A Survey of the Trust Region Subproblem Within a Semidefinite Programming Framework

Charles Fortin * Henry Wolkowicz †

August 27, 2002

University of Waterloo
Department of Combinatorics & Optimization
Waterloo, Ontario N2L 3G1, Canada
Research Report CORR 2002-22

Key words: Trust Regions, Semidefinite programming, Duality, Unconstrained Minimization.

Abstract

The trust region 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 trust region algorithms for function minimization. Trust region algorithms

*McGill University, Department of Mathematics and Statistics, Montréal, Québec, Canada, H3A 2T5. 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. E-mail fortin@math.mcgill.ca. This work was part of this author's Master's thesis while at the University of Waterloo.

†University of Waterloo, Department of Combinatorics and Optimization, Waterloo, Ontario, Canada, N2L 3G1. E-mail hwolkowicz@uwaterloo.ca.

[0] This report is available with URL:
orion.math.uwaterloo.ca/~hwolkowi/henry/reports/ABSTRACS.html

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.

This paper 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 solves TRS efficiently; and 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. This is contrary to statements in the literature which label TRS ill-posed or degenerate, if the so-called hard case holds. 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.

Contents

1 Introduction

We are concerned with the following quadratic minimization problem:

$$\text{(TRS)} \qquad \begin{array}{rll} q^* = & \min & q(x) := x^T A x - 2a^T x \\ & \text{s.t.} & \|x\| \leq s. \end{array}$$

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 referred as the trust region subproblem (denoted TRS). This problem has many applications in e.g.: forming subproblems for constrained optimization [4], regularization of ill-posed problems [32], and regularization for ill-conditioned linear regression problems (called ridge regression, [15]). In addition, it is important in a class of optimization methods called *trust region (TR) 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 trust region. We do not discuss scaled, ellipsoidal, trust regions.) 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. Trust region methods have advantages over e.g., quasi-Newton methods. Under mild assumptions, the trust region algorithms produce a sequence of iterates with an accumulation point that satisfies *both* first and second order necessary optimality conditions (e.g. [8]). 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. the recent books [22, 4].)

However, the popularity of trust region 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, sparsity of the Hessian was not exploited efficiently, whereas algorithms such as *limited memory quasi-Newton methods* proved to be successful, e.g. [18, 21].

Though TRS appears to be a simple problem, there is a long history of elegant theory and algorithms. (The recent books [3, 4] contain extensive bibliographies. See also the bibliographical database for [4] at URL

www.fundp.ac.be/~phtoint/pht/trbook.bib.) In this paper, we consider TRS using 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 1983 (MS), the semidefinite programming (SDP) based algorithm of Rendl-Wolkowicz 1997 (RW), and the *generalized Lanczos trust region method* 1999 (GLTR) of Gould, Lucidi, Roma and Toint [13]. The classical (MS) algorithm [20] was the first algorithm able to handle the hard case efficiently. (The algorithm of Gay [9] also treats the hard case.) We revisit and modify the RW primal-dual algorithm [27] which is based on SDP and duality and designed specifically to handle large sparse problems; it also handles the hard case efficiently. The SDP formulation allows for a (convex) pair of primal-dual programs that are equivalent to (TRS). Therefore, one considers iterations using the modern elegant approach of primal-dual optimality conditions and duality gaps. In particular, the SDP and duality approach illustrates that, contrary to statements in the literature (e.g. [13]) TRS in the hard (or near hard) case is *not* an ill-posed problem. (The hard case occurs when the vector from the linear term a is orthogonal to the null space of $A - \lambda^* I$, where λ^* is the optimal Lagrange multiplier. Note that $a = 0$ is a hard case instance, but this is a symmetric eigenvalue problem with condition number 1, e.g. [6].) The optimal value and an optimum can be found to a specified accuracy, though the optimum may not be unique for hard case examples. In fact, the SDP pair both satisfy the Slater constraint qualification and strict complementarity; thus they are stable convex programs. (Though the optimum for TRS may not be unique, the optimal set for the SDP formulation is convex and bounded and the optimum found by an interior-point method is the analytic center, which is unique.) However, the formulation used for many algorithms results in an ill-posed problem. (See Section 2.1.)

GLTR (or coincidently GLRT) is the last algorithm we look at, see [13]. This algorithm uses the Lanczos procedure to obtain a restricted TRS problem with a tridiagonal matrix. This subproblem can can be solved quickly using the MS algorithm.

Several other recent approaches deserve mention. The method by Sorensen [5] is similar to the RW algorithm in that it uses a parametric eigenvalue approach. The DC (difference of convex functions) method of An and Tao [31] and the method of Hager [14] are both designed to exploit sparsity. The method in [14] 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 [37] exploits a new efficient line search technique.

We include several dual programs to TRS. Surprisingly, strong Lagrangian duality holds for TRS, a nonconvex program. Thus TRS sits on the boundary between convex and nonconvex programs and is an important theoretical tool, see e.g. [23]. With strong duality, we can use a modern primal-dual approach to derive algorithms for TRS. Advantages and disadvantages can be viewed through the choices of using the primal and/or dual programs, e.g. using a pure dual algorithm allows for sparsity considerations. Semidefinite programming (SDP) and the primal-dual methods will be the link between the above mentioned MS, RW, GLTR algorithms.

One contribution of this paper is a novel approach to handling the hard case using a shift of the eigenvalues and deflation. More precisely, if the hard case holds and A is not positive definite, then the optimum occurs on the boundary of the ball. We can therefore shift A to make it positive semidefinite, and also deflate eigenvectors that arise that are orthogonal to a. We thus obtain an equivalent stable convex program with Slater's condition and with a unique Lagrange multiplier.

We also include numerical comparisons between the algorithms and examples that illustrate the performance on the hard case. In particular, we try to answer questions posed in [13] about the desired accuracy in solving the TRS within a TR minimization algorithm.

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 the equivalent well-posed convex program. The MS algorithm is described in Section 3, while 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. In Section 6 we present the SDP frameworks for both the MS and GLTR algorithms. The RW algorithm with our modifications is presented in detail in Section 7. The numerical tests appear in Section 8 with special hard case instances in 8.1. Concluding remarks are given in Section 9.

2 Optimality Conditions

It is known (see [10] and [28]) that x^* is a solution to TRS if and only if

$$\left.\begin{array}{c}(A-\lambda^* I)x^* = a \\ A-\lambda^* I \succeq 0, \lambda^* \leq 0\end{array}\right\} \quad \text{dual feasibility}$$
$$\|x^*\|^2 \leq s^2 \quad \text{primal feasibility} \tag{2.1}$$
$$\lambda^*(s^2 - \|x^*\|) = 0, \quad \text{complementary slackness}$$

for some (Lagrange multiplier) λ^*. These conditions are surprising in two respects. First, these conditions characterize optimality of a possibly non-convex 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.

Remark 2.1. *We have added the descriptive three phrases in* (2.1) *since this coincides with the framework in [27] and with the modern primal-dual optimization approach, though no dual program appeared in the earlier papers [10, 28]. It is interesting to start with these equations and derive a possible dual program for TRS, i.e. for some function* $h(x)$ *the dual program would be*

$$\max h(\lambda) \ \ s.t. \ \ (A-\lambda I)x = a, \ \ A-\lambda I \succeq 0, \lambda \leq 0. \tag{2.2}$$

The standard paradigm is that the duality gap is zero if and only if complementary slackness holds. Here this means that for a feasible primal dual pair we have

$$h(\lambda) - q(x) = \lambda(s^2 - \|x\|^2),$$

i.e. we can substitute using the dual feasibility equation and get

$$\begin{aligned} h(\lambda) &= x^T A x - 2a^T x + \lambda s^2 - \lambda x^T x \\ &= x^T(A-\lambda I)x - 2a^T x + \lambda s^2 \\ &= -a^T(A-\lambda I)^\dagger a + \lambda s^2, \end{aligned}$$

where $\cdot^\dagger$ *denotes the Moore-Penrose generalized inverse. (In fact, any generalized inverse would do here.) This is the maximization of a concave function of one variable over an interval. Can we determine whether this is a correct dual program for TRS? We could then apply modern interior-point methods; i.e. we would apply a damped Newton method to the primal-dual optimality*

conditions after perturbing the right hand side of the complementary slackness equation to $\rho > 0$; simultaneously, we would drive ρ, the measure of the duality gap, down to 0.

However, though the equations (2.1) *characterize optimality of TRS, one glaring missing equation here is a complementary slackness equation for the dual semidefinite inequality constraint. A true primal-dual path-following method exploits such an equation to speed up convergence. This suggests a weakness in this characterization of optimality. This is discussed further below in the sections on SDP.*

2.1 The Hard Case

If $A - \lambda^* I \succ 0$ in (2.1), then x^* is the unique solution to TRS, (this is true generically) i.e. $x^* = (A - \lambda^* I)^{-1} a$. In general, we denote

$$x(\lambda) = (A - \lambda I)^{\dagger} a. \tag{2.3}$$

Also, there will be a solution to TRS on the boundary of the ball $\{x : \|x\|^2 \leq s^2\}$ unless A is positive definite and $\|A^{-1}a\| < s$, i.e. unless the unique unconstrained minimizer of q lies in the interior of the ball, $\|x^*\| < s$. If we assume that there exists a solution to TRS on the boundary of the ball, it would seem natural to search for a λ such that

$$\|(A - \lambda I)^{-1} a\| = s, \quad \lambda \leq \lambda_1(A),\ \lambda \leq 0, \tag{2.4}$$

using a fast algorithm such as Newton's method. ($\lambda_k(A)$ denotes the k-th smallest eigenvalue of A. This is to distinguish between the k-th iterate λ_k for estimating λ^*.) This raises the question of singularity of $A - \lambda^* I$. The conditions for this to occur are well known in the literature. We let $\mathcal{N}(\cdot)$ denote the null space, and $\mathcal{R}(\cdot)$ denote the range space. We define the following (see also Table 2.1):

1. **Easy Case**: If a is not perpendicular to $\mathcal{N}(A - \lambda_1(A)I)$ (equivalently $a \notin \mathcal{R}(A - \lambda_1(A)I)$), then we have the easy case; this implies $A - \lambda^* I \succ 0$. In particular $A - \lambda^* I$ is invertible and we can therefore work with (2.4). The optimum $x^* = x(\lambda^*)$ is unique.

2. **Hard Case**: If a is perpendicular to $\mathcal{N}(A - \lambda_1(A)I)$ (equivalently $a \in \mathcal{R}(A - \lambda_1(A)I)$), two possibilities can occur:

(a) **Hard Case (case 1)**: If $\lambda^* < \lambda_1(A)$, no obvious difficulties occur. We can still work with (2.4). The optimum $x^* = x(\lambda^*)$ is unique.

(b) **Hard Case (case 2)**: If $\lambda^* = \lambda_1(A)$, then there are two possibilities:

i. When $\|(A - \lambda^* I)^\dagger a\| = s$ or $\lambda^* = 0$, then the pair $x^* = x(\lambda^*), \lambda^*$ satisfies the optimality conditions (2.1). Therefore, We can still work with (2.4) and use the Moore-Penrose generalized inverse for the singular case.

ii. When $x(\lambda^*) = \|(A - \lambda^* I)^\dagger a\| < s, \lambda^* < 0$, it almost appears that the optimality conditions fail. However, we can choose an eigenvector $z \in \mathcal{N}(A - \lambda^* I)$ and calculate τ appropriately so that $s = \|x(\lambda^*) + \tau z\|$. (Note that $x(\lambda^*) \perp z$, i.e. $s^2 = \|x(\lambda^*)\|^2 + \|\tau z\|^2$.) This is the only case where the optimum is *not* unique. The manifold of optimal solutions is given by the intersection of the *generalized eigenspace* $\{x : (A - \lambda^* I)x - a = 0\}$ with the boundary of the ball, $\{x : \|x\| = s\}$.

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) $\|(A - \lambda^* I)^\dagger a\| < s, \lambda^* < 0$

Table 2.1: The three different cases for the trust region subproblem. We include two subcases (i) and (ii) for the hard case (case 2).

2.1.1 Shift, Deflation, 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 $\lambda^* = 0$ and the hard case (case 2) holds, 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 handling the hard case. It shows that we can *deflate and/or shift* eigenspaces that are orthogonal to the linear term a.

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

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

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

1. *If* $S_3 \neq \emptyset$ *(easy case), then*

$$(x^*, \lambda^*) \text{ solves TRS}$$
$$\textbf{iff}$$
$$(x^*, \lambda^*) \text{ solves TRS when } A \text{ is replaced by } A_1 + A_3.$$

2. *Suppose* $S_3 = \emptyset$ *(hard case). Then* $i_0 := 1 \in S_4$ *and*

$$(x^*, \lambda^*) \text{ solves TRS}$$
$$\textbf{iff}$$
$$(x^*, \lambda^*) \text{ solves TRS when } A \text{ is replaced by } A_1 + \lambda_{i_0} v_{i_0} v_{i_0}^T.$$

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

$$(x^*, \lambda^*), \text{ with } x^* = u^* + z, z \in \mathcal{N}(A - \lambda^* I) \text{ solves TRS}$$
$$\textbf{iff}$$
$$(u^*, \lambda^* - \lambda_1(A)) \text{ solves TRS when } A \text{ is replaced by } A - \lambda_1(A) I.$$

4. *Suppose that* $\lambda_1(A) \geq 0$*. Then*

$$(x^*, \lambda^*) \text{ solves TRS}$$
$$\textbf{iff}$$
$$(x^*, \lambda^*) \text{ solves TRS when } A \text{ is replaced by } A + \sum_{i \in S_4} \alpha_i v_i v_i^T, \text{ with } \alpha_i \geq 0.$$

5.

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

iff

the hard case (case 2(ii)) holds.

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

$$(\mathrm{TRS}_P) \quad q^* = \begin{array}{ll} \min & (P^T x)^T \Lambda (P^T x) - 2\bar{a}^T (P^T x) = w^T \Lambda w - 2\bar{a}^T w \\ \text{s.t.} & \|w\| \leq s, \quad w = P^T x. \end{array} \tag{2.5}$$

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 $w^* = P^T x^*$ solves (2.5). We set $w^* = P^T x^*$ and, for $k = 1, 2, 3, 4$, $E_k = \sum_{i \in S_k} e_i e_i^T$, $\Lambda_k = \sum_{i \in S_k} \lambda_i e_i e_i^T$, $x_k^* := P_k x^*, w_k^* := E_k w^*$, where the e_i are unit vectors, and $E_k = \Lambda_k = 0, x_k = w_k = 0$, if $S_k = \emptyset$. In addition,

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

1. Necessity: Assume that (x^*, λ^*) solves TRS. From the definitions, $\bar{a}_i = 0, \forall i \in S_2 \cup S_4$. Since $S_3 \neq \emptyset$, the easy case holds and $w^* = (\Lambda - \lambda^* I)^{-1} \bar{a}$. We conclude $w_2^* = w_4^* = 0$. It follows from the optimality conditions that

$$\begin{array}{c} w^* = w_1^* + w_3^* = (\Lambda - \lambda^* I)^{-1} \bar{a} = (\Lambda_1 + \Lambda_3 - \lambda^* I)^{-1} \bar{a}, \\ \|w^*\| = s, \quad (\Lambda - \lambda^* I) \succeq 0, \quad (\Lambda_1 + \Lambda_3 - \lambda^* I) \succeq 0. \end{array} \tag{2.6}$$

   Therefore, w^* is still optimal for TRS_P if we set $\Lambda_2 = \Lambda_4 = 0$, i.e. x^* is still optimal if we set $A_2 = A_4 = 0$.

