda). \tag{23}$$

In the easy case and hard case (*case* 1), *the* sup *can be replaced by a* max.

5.2 Unconstrained and Linear Duals

We now present an unconstrained concave maximization problem and a pair of linear SDPs, all of which are equivalent to TRS$_=$, see Ref. [11] for the details. Define

$$D(t) = \begin{pmatrix} t & -a^{\mathrm{T}} \\ -a & A \end{pmatrix}, \quad k(t) := (s^2 + 1)\lambda_1(D(t)) - t. \tag{24}$$

Then we have the following unconstrained dual problem for TRS$_=$:

$$q^* = \max_t k(t). \tag{25}$$

It is well known that $\lambda_1(D(\cdot))$ is a concave function, and therefore $k(\cdot)$ is concave as well. Thus, using duality, TRS$_=$ is equivalent to an *unconstrained* concave maximization problem in *one* variable. We can also rewrite Eq. (25) in the following way so that it becomes a linear semidefinite program:

$$\max_{D(t) \succeq \lambda I} (s^2 + 1)\lambda - t. \tag{26}$$

Equivalently,

$$\begin{aligned} q^* = \max \quad & (s^2 + 1)\lambda - t \\ \text{s.t.} \quad & \lambda I - t E_{00} \preceq D(0), \end{aligned} \tag{27}$$

where E_{00} is the zero matrix except for 1 in the top left corner. Because Slater's constraint qualification holds for this problem, one can take the Lagrangian dual and get a semidefinite equivalent for TRS$_=$

$$\begin{aligned} q^* = \min \quad & \operatorname{trace} D(0)Y \\ \text{s.t.} \quad & \operatorname{trace} Y = s^2 + 1 \\ & -Y_{00} = -1 \\ & Y \succeq 0. \end{aligned} \tag{28}$$

THEOREM 5.2 *The three programs*: (25), (27), (28), *are equivalent to TRS$_=$. Moreover, the Slater constraint qualification and strict complementarity hold for the primal–dual SDP pair* (27) *and* (28).

Proof The equivalence with TRS$_=$ was already shown above.

That Slater's constraint qualification holds is clear, i.e., choose λ and Y appropriately.

Now suppose that λ, t, Y are optimal for the SDP pair (27) and (28). Then $Z := D(t) - \lambda I$ is positive semidefinite and singular. Let k be the multiplicity of $\lambda_1(D(t))$ and $y_1, \dots, y_k$ be an orthonormal basis for its eigenspace. Set $V = [y_1 \ \cdots \ y_k]$ and redefine $Y \leftarrow Y + VV^{\mathrm{T}}$. We can scale *DYD* using a diagonal matrix D to guarantee that Y is feasible for Eq. (28). By construction

$$ZY = 0, \quad Z + Y \succ 0.$$

COROLLARY 5.2 *The SDP* (28) *has a rank one optimal solution Y^*.*

Proof Let x^* be an optimum for TRS and

$$y^* = \begin{pmatrix} 1 \\ x^* \end{pmatrix}, \quad Y^* = y^*(y^*)^{\mathrm{T}}.$$

Remark 5.1 The primal–dual linear SDP pair can be solved to any desired accuracy in polynomial time, see e.g. Ref. [31]. This emphasizes that TRS is a well-posed problem.

6 SEMIDEFINITE FRAMEWORK

From above (Theorem 5.2), we saw that TRS$_=$ is equivalent to a primal–dual pair of SDPs which could be solved using primal–dual interior-point methods. These methods have revolutionized our view of optimization during the last 15 years. In particular, path-following methods have proven to be an efficient approach for many classes of optimization problems. The main idea for these methods is to apply Newton's method to a perturbation of the primal–dual optimality conditions. Using *both* the primal and dual equations and variables makes for efficient, robust algorithms. (The recent books [32,33] describe this approach for both linear and semidefinite programming.) Often, compromises have to be made to deal with large sparse problems. In particular, for SDP one often uses a dual based method to exploit sparsity, since the primal variable is often large and dense, see e.g., Refs. [34,35].

We previously motivated the MS algorithm using duality. We now describe the MS and GLTR algorithms using SDP and the modern primal–dual approach. We see that compromises are made and a full primal–dual path-following method is not used.

6.1 A Semidefinite Framework for the Moré–Sorensen Algorithm

For simplicity, we consider the case $a \neq 0$. We follow Section 5.2 [see also Ref. 11] and use the following pair of SDP dual programs. (D) is the dual of TRS and (DD) is the dual of (D):

$$\text{(D)} \quad q^* = \sup_{A-\lambda I \succ 0} h(\lambda) \tag{29}$$

$$\begin{aligned} \text{(DD)} \quad q^* = \inf \quad & h(\lambda) + \operatorname{trace}(X(A-\lambda I)) \\ \text{s.t.} \quad & s^2 - a^{\mathrm{T}}((A-\lambda I)^{\dagger})^2 a - \operatorname{trace} X = 0 \\ & \lambda < \lambda_l(A) \\ & \operatorname{trace} X \leq s^2 \\ & X \succeq 0, \end{aligned} \tag{30}$$

where $\lambda_l(A)$ is the smallest eigenvalue such that $a \not\perp \mathcal{N}(A - \lambda_l(A)I)$.

In the easy case and the hard case (case 1), we use the dual program (D). The supremum in Eq. (29) is attained at the stationary point $\lambda^* \in (-\infty, \lambda_1(A))$,

$$h'(\lambda^*) = -a^{\mathrm{T}}((A-\lambda^* I)^{-1})^2 a + s^2 = -\|(A-\lambda^* I)^{-1} a\|^2 + s^2 = 0.$$

Newton's method is applied to the equivalent root finding problem $\phi(\lambda) = 0$. Safeguarding guarantees that λ^* stays in the proper interval. Though Newton's method guarantees q-quadratic convergence, this may only happen after many Newton and backtracking steps. The semidefinite constraint is not used explicitly in choosing the Newton direction or the step length.

Things are different in the hard (or near hard) case (case 2), i.e., this is the case when the current estimates satisfy primal feasibility $\|x(\lambda)\| < s$. In this case, MS uses a dual–primal approach. Given such a λ, we now use (DD) to reduce the duality gap, $\operatorname{trace}(X(A-\lambda I))$, between (D),(DD); and we simultaneously reduce the objective value of TRS. To do this we find z to move to the boundary (i.e., the primal step is $\|x+z\|^2 = s^2$). The SDP (30) suggests how such a z should be chosen.

$$\begin{aligned} q(x+z) &= (x+z)^{\mathrm{T}} A(x+z) - 2a^{\mathrm{T}}(x+z) + \lambda s^2 - \lambda\|x+z\|^2 \\ &= \lambda s^2 + x^{\mathrm{T}}(A-\lambda I)x + 2x^{\mathrm{T}}(A-\lambda I)z + z^{\mathrm{T}}(A-\lambda I)z - 2a^{\mathrm{T}}x - 2a^{\mathrm{T}}z \\ &= h(\lambda) + z^{\mathrm{T}}(A-\lambda I)z \\ &= h(\lambda) + \operatorname{trace}(zz^{\mathrm{T}}(A-\lambda I)). \end{aligned}$$

To summarize, note that in the MS algorithm, $\|Rz\|^2 = z^{\mathrm{T}}(A-\lambda I)z$. Therefore, when a z is found such that $\|x+z\|^2 = s^2$ and $\|Rz\|$ is small, the algorithm is trying to reduce the duality gap between Eqs. (29) and (30), while maintaining feasibility for Eq. (30).

6.2 A Semidefinite Framework for the Generalized Lanczos Trust Region Method

As in the MS algorithm, we now show that the GLTR algorithm can also be explained using the Lagrangian dual Eq. (29). Here, we outline how their stopping criterion is measuring the duality gap between $\text{TRS}_=$ and this Lagrangian dual. (A detailed discussion is given in Ref. [1].)

Each iteration of the algorithm returns a feasible point x_k for TRS and a corresponding Lagrange multiplier λ_k which are optimal for the subproblem (18). The algorithm stops when stationarity is satisfied up to a tolerance, i.e., when

$$\|(A-\lambda_k I)x_k - a\| \tag{31}$$

becomes small. When the λ_k are feasible for Eq. (29) and bounded, then it is possible to show

$$q(x_k) - h(\lambda_k) = \mathrm{O}(\|(A - \lambda_k I)x_k - a\|^2),$$

i.e., the duality gap is bounded by a quantity proportional to the square of Eq. (31). Therefore, convergence of Eq. (31) to zero implies a zero duality gap.

Though the GLTR algorithm is a primal algorithm, since simpler primal problems are solved to approximate the solution to TRS, the strength of the approximation is directly linked to the duality gap between TRS and the dual problem (29).

7 THE RENDL–WOLKOWICZ ALGORITHM, MODIFIED

This algorithm both exploits the sparsity of A and handles the hard case. The algorithm is based on using various primal and dual steps to reduce the interval of uncertainty for the optimum (maximum) of the unconstrained dual program (25). We exploit the properties of the eigenvalues and eigenvectors of the parametric matrix $D(t)$. Many ideas from the MS algorithm are transformed to the large sparse case, e.g., the primal step to the boundary and the secular function. We also exploit information from the primal–dual pair of linear SDPs (27) and (29). We outline the algorithm with a flowchart in Section 7.2. In Section 7.3 we list new heuristics that take advantage of the structure of $k(\cdot)$, accelerate convergence, and facilitate the handling of the hard case.

7.1 Three Useful Functions

Graphs, illustrating the properties of these functions, appear in Ref. [13].

7.1.1 $k(t) = (s^2 + 1)\lambda_1(D(t)) - t$

This is the function that we (implicitly) maximize to solve TRS, see Eq. (25).
Since

$$\lim_{t\to\infty} \lambda_1(D(t)) = \lambda_1(A) \text{ and } \lim_{t\to-\infty} (\lambda_1(D(t)) - t) = 0,$$

the asymptotic behavior of $k(t)$ is

$$\begin{aligned} k(t) &\sim (s^2 + 1)\lambda_1(A) - t, && \text{as } t \to \infty \text{ (linear with slope } -1), \\ k(t) &\sim s^2 t, && \text{as } t \to -\infty \text{ (linear with slope } s^2), \end{aligned}$$

i.e., $k(t)$ is linear as $|t| \to \infty$. Since $\lambda_1(D(t))$ is concave, so is $k(t)$. In the easy case, the function is differentiable and strictly concave. In the hard case, loss of differentiability occurs when the multiplicity of the smallest eigenvalue for $\lambda_1(D(t))$ changes. The following theorem, based on Ref. [11, Proposition 8, Lemmas 9 and 15], tells us when this happens.

THEOREM 7.1 *Let $A = P\Lambda P^{\mathrm{T}}$ be an orthogonal diagonalization of A. Let $\lambda_1(A)$ have multiplicity i and define*

$$t_0 := \lambda_1(A) + \sum_{j\in\{k|(P^T a)_k\neq 0\}} \frac{(P^{\mathrm{T}}a)_j^2}{\lambda_j(A) - \lambda_1(A)}.$$

Then:

1. *In the easy case, for all* $t \in \mathbb{R}$, $\lambda_1(D(t)) < \lambda_1(A)$ *and* $\lambda_1(D(t))$ *has multiplicity* 1.
2. *In the hard case*:
   (a) *for* $t < t_0$, $\lambda_1(D(t)) < \lambda_1(A)$ *and* $\lambda_1(D(t))$ *has multiplicity* 1;
   (b) *for* $t = t_0$, $\lambda_1(D(t)) = \lambda_1(A)$ *and* $\lambda_1(D(t))$ *has multiplicity* $1 + i$;
   (c) *for* $t > t_0$, $\lambda_1(D(t)) = \lambda_1(A)$ *and* $\lambda_1(D(t))$ *has multiplicity* i.

COROLLARY 7.1 *In the hard case, for* $t \geq t_0$, $k(t) = (s^2 + 1)\lambda_1(A) - t$.

Note that the maximum $t^* \leq t_0$. Difficulties with differentiability arise when t^* is close to t_0.

7.1.2 $k'(t) = (s^2 + 1)y_0(t)^2 - 1$

Maximizing the concave function $k(t)$ is equivalent to finding $0 \in \partial k(t)$ (subgradient). Recall that $y(t)$ is a normalized eigenvector for $\lambda_1(D(t))$ and $y_0(t)$ is its first component. If

$$y(t) = \begin{pmatrix} y_0(t) \\ x(t) \end{pmatrix},$$

then, in the differentiable case,

$$\frac{1}{y_0(t)^2}\|x(t)\|^2 = \frac{1 - y_0(t)^2}{y_0(t)^2} = s^2 \quad \text{if and only if } k'(t) = 0,$$

i.e., this is equivalent to primal feasibility (cf. $h'(\lambda) = 0$ in MS algorithm). To obtain conditions for $y_0(t) \neq 0$, we use the following from Ref. [11, Lemma 12, Lemma 15].