   Conversely, suppose that x^* is optimal if we set $A_2 = A_4 = 0$. Then (2.6) still holds, i.e. w^* is optimal for TRS_P.

   This completes the proof of Item 1.

2. Note that $S_3 = \emptyset$ implies $i_0 \in S_4$, since $S_3 \cup S_4 = \{i : \lambda_i(A) = \lambda_1(A)\}$. Necessity: Assume that (x^*, λ^*) with $x^* = u^* + z, z = \alpha v_{i_0} \in \mathcal{N}(A - \lambda^* I)$, solves TRS. Therefore, $w^* = P^T u^* + P^T z, P^T z \in \mathcal{N}(\Lambda - \lambda^* I)$

solves TRS_P. From the definitions, $\bar{a}_i = 0, \forall i \in S_2 \cup S_4$. It follows from the optimality conditions that

$$\begin{array}{c} w^* = P^T u^* + P^T z = (\Lambda_1 + \lambda_{i_0} e_{i_0} e_{i_0}^T - \lambda^* I)^\dagger \bar{a} + P^T z, \\ \lambda^* \left(\|P^T u^*\|^2 + \|P^T z\|^2 - s^2 \right) = 0, \quad (\Lambda - \lambda^* I) \succeq 0. \end{array} \tag{2.7}$$

The conclusion follows.

To see the converse, we reverse the above steps. 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 $w^* = (\Lambda - \lambda^* I)^\dagger \bar{a} + P^T z, P^T z \in \mathcal{N}(\Lambda - \lambda^* I)$. It follows from the optimality conditions that

$$\begin{array}{c} w^* = (\Lambda - \lambda^* I)^\dagger \bar{a} + P^T z = P^T u^* + P^T z, \\ \lambda^* \left(\|P^T u^*\|^2 + \|P^T z\|^2 - s^2 \right) = 0, \quad (\Lambda - \lambda^* I) \succeq 0. \end{array} \tag{2.8}$$

By adding and subtracting $\lambda_1(A)$, we see that $(P^T u^*, \lambda^* - \lambda_1(A))$ 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

$$P^T u^* = (\Lambda - \lambda^* I)^\dagger \bar{a}, \quad \|u^*\| \leq s, \quad (\Lambda - \lambda_1(A) I) \succeq 0. \tag{2.9}$$

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.

4. Note that if the hard case holds, since we assume $\lambda_1(A) \geq 0$, this implies $\lambda^* = 0$ and the hard case (case 2(ii)) does not hold. Therefore, the equivalence is clear by considering TRS_P, i.e. $w^* = (\Lambda - \lambda^* I)^\dagger \bar{a} = w^* = (\Lambda_1 + \Lambda_3 - \lambda^* I)^\dagger \bar{a}$. (See also the proof of Item 1.)

5. Assume that $x^* = P w^*$ solves TRS and $v_i^T x^* \neq 0$, for some $i \in S_4$. Equivalently, $e_i^T w^* \neq 0$, 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 $E_4 v \neq 0$. The conclusion follows.

■

Remark 2.2. *Assume that $\lambda_1(A) < 0$. We can draw several conclusions from the above lemma. In particular, we can shift using the eigenvectors and avoid the hard case (case 2). The items from the Lemma yield:*

1. *We can ignore eigenvectors that satisfy $v_i \perp a$ if $\lambda_i(A) > \lambda_1(A)$, e.g. we can set them to 0. (We can deflate all of them.)*

2. *We can ignore eigenvectors that satisfy $v_i \perp a$ if $\lambda_i(A) = \lambda_1(A), i \neq 1$, e.g. we can set them to 0. (We can deflate all but one of them.)*

3. *We can shift the eigenvalues by replacing A with $A - \lambda_1(A)I$. This guarantees that the hard case (case 2(ii)) does not hold. (See also Lemma 2.3 below.)*

4. *After the shift, we can further perturb A to obtain $S_4 = \emptyset$ so that the hard case (case 2) does not hold.*

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

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

Proof:

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

This proves the right-hand-side inequality.
Since x^* is optimal for TRS and on the boundary of the ball, we get

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

■

The literature often labels the hard case (case 2) as an ill-posed or degenerate problem, e.g. [13, 14]. Suppose that we consider the perturbed problem

$$(\mathrm{TRS}_\epsilon) \qquad \begin{array}{rll} q^*(\epsilon) := & \min & q_\epsilon(x) := x^T A(\epsilon) x - 2a(\epsilon)^T x \\ & \text{s.t.} & \|x\| \leq s(\epsilon), \end{array} \tag{2.10}$$

where the perturbed data $v(\epsilon) = (A(\epsilon), a(\epsilon), s(\epsilon))$ are sufficiently smooth functions of the perturbation vector ϵ. Except for the hard case (case 2), we get a unique solution from the linear equation

$$(A(\epsilon) - \lambda^*(\epsilon) I) x^*(\epsilon) = a(\epsilon), \tag{2.11}$$

since $(A(\epsilon) - \lambda^*(\epsilon) I) \succ 0$. This appears to imply that we can find a condition number for TRS based on the condition number of the Hessian of the Lagrangian, $\mathrm{cond}(A(0) - \lambda^*(0) I)$. If this were true, then we would conclude that TRS in the near hard case (case 2) is an ill-conditioned problem. However, this ignores the norm constraint $\|x^*\| \leq s$. For example, when $\lambda_1(A(0)) < 0$ and ϵ is small, the norm constraint implies that

$$\frac{\|x^*(\epsilon) - x(0)\|}{\|x(0)\|} \leq \frac{2(s + \epsilon)}{s},$$

independent of the other data. More precisely, suppose that the easy case holds. As s increases, $\lambda^* \to \lambda_1$, $\|x^*\| \to \infty$ and the relative error of the system (2.11) stays bounded. If the hard case holds, then as s increases we still have $\lambda^* \to \lambda_1$. However, there exists an $\bar{s}$ such that $s \geq \bar{s}$ implies $\lambda^* = \lambda_1$ and the optimum x^* can be found from the *bounded* best least squares solution $\bar{x} = (A - \lambda_1(A) I)^\dagger a$ plus a scaled eigenvector z.

Adding a norm constraint to an ill-posed problem is a well-known regularization processes, e.g. [32]. 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, as mentioned above, the symmetric eigenvalue problem has a condition number of 1, [6]. Then TRS can be shown to be equivalent to a linearly constraint convex programming problem, see [2]. The following lemma and example illustrate that TRS is always a stable program.

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

$$(TRS_s) \qquad \begin{array}{rll} q_s^* := & \min & q_s(x) := x^T (A - \lambda_1(A) I) x - 2a^T x + s^2 \lambda_1(A) \\ & s.t. & \|x\| \leq s. \end{array} \tag{2.12}$$

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 CQ holds are called *stable*, e.g. [11, 7]. 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. [7, 11].

We now summarize a procedure for solving the hard case TRS. We first find $\lambda_1(A)$ (a well-conditioned problem). If $\lambda_1(A) > 0$, then the hard case, case 2 does not hold. If $\lambda_1(A) \leq 0$, then we shift and obtain the convex program TRS_s. We then solve the *best least squares problem* (least squares solution of minimum norm) and check $\|u^*\| = \|(A - \lambda_1(A)I)^\dagger a\| < s$. If not, then the hard case, case 2(ii) does not hold. Otherwise, we find an eigenvector $z \in \mathcal{N}(A - \lambda^* I)$ so that $\|u^* + z\| = s$. Thus, the condition number of TRS in the hard case depends on the condition number of finding an eigenvector z and the best least squares solution u^*. For example, we know that $A - \lambda_1(A)I$ is singular. If the next eigenvalue is well separated, $\lambda_2(A) >> \lambda_1(A)$, then u^* can be found and the problem is well-conditioned. (See e.g. [6, Section 3.5.1] for a detailed discussion.)

Example 2.1. *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

$$x^* = x^*(0) = \bar{x} + \begin{bmatrix} 0 \\ \pm 1 \end{bmatrix} = \begin{bmatrix} 1 \\ \pm 1 \end{bmatrix}. \tag{2.13}$$

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

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

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

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

Depending on the choice of sign, these solutions converge to a solution in (2.13), as the perturbations converge to zero. Moreover, a Taylor series expansion shows that $\|x^*(0) - x^*(\alpha,\gamma,\delta)\| \leq 2(|\alpha|+|\gamma|+|\delta|)$ for small perturbations.

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

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

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 $\frac{2+\alpha}{1+\gamma-\lambda^*} \to 1$, as the perturbations go to 0, we see that the optimal solutions converge appropriately.

3 The Moré-Sorensen (MS) Algorithm

This algorithm features efficient handling of the easy and the hard case. It is dual-based in that it iterates on the dual variable λ. The main work in the

iterations is a Cholesky factorization used in the evaluation of a Newton step for λ, as well as in a 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. In the latter case, 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

$$q(\bar{x}) - q^* \leq \sigma_1(2-\sigma_1)\max\{|q^*|, \sigma_2\} \quad , \quad \|\bar{x}\| \leq (1+\sigma_1)\Delta. \tag{3.1}$$

Outline:
(i) Use a safeguarding and updating procedure to reduce the interval of uncertainty $[\lambda_L, \lambda_U]$ for λ^* and improve the upper bound λ_S for $\lambda_1(A)$.
(ii) Take a Newton step to implicitly solve $\|(A-\lambda I)^{-1}a\| = s$ for λ.
(iii) If the possibility of the hard case (case 2) is detected ($\|x(\lambda)\| < s$), take a *primal step* to the boundary while simultaneously reducing the objective function.
(iv) If a unique unconstrained minimizer exists and is in the strict interior of the trust region, then in at most two iterations the algorithm will terminate and find this optimal solution.

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

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

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

$$\phi(\lambda) := \frac{1}{s} - \frac{1}{\|x(\lambda)\|} = 0. \tag{3.2}$$

(See Reinsch [24],[25] and Hebden [26].) 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 $(-\infty, \lambda_1(A))$.

In practice, the MS algorithm uses the algorithm below to compute the Newton sequence $\{\lambda_k\}$ whenever $\lambda_k < 0$ and strictly satisfies the strengthened second order optimality conditions $A - \lambda_k I \succ 0$ (so that the Cholesky factorization can be used).

Algorithm 3.1. *Assume $\lambda_k \leq 0$ and $A - \lambda_k I \succ 0$ (i.e. $\lambda_k < \lambda_1(A)$).*

1. *Factor $A - \lambda_k I = R^T R$ (Cholesky factorization).*
2. *Solve, for x, $R^T R x = a$ ($x = x(\lambda_k)$).*
3. *Solve, for y, $R^T y = x$.*
4. *Let $\lambda_{k+1} = \lambda_k - \left[\frac{\|x\|}{\|y\|}\right]^2 \left[\frac{(\|x\|-s)}{s}\right]$ (Newton step).*

If a unique unconstrained minimizer exists in the strict interior of the trust region, the optimal λ^* is 0. If the initial λ is strictly less than 0, since A is positive definite and $A - \lambda I \succ 0$, the convexity and monotonicity of ϕ implies that the next Newton iterate for λ is strictly greater than 0. The safeguarding scheme then resets λ to the upper bound λ_U which is now 0 and the algorithm terminates in at most two iterations ([20, Pg 562]).

If the optimal solution is on the boundary of the trust region and one can find a λ_0 such that $\lambda_0 < \lambda_1(A)$ and $\phi(\lambda_0) > 0$, then Algorithm 3.1 has q-quadratic convergence asymptotically, again by the convexity and monotonicity of ϕ on $(-\infty, \lambda_1(A))$ and since ϕ is negative for large negative values of λ. Hence it has a unique root and Newton's method converges monotonically when initiated from λ_0. We illustrate this in Figure 3.1. The current intersection with the x-axis of the tangent line to the convex function yields the next iterate in Newton's method. However, if $\phi(\lambda_0) < 0$, then the Newton step may not provide a good prediction of a root, since the dual constraints on λ are not taken into account when evaluating the Newton step. (This is discussed in the RW approach below.)

Generically it is always possible to find a $\lambda_0 < \lambda_1(A)$ with $\phi(\lambda_0) > 0$, since small perturbations of the data avoid the orthogonality condition of the hard case. In the easy case $\|x(\lambda)\|$ is a function that takes all values from 0 to ∞ when λ varies from $-\infty$ to $\lambda_1(A)$, and thus, for $\lambda^* < \lambda < \lambda_1(A)$, $\phi(\lambda)$ is

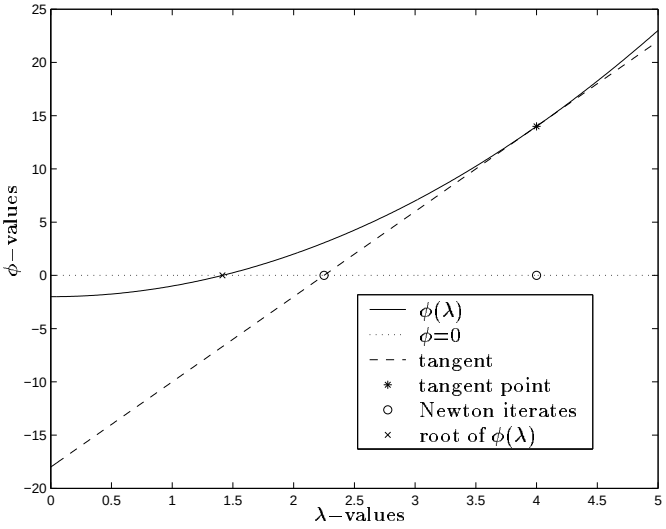

Figure 3.1: Newton's method with the secular function, $\phi(\lambda)$.

positive. Yet we have to be careful when the orthogonality condition almost holds, the so-called *almost hard case*. In the almost hard case, problems occur if λ^* is close to $\lambda_1(A)$, since the λ for which $\lambda < \lambda_1(A)$ and $\phi(\lambda) > 0$ are contained in a small interval, i.e. it is hard to find the λ_0 discussed above. Furthermore, Algorithm 3.1 may have computational difficulties, since the matrices $A - \lambda_k I$ become ill-conditioned. On the other hand, in the easy case, the function is smooth near λ^*. Fortunately, the MS algorithm has an efficient way to handle the hard case.

3.1 Handling the Hard Case in MS Algorithm

From Figure 3.1, we get the following indicator of the easy case for TRS.

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

However, Figure 3.1 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 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

$$(A - \lambda_1(A)I)x = a \tag{3.3}$$

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.

$$x^* = x(\lambda_1(A)) + \tau z \tag{3.4}$$

satisfies the optimality conditions. The following lemma by Moré-Sorensen [20] 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*

$$A - \lambda I = R^T R, \quad (A - \lambda I)x = a, \quad \lambda \leq 0. \tag{3.5}$$

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

$$\|x + z\|^2 = s^2 \tag{3.6}$$

and

$$\|Rz\|^2 \leq \sigma(\|Rx\|^2 - \lambda s^2). \tag{3.7}$$

Then

$$|q(x + z) - q(x^*)| \leq \sigma |q(x^*)|. \tag{3.8}$$

where x^* is optimal for TRS. ■

We will get back to this Lemma in Section 6.1 below, where we show that this lemma is measuring a duality gap. Note that (3.5) and (3.6) guarantee the dual and primal feasibility constraints in the optimality conditions (2.1).