THEOREM 7.2 *Let* $y(t)$ *be a normalized eigenvector for* $\lambda_1(D(t))$ *and let* $y_0(t)$ *be its first component. Then*:

1. *In the easy case*: *for* $t \in \mathbb{R}$, $y_0(t) \neq 0$.
2. *In the hard case*:
   (a) *for* $t < t_0$: $y_0(t) \neq 0$;
   (b) *for* $t > t_0$: *there exists a basis of eigenvectors for the eigenspace of* $\lambda_1(D(t))$ *such that each eigenvector in the basis has a zero first component* ($y_0(t) = 0$) *and the vector composed of the last n components is an eigenvector for* $\lambda_1(A)$;
   (c) *for* $t = t_0$: *there exists a basis of eigenvectors for the eigenspace of* $\lambda_1(D(t_0))$, *such that one eigenvector of this basis*, ω, *has a nonzero first component* ($\omega_0 \neq 0$) *and each of the other eigenvectors in the basis has a zero first component* ($y_0(t) = 0$) *and the vector composed of the last n components is an eigenvector for* $\lambda_1(A)$.

It is known that the function $\lambda_1(D(t))$ is differentiable at points where the multiplicity of the eigenvalue is 1. Its derivative is given by $y_0(t)^2$, where $y(t)$ is a normalized eigenvector for $\lambda_1(D(t))$, i.e., $\|y(t)\| = 1$ [see Ref. 36]. Therefore, Theorems 7.1 and 7.2 yield the following.

COROLLARY 7.2

1. *In the easy case*: $k(\cdot)$ *is differentiable and* $k'(t) = (s^2 + 1)y_0(t)^2 - 1$.
2. *In the hard case*:
   (a) *for* $t < t_0$, $k(\cdot)$ *is differentiable and* $k'(t) = (s^2 + 1)y_0(t)^2 - 1$;
   (b) *for* $t = t_0$, $k(\cdot)$ *is nondifferentiable and the directional derivatives from the left and right are, respectively*, $k'_-(t_0) = (s^2 + 1)\omega_0^2 - 1$ *and* $k'_+(t_0) = -1$;
   (c) *for* $t > t_0$, $k(\cdot)$ *is differentiable and* $k'(t) = -1$.

The structure of the eigenvectors along with the shift and deflation Lemma 2.1 can be used to avoid the hard case, i.e., if $y_0(t)$ is *small*, then we can deflate using the corresponding eigenvector. Lemma 2.2 shows that the deviation from the original problem is *small*.

7.1.3 $\psi(t) = \sqrt{s^2+1} - 1/y_0(t)$

Solving $\psi(t) = 0$ is equivalent to solving $k'(t) = 0$. The advantage is that $\psi(t)$ is less nonlinear (cf. replacing h' with ϕ in the MS algorithm). It can be shown that $\psi(t)$ is strictly decreasing and converges to $\sqrt{s^2+1} - 1$ as $t \to -\infty$. In the easy case, $\psi(t)$ goes to $-\infty$ as $t \to \infty$. In the hard case, $\psi(t)$ is undefined for $t > t_0$.

7.2 Flowchart

In the sequel, the set of $\{t: k'(t) < 0\}$ is referred to as the *easy side* and its complement as the *hard side*. The details in the flowchart follow in Section 7.3.

- INITIALIZATION:
  1. Compute $\lambda_1 = \lambda_1(A)$ and corresponding eigenvector v_1. If $\lambda_1(A) < 0$, shift $A \leftarrow A - \lambda_1(A)I$. ($\lambda_1(A)\|x^*\|^2$ is added back to the objective value at the end.)
     If $a^{\mathrm{T}}v_1$ is small (near hard case), then deflate, i.e., set
     $$\mathcal{Y} = \{y_1\} = \left\{\begin{pmatrix} 0 \\ v_1 \end{pmatrix}\right\}.$$
     .
  2. Obtain bounds on q^*, λ^*, and t^*.
     If $\lambda_1 > 0$, EXIT if the optimum is an unconstrained minimizer.
  3. Initialize parameters and the stopping criteria; this is based on the optimality conditions, duality gap, and intervals of uncertainty.
- ITERATION LOOP: (until convergence to the desired tolerance or until we find the solution is the unconstrained minimizer)
  1. FIND a NEW VALUE of t.
     (a) Set t using Newton's method on $k(t) - M_t = 0$ if the iterate falls into the interval of uncertainty for t; otherwise set it to the the midpoint (default) of the interval of uncertainty.
     (b) If points from the hard and easy side are available:
        i. Do TRIANGLE INTERPOLATION (Update upper bound on q^* and set t, if possible.)
        ii. Do VERTICAL CUT (Update lower or upper bound for interval of uncertainty for t.)
     (c) Do INVERSE INTERPOLATION (Set t, if possible)
  2. UPDATE
     (a) With new t, compute (with restarts using a previous eigenvector) $\lambda = \lambda_1(D(t))$ and corresponding eigenvector y with $y_0 \geq 0$. (Use y orthogonal to the vectors in $\mathcal{Y}$, if possible.)
     (b) If $\lambda > 0$ and $y_0^2 > 1/(s^2+1)$ then the solution is the unconstrained minimizer. Use Conjugate Gradients and EXIT.
     (c) Update bounds on interval of uncertainty of q^*.
     (d) i. If y_0 is small, then deflate, i.e., add y to $\mathcal{Y}$.

   ii. elseif t is on the easy side, update parameters. Take a primal step to the boundary if a hard side point exists.
   iii. elseif t is on the hard side, update parameters. Take a primal step to the boundary from this hard side point.
  (e) Save new bounds and update stopping criteria.
- END LOOP

7.3 Techniques Used in the Algorithm

7.3.1 Newton's Method on $k(t) - M_t = 0$

We use the upper and lower bounds on $k(t)$ and the Newton type method presented in Ref. [37]. Note that Newton's method applied to $k(t) - M_t = 0$ at t_c yields

$$t_+ = t_c - \frac{k(t_c) - M_t}{k'(t_c)} = \frac{(s^2+1)(t_c y_0^2(t_c) - \lambda_1(D(t_c))) + M_t}{(s^2+1)y_0(t_c)^2 - 1}.$$

One advantage of this method over solving $k' = 0$ is that the second derivative k'' is not needed. We use this iteration for appropriate choices of M_t in cases where the inverse iteration on ψ fails, i.e., if the hard case holds.