A consequence of this lemma is that if we can choose a z with a corresponding σ that is small, then $x + z$ is nearly optimal. In the almost hard case (case 2), when λ is close to $\lambda_1(A)$, we can expect this to happen as R will be nearly singular. The authors used a LINPACK (now upgraded to LAPACK) technique (see [20, p.571], and NMTR at NEOS: www-neos.mcs.anl.gov/neos/solvers/UCO:NMTR/) to construct a vector $\hat{z}$ such that $\|R\hat{z}\|$ is small. A vector e, where each component of e has an absolute

value of 1, is constructed such that the solution w to $R^T w = e$ is large. Then, a solution v to $Rv = w$ is obtained and $\hat{z} := v/\|v\|$. Recall $A - \lambda I = R^T R$. If R is singular, $R\hat{z} = 0$ is obtained. Otherwise

$$\|R\hat{z}\| \leq \sqrt{n}(1+\rho)\min\{R_{kk} : 1 \leq k \leq n\}, \tag{3.9}$$

where n is the dimension of A, ρ is an upper bound on the sum of the absolute value of the components of R and the R_{kk}'s are the diagonal elements of R. If λ is near $\lambda_1(A)$, some element on the diagonal of R is near zero and $\|R\hat{z}\|$ is near zero. Moreover, given a feasible solution inside the trust region, the direction $\pm\hat{z}$ which improves the objective function value is chosen.

3.2 Summary of MS Algorithm

The MS Algorithm tries to solve TRS using Newton's method Algorithm 3.1 applied to the secular equation $\phi(\lambda) = 0$, (3.2). This is a dual algorithm in the sense that it is iterating on the dual variable λ. However, the two dual inequality constraints in the optimality conditions (2.1) are not taken into account when finding the Newton step. Instead, a safeguarding and updating scheme either reduces the interval of uncertainty for λ or finds a value of λ where quadratic convergence takes over; in both cases these inequality constraints are eventually satisfied up to a tolerance, i.e. the algorithm maintains dual feasibility using heuristic schemes. Furthermore, the algorithm is such that if the solution to TRS is unique and lies inside the trust region (i.e. it is a unique unconstrained minimum), then after at most two iterations, $\lambda = 0$ is tried and an optimal solution is found. If the almost hard case (case 2) is detected, i.e. $x(\lambda)$ lies in the interior of the trust region, then a primal step to the boundary is taken and the stopping criteria is checked. Few iterations are needed in practice if this case occurs. (Typically, only 2-3 iterations are needed for low accuracy solutions. However, high accuracy solutions can result in many backtracking steps since the information from the primal steps are lost at the end of each iteration.) The algorithm has been successfully implemented and is available using the NEOS Server with URL: www-neos.mcs.anl.gov/neos/solvers/UCO:NMTR/.

Many advances have been made in recent years in the efficiency of sparse Cholesky factorizations. This was stimulated by its use within interior-point methods. However, this step can still be a bottleneck for large sparse problems. For example, if a few dense rows are not handled properly, then the factorization can be dense, see e.g. [12].

Further details are included within the semidefinite framework in Section 6.1 below, as well as in the original paper [20] and in [4, Chapter 7]. Comparisons with the RW algorithm are included in Section 7 below.

4 The Generalized Lanczos Trust Region (GLTR) Algorithm

We now present the GLTR method of Gould, Lucidi, Roma and Toint [13]. As mentioned above, current attempts to solve TRS focus on exploiting sparsity. The main work of the GLTR algorithm involves a Lanczos tridiagonalization of the matrix A. Therefore, this method requires only matrix-vector multiplications and exploits sparsity in A. The method solves a *sequence* of restricted problems

$$\begin{array}{ll} \min & q(x) \\ \text{s.t.} & \|x\| \leq s \\ & x \in S, \end{array} \tag{4.1}$$

where the S are specially chosen (Krylov) subspaces of $\mathbb{R}^n$. (A similar approach based on Krylov subspaces is presented in [14].) The way S is chosen is inspired by the Steihaug algorithm of [29] (see also [13] and [33]). The conjugate-gradient method is applied to the quadratic function $q(\cdot)$ until the piecewise linear path, formed by connecting the conjugate-gradient iterates, hits the boundary. This will occur unless a global minimizer exists for $q(\cdot)$ and lies in the interior of the trust region or unless a direction of nonpositive curvature is detected for the Hessian A (i.e. for some direction p, $\langle p, Ap\rangle \leq 0$). The algorithm then returns either an approximate global minimizer or, in the latter case, moves to the boundary in the direction of the nonpositive curvature.

Suppose that x_k lies inside the trust region boundary, i.e. it is obtained using the conjugate gradient method and suppose that the method cannot produce another iterate inside the trust region (either a direction of negative curvature is detected or the next iterate lies outside of the trust region). Then the GLTR method continues its search of an approximate solution to the TRS by solving the subproblem

$$\begin{array}{ll} \min & q(x) \\ \text{s.t.} & \|x\| \leq s \\ & x \in S \equiv \mathcal{K}_k, \end{array} \tag{4.2}$$

where

$$\mathcal{K}_k := \text{span}\{a, Aa, A^2a, A^3a, \ldots, A^ka\}. \tag{4.3}$$

It then keeps increasing k, or equivalently the size of the Krylov subspace S, and solves problems of the type (4.2) and terminates when a good approximate solution to the TRS is found (see the stopping criteria in Algorithm 4.1). The efficiency of the algorithm is based on efficient approximate solutions of (4.2). Using the basis of $\mathcal{K}$ given by the orthonormal columns of the matrix Q_k in (4.5) below, and the substitution $x = Q_k h$, [13] shows that problem (4.2) is equivalent to the following tridiagonal trust region subproblem

$$\begin{array}{ll} \min & h^T T_k h - 2\gamma_0 e_1^T h \\ \text{s.t.} & \|h\| \leq s, \end{array} \tag{4.4}$$

where e_1 is the first column of the identity matrix of dimension $k+1$, T_k is the tridiagonal matrix

$$T_k = \begin{bmatrix} \delta_0 & \gamma_1 & & & \\ \gamma_1 & \delta_1 & . & & \\ & . & . & . & \\ & & . & \delta_{k-1} & \gamma_k \\ & & & \gamma_k & \delta_k \end{bmatrix},$$

$\delta_k = q_k^T A q_k$, $\gamma_0 = \|a\|$, $q_{-1} = 0$, $q_0 = a/\|a\|$, $\gamma_k = \|(A - q_{k-1}^T A q_{k-1} I)q_{k-1} - \gamma_{k-1} q_{k-2}\|$, and $q_k = ((A - q_{k-1}^T A q_{k-1} I)q_{k-1} - \gamma_{k-1} q_{k-2})/\gamma_k$, for $k \geq 1$. An optimal solution h_k to (4.4) yields an optimal solution $x_k = Q_k h_k$, where

$$Q_k := (q_0, q_1, \ldots, q_k) \tag{4.5}$$

and where $\{q_0, q_1, \ldots, q_k\}$ is an orthonormal basis for $S \equiv \mathcal{K}_k$. The Lanczos method (see [12]) can be used to build the *Lanczos vectors* q_i for $i = 1 \ldots k$, but as mentioned in [13], these vectors can be recovered from the conjugate gradient iterates (see equation (4.9) below), since the conjugate gradient and Lanczos method are intimately linked and generate different bases of the same Krylov space $\mathcal{K}_k$.

The advantage of solving problem (4.4) is that Algorithm 3.1 may be used to find the approximate optimum, even for large problems, since the Cholesky factorization can take advantage of the tridiagonal form of T_k to preserve sparsity. The algorithm implicitly searches for a λ_k and h_k which satisfy the conditions

$$(T_k + \lambda_k I_{k+1})h_k = \gamma_0 e_1 \quad \|h_k\| = s,$$

where $T_k + \lambda_k I_{k+1}$ is positive definite. The authors use heuristics to find a starting value of λ, for Algorithm 3.1, which satisfies

$$\lambda \leq 0, \lambda \leq \lambda_1(T_k) \quad \text{and} \quad \frac{1}{s} - \frac{1}{\|(T_k - \lambda I)^{-1}\gamma_0 e_1\|} > 0. \tag{4.6}$$

Such a starting value of λ ensures asymptotic q-quadratic convergence of the Newton iteration to λ_k.

It is shown in [13] that

$$(A - \lambda_k I)x_k - a = \gamma_{k+1} e_{k+1}^T h_k q_{k+1} \tag{4.7}$$

$$\text{and} \quad \|(A - \lambda_k I)x_k - a\| = \gamma_{k+1}|e_{k+1}^T h_k|, \tag{4.8}$$

where e_{k+1} is the last column of the identity matrix of dimension $k+1$. Therefore, when $\gamma_{k+1}|e_{k+1}^T h_k|$ is small, the relationship $(A - \lambda_k I)x_k = a$ is almost satisfied and x_k is considered a good approximation to an optimal solution of TRS.

4.1 Handling the Near Hard Case in GLTR Algorithm

If the near hard case (case 2) occurs for (4.4), then finding a λ that satisfies (4.6) may be difficult and time consuming, since the interval for the λ which satisfy the conditions (4.6) can be small. This is illustrated in Figure 4.1. (The GLTR algorithm fails if the hard case (case 2) occurs.) In addition, ill-conditioning will slow down both the Algorithm 3.1 and therefore the GLTR method (see [13], pp. 515).

4.2 Summary of GLTR Algorithm

Algorithm 4.1 (GLTR Method). *Let $x_0 = 0$, $g_0 = -a$, $\gamma_0 = \|g_0\|$ and $p_0 = -g_0$. Set the flag INTERIOR as true. For $k = 0, 1, \ldots$ until convergence, perform the iteration:*

$\alpha_k = \|g_k\|^2/(p_k^T A p_k)$.
Obtain T_k from T_{k-1}.
If INTERIOR is true, but $\alpha_k \leq 0$ or $\|x_k + \alpha_k p_k\|^2 \geq s^2$,
reset INTERIOR to false.
If INTERIOR is true,
$x_{k+1} = x_k + \alpha_k p_k$,

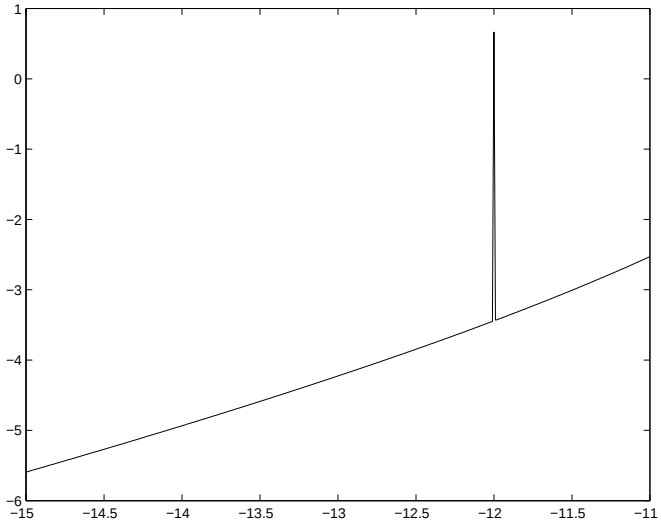

Figure 4.1: An example of the function $\phi(\lambda)$, for the problem (4.4) in the near hard case (case 2). In this example, $\lambda_1(T_k) = -12$. The figure illustrates for this case that the values of λ for which $\phi(\lambda) > 0$ are contained in a small interval.

else

solve the tridiagonal trust region subproblem (4.4), using Algorithm 3.1, to obtain h_k.

end if

$g_{k+1} = g_k + \alpha_k A p_k$.

If INTERIOR is true,

stop if $\|g_{k+1}\| < \max(10^{-8}, 10^{-5}\|a\|)$,

else

stop if $\gamma_{k+1}|e_{k+1}^T h_k| < \max(10^{-8}, 10^{-5}\|a\|)$.

end if

$\beta_k = \|g_{k+1}\|^2 / \|g_k\|^2$.

$p_{k+1} = -g_{k+1} + \beta_k p_k$.

If INTERIOR is false, set $x_k = Q_k h_k$.

The Lanczos vectors are obtained from the relation

$$q_k = \sigma_k g_k / \|g_k\|, \text{ where } \sigma_k = -\text{sign}(\alpha_{k-1})\sigma_{k-1} \text{ and } \sigma_0 = 1. \tag{4.9}$$

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

As mentioned above, the method in [14] is similar in spirit to GLTR. The main difference is the choice of vectors to put into the 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. In this section, we focus our attention on different dual programs associated with TRS. We will show below that both the MS and GLTR algorithms can be explained using dual programs and that these explanations reveal some of the weaknesses of the 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 $\lambda \leq 0$.) Precisely, consider the slightly different problem

$$(\mathrm{TRS}_{=}) \qquad \begin{array}{rll} q^* = & \min & q(x) \\ & \text{s.t.} & \|x\| = s. \end{array} \tag{5.1}$$

A key result shown in [30] is that strong Lagrangian duality holds for this problem (see also [31]). In fact, [30] shows the deeper result that strong duality holds if the constraint in $\mathrm{TRS}_{=}$ is replaced by the more general constraint $\beta \leq x^T C x \leq \alpha$, where C is a symmetric matrix (no definiteness is assumed) and where β and α are constants such that $\beta < \alpha$. (This allows for indefinite inner product scaling. See also [19].) This property of trust region subproblems permits the construction of equivalent dual problems. These duals, closely related to semidefinite programs, are the key for new insights into the previous algorithms. The duals have been used in [27] to explain the steps of the MS algorithm. We recall some of their results in Section 6.1. This can also be done for the GLTR method, see Section 6.2, where we show that their stopping criteria is in fact measuring a duality gap between a pair of dual programs.

5.1 Lagrangian Duality and SDP

From [30], we know that strong Lagrangian duality holds for $\text{TRS}_=$, i.e.

$$q^* = \nu^* := \max_{\lambda} \min_{x} L(x, \lambda), \tag{5.2}$$

where the Lagrangian is $L(x, \lambda) := x^T(A - \lambda I)x - 2a^T x + \lambda s^2$. Since $L(x, \lambda)$ is a quadratic, the inner min problem is unbounded below unless the hidden constraints,

$$A - \lambda I \succeq 0, \quad a \in \mathcal{R}(A - \lambda I), \tag{5.3}$$

hold. (This can be seen by moving in a direction of an eigenvector corresponding to a negative eigenvalue or, if $A - \lambda I \succeq 0, \quad a \notin \mathcal{R}(A - \lambda I)$, then moving in a direction $d \in \mathcal{N}(A - \lambda I)$ such that $d^T a > 0$.) 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 min is attained if bounded, i.e. define (Recall Remark 2.1 and the definition of the Moore-Penrose generalized inverse.)

$$h(\lambda) := \lambda s^2 - a^T(A - \lambda I)^\dagger a.$$

We get

$$\min_{x} L(x, \lambda) = L((A - \lambda I)^\dagger a, \lambda) = h(\lambda), \quad \text{if } A - \lambda I \succeq 0, a \in \mathcal{R}(A - \lambda I). \tag{5.4}$$

Therefore we have to modify the conjectured dual (2.2) (in the $\text{TRS}_=$ case), i.e. we use (5.4) and get an equivalent expression to the strong duality (5.2)

$$q^* = \max h(\lambda) \text{ s.t. } A - \lambda I \succeq 0, a \in \mathcal{R}(A - \lambda I). \tag{5.5}$$

In the hard case, $a \in \mathcal{R}(A - \lambda_1(A)I)$, we deduce

$$q^* = \max_{A - \lambda I \succeq 0} h(\lambda), \quad \text{if the hard case (case 2) holds.} \tag{5.6}$$

However, possible discontinuity in $h(\lambda)$ at $\lambda = \lambda_1(A)$, in the case $A - \lambda^* I \succ 0$, requires the equivalent dual

$$q^* = L(x^*, \lambda^*) = \max_{\lambda} \min_{x} L(x, \lambda) = \sup_{A - \lambda I \succ 0} \min_{x} L(x, \lambda) = \sup_{A - \lambda I \succ 0} h(\lambda). \tag{5.7}$$

Problems (5.6) and (5.7) are similar and the next theorem, which can be found in [30], combines both problems into a single one.

Theorem 5.1. *The Lagrangian dual for $TRS_=$ is*

$$(D) \qquad q^* = \sup_{A-\lambda I \succ 0} h(\lambda). \tag{5.8}$$

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

Proof: Let λ_l be the smallest eigenvalue of A such that $a \not\perp \mathcal{N}(A-\lambda_l(A)I)$. Such a λ_l may not exist, but this implies that $a = 0$ and $h(\lambda) = \lambda s^2$ and the theorem trivially holds. Thus, assume λ_l exists. Note that in the easy case, $\lambda_l = \lambda_1(A)$. Also, let $A = Q\Lambda Q^T$, where Q is an orthogonal matrix with columns formed from the normalized eigenvectors of A, and Λ is a diagonal matrix having the eigenvalues of A on its diagonal in nondecreasing order, that is

$$\Lambda_{11} = \lambda_1(A) \leq \ldots \leq \Lambda_{nn} = \lambda_n(A).$$

Since $a \perp \mathcal{N}(A - \lambda_j(A)I)$ for $j = 1 \ldots l-1$, we see that a is perpendicular to $q_1 \ldots q_{l-1}$, where q_j is the j-th column of Q. Hence, $(Q^T a)_j = \gamma_j = 0$, for $j = 1 \ldots l-1$. Now let (λ_w) be a sequence converging to $\tilde{\lambda} \in (-\infty, \lambda_l]$ such that, for all w, $\lambda_w \in (-\infty, \lambda_l)$ and $A - \lambda_w I$ is invertible. Then

$$h(\lambda_w) = -a^T(A-\lambda_w I)^{-1}a + \lambda_w s^2 = -(Q^T a)^T(\Lambda - \lambda_w I)^{-1}Q^T a + \lambda_w s^2$$

$$= -\sum_{j=l}^{n} \frac{\gamma_j^2}{(\lambda_j - \lambda_w)} + \lambda_w s^2. \tag{5.9}$$

Note that $\lambda_j - \lambda_w > 0$ for $j = l \ldots n$. Note also that by the definition of λ_l, if r is the multiplicity of λ_l (i.e. $\lambda_{l-1} < \lambda_l = \lambda_{l+1} = \ldots = \lambda_{l+r-1} < \lambda_{l+r}$), then there exist $\gamma_j \neq 0$ for $j \in \{l, l+1 \ldots l+r-1\}$. As a consequence, when $\tilde{\lambda} = \lambda_l$, we have $h(\lambda_w) \to -\infty$. Therefore, $h(\cdot)$ has a vertical asymptote and is not continuous in λ_l.
When $\tilde{\lambda} \in (-\infty, \lambda_l)$, by equation (5.9) and since $\lambda_j - \tilde{\lambda} > 0$ for $j = l \ldots n$ we have

$$-a^T(A-\lambda_w I)^{-1}a + \lambda_w s^2 \to -a^T(A-\tilde{\lambda}I)^{\dagger}a + \tilde{\lambda}s^2,$$

i.e. $h(\cdot)$ is a continuous function over $(-\infty, \lambda_l)$. In the hard case, since $\lambda_l > \lambda_1(A)$, then $h(\cdot)$ is a continuous function over $(-\infty, \lambda_1(A)]$. Therefore

$$q^* = \max_{A-\lambda I \succeq 0} h(\lambda) = \sup_{A-\lambda I \succ 0} h(\lambda).$$

Combining this result with (5.7) yields the dual program (D). Equation (5.7) also implies that in the easy case and the hard case (case 1) the sup can be replaced by a max. ■

Corollary 5.1. *The Lagrangian dual for TRS is equivalent to*

$$(D) \qquad q^* = \sup_{\substack{A - \lambda I \succ 0 \\ \lambda \leq 0}} h(\lambda) \tag{5.10}$$

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

■

Note that $h(\cdot)$ is a concave function on $(-\infty, \lambda_l)$, which diverges to $-\infty$ as $\lambda \to \lambda_l$, where λ_l is as in the above proof, i.e.

$$h'(\lambda) = s^2 - a^T(A - \lambda I)^{-2}a = s^2 - \sum_{j=l}^{n} \frac{\gamma_j^2}{(\lambda_j - \lambda)^2},$$

$$h''(\lambda) = -2a^T(A - \lambda I)^{-3}a = -2\sum_{j=l}^{n} \frac{\gamma_j^2}{(\lambda_j - \lambda)^3}.$$

For $\lambda_j - \lambda > 0$, we conclude that

$$h''(\lambda) = -2a^T((A - \lambda I)^\dagger)^3 a \leq 0.$$

This shows that $\text{TRS}_=$ is equivalent to finding the supremum of a concave function over an open interval.

In addition, if the current point λ_c is on the *hard side* ($h'(\lambda_c) > 0$), we could use the derivative information to verify the existence of an easy side point, i.e. a point λ with $h'(\lambda) < 0$ is guaranteed to exist if

$$h'(\lambda_c) + h''(\lambda_c)(\lambda_1(A) - \lambda_c) < 0. \tag{5.11}$$

5.2 Dual of Dual in the Hard Case

The next dual we consider is the Lagrangian dual of the Lagrangian dual (5.1) when the hard case holds, i.e.

$$\begin{aligned} q^* &= \max_{A-\lambda I \succeq 0} h(\lambda) \\ &= \min_{X \succeq 0} \max_{\lambda < \lambda_l} h(\lambda) + \operatorname{trace} X(A - \lambda I). \end{aligned} \tag{5.12}$$

The inner max is a maximum of a concave function over an open interval. Therefore, we can differentiate with respect to λ and add the stationarity condition to replace the sup, i.e. we get the Wolfe type dual

$$(DD) \qquad \begin{array}{lll} q^* = & \inf & h(\lambda) + \operatorname{trace}(X(A - \lambda I)) \\ & \text{s.t.} & s^2 - a^T((A - \lambda I)^\dagger)^2 a - \operatorname{trace}(X) = 0 \\ & & \lambda < \lambda_l \\ & & \operatorname{trace}(X) \leq s^2 \\ & & X \succeq 0. \end{array} \tag{5.13}$$

Remark 5.1. *The equality constraint allows for* $s^2 - \|x(\lambda)\|^2 = \operatorname{trace} X > 0$, *which occurs in the hard case even at* $\lambda = \lambda^*$. *As mentioned above, we could add the constraint* $\lambda \leq 0$ *to get the corresponding dual for TRS.*

5.3 Unconstrained and Linear Duals

The dual problem (5.8) shows that $TRS_=$ is equivalent to finding the supremum of a concave function over an open interval. This dual problem will show that $\text{TRS}_=$ is also equivalent to the maximization of a single variable concave function over $\mathbb{R}$, i.e. an unconstrained concave maximization problem.

We start by homogenizing $\text{TRS}_=$ and obtain

$$\begin{array}{lll} q^* = & \min & x^T A x - 2 y_0 a^T x \\ & \text{s.t.} & \|x\|^2 = s^2, \\ & & y_0^2 = 1. \end{array} \tag{5.14}$$

Therefore

$$\begin{aligned}
q^* &= \max_t \min_{\|x\|^2=s^2, y_0^2=1} x^T Ax - 2y_0 a^T x + t(y_0^2 - 1) \\
&\geq \max_t \min_{\|x\|^2+y_0^2=s^2+1} x^T Ax - 2y_0 a^T x + t(y_0^2 - 1) \qquad (5.15) \\
&\geq \sup_{t,\lambda} \min_{x,y_0} x^T Ax - 2y_0 a^T x + t(y_0^2 - 1) + \lambda(\|x\|^2 + y_0^2 - s^2 - 1) \\
&= \sup_{r,\lambda} \min_{x,y_0} x^T Ax - 2y_0 a^T x + r(y_0^2 - 1) + \lambda(\|x\|^2 - s^2) \\
&= \sup_{\lambda} \left(\sup_{r} \min_{x,y_0} x^T Ax - 2y_0 a^T x + r(y_0^2 - 1) + \lambda(\|x\|^2 - s^2) \right),
\end{aligned}$$

where $r = t + \lambda$. Because strong duality holds ([30, Theorem 5.1]),

$$\begin{aligned}
q^* &= \sup_{\lambda} \left(\min_{x,y_0} \sup_{r} x^T Ax - 2y_0 a^T x + r(y_0^2 - 1) + \lambda(\|x\|^2 - s^2) \right) \\
&= \sup_{\lambda} \min_{x,y_0^2=1} x^T Ax - 2y_0 a^T x + \lambda(\|x\|^2 - s^2) \\
&= \min_{x,y_0^2=1} \sup_{\lambda} x^T Ax - 2y_0 a^T x + \lambda(\|x\|^2 - s^2) \\
&= \min_{\|x\|^2=s^2, y_0^2=1} x^T Ax - 2y_0 a^T x \\
&= q^*.
\end{aligned}$$

So all the above turn out to be equal. Now, if we consider (5.15), then

$$q^* = \max_t \min_{\|x\|^2+y_0^2=s^2+1} x^T Ax - 2y_0 a^T x + t(y_0^2 - 1)$$

$$= \max_t \min_{\|z\|^2=s^2+1} z^T D(t) z - t = \max_t (s^2+1)\lambda_1(D(t)) - t,$$

where $z = \begin{pmatrix} y_0 \\ x \end{pmatrix}$ and $D(t) = \begin{pmatrix} t & -a^T \\ -a & A \end{pmatrix}$. If we define

$$k(t) := (s^2+1)\lambda_1(D(t)) - t, \qquad (5.16)$$

then we have the following unconstrained dual problem for TRS:

$$\max_t k(t). \qquad (5.17)$$

One can show $\lambda_1(D(\cdot))$ is a concave function and therefore $k(\cdot)$ is concave as well. This shows that $\text{TRS}_=$ is equivalent to an unconstrained concave

maximization problem in one variable. We can also rewrite (5.17) in the following way so that it becomes a linear semidefinite program:

$$\max_{D(t)\succeq\lambda I} (s^2+1)\lambda - t. \tag{5.18}$$

Equivalently,

$$\begin{array}{rll} q^* = & \max & (s^2+1)\lambda - t \\ & \text{s.t.} & \lambda I - tE_{00} \preceq D(0), \end{array} \tag{5.19}$$

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 $\text{TRS}_=$ (which correspond to its semidefinite relaxation, but here the relaxation is exact, see e.g. [27]).

$$\begin{array}{rll} q^* = & \min & \operatorname{trace} D(0)Y \\ & \text{s.t.} & \operatorname{trace} Y = s^2+1 \\ & & -Y_{00} = -1 \\ & & Y \succeq 0. \end{array} \tag{5.20}$$

Theorem 5.2. *The Slater constraint qualification and strict complementarity hold for the primal-dual SDP pair* (5.19) *and* (5.20).

Proof: That Slater's CQ holds is clear, i.e. choose λ and Y appropriately. Now suppose that λ, t, Y are optimal for the SDP pair (5.19) and (5.20). Then $Z := D(t) - \lambda I \succeq 0$ and singular. Let k be the multiplicity of $\lambda_1(D(t))$ and $y_1, \ldots, y_k$ be an orthonormal basis for its eigenspace. Set $V = [y_1 \ \ldots \ y_k]$ and redefine $Y \leftarrow Y + VV^T$. We can scale DYD using a diagonal matrix D to guarantee that Y is feasible for (5.20). By construction $ZY = 0, Z + Y \succ 0$. ■

Corollary 5.2. *The SDP* (5.20) *has a rank one optimal solution* Y^*.

Proof: Let x^* be an optimum for TRS, $y^* = \begin{pmatrix} 1 \\ x^* \end{pmatrix}$, and $Y^* = y^*(y^*)^T$. ■

Remark 5.2. *This primal-dual linear pair provides important information for maximizing the concave function $k(t)$. In particular, it may be too expensive to find $\lambda_1(D(t))$ accurately. However, Slater's CQ and strict complementarity hold for both primal-dual SDP programs. Therefore, these SDPs can be solved to any desired accuracy in polynomial time, see e.g. [35], i.e. this shows that TRS is a well-posed problem, contrary to statements made in the literature. (See e.g. [13].) In fact, from the formulation* (5.17)*, solving TRS involves a parametric eigenvalue problem. Though the formulation* (5.17) *can involve a nondifferentiable concave function if the smallest eigenvalue is not a singleton, using* (5.19) *is a stable reformulation of TRS.*

6 Semidefinite Framework

Primal-dual interior-point 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. (The recent books [34, 36] 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 usually dense, see e.g. [1].

We can 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

We follow [27] and use the pair of dual programs

$$(D) \qquad q^* = \sup_{A-\lambda I \succ 0} h(\lambda) \tag{6.1}$$

and

$$
\begin{array}{rll}
q^* = & \inf & h(\lambda) + \operatorname{trace}(X(A-\lambda I)) \\
& \text{s.t.} & s^2 - a^T((A-\lambda I)^\dagger)^2 a - \operatorname{trace}(X) = 0 \\
(DD) & & \lambda < \lambda_l \\
& & \operatorname{trace}(X) \leq s^2 \\
& & X \succeq 0.
\end{array} \tag{6.2}
$$

We use (D) to explain the main step, i.e. how we change λ at each iteration of the MS algorithm. This completely explains the steps in the easy case. For the additional step done in the (near) hard case, we use the duality gap $\operatorname{trace}(X(A-\lambda I))$ between (D) and (DD).

In the easy case and the hard case (case 1), the sup in (6.1) can be replaced by a max, i.e. we are finding an unconstrained maximum in the open interval for λ defined by the positive definite constraint. Therefore we need to find a stationary point for the concave objective function in λ. We can differentiate and obtain, for $\lambda^* \in (-\infty, \lambda_1(A))$,

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

i.e. stationarity for (6.1) corresponds to feasibility for $\text{TRS}_=$. The MS algorithm solves this equation for λ^* by applying Newton's method on the function $\phi(\lambda)$ (see (3.2)), a less nonlinear function. As discussed in Section 3, if we have an iterate with $\phi(\lambda) > 0$, then we obtain quadratic (monotonic) convergence and $x(\lambda)$ is infeasible during the iterations. Therefore, we do not discuss a duality gap in this case.