7.3.2 Triangle Interpolation

Given $k(t)$, if we have values of t from the easy and hard sides, t_e and t_h, then we try to find a better approximation t_{new} to the maximum of $k(t)$ using a technique we call *triangle interpolation*, i.e., we find the coordinate of the point where the secant lines intersect. (We use a tangent line on the side where there is only one point.) In addition, we also obtain upper bounds q_{up} to $q^* := k(t^*)$ from the point where the secant lines intersect.

7.3.3 Vertical Cut

Suppose we have two values of t, t_e, and t_h, with $k(t_e) < k(t_h)$. (A similar argument holds for the reverse inequality.) Then we can use the concavity of k to reduce the interval of uncertainty for t. We find the intersection of the horizontal line through $(t_h, k(t_h))$ with the tangent line at the point $(t_e, k(t_e))$, i.e.,

$$t_{\text{high}} = t_e + \frac{k(t_h) - k(t_e)}{k'(t_e)},$$

where t_{high} is the upper bound on t^*.

7.3.4 Inverse Interpolation

We use (quadratic or linear) inverse interpolation on $\psi(t) = 0$, in the case that $y_0(t) \neq 0$. Since $\psi(t)$ is a strictly decreasing function, we can consider its inverse function, say $t(\psi)$. We use (concave) quadratic interpolation when possible, i.e., suppose the points (ψ_i, t_i), $i = 1, 2, 3$. Then we solve the system

$$\begin{bmatrix} \psi_1^2 & \psi_1 & 1 \\ \psi_2^2 & \psi_2 & 1 \\ \psi_3^2 & \psi_3 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ t_{\text{new}} \end{bmatrix} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

and get the new estimate t_{new}. We use the top right 2×2 system in the linear interpolation case.

In the hard case (case 2), inverse interpolation is not well defined. To avoid an erroneous step, we maintain λ on the correct side of $\lambda_1(A)$.

7.3.5 Recognizing an Unconstrained Minimum

Problem (TRS) has an inequality constraint, but generally the optimum lies on the boundary and we can solve the same problem with the equality constraint to get the optimum. This does not hold if and only if the matrix A is positive definite and the unconstrained minimum lies inside the TR. The following theorem is the key to recognizing this case.

THEOREM 7.3 *Let $\bar{x}$ be a solution to $(A - \lambda I)x = a$ with $(A - \lambda I)$ positive semidefinite. If $\lambda \leq 0$, then $\bar{x}$ is a solution to* $\min\{x^{\mathrm{T}}Ax - 2a^{\mathrm{T}}x\colon \|x\| \leq \|\bar{x}\|\}$. *If $\lambda \geq 0$, then $\bar{x}$ is a solution to* $\min\{x^{\mathrm{T}}Ax - 2a^{\mathrm{T}}x\colon \|x\| \geq \|\bar{x}\|\}$.

Proof The first part follows from the necessary and sufficient optimality conditions and the second part follows easily knowing that the sign of λ plays no role in the positive semidefiniteness of the matrix $A - \lambda I$ when proving these optimality conditions. ■

In our algorithm, we successively obtain solutions x_k to $(A - \lambda_k I)x_k = a$ with $A - \lambda_k I \succeq 0$. Therefore each x_k is a solution to $\min\{x^{\mathrm{T}}Ax - 2a^{\mathrm{T}}x\colon \|x\| = \|x_k\|\}$. Checking the sign of the multiplier λ_k tells us whether x_k is a solution to $\min\{x^{\mathrm{T}}Ax - 2a^{\mathrm{T}}x\colon \|x\| \leq \|x_k\|\}$ or $\min\{x^{\mathrm{T}}Ax - 2a^{\mathrm{T}}x\colon \|x\| \geq \|x_k\|\}$. If the latter case holds and $\|x_k\| \leq s$, then we know the unconstrained minimum lies in the TR.

7.3.6 Shift and Deflate

Let us consider the case when the optimum is on the boundary of the ball. Once the smallest eigenvalue $\lambda_1(A)$ is found in the initialization step, we can use the shift in Lemma 2.1, Item 3 and Lemma 2.3. Therefore, for simplicity, we can assume that $\lambda_1(A) = 0$.

During the algorithm we *deflate* eigenvectors $y = (y_0 v^{\mathrm{T}})^{\mathrm{T}}$ if y_0 is small (essentially 0). This indicates that $a^{\mathrm{T}}v$ is small. We perturb $a \leftarrow a - a^{\mathrm{T}}vv$ and deflate using $A \leftarrow A + \alpha vv^{\mathrm{T}}$.

7.3.7 Taking a Primal Step to the Boundary

The interpolation and heuristics are used to find a new point t and then a corresponding λ for the dual problem, i.e., they are used to take a dual step. Once the λ is found, we can find a corresponding primal point $x(\lambda)$ for the primal problem. This point will be primal feasible if t (equivalently λ) is on the hard side and it will be primal infeasible on the other (easy) side.

In either case, we now show that we can take an inexpensive primal step, i.e., move to the boundary and improve the objective value. Thus, we get a primal–dual algorithm.

In the easy case, the step is motivated by the following lemma (for a proof see Refs. [1] or [38]).

LEMMA 7.1 *Let* $0 < s_1 < s < s_2$ *and*

$$x_h \in \operatorname{argmin}\{q(x)\colon \|x\| \le s_1^2\},\, x_e \in \operatorname{argmin}\{q(x)\colon \|x\| \le s_2^2\}.$$

Suppose that $\|x_h\| = s_1$, $\|x_e\| = s_2$, $x_h^T(x_e - x_h) \neq 0$ *and the Lagrange multiplier* λ_h *for* x_h *satisfies* $A - \lambda_h I \succ 0$. *Furthermore, let* $m(\alpha) := q(x_h + \alpha(x_e - x_h))$. *Then* $m'(\alpha) \le 0$, for $\alpha \in [0, 1]$.