Things are different in the hard (or near hard) case (case 2). The indication for this is $\phi(\lambda) < 0$ at the current iterate λ. Equivalently, this means that $\|x(\lambda)\| < s$, i.e. we have a strictly feasible iterate $x(\lambda)$ for the inequality constrained form TRS. Therefore, we can reduce the duality gap between our pair of dual programs. We use the same strategy to predict the new value for λ with safeguarding to maintain positive definiteness of $A - \lambda I$. However, as mentioned in (3.3) and (3.4), if we are given a feasible vector $x = x(\lambda_1(A)) = (A - \lambda_1(A)I)^\dagger a$, the idea to handle this case is to find a proper z to reduce the primal objective and move to the boundary (i.e. $\|x+z\|^2 = s^2$). The framework of Section (5.1) suggests how such a z should be chosen and the result follows from the equation

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

Using $x = (A-\lambda I)^\dagger a$ we get

$$q(x+z) = h(\lambda) + z^T(A-\lambda I)z.$$

This implies that z should be chosen to make $z^T(A-\lambda I)z$ small. For a fixed feasible λ for (6.1), the duality gap between (6.1) and (6.2) is dependent on X and equals trace $(X(A-\lambda I))$. If we set $X = zz^T$, then trace $(X(A-\lambda I)) = z^T(A-\lambda I))z$. Note that in the MS algorithm, $\|Rz\|^2 = z^T(A-\lambda I))z$. 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 (6.1) and (6.2), while maintaining feasibility for (6.2).

6.2 A Semidefinite Framework for the GLTR method

As in the previous section, we now show that the Lanczos method can also be explained using the Lagrangian dual (6.1). Here we show that their stopping criteria is in fact measuring the duality gap between TRS$_=$ and this Lagrangian dual.

Recall that the GLTR method solves a problem of the type (4.2) when it is known that the solution to the TRS lies on the boundary and that TRS$_=$ needs to be solved. In the GLTR algorithm, a solution x_k of (4.2) is said to be a good approximation of TRS if $\gamma_{k+1}|e_{k+1}^T h_k|$ is small (see (4.8)) and this is used as a stopping criteria for the algorithm. We now show the relationship between the duality gap and this stopping criteria.

Recall λ_k in the GLTR method is the root of the function $\phi(\lambda)$ obtained by applying Algorithm 3.1 to the problem (4.4). Since $\|x_k\|^2 = s^2$, we have

$$\begin{aligned} q(x_k) &= x_k^T A x_k - 2a^T x_k - \lambda_k(\|x_k\|^2 - s^2) \\ &= x_k^T(A-\lambda_k I)x_k - 2a^T x_k + \lambda_k s^2. \end{aligned}$$

If λ_k is feasible for (6.1), then equation (4.7) implies that $x_k = (A - \lambda_k I)^\dagger a + \gamma_{k+1} e_{k+1}^T h_k (A - \lambda_k I)^\dagger q_{k+1}$ and therefore

$$\begin{aligned} q(x_k) &= a^T (A - \lambda_k I)^\dagger (A - \lambda_k I)(A - \lambda_k I)^\dagger a \\ &\quad + \gamma_{k+1}^2 (e_{k+1}^T h_k)^2 q_{k+1}^T (A - \lambda_k I)^\dagger (A - \lambda_k I)(A - \lambda_k I)^\dagger q_{k+1} \\ &\quad + 2\gamma_{k+1} e_{k+1}^T h_k a^T (A - \lambda_k I)^\dagger (A - \lambda_k I)(A - \lambda_k I)^\dagger q_{k+1} \\ &\quad - 2a^T (A - \lambda_k I)^\dagger a - 2\gamma_{k+1} e_{k+1}^T h_k a^T (A - \lambda_k I)^\dagger q_{k+1} + \lambda_k s^2. \end{aligned}$$

Some simplification and the properties of the generalized inverse yield

$$\begin{aligned} q(x_k) &= -a^T (A - \lambda_k I)^\dagger a + \lambda_k s^2 + \gamma_{k+1}^2 (e_{k+1}^T h_k)^2 q_{k+1}^T (A - \lambda_k I)^\dagger q_{k+1} \\ &= h(\lambda_k) + (\gamma_{k+1} |e_{k+1}^T h_k|)^2 q_{k+1}^T (A - \lambda_k I)^\dagger q_{k+1}. \end{aligned}$$

Thus

$$q(x_k) - h(\lambda_k) = (\gamma_{k+1} |e_{k+1}^T h_k|)^2 q_{k+1}^T (A - \lambda_k I)^\dagger q_{k+1}.$$

If we assume λ_k feasible for (6.1), then the right hand term is the duality gap between $\text{TRS}_=$ and (6.1). When this term is small, we can therefore expect x_k to be almost optimal for TRS. This is in agreement with the stopping criteria for the GLTR method, since $\gamma_{k+1} |e_{k+1}^T h_k|$ appears in the duality gap. Note though that $q_{k+1}^T (A - \lambda_k I)^\dagger q_{k+1}$ is not taken into account in the measurement of the gap. Furthermore, for M a positive definite symmetric matrix, define the M-norm of a vector x as

$$\|x\|_M^2 := x^T M x.$$

Then, for $M = A - \lambda_k I$, the duality gap can be written in the form

$$q(x_k) - h(\lambda_k) = \|(\gamma_{k+1} |e_{k+1}^T h_k|) q_{k+1}\|_M^2 \ .$$

We see the stopping criteria used by the Lanczos method is an incomplete measure of the duality gap. This suggests that one might find some example where the GLTR method stops, but where $\|q_{k+1}\|_M^2$ is large enough so that the duality gap is also large. The semidefinite framework we showed here for the Lanczos Method, presents a clearer way to understand the algorithm. It shows that it is mostly a primal algorithm, since simpler primal problems are solved to approximate the solution to TRS. Yet, the strength of the approximation is directly linked to the duality gap between TRS and the dual problem (6.1). Furthermore, at each iteration, since feasibility is not assured for λ_k, the algorithm compares to a primal-dual infeasible algorithm.

7 The Rendl-Wolkowicz (RW) Algorithm Revisited

We now present our modified form of the RW algorithm. This algorithm exploits both the sparsity of A and handles the hard case (case 2). The algorithm is based on the unconstrained dual program (5.17) and exploits 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. We include information from the primal-dual pair of linear SDPs (5.19),(5.20). In Section 7.4 we also include new heuristics that take advantage of the structure of $k(\cdot)$, accelerate convergence, and facilitate the handling of the hard case. We summarize the algorithm in Sections 7.3, 7.4.

7.1 Three Useful Functions

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 (5.17). There is no equivalent function used in the MS algorithm, but we do show a connection using the derivative of $k(t)$. 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

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

Therefore, asymptotically, as $|t| \to \infty$, the behavior of $k(t)$ is linear. Since $\lambda_1(D(t))$ is concave, so is $k(t)$. In the easy case, the function is also differentiable. In the hard case, loss of differentiability occurs when the multiplicity of the smallest eigenvalue for $\lambda_1(D(t))$ changes. The three Figures 7.1, 7.2, 7.3 illustrate the three different situations. The following theorem tells us when this happens.

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

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

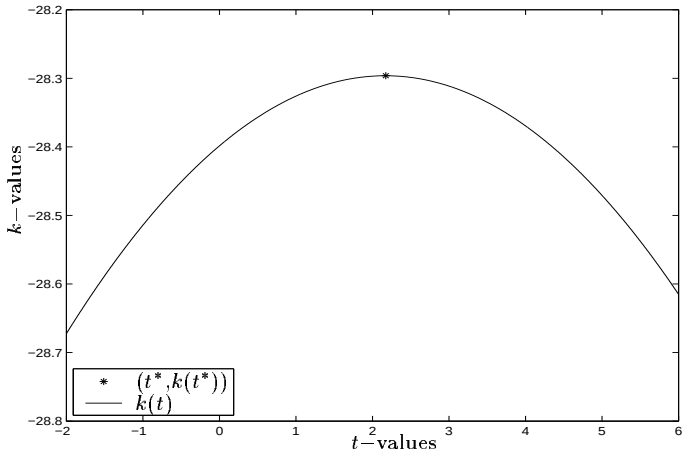

Figure 7.1: $k(t)$ in the easy case

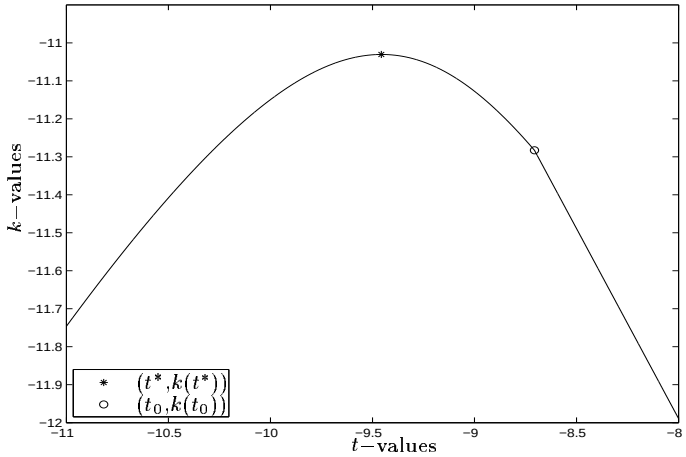

Figure 7.2: $k(t)$ in the hard case (case 1)

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* $1 + i$*;*

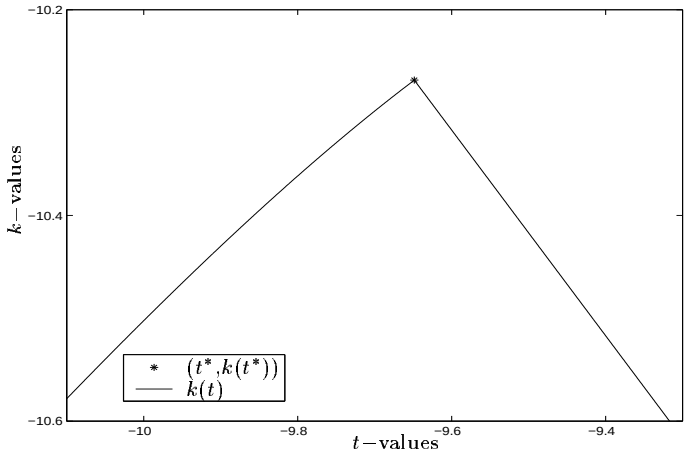

Figure 7.3: $k(t)$ in the hard case (case 2)

(c) for $t > t_0$, $\lambda_1(D(t)) = \lambda_1(A)$ and $\lambda_1(D(t))$ has multiplicity i.

Proof: We can assume without loss of generality that A is a diagonal matrix with diagonal elements α_k and i is the multiplicity of $\lambda_1(A)$, i.e.

$$\alpha_1 = \alpha_2 = \ldots = \alpha_i < \alpha_{i+1} \leq \ldots \leq \alpha_n.$$

In particular, we get the easy case if and only if $\exists j \in \{1 \ldots i\}$ such that $a_j \neq 0$. We then have

$$\det(D(t) - \lambda I) = (t - \lambda) \prod_{k=1}^{n} (\alpha_k - \lambda) - \sum_{k=1}^{n} \left(a_k^2 \prod_{j \neq k}^{n} (\alpha_j - \lambda) \right).$$

Let $J = \{i | a_i \neq 0\}$. Then

$$\det(D(t) - \lambda I) = \left(t - \left(\lambda + \sum_{j \in J} \frac{a_j^2}{\alpha_j - \lambda} \right) \right) \prod_{j=1}^{n} (\alpha_j - \lambda), \quad \text{for } \lambda \notin \{\alpha_j | j \in J\}.$$

For $\lambda \notin \{\alpha_j | j \in J\}$, define

$$d(\lambda) := \lambda + \sum_{j \in J} \frac{a_j^2}{\alpha_j - \lambda}.$$

Then

$$\det(D(t) - \lambda I) = (t - d(\lambda)) \prod_{j=1}^{n} (\alpha_j - \lambda), \quad \text{for } \lambda \notin \{\alpha_j | j \in J\}. \tag{7.1}$$

Note that the eigenvalues of A, the α_k's for $k \in \{1 \dots n\}$, are not necessarily eigenvalues for $D(t)$ since $d(\cdot)$ might not be defined for any of these values. Yet, if $\alpha_k \notin \{\alpha_j | j \in J\}$, then α_k is an eigenvalue for $D(t)$, since $d(\cdot)$ is well defined at α_k. In the easy case, there exists $\alpha_h \in \{\alpha_j | j \in J\}$ with $h \in \{1 \dots i\}$. Without loss of generality, assume $\alpha_1 \in \{\alpha_j | j \in J\}$. Therefore

$$\lim_{\lambda \to \alpha_1, \lambda < \alpha_1} d(\lambda) = \infty$$

and

$$\lim_{\lambda \to -\infty} d(\lambda) = -\infty.$$

Moreover,

$$d'(\lambda) = 1 + \sum_{j \in J} \frac{a_j^2}{(\alpha_j - \lambda)^2} > 0 \tag{7.2}$$

and

$$d''(\lambda) = \sum_{j \in J} \frac{a_j^2}{(\alpha_j - \lambda)^3} > 0, \quad \text{if } \lambda < \alpha_1.$$

Therefore, $d(\cdot)$ is strictly monotonically increasing and convex on $(-\infty, \alpha_1)$. In the hard case, $\alpha_h \notin \{\alpha_j | j \in J\}$, for $h \in \{1 \dots i\}$ and so, in particular, $d(\alpha_1)$ is well defined. If $\alpha_l := \min(\alpha_j | j \in J)$, then a similar analysis shows that $d(\cdot)$ is strictly monotonically increasing and convex on $(-\infty, \alpha_l)$. Figures 7.4 and 7.5 give a representation of $d(\cdot)$ in the easy and hard case, respectively. We conclude from this analysis of $d(\cdot)$, that in the easy case, for $t \in \mathbb{R}$, the equation $t - d(\lambda) = 0$ always has a solution $\lambda < \alpha_1$ and that this solution is unique. Since the eigenvalues of A and D(t) interlace, if $\lambda < \alpha_1$ and $t - d(\lambda) = 0$, the equation (7.1) implies $\lambda = \lambda_1(D(t))$. Since λ is the unique solution less than α_1, $\lambda_1(D(t))$ has multiplicity one. Thus, in the easy case, for any t, $\lambda_1(D(t))$ has multiplicity 1.
By (7.1), In the hard case with $t < t_0$, the equation $t - d(\lambda) = 0$ also has a unique solution $\lambda_1(D(t)) < \alpha_1$. Therefore, in this case $\lambda_1(D(t)) < \alpha_1$ with multiplicity 1. When $t = t_0$, we first note from (7.2) that $d'(\lambda) > 0$, if $\lambda < \alpha_1$. Therefore,

$$\lambda < \alpha_1 \Rightarrow d(\lambda) < d(\alpha_1) \Rightarrow 0 < t_0 - d(\lambda).$$

Since

$$\det(D(t_0) - \lambda I) = (t_0 - d(\lambda)) \prod_{j=1}^{n} (\alpha_j - \lambda), \tag{7.3}$$

we get for $\lambda < \alpha_1$, that $\det(D(t_0) - \lambda I) > 0$. But $\det(D(t_0 - \alpha_1 I) = 0$, and therefore $\lambda_1(D(t)) = \alpha_1$ with multiplicity $i + 1$ by (7.3). When $t > t_0$, the equation $t - d(\lambda) = 0$ does not have a solution $\lambda < \alpha_1$. Since α_1 is a solution to $\det(D(t) - \lambda I) = 0$, then again by equation (7.1) we get that $\lambda_1(D(t) = \alpha_1$ with multiplicity i. ■

With the previous theorem in hand, we know that, in the hard case, loss of

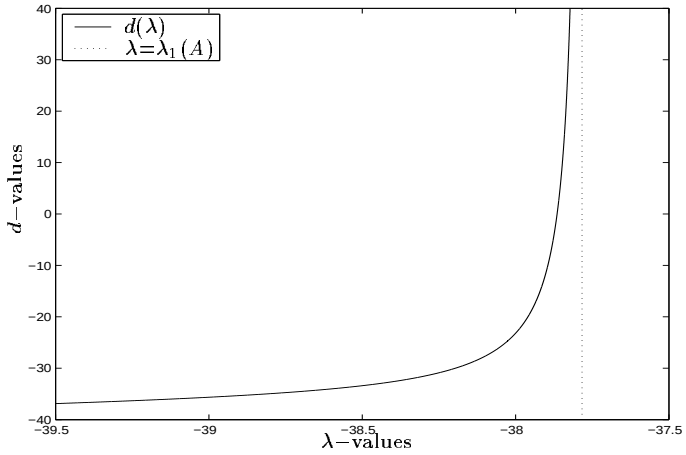

Figure 7.4: $d(\lambda)$ in the easy case

differentiability will occur for $k(t)$ at t_0. If we define t^* by

$$t^* \equiv \quad \arg\max_t \quad k(t), \tag{7.4}$$

then, in the hard case, we have shown that either $t^* < t_0$ (which is referred to as *case 1*) or $t^* = t_0$ (which is referred to as *case 2*). Figures 7.1, 7.2 and 7.3 illustrate the behavior of $k(t)$.

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

We let $y(t)$ be a normalized eigenvector for $\lambda_1(D(t))$ and let $y_0(t)$ be its first component. If $y(t) = \begin{pmatrix} y_0(t) \\ x(t) \end{pmatrix}$, then $\frac{1}{y_0(t)}\|x(t)\| = s$ if and only if $k'(t) = 0$.

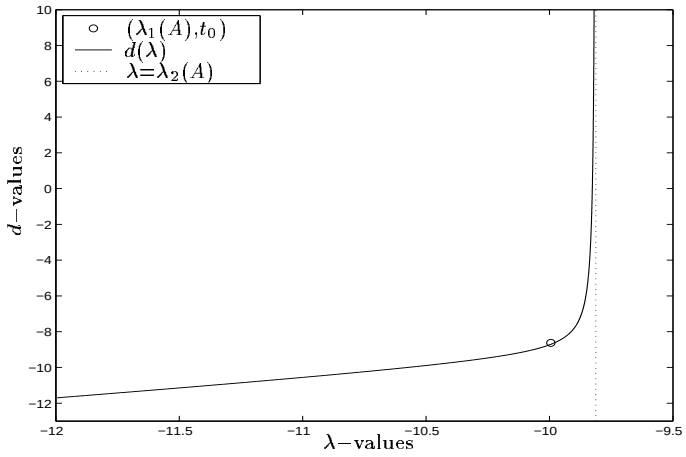

Figure 7.5: $d(\lambda)$ in the hard case

To maximize the concave function $k(t)$ we need to set its derivative (subgradient) to 0. We see below that this is equivalent to setting the derivative of $h(\lambda)$ to 0 or to setting $\phi(\lambda)$ to 0. We now look at the differentiability of the function $k(\cdot)$. For this we use the above results on the eigenvectors of $\lambda_1(D(t))$. We obtain the following, [27, 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** $t > t_0$: $y_0(t) = 0$;
   (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 non-zero first component ($\omega_0 \neq 0$) and the other eigenvectors in the basis have a zero first component ($y_0(t) = 0$).*

■

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 [16]). We know from Theorem 7.1 that $\lambda_1(D(t))$ has multiplicity 1 both in the easy case and, when $t < t_0$, in the hard case. Hence, for these two cases,

$$k'(t) = (s^2 + 1)y_0(t)^2 - 1. \tag{7.5}$$

In the hard case, when $t > t_0$, equation (7.5) still holds. It is well defined because $y_0(t) = 0$ for all eigenvectors of $\lambda_1(D(t))$. When $t = t_0$ in the hard case, $k(\cdot)$ is not differentiable and this is caused by a change in the multiplicity of the eigenvalue $\lambda_1(D(t))$. The directional derivative from the left is ω_0^2, where ω is defined as in Theorem 7.2; while the directional derivative from the right is -1.

Since $k(\cdot)$ is a concave and coercive (i.e. diverges to $-\infty$ as $|t| \to \infty$) function, to solve the dual (5.17) in the differentiable case we need simply solve $k'(t) = 0$. This will always be possible except when the maximum occurs at t_0, i.e. where $k(\cdot)$ is not differentiable. In the next section we will see that $k'(t) = 0$ always has a solution in the easy case and the hard case (case 1) and that otherwise, the hard case (case 2) occurs. To maximize the concave function $k(t)$ we need to use its derivative (subgradient). To establish the behavior of $k'(t)$, we start by considering the function $y_0(t)$, where $y_0(t)$ is the first component of the eigenvector for the smallest eigenvalue of $D(t)$. In the easy case, [27, lemma 13] proves that:

$y_0(t)$ is strictly monotonically decreasing,
$y_0(t) \to 1$ as $t \to -\infty$
and $y_0(t) \to 0$ as $t \to \infty$.

In particular, $y_0(t) \neq 0$ for all $t \in \mathbb{R}$. Figure 7.6 gives the representation $y_0(t)$ for this case.

In the hard case, for $t < t_0$, the graph of $y_0(t)$ is similar. For $t = t_0$, we know by [27, Lemma 15] that there exists an eigenvector for $\lambda_1(D(t))$ with $y_0(t) \neq 0$. To analyze what happens when $t > t_0$, note that if v_k is an eigenvector for the smallest eigenvalue $\lambda_1(A)$, then

$$D(t)\begin{bmatrix} 0 \\ v_k \end{bmatrix} = \begin{bmatrix} t & -a^T \\ -a & A \end{bmatrix}\begin{bmatrix} 0 \\ v_k \end{bmatrix} = \begin{bmatrix} -a^T v_k \\ Av_k \end{bmatrix} = \alpha_1 \begin{bmatrix} 0 \\ v_k \end{bmatrix}$$

This follows since $a^T v_k = 0$ is the condition for the hard case to occur. These last equations and Theorem 7.1 tell us that $\begin{bmatrix} 0 \\ v_k \end{bmatrix}$ is an eigenvector for the

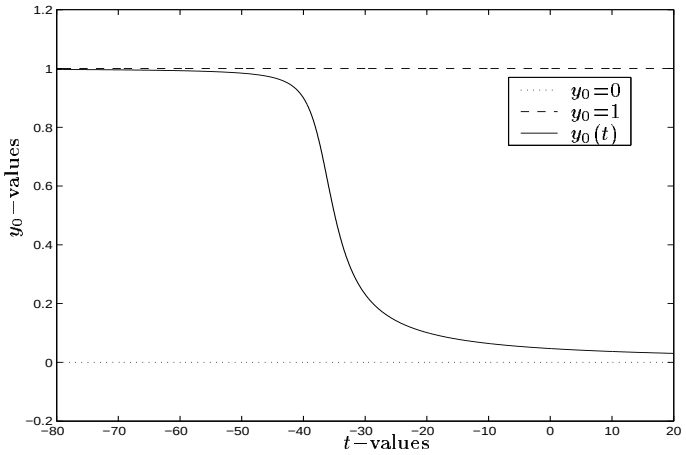

Figure 7.6: $y_0(t)$ in the easy case

smallest eigenvalue of $D(t)$. So for $t = t_0$, a basis for the space generated by the eigenvectors of the smallest eigenvalue of $D(t_0)$ is $\{\begin{bmatrix} 0 \\ v_1 \end{bmatrix}, \ldots, \begin{bmatrix} 0 \\ v_i \end{bmatrix}, \omega\}$, where $\{v_1, \ldots, v_i\}$ is a basis for the space generated by the smallest eigenvalue of A and where ω is an eigenvector of $D(t)$ with first component nonzero. The existence of ω is assured by [27, Lemma 15]. For $t > t_0$, the multiplicity of the smallest eigenvalue of D(t), α_1 by Theorem 7.1, is i and a basis for the space generated by its eigenvectors, is $\{\begin{bmatrix} 0 \\ v_1 \end{bmatrix}, \ldots, \begin{bmatrix} 0 \\ v_i \end{bmatrix}\}$. Therefore $y_0(t)$ is well defined for $t > t_0$ and is equal to 0. Figure 7.7 illustrates the hard case for $y_0(t)$. Since $k'(t) = (s^2+1)y_0(t)^2 - 1$ and since $y_0(t) \in [0,1]$, then $y_0(t)^2$ has a similar behavior to $y_0(t)$ and therefore, so does $k'(t)$. In the easy case we see this in Figure 7.8. Depending if the equation $k'(t) = 0$ has a solution or not, we have two cases to consider for the hard case. If the equation has a solution, then we must have $k'(t_0) < 0$ and the following.

$$\begin{aligned} k'(t_0) < 0 &\Leftrightarrow (s^2+1)y_0^2(t_0) - 1 < 0 \\ &\Leftrightarrow y_0^2(t_0)s^2 < 1 - y_0^2(t) \\ &\Leftrightarrow s^2 < \tfrac{1-y_0^2(t)}{y_0^2(t_0)}. \end{aligned}$$

which is equivalent to *case 1* of the hard case (see [27, pg. 288]). Similarly, the nonexistence of a solution implies we are dealing with *case 2* of the hard case (see [27, pg 289]). See Figure 7.9 for the behavior of $k'(t)$ in the hard case.

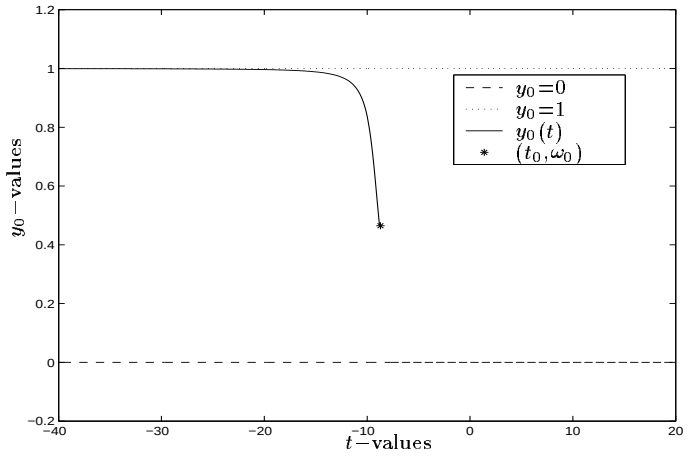

Figure 7.7: $y_0(t)$ in the hard case

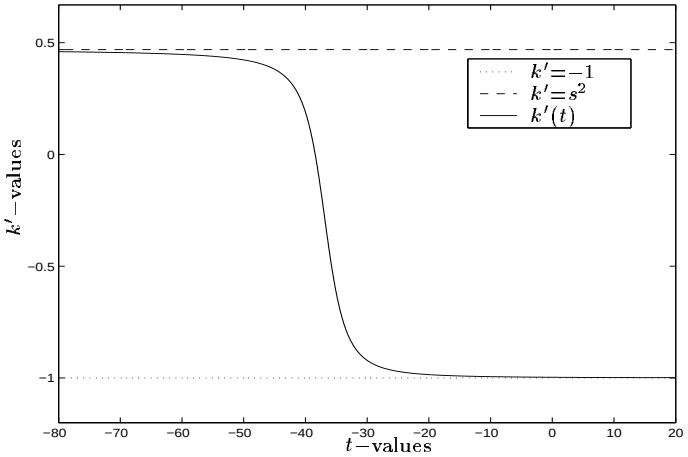

Figure 7.8: $k'(t)$ in the easy case

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

Solving $\psi(t) = 0$ or $k'(t) = 0$ is equivalent. There is always a solution in the easy case, though not necessarily in the hard case. As we already know the behavior of $y_0(t)$ for the easy and hard case, Figures 7.10, 7.11, 7.12 easily follow. $\psi(t)$ converges to $\sqrt{x^2+1} - 1$ as $t \to -\infty$. In the easy case, $\psi(t)$ goes to $-\infty$ as $t \to \infty$ in t. In the hard case, $\psi(t)$ is undefined for $t > t_0$.

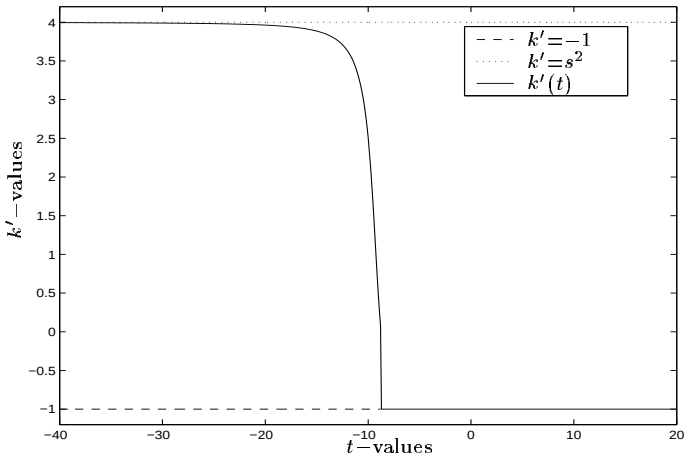

Figure 7.9: $k'(t)$ in the hard case (*case 1*)

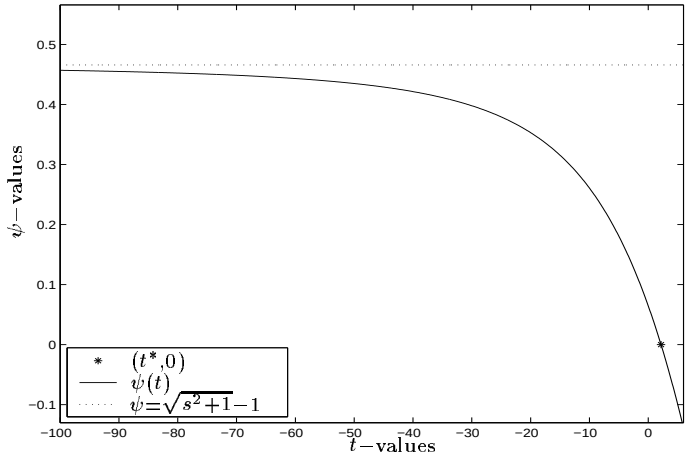

Figure 7.10: $\psi(t)$ in the easy case

7.2 Unconstrained Maximization of $k(t)$

In this section, we see how the RW algorithm solves TRS$_=$ using problem (5.17). The easy case is solved in a way similar to the MS algorithm, except that the function used is not $h(\cdot)$, but $k(\cdot)$ for sparsity considerations. The almost hard case (case 2) does not use the LAPACK routine to find the step direction to the boundary. Instead, the direction is obtained from an eigenvector previously computed. We use the following indicators of the easy