Thus, we find $\bar{\alpha}$ so that $\|x_h + \bar{\alpha}(x_e - x_h)\| = s$. We use two values, t_h and t_e, respectively on the hard side and the easy side, with

$$x_h := \frac{1}{y_0(t_h)} x(t_h), \quad x_e := \frac{1}{y_0(t_e)} x(t_e).$$

Then if $x_h^T(x_e - x_h) \neq 0$ and $\bar{\alpha}$ is defined as in the above lemma, taking a step to the boundary from x_h to $x_h + \bar{\alpha}(x_e - x_h)$ will decrease the objective function. We assumed the easy case so that $y_0(t_e) \neq 0$. However, in the hard case (case 2) $y_0(t_e) = 0$. As in the MS algorithm, we take a step to the boundary. From x_h, we use an eigenvector z for the eigenvalue $\lambda_1(A)$ as the direction to the boundary. This choice is motivated by Lemma 3.2 and the desire to make the quadratic form $z^T(A - \lambda_1(D(t_h))I)z$ small. We take the step $x_h + \tau z$ with τ chosen to reduce the objective function and satisfy $\|x_h + \tau z\| = s$. The explicit expression for τ is

$$\tau = \frac{s^2 - \|x_h\|^2}{x_h^T z + \operatorname{sgn}(x_h^T z)\sqrt{(x_h^T z)^2 + (s^2 - \|x_h\|^2)}},$$

where $\operatorname{sgn}(\cdot)$ equals 1 if its argument is nonnegative and -1 otherwise. Given a direction z, there are two values of τ for which $x_h + \tau z$ reaches the boundary. Reference [10] proves that to improve the objective, we should pick the one with smallest magnitude.

8 NUMERICAL EXPERIMENTS

8.1 The Hard Case

We now provide numerical evidence that our modified RW algorithm is better suited to handle the hard case (case 2) than the MS algorithm. It is stated in Ref. [10] that the latter algorithm requires few iterations (2–3) in the hard case. However, this appears to hold only when the desired accuracy is low. Many more iterations are required when higher accuracy is desired. Our tests were done using MATLAB 6.1 on a SUNW Ultra−5_10 with 1 GIG RAM.

Let q^* be the optimal objective value of TRS and $\tilde{q}$ be an approximation for q^*. The MS algorithm returns an approximate solution that satisfies $\tilde{q} \le (1 - \sigma)^2 q^*$, where $0 \le \sigma < 1$ is an input to the algorithm, and the approximate solution of the RW algorithm satisfies

$\tilde{q} \leq 1/(1 + 2\text{dgaptol})q^*$, where dgaptol is the desired relative duality gap tolerance. Hence, to get equivalent accuracy, we choose $\sigma = 1 - \sqrt{1/(1 + 2\text{dgaptol})}$.

We used randomly generated sparse hard case (case 2(ii)) TR subproblems, where the density is order $1/(20n \log n)$. The tolerance parameter dgaptol was set to 10^{-12}. Each row in Table II gives the average number of iterations (iter) and computation time (cpu) for 10 problems of size n. We could not go beyond $n = 640$ for the MS algorithm due to the large cpu that arise. The results, given in Table II illustrate the improved performance in both the iter and the cpu.

Note that the GLTR algorithm does not appear in the above comparison since the algorithm was not designed to handle this case.

8.2 Rendl–Wolkowicz and Generalized Lanczos Trust Region Algorithms in Trust Region Framework

In Ref. [12], the GLTR method is stopped early after a limited extra number of iterations, say N, once the solution is known to lie on the boundary of the TR. More precisely, the algorithm stops if the subspace S in Eq. (18) is increased in dimension by N once the solution is known to be on the boundary of the TR and problems of the type (18) are solved. The reason for limiting the size of the subspaces S once the boundary has been reached is motivated by the fact that the authors in Ref. [12] question whether high accuracy is needed for the TRS within a TR framework. We argue that increased accuracy is needed for TRS just as for the solution of the Newton equation when using inexact Newton methods, as the iterates approach a stationary point, see e.g. Refs. [7,39].

For the upcoming test problems, we used $N = 2, 6$, and n, where n is the problem dimension. Our test problems are of the following form

$$\min_{x \in \mathbb{R}^n} f(x) := \frac{x^{\mathrm{T}} A x}{x^{\mathrm{T}} B x}, \tag{32}$$

where B is a positive definite matrix and A and B are generated randomly. The minimum is attained at x^*, a generalized eigenvector corresponding to the smallest eigenvalue of the generalized eigenvalue problem

$$Ax = \lambda Bx.$$

The optimal value is equal to $\lambda_1(B^{-1/2} A B^{-1/2})$.

To solve each problem we used the TR method described by Algorithm 8.1 on the same machine as in Section 8.1. In this algorithm, f represents the function to be minimized, x_j is an approximation of a minimizer after j iterations and s_j is the radius of the TR at iteration j.

TABLE II Modified-RW and MS algorithms; hard case (case 2(ii))

Dim. (n)	*MS iters*	*RW iters*	*MS cpu*	*RW cpu*
40	36.4	6.4	0.79	0.55
80	34.4	7.6	1.0	0.57
160	39.2	7.2	6.49	0.61
320	33.8	7.4	23.36	0.77
640	37.8	5.0	149.36	0.78
1280	–	7.6	–	2.06
2560	–	5.0	–	3.18

ALGORITHM 8.1 *Trust Region Method*

1. *Given x_j and s_j, calculate $\nabla f(x_j)$ and $\nabla^2 f(x_j)$. Stop if*

$$\frac{\|\nabla f(x_j)\|}{1+|f(x_j)|} < \text{gradtol}. \tag{33}$$

2. *Find δ_j to a given tolerance in the TRS*

$$\begin{aligned} \delta_j \in \text{argmin} \quad & q_j(\delta) := \nabla f(x_j)^{\mathrm{T}}\delta + \frac{1}{2}\delta^{\mathrm{T}}\nabla^2 f(x_j)\delta \\ \text{s.t.} \quad & \|\delta\|^2 \le s_j^2. \end{aligned} \tag{34}$$

3. *Evaluate $r_j = (f(x_j) - f(x_j+\delta_j))/(q_j(0) - q_j(\delta_j))$.*

4. (a) *If $r_j > 0.95$, set $s_{j+1} = 2s_j$ and $x_{j+1} = x_j + \delta_j$.*
   (b) *If $0.01 \le r_j < 0.95$, set $s_{j+1} = s_j$ and $x_{j+1} = x_j + \delta_j$.*
   (c) *If $r_j < 0.01$, set $s_{j+1} = 0.5s_j$ and $x_{j+1} = x_j$.*