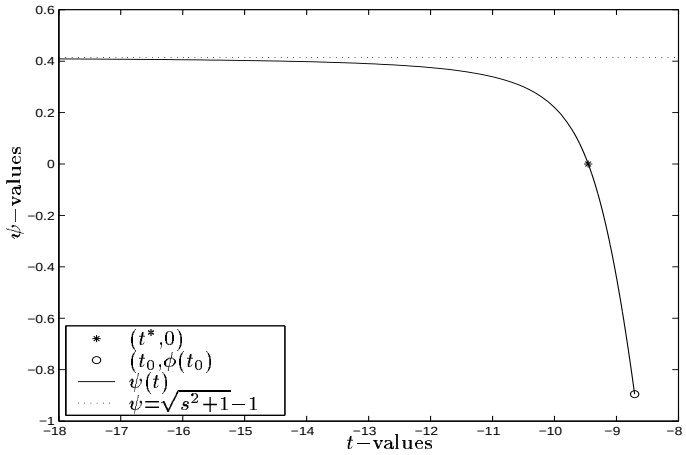

Figure 7.11: $\psi(t)$ in the hard case (*case 1*)

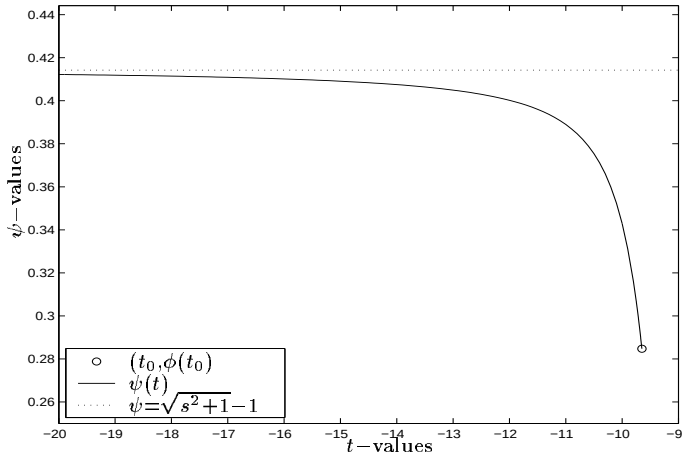

Figure 7.12: $\psi(t)$ in the hard case (*case 2*)

case. We assume that $y_0 > 0$.

$$\textbf{Easy Case Indicators:} \quad \begin{cases} \phi(\lambda) & > & 0 \\ \psi(t) & < & 0 \\ k'(t) & < & 0 \quad ((s^2+1)y_0^2 - 1 < 0) \\ y_0 & < & \frac{1}{\sqrt{s^2+1}} \\ t & > & t^* \\ \lambda & > & \lambda^* \\ \|x(\lambda)\| & > & s \\ h'(\lambda) & < & 0. \end{cases} \tag{7.6}$$

7.2.1 The Easy Case and the Hard Case (case 1)

We start this section with a theorem, derived from [27, Theorem 14], showing that, in these two cases, at each t there corresponds a solution to $\text{TRS}_=$ with a certain radius, as well as solutions to the SDP pair (5.19),(5.20) with s^2 changed appropriately.

Theorem 7.3. *Let $t \in \mathbb{R}$ and suppose $y(t) = [y_0(t), z(t)^T]^T$ is a normalized eigenvector of $D(t)$ corresponding to $\lambda_1(D(t))$. Suppose $y_0(t) \neq 0$. Let $x^* = \frac{1}{y_0(t)} z(t)$, $y = \begin{pmatrix} 1, x^* \end{pmatrix}$, and $\bar{s} = \|x^*\|$. Then $\bar{s} = \sqrt{\frac{1-y_0(t)^2}{y_0(t)^2}}$ and:*

1. *x^* is an optimal solution of the $\text{TRS}_=$ with s replaced by $\bar{s}$, i.e.*

$$\min \{q(x) : \ \|x\| = \bar{s}\}$$

   and $\lambda^* = \lambda_1(D(t))$ is its Lagrange multiplier.

2. *Let s be replaced by $\bar{s}$ and let $Y^* = yy^T$. Then $\lambda = \lambda_1(D(t)), t$ solve (5.19) and Y^* solves (5.20).*

■

From [27, Lemma 13], if the easy case holds and $y(t)$ is a normalized eigenvector with $y_0(t) > 0$, then $y_0(t)$ is a function mapping $\mathbb{R} \to (0, 1)$ which is strictly monotonically decreasing. It is then easy to show that the function $\frac{1-y_0(t)^2}{y_0(t)^2}$ is a strictly decreasing function mapping $(0, 1) \to (0, \infty)$. Hence for a given s, we can solve $\text{TRS}_=$ in the easy case by finding a t such that

$$s^2 = \frac{1 - y_0(t)^2}{y_0(t)^2}. \tag{7.7}$$

Now note that $k'(t) = 0$ if and only if the previous equation is satisfied. Hence $\text{TRS}_=$ in the easy case can be solved by finding t^* that satisfies (7.7). Setting

$$x^* := \frac{1}{y_0(t^*)} z(t^*) \quad \text{and} \quad \lambda^* = \lambda_1(D(t^*))$$

gives an optimal solution to $\text{TRS}_=$ and its Lagrange multiplier, according to Theorem 7.3.

In the hard case, we can use the same approach, if at t^* the function $k(\cdot)$ is differentiable; hence $k'(t^*) = 0$. Since $k(\cdot)$ is not differentiable only at t_0, and since the directional derivatives from the left and right are, respectively,

$$k'(t_0^-) = (s^2+1)\omega_0^2 - 1 \quad \text{and} \quad k'(t_0^+) = -1,$$

we get by the concavity of $k(\cdot)$, that this function is differentiable at the optimum if and only if the directional derivative from the left of t_0 is negative, i.e.

$$(s^2+1)\omega_0^2 - 1 < 0 \Leftrightarrow \frac{1-\omega_0^2}{\omega_0^2} > s^2.$$

This implies that $t^* < t_0$. Since the function $y_0(\cdot)^2$ is strictly positive on the interval $(-\infty, t_0)$ and is the derivative of the function $\lambda_1(D(\cdot))$, then the latter is strictly increasing on the interval $(-\infty, t_0)$. $\lambda_1(D(\cdot))$ is also a continuous function, hence it is strictly increasing on the interval $(-\infty, t_0]$. Therefore

$$t^* < t_0 \Rightarrow \lambda_1(D(t^*)) < \lambda_1(t_0).$$

The right inequality implies $\lambda^* < \lambda_1(A)$ and the hard case (case 1) occurs.

This shows we can solve $\text{TRS}_=$ in the easy case and the hard case (case 1) by solving the equation

$$k'(t) = (s^2+1)y_0(t)^2 - 1 = 0.$$

The RW algorithm does this by finding the zero of the function

$$\psi(t) := \sqrt{s^2+1} - \frac{1}{y_0(t)}. \tag{7.8}$$

Note that this trick is analogous to the use of the function $\phi(\cdot)$ in the MS algorithm. The function $\psi(\cdot)$ has the advantage of being less nonlinear near t^* and therefore interpolating to find t such that $\psi(\cdot)$ equals 0 will be more efficient.

7.2.2 The (Near) Hard Case (case 2)

In the hard case, when

$$\frac{1-\omega_0^2}{\omega_0^2} < s^2,$$

then $k'(\cdot)$ is positive to the left of t_0 and is -1 on its right. Hence, by the concavity of $k(\cdot)$, its maximum occurs at t_0 and so $t^* = t_0$. Note also that $\lambda^* = \lambda_1(D(t_0)) = \lambda_1(A)$. This is true since for $\omega = [\omega_0, \tilde{\omega}]^T$, where ω is as in Theorem 7.2 and where $\tilde{\omega} \in \mathbb{R}^n$, $\tilde{\omega}$ is by construction perpendicular to the eigenvectors of $\lambda_1(A)$ and a short computation shows that

$$x^* := \frac{\tilde{\omega}}{\omega_0} + \sqrt{s^2 - \frac{1-\omega_0^2}{\omega_0^2}} z,$$

with $z \in S_1$, satisfies the optimality conditions for $\text{TRS}_=$ with Lagrange multiplier $\lambda^* := \lambda_1(A)$ and we are in the hard case (case 2).

Now, we cannot solve $k'(t) = 0$ anymore to find the optimum of $k(\cdot)$ and the function $\psi(\cdot)$ is positive on the interval $(-\infty, t_0)$ and does not exists for higher values of t. To handle this case, we take a primal step to the boundary. Let t_e be such that $k'(t_e) < 0$, then t_e is defined to be on the *easy side*, since the easy side is where we want to be in the MS algorithm, that is when $\phi(\lambda) > 0$. Similarly, if for t_h we have $k'(t_h) > 0$, then t_h is defined to be on the *hard side*. If we have a point t_h from the hard side, then $t_h < t_0$ and this implies that $y_0(t_h) \neq 0$. Let $y(t) = [y_0(t), z(t)]^T$. We have

$$k'(t_h) > 0 \Leftrightarrow (s^2+1)y_0(t_h)^2 - 1 > 0 \Leftrightarrow \frac{1-y_0(t_h)^2}{y_0(t_h)^2} < s^2 \Leftrightarrow \|\frac{1}{y_0(t_h)} z(t_h)\|^2 < s^2.$$

Theorem 7.2 implies that

$$x_h := \frac{1}{y_0(t_h)} z(t_h) \tag{7.9}$$

minimizes $q(\cdot)$ on the sphere of radius $\sqrt{\frac{1-y_0(t_b)^2}{y_0(t_b)^2}}$, which is less than s.

In the next section, we will show, if $\lambda_1(D(t_h)) > 0$ for a t_h on the hard side, then an unconstrained minimum lies within the trust region and we can solve TRS with the conjugate gradient method. For this section, we assume $\lambda_1(D(t_h)) \leq 0$. We can now apply Lemma 3.2 with the feasible solution x_h, since $(A - \lambda_1(D(t_h))I)x_h = a$ holds as well. Yet, we need to find a vector z, with $\|z\| = 1$, such that $z^T(A - \lambda_1(D(t_h))I)z$ is small. Let t_e be a point from the easy side. Then, by Theorem 7.2, in the hard case (case 2), for any eigenvector $y(t_e)$ of $\lambda_1(D(t_e))$, $z(t_e)$ is an eigenvector for $\lambda_1(A)$ and has a unit norm, since $y_0(t_e) = 0$ (notice that in the hard case (case 2), $t_e > t_0$).

Hence

$$\begin{aligned} z(t_e)^T(A - \lambda(D(t_h))I)z(t_e) &= z(t_e)^T(A - \lambda_1(A)I + (\lambda_1(A) - \lambda(D(t_h)))I)z(t_e) \\ &= (\lambda_1(D(t_0)) - \lambda(D(t_h))). \end{aligned}$$

Therefore, for t_h close to t_0, $z(t_e)^T(A - \lambda(D(t_h))I)z(t_e)$ will be small and $z(t_e)$ is taken as a step direction to the boundary.

As in the MS algorithm, every time a feasible solution to TRS is obtained (each new point t_b from the easy side gives us a new feasible solution), if we have a point t_g from the hard side, we take a primal step to the boundary. This handles the almost hard case (case 2), but may also prove to be of use in the two other cases if a decrease in the objective function is obtained.

7.3 Flowchart

The details for the individual steps follow in Section 7.4 below.

INITIALIZATION:

1. *Compute $\lambda_1(A)$ and corresponding eigenvector v_1. If $\lambda_1(A) < 0$, shift $A \leftarrow A - \lambda_1(A)I$. ($\lambda_1(A)\|x^*\|^2$ added to objective value at end.)*

   If $a^T v_1$ is small (near hard case), then deflate, i.e. add the vector $y_1 = \begin{pmatrix} 0 \\ v_1 \end{pmatrix}$ to the set $\mathcal{Y}$.

2. *Obtain bounds on q^*, λ^* and t^*.*

   If $\lambda_1 > 0$, test if the optimum is an unconstrained minimizer. EXIT here if so.

3. *Initialize parameters and the stopping criteria 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 $k(t) - M_t$ if possible; otherwise set it to the the midpoint (default) of the interval of uncertainty for t.*

(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 $y_0 > 1/(s^2+1)$ then the solution is the unconstrained minimizer. Use CONJUGATE GRADIENT 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 $t_i, i = 1..5$*

END LOOP

7.4 Techniques used in the algorithm

In this section, we clarify the techniques used in the algorithm for solving (TRS).

7.4.1 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 [17]. Note that Newton's method applied to $k(t) - M_t = 0$ at t_c yields

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

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.

7.4.2 Triangle Interpolation

Given $k(t)$, if we have two values of t, t_e and t_h such that $k'(t_e) < 0$ and $k'(t_h) > 0$, then we try to find a better approximation t_{new} to the solution of the program (7.4). The set of $\{t : k'(t) < 0\}$ is referred to as the *easy side* and its complement as the *hard side*. A better approximation is found using a technique we call *triangle interpolation*, i.e. we find the t coordinate of the point where the secant lines intersect. (We use a tangent line on the side where there is only one point.) This becomes our new approximation t_{new}. We also obtain an upper bound q_{up} to $q^* := k(t^*)$ from the k coordinate of the point where the secant lines intersect, see Figure 7.13.

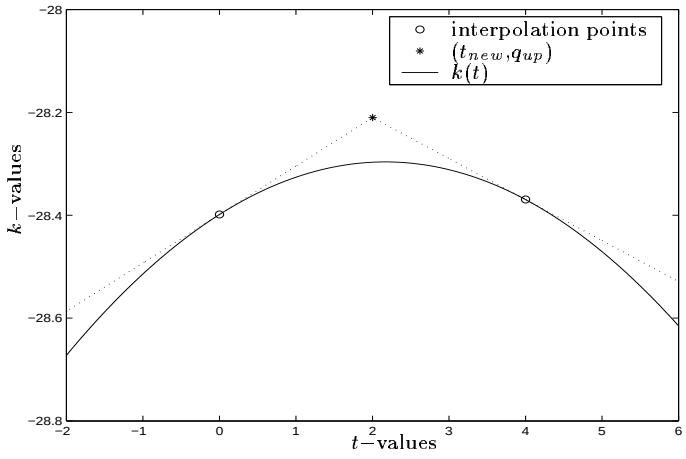

Figure 7.13: Triangle interpolation

7.4.3 Vertical Cut

Suppose we have two values of t again, 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.

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

where t_{high} is the upper bound on t^*. See Figure 7.14.

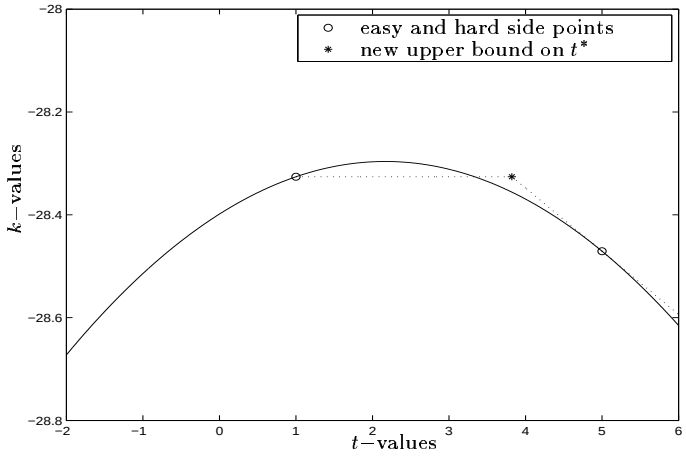

Figure 7.14: Vertical cut

7.4.4 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 monotonically decreasing function, we can consider its inverse function, say $t(\psi)$. We use (concave) quadratic interpolation when possible, i.e. suppose the points $(\psi_i, t_i), i = 1, 2, 3$. 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 2×2 system in the linear interpolation case. Figures 7.15 and 7.16 illustrate the process.