Except for the stopping criteria (33) which has been scaled here, this algorithm is Algorithm 6.1 in Ref. [12]. We chose x_0 randomly and have fixed $s_0 = 1$, gradtol $= 10^{-2}$. We ran five random problems for each problem size $n = 20, 25$, and 30, where n is the size of the square matrices A and B. If the RW algorithm solves Eq. (34) and the solution is on the boundary of the TR, we stop if the duality gap, dgaptol, (between TRS and Eq. (25)) satisfies

$$\sqrt{\text{dgaptol}} \le \min\{0.1, \max\{10^{-8}, 10^{-5}\|\nabla f(x_j)\|\}\}. \tag{35}$$

Otherwise, the solution is in the interior and we stop with an approximate solution δ_j which satisfies

$$\|\nabla f(x_j) + \nabla^2 f(x_j)\delta_j\| \le \min\{0.1, \max\{10^{-8}, 10^{-5}\|\nabla f(x_j)\|\}\}, \tag{36}$$

If the GLTR algorithm is used, we stop within this algorithm if N iterations have been done after knowing the solution lies on the boundary of the TR (see Ref. [12]) or if

$$\|(A - \lambda_k I)x_k - a\| < \min\{0.1, \max\{10^{-8}, 10^{-5}\|\nabla f(x_j)\|\}\} \tag{37}$$

(see Eq. (30)) or Eq. (36) is satisfied, depending if the solution is on the boundary of the TR or not.

Equations (35) and (37) yields approximately the same accuracy in terms of the duality gap. Recall from Section 6.2 that $\|(A - \lambda_k I)x_k - a\|$ is an approximation for the square root of the duality gap between TRS and its dual (28). The stopping criteria (35) and (37) are set to reflect this relationship.

For each problem, we give the number of iterations (iter) taken by the TR method (32). If the GLTR algorithm is used to solve the TRS (34), we give as well the number of iterations within Algorithm 8.1 where the GLTR algorithm failed to solve Eq. (34) because it was unable to solve the restricted problem (18). This last output (*hc2*) is an indicator of the (almost) hard case (case 2).

The results given in Tables III–V, show that the (almost) hard case (case 2) occurs in many problems and Algorithm 8.1, using the RW algorithm for Eq. (34), takes fewer iterations to converge compared to the GLTR algorithm. This is independent of N. This suggests that handling the hard case (case 2) should be an essential feature for a robust TR method. However, we reach the same conclusions mentioned in Ref. [12] when the hard case (case 2) does not

TABLE III The RW and GLTR; TR framework; size $n = 20$

	Algorithm used for solving the TRS (34)						
	RW	*GLTR with N = 2*		*GLTR with N = 6*		*GLTR with N = n*	
Problem	*iter*	*iter*	*hc2*	*iter*	*hc2*	*iter*	*hc2*
1	16	36	5	25	0	16	0
2	33	22	0	55	19	48	16
3	50	21	0	15	0	52	16
4	41	39	8	45	16	32	3
5	25	45	10	19	0	25	0

occur. We observe the surprising fact that inexact solutions may indeed lead to less iterations in some cases. This may be due to the fact that the TR becomes inactive early and Newton's method takes over. This clearly requires more study.

As we may expect, when the hard case (case 2) does not occur and $N = n$, a TR method, using either the RW algorithm or the GLTR algorithm to solve the TRS (34), takes more or less the same iter. This should be the case since we are asking for the same accuracy in Eqs. (35) and (37).

8.3 Accuracy of Trust Region Subproblem in a Trust Region Method

Inexact Newton methods can obtain q-superlinear and even q-quadratic convergence rates if the accuracy of the Newton equation increases appropriately as the iterates approach a stationary point, e.g., Ref. [6]. Trust region methods such as Algorithm 8.1 are expected to reduce to Newton's method asymptotically, i.e., the TR constraint is expected to become inactive for most problems. This happens for example when the second order sufficient optimality conditions (positive definite Hessian) holds at the limit point. In either case, i.e., whether or not the TR constraint becomes inactive, the accuracy for solving TRS must increase as we approach the stationary point. We now investigate the number of iterations the TR Algorithm 8.1 takes to solve an unconstrained minimization problem as the accuracy of the solutions of the TRS (34) varies using MATLAB 6.5 on a Sun Fire 280R (UltraSPARC-III) with 2 GIGs RAM.

We solve the problem $\min_{x\in\mathbb{R}^n} f(x)$ to accuracy given by varying values of gradtol in the inequality Eq. (33). The TRS (34) is solved using the modified RW algorithm to accuracy

$$\text{dgaptol} = \max\left\{\text{tol}, 10^{-6}\min\left\{1, \frac{\|\nabla f(x_j)\|}{1+|f(x_j)|}\right\}^{1/2}\right\}, \tag{38}$$

TABLE IV The RW and GLTR; TR framework; size $n = 25$

	Algorithm used for solving the TRS (34)						
	RW	*GLTR with N = 2*		*GLTR with N = 6*		*GLTR with N = n*	
Problem	*iter*	*iter*	*hc2*	*iter*	*hc2*	*iter*	*hc2*
1	31	48	17	34	5	32	2
2	38	26	0	27	1	32	2
3	22	25	0	22	0	22	0
4	20	39	4	31	1	20	0
5	25	26	0	22	0	22	0

TABLE V The RW and GLTR; TR framework; size $n = 30$

	Algorithm used for solving the TRS (34)						
	RW	*GLTR with $N = 2$*		*GLTR with $N = 6$*		*GLTR with $N = n$*	
Problem	*iter*	*iter*	*hc2*	*iter*	*hc2*	*iter*	*hc2*
1	61	14	0	36	7	57	24
2	38	50	17	35	2	37	6
3	34	22	0	27	1	45	17
4	34	19	14	25	0	36	8
5	36	38	7	26	0	32	3

TABLE VI The TR on $f(x) = \sin(x_1 - 1) + \sum_{i=2}^{1000} 100\ \sin(x_i - x_{i-1}^2)$ [see Ref. 40]

	gradtol						
tol	*10^{-3}*	*10^{-4}*	*10^{-5}*	*10^{-6}*	*10^{-7}*	*10^{-8}*	*10^{-9}*
10^{-1}	42	117	$\geq$1000	$\geq$1000	$\geq$1000	$\geq$1000	$\geq$1000
10^{-3}	15	21	50	$s \cong 0$	$s \cong 0$	$s \cong 0$	$s \cong 0$
10^{-5}	12	16	18	88	$s \cong 0$	$s \cong 0$	$s \cong 0$
10^{-7}	12	22	38	52	67	$s \cong 0$	$s \cong 0$
10^{-9}	12	16	18	22	24	50	$s \cong 0$
10^{-11}	12	22	38	53	68	83	96

for varying values of tol. We also terminate Algorithm 8.1 if more than 1000 iterations are necessary or if the TR radius s_j becomes smaller than 10^{-10} (this case is indicated by $s \cong 0$). The results for our two examples are given in Tables VI and VII, where the entries are the iter taken by Algorithm 8.1.

From these results we observe two things. First, as the accuracy on the norm of the gradient is decreasing, the TR method (32) using low accuracy solutions for the TRS is eventually outperformed by higher accuracy solutions. Second, the results of Table VI indicate, for a fixed tolerance gradtol on the norm of the gradient, that more accurate solution of the TRS does not necessarily imply fewer iterations. Therefore, for robustness and as for inexact Newton methods, it is beneficial to use increased accuracy for the TRS when approaching the minimum of the objective function f.

TABLE VII The TR on $f(x) = \sum_{i=1}^{1000} 6x_i^2 + \sum_{i=503}^{n} x_i + \sum_{i=1}^{998} (x_i x_{i+2} - 4x_i x_{i+1}) - 3x_1 + x_2 + x_{499} - 3x_{500} + 4x_{501}$ [see Ref. 41]

	gradtol						
tol	*10^{-3}*	*10^{-4}*	*10^{-5}*	*10^{-6}*	*10^{-7}*	*10^{-8}*	*10^{-9}*
10^{-1}	9	10	$s \cong 0$	$s \cong 0$	$s \cong 0$	$s \cong 0$	$s \cong 0$
10^{-3}	4	5	$s \cong 0$	$s \cong 0$	$s \cong 0$	$s \cong 0$	$s \cong 0$
10^{-5}	4	5	6	$s \cong 0$	$s \cong 0$	$s \cong 0$	$s \cong 0$
10^{-7}	4	5	6	7	8	$s \cong 0$	$s \cong 0$
10^{-9}	4	5	6	7	8	$s \cong 0$	$s \cong 0$
10^{-11}	4	5	6	7	8	$s \cong 0$	$s \cong 0$

TABLE VIII Modified RW algorithm on TRS; $n = 100{,}000$

	dgaptol					
Density	*10^{-12}*		*10^{-10}*		*10^{-8}*	
10^{-8}	cpu:	18.2	cpu:	14.5	cpu:	13.5
	mvm:	185.6	mvm:	150.0	mvm:	140.0
	iter:	6.2	iter:	5.4	iter:	4.8
10^{-6}	cpu:	21.1	cpu:	19.2	cpu:	20.0
	mvm:	210.0	mvm:	196.0	mvm:	204.0
	iter:	6.4	iter:	5.4	iter:	5.6
10^{-4}	cpu:	91.7	cpu:	78.8	cpu:	76.0
	mvm:	341.6	mvm:	294.0	mvm:	276.0
	iter:	5.8	iter:	5.0	iter:	5.6

TABLE IX Modified RW algorithm on TRS; $n = 1{,}000{,}000$

	dgaptol					
	10^{-2}	*10^{-4}*	*10^{-6}*	*10^{-8}*	*10^{-10}*	*10^{-12}*
cpu	985.1	985.1	1082.8	1082.8	1183.6	1282.9
mvm	240	240	260	260	280	300
iter	3	3	4	4	5	6

8.4 Large Sparse Trust Region Subproblem

The results of Table VIII show how we used the RW algorithm to solve problems of size $n = 100{,}000$ of different density. Precisely, each row in this table corresponds to the density of the problems (for example, if the density is 10^{-6}, then at most $n^2 \times 10^{-6}$ entries in the matrix A are nonzero) and each column to the value of the parameter dgaptol. In each entry of the table we give the average taken over five random TR subproblems of the cpu seconds, the number of matrix-vector multiplications (mvm) and the number of iterations taken by the RW algorithm to find an approximate solution. We have been using for this section MATLAB 6.1 on a Pentium III with 4 GIGs RAM.

As we may expect, the cpu and the mvm increase as the density increases and the duality gap tolerance decreases. Furthermore, considering the reasonable length of the cpu taken to solve such TRS, we conclude that it is now within our reach to use TR methods to minimize functions with hundreds of thousands of variables assuming the Hessian has a sparse structure. In Table IX, we considered a TRS of size $n = 10^6$ with 11 million nonzeros in the sparse matrix A. The results show that higher accuracy solutions require minimal extra iterations of the RW algorithm.

9 CONCLUSION

In this article, we have studied the TRS with emphasis on robustness and solving large sparse problems. We focused on three dual based algorithms: the classical MS algorithm and the recent RW and GLTR algorithms, designed to solve large and sparse TRS.

We also studied many duals to TRS which can be formulated as semidefinite programs. We have seen how SDP arises naturally for TRS and provides a clear and simple unifying analysis between the different algorithms. In addition, this framework provides insights to the strengths and weaknesses of the algorithms.

In addition, we presented a modified/enhanced RW algorithm with new heuristics and techniques, in particular for taking a primal step to the boundary. However, the main improvement came from a new way of treating the (near) hard case based on Lemma 2.1. Surprisingly, the lemma shows that for each TRS, it is possible to consider an equivalent TRS where the hard case (case 2) does not occur.

Our final section included numerics which showed the advantage of using the modified RW algorithm over the MS algorithm in treating the hard case when high accuracy approximations are needed. We have also shown that handling the hard case in the TRS within a TR method may have an impact on the total iter if the hard case occurs frequently enough. Thus, the robustness of a TRS algorithm is indeed an important feature, in particular when the TR constraint stays active close to the optimal solution. Finally, we showed it is possible to solve large sparse TRS to high accuracy with hundreds of thousands of variables in a small number of iterations.

Acknowledgments

The authors are indebted to Florian Jarre for many helpful discussions and suggestions. Research supported by a ES-A scholarship of the Natural Sciences and Engineering Research Council of Canada and a doctoral scholarship (B2) from the Fonds de recherche sur la nature et les technologies du Québec.

References

[1] C. Fortin and H. Wolkowicz (2002). A survey of the trust region subproblem within a semidefinite framework. Technical Report CORR 2002-22, University of Waterloo, Waterloo, Canada. URL:orion.math.uwaterloo.ca:80/hwolkowi/henry/reports/ABSTRACTS.html#surveytrs.

[2] A.R. Conn, N.I.M. Gould and P.L. Toint (2000). *Trust-Region Methods*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.

[3] A.N. Tikhonov and V.Y. Arsenin (1977). *Solutions of Ill-Posed Problems*. V.H. Winston & Sons, John Wiley & Sons, Washington DC (Translation editor Fritz John).

[4] A.E. Hoerl and R.W. Kennard (1970). Ridge regression: biased estimation of nonorthogonal problems. *Technometrics*, **12**, 55–67.

[5] R. Fletcher (1987). *Practical Methods of Optimization*, 2nd ed. John Wiley and Sons, New York.

[6] J. Nocedal and S.J. Wright (1999). *Numerical Optimization*. Springer-Verlag, New York.

[7] J.L. Morales and J. Nocedal (2000). Automatic preconditioning by limited memory quasi-Newton updating. *SIAM J. Optim.*, **10**(4), 1079–1096 (electronic).

[8] S.G. Nash and J. Nocedal (1991). A numerical study of the limited memory BFGS method and the truncated-Newton method for large scale optimization. *SIAM J. Optim.*, **1**(3), 358–372.

[9] Å. Björck (1996). *Numerical Methods for Least Squares Problems*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia.

[10] J.J. Moré and D.C. Sorensen (1983). Computing a trust region step. *SIAM J. Sci. Stat. Comp.*, **4**(3), 553–572.

[11] F. Rendl and H. Wolkowicz (1997). A semidefinite framework for trust region subproblems with applications to large scale minimization. *Mathematical Programming Series B*, **77**(2), 273–299.

[12] N.I.M. Gould, S. Lucidi, M. Roma and P.L. Toint (1999). Solving the trust-region subproblem using the lanczos method. *SIAM J. Optimiz.*, **9**(2), 504–525.

[13] D.M. Gay (1981). Computing optimal locally constrained steps. *SIAM J. Sci. Statist. Comput.*, **2**, 186–197.

[14] D.C. Sorensen (1997). Minimization of a large-scale quadratic function subject to a spherical constraint. *SIAM J. Optimiz.*, **7**(1), 141–161.

[15] P.D. Tao and L.T.H. An (1996). Difference of convex functions optimization algorithms (dca) for globally minimizing nonconvex quadratic forms on euclidean balls and spheres. *Oper. Res. Lett.*, **19**(5), 207–216.

[16] W.W. Hager (2000). Minimizing a quadratic over a sphere. Technical report, University of Florida, Gainsville, FA.

[17] Y. Ye (1994). Combining binary search and Newton's method to compute real roots for a class of real functions. *J. Complexity*, **10**, 271–280.

[18] D.M. Gay (1981). Computing optimal locally constrained steps. *SIAM J. Sci. Stat. Comp.*, **2**(2), 186–197.

[19] D.C. Sorensen (1982). Newton's method with a model trust region modification. *SIAM J. Numer. Anal.*, **19**(2), 409–426.

[20] J.W. Demmel (1997). *Applied Numerical Linear Algebra*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.

[21] A. Ben-Tal and M. Teboulle (1996). Hidden convexity in some nonconvex quadratically constrained quadratic programming. *Math. Programming*, **72**(1, Ser. A), 51–63.
[22] E.G. Gol'stein (1972). *Theory of Convex Programming*. American Mathematical Society, Providence, RI.
[23] A.V. Fiacco (1983). *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming. Mathematics in Science and Engineering*, Vol. 165. Academic Press, 1983.
[24] R.J. Stern and H. Wolkowicz (1995). Indefinite trust region subproblems and nonsymmetric eigenvalue perturbations. *SIAM J. Optimiz.*, **5**(2), 286–313.
[25] C. Reinsch (1967). Smoothing by spline functions. *Numer. Math.*, **10**, 177–183.
[26] C. Reinsch (1971). Smoothing by spline functions ii. *Numer. Math.*, **16**, 451–454.
[27] M.D. Reinsch (1973). An algorithm for minimization using exact second derivatives. Technical Report 515, Harwell Laboratory, Harwell, Oxfordshire, England.
[28] G.E. Forsythe and G.H. Golub (1965). On the stationary values of a second-degree polynomial on the unit sphere. *J. Soc. Indust. Appl. Math.*, **13**, 1050–1068.
[29] T. Steihaug (1983). The conjuguate gradient method and trust regions in large scale optimization. *SIAM J. Numer. Anal.*, **20**(3).
[30] P.L. Toint (1981). Towards an efficient sparsity exploiting Newton method for minimization. *Sparse Matrices and Their Uses*, pp. 57–88. I.S. Duff, Academic Press Edition.
[31] H. Wolkowicz, R. Saigal and L. Vandenberghe (Eds.) (2000). Handbook of Semidefinite Programming: Theory, Algorithms, and Applications, xxvi+ p. 654. Kluwer Academic Publishers, Boston, MA.
[32] R.J. Vanderbei (1998). *Linear Programming: Foundations and Extensions*. Kluwer Acad. Publ., Dordrecht.
[33] S. Wright (1996). *Primal-Dual Interior-Point Methods*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.
[34] C. Helmberg and F. Rendl. A spectral bundle method for semidefinite programming. *SIAM J. Optimiz.*, **10**(3), 673–696.
[35] S.J. Benson, Y. Ye and X. Zhang (2000). Solving large-scale sparse semidefinite programs for combinatorial optimization. *SIAM J. Optim.*, **10**(2), 443–461 (electronic).
[36] R.A. Horn and C.R. Johnson (1987). *Matrix Analysis*. Cambridge University Press, Cambridge.
[37] Y. Levin and A. Ben-Israel (2002). The Newton bracketing method for convex minimization. *Comput. Optimiz. Appl.*, **21**, 213–229.
[38] C. Fortin (2000). *A survey of the trust region subproblem within a semidefinite framework*. Master's thesis, University of Waterloo.
[39] R.S. Dembo and T. Steihaug (1963). Truncated Newton algorithms for large scale unconstrained optimization. *Math. Program.*, **26**(72), 190–212.
[40] A. Bouriacha. Private communication.
[41] Y. Lin and J. Pang (1987). Iterative methods for large convex quadratic programs: A survey. *SIAM J. Control Optim.*, **25**.