We see that in the hard case (case 2), inverse interpolation doesn't provide us with useful informations on t^* and the estimate is somewhat irrelevant, since $\psi(t)$ is not defined for $t > t_0$ and is positive for $t < t_0$. So when we extrapolate in this case, we extrapolate outside the domain of the function $\psi(t)$. Since we cannot tell that this case 2 holds, if we are currently on the hard side, this interpolation must be safeguarded, i.e. we have to maintain that t and λ stay on the correct side of $\lambda_1(A)$.

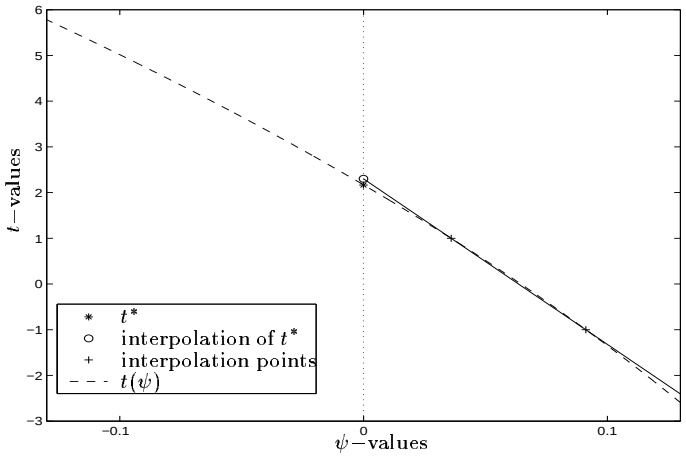

Figure 7.15: Inverse interpolation in the easy case and hard case (*case 1*)

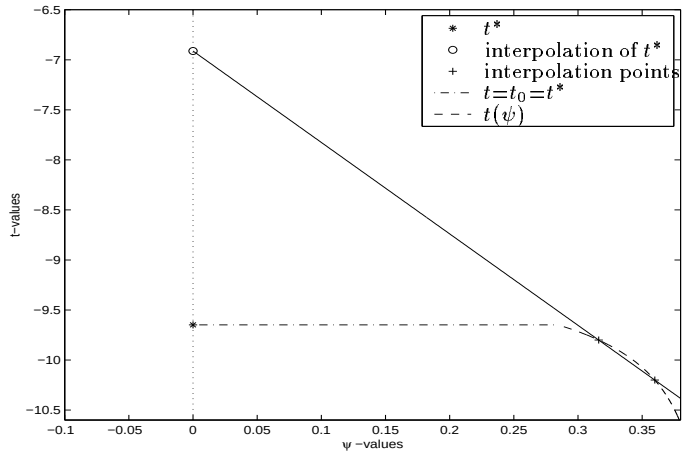

Figure 7.16: Inverse interpolation in the hard case (*case 2*)

In the easy case and case 1 of the hard case, as we get values of t such that ψ approaches 0 for those value, inverse interpolation will give a better approximation for t^*.

7.4.5 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 trust region. The following theorem is the key to recognizing this case.

Theorem 7.4. *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 $\min\{x^T Ax - 2a^T x : \|x\| \leq \|\bar{x}\|\}$. If $\lambda \geq 0$, then $\bar{x}$ is a solution to $\min\{x^T Ax - 2a^T x : \|x\| \geq \|\bar{x}\|\}$.*

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. ■

Now suppose that $\bar{x}$ satisfies $(A - \lambda I)\bar{x} = a$ with $A - \lambda I \succeq 0$, that $\|\bar{x}\| \leq s$ and that $\lambda > 0$. Then, by Theorem 7.4, $\bar{x}$ is a solution to $\min\{x^T Ax - 2a^T x : \|x\| \geq \|\bar{x}\|\}$ and $A - \lambda I$ is positive semidefinite, then A is positive definite. Moreover, by the optimality of $\bar{x}$, we know that the unconstrained minimum lies inside the region $\{x : \|x\| \leq \|\bar{x}\|\}$ which is included in the trust region $\{x : \|x\| \leq s\}$. Therefore we can set the optimum to $A^{-1}a$.

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 $\min\{x^T Ax - 2a^T x : \|x\| = \|x_k\|\}$. Checking the sign of the multiplier λ_k tells us if x_k is a solution to $\min\{x^T Ax - 2a^T x : \|x\| \leq \|x_k\|\}$ or $\min\{x^T Ax - 2a^T x : \|x\| \geq \|x_k\|\}$. If the later case holds and $\|x_k\| \leq s$, then we know the unconstrained minimum lies in the trust region.

7.4.6 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 $y = (y_0 \ v^T)^T$ if y_0 is small (essentially 0). This indicates that $a^T v$ is small. We perturb $a \leftarrow a - a^T v v$ and deflate using $A \leftarrow D + \alpha y y^T$. This is done by adding a threshold value times the matrix yy^T during the Lanczos process. This uses the deflation in Lemma 2.1 Item 4 and the perturbation analysis in Lemma 2.2.

7.4.7 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. we can consider this as 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; it will be infeasible on the other (easy) side. In either case, we can now take a primal step, i.e. move to the boundary and improve the objective value.

There are two similar steps to the boundary we can take depending if the (almost) hard case holds or not. In the easy case, the step is motivated by the following lemma:

Lemma 7.1. *Let* $0 < s_1 < s < s_2$ *and let*

$$x_h \in \operatorname{argmin}\{q(x) : \|x\| \leq s_1^2\}$$
$$x_e \in \operatorname{argmin}\{q(x) : \|x\| \leq s_2^2\}.$$

Assume $\|x_h\| = s_1$, $\|x_e\| = s_2$, $x_h^T(x_e - x_h) \neq 0$ *and the Lagrange multiplier* λ *for* x_h *satisfies* $A - \lambda I \succ 0$. *Let*

$$m(\alpha) := q(x_h + \alpha(x_e - x_h)).$$

Then

$$m'(\alpha) \leq 0 \quad \text{for } \alpha \in [0, 1]$$

and therefore

$$q(x_h + \alpha(x_e - x_h)) \leq q(x_h) \quad \text{for } 0 \leq \alpha \leq 1.$$

In particular, for $\bar{\alpha} \in [0, 1]$ *such that* $\|x_e + \bar{\alpha}(x_e - x_h)\| = s$ *we have that* $x_h + \bar{\alpha}(x_e - x_h)$ *is feasible for TRS and has a smaller objective value than* x_h.

Proof: Since $\|x_h\|^2 = s_1^2 < s_2^2$, it is possible to find 2 different values α_1 and α_2 such that

$$\|x_h + \alpha_i(x_e - x_h)\|^2 = s_2^2 \quad i = 1, 2. \tag{7.10}$$

A short computation shows that

$$\alpha_1 = \frac{-x_h^T(x_e - x_h) + \sqrt{(x_h^T(x_e - x_h))^2 + \|x_e - x_h\|^2(s_2^2 - \|x_h\|^2)}}{\|x_e - x_h\|^2},$$

$$\alpha_2 = \frac{-x_h^T(x_e - x_h) - \sqrt{(x_h^T(x_e - x_h))^2 + \|x_e - x_h\|^2(s_2^2 - \|x_h\|^2)}}{\|x_e - x_h\|^2}.$$

Now suppose $x_h^T(x_e - x_h) < 0$, then $\alpha_1 > 0$, $\alpha_2 < 0$ and $|\alpha_2| < |\alpha_1|$. By (7.10) and using $(A - \lambda I)x_h = a$ we get for $i = 1, 2$

$$q(x_h + \alpha_i(x_e - x_h)) = -(x_h^T(A - \lambda I)x_h - \lambda s_2^2) + \alpha_i^2(x_e - x_h)^T(A - \lambda I)(x_e - x_h). \tag{7.11}$$

Since $\|x_e\|^2 = s_2^2$, then $\alpha = 1$ solves $\|x_h + \alpha(x_e - x_h)\|^2 = s_2^2$ and therefore $\alpha_1 = 1$. By the optimality of x_e, by equation (7.11) and because $A - \lambda I \succ 0$, we must have $|\alpha_1| \leq |\alpha_2|$. This is a contradiction, hence $x_h^T(x_e - x_h) > 0$.
Now let

$$f(\epsilon) := \|x_h - \epsilon(x_e - x_h)\|^2.$$

Note $f(0) = s_1^2$. Then

$$f'(\epsilon) = 2\epsilon\|x_e - x_h\|^2 - 2x_h^T(x_e - x_h).$$

Therefore for ϵ small enough $f'(\epsilon) < 0$ and this yields $\|x_h - \epsilon(x_e - x_h)\|^2 < s_1^2$. Since

$$m(-\epsilon) = m(0) - m'(0)\epsilon + o(\epsilon) = q(x_h) - m'(0)\epsilon + o(\epsilon),$$

if $m'(0) > 0$, then $q(x_h - \epsilon(x_e - x_h)) < q(x_h)$ for $\epsilon > 0$ small enough. This contradicts the optimality of x_h. Hence $m'(0) \leq 0$.
Let

$$w(\epsilon) := \|x_h + (1 - \epsilon)(x_e - x_h)\|^2.$$

Note $w(0) = s_2^2$. Then

$$w'(\epsilon) := -2(1 - \epsilon)\|x_e - x_h\|^2 - 2x_h^T(x_e - x_h).$$

For $0 < \epsilon < 1$, $w'(\epsilon) < 0$, hence $\|x_h + (1 - \epsilon)(x_e - x_h)\|^2 < s_2^2$. Now since

$$m(1 - \epsilon) = m(1) - m'(1)\epsilon + o(\epsilon) = q(x_e) - m'(1)\epsilon + o(\epsilon),$$

again, by the optimality of x_e, we have $m'(1) \leq 0$.
Since $q(\cdot)$ is a quadratic function, then $m(\cdot)$ is a parabola, i.e.

$$m(\alpha) = a\alpha^2 + b\alpha + c, \quad \text{where } a, b, c \in \mathbb{R},$$

and we have

$$m'(0) = b \leq 0 \quad \text{and} \quad m'(1) = 2a + b \leq 0.$$

Let $0 < \alpha < 1$, then

$$\begin{aligned}
&\text{if} \quad a > 0, \text{ we have: } \alpha < 1 \Rightarrow m'(\alpha) = 2a\alpha + b < 2a + b \leq 0; \\
&\text{if} \quad a < 0, \text{ we have: } 0 < \alpha \Rightarrow m'(\alpha) = 2a\alpha + b < b \leq 0; \\
&\text{if} \quad a = 0, \text{ we have: } m'(\alpha) = b \leq 0.
\end{aligned}$$

Therefore, these inequalities with $m'(0) \leq 0$ and $m'(1) \leq 0$ show that

$$m'(\alpha) \leq 0 \quad \text{for } 0 \leq \alpha \leq 1.$$

Hence, by the definition of $m(\cdot)$,

$$\frac{d}{d\alpha} q(x_h + \alpha(x_e - x_h)) \leq 0 \quad \text{for } 0 \leq \alpha \leq 1$$

and the lemma follows. ■

The use of the previous lemma in the easy case is now clear: given two values of the variable t, t_h and t_e, respectively on the hard side and the easy side, assume $\lambda_1(D(t_h)) \leq 0$ and $\lambda_1(D(t_e)) \leq 0$ (note that $\lambda_1(D(t_h))$ is the Lagrange multiplier for x_h and that $A - \lambda_1(D(t_h))I \succ 0$ holds) , and let

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

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 value.

Note that we assume the easy case to be able to justify $y_0(t_e) \neq 0$. We may also extend this step to the boundary in the hard case (case 1) when this condition is satisfied. However, in the hard case (case 2) $y_0(t_e) = 0$ always holds. It seems we need a new strategy, but again, as we will explain later, Lemma 7.1 is again involved. In section 7.2.2 we explained that from x_h, we use $z := z(t_e)$ as a step direction to the boundary (if no point on the easy side has yet been found, then we use the eigenvector of A found during the initialization step). This choice was motivated by Lemma 3.2 and the desire to make the quadratic form $z^T(A - \lambda(D(t_h))I)z$ small. Precisely, we take the step $x_h + \tau z$ with τ chosen to reduce the objective function and satisfy

$$\| \frac{1}{y_0(t_h)} z(t_h) + \tau z \| = s.$$

The explicit expression for τ is

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

where $sgn(\cdot)$ 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. [20] proves that to improve the objective, we should pick the one with smallest magnitude. This is how τ is built in the previous expression.

Now let's assume again the easy case holds and consider the sequence of points

$$x(t_i) := \frac{z(t_i)}{y_0(t_i)},$$

where $t_i > t^*$ (so $\{t_i\}$ are on the easy side, i.e. $\|x(t_i)\| > s$) and $t_i \to \infty$. As $t_i \to \infty$, by [27, lemma 13] $y_0(t_i) \to 0$, $\lambda_1(D(t_i)) \to \lambda_1(A)$ and the boundedness of the $z(t_i)$ implies that a subsequence converges. Let $\tilde{z}$ be the limit. Since

$$\begin{aligned} D(t_i)\begin{bmatrix} y_0(t_i) \\ z(t_i) \end{bmatrix} &= \begin{bmatrix} t_i & -a^T \\ -a & A \end{bmatrix}\begin{bmatrix} y_0(t_i) \\ z(t_i) \end{bmatrix} = \begin{bmatrix} t_i y_0(t_i) - a^T z(t_i) \\ -y_0(t_i)a + Az(t_i) \end{bmatrix} \\ &= \lambda_1(D(t_i))\begin{bmatrix} y_0(t_i) \\ z(t_i) \end{bmatrix}, \end{aligned}$$

$\tilde{z}$ is therefore an eigenvector of $\lambda_1(A)$. Assuming $\lambda_1(D(t_i)) \leq 0$, i.e. the multipliers have the correct sign, we have

$$x(t_i) \in \text{argmin}\{q(x) : \|x\| \leq \|x(t_i)\|\}.$$

If we assume also for all i that $x_h^T(x(t_i) - x_h) \neq 0$, Lemma 7.1 says we may use

$$d := \frac{z(t_i)}{y_0(t_i)} - \frac{z(t_h)}{y_0(t_h)}$$

as a direction for the step to the boundary. Now taking the limit of the normalized direction as $t_i \to \infty$, then

$$\lim_{t_i \to \infty} \frac{d}{\|d\|} = \frac{\tilde{z}}{\|\tilde{z}\|}.$$

Hence, when we use an eigenvector for $\lambda_1(A)$ (as z above) as a direction for a step to the boundary from a hard side point, it is as if we are using in

Lemma 7.1 a point from the easy side located at infinity. This also justifies the use of a different step to the boundary in the ease case and potentially the hard case (case 1), since using z in these cases for a step direction to the boundary, as it was suggested in [27], does not take into account the local information we have in hand, i.e. the information from the easy side.

To summarize, given two values of t, t_h and t_e, respectively from the hard and easy side, we monitor $y_0(t_e)$. If it is significantly above 0, we use the direction $x_e - x_h$ to move to the boundary. Otherwise, we use an eigenvector z corresponding to $\lambda_1(A)$) as a direction to the boundary. If we have a value of t, say t_e, from the easy side, we use $z = z(t_e)$, otherwise we use the eigenvector for the smallest eigenvalue of A found during the initialization step.

If no points have so far been found on the hard side, and a point from the easy side has been found with correct Lagrange multiplier, then it does not hurt to project this point on the feasible set and obtain a new feasible point.

Let t_e be the value of t for the last point found on the easy side and λ_e its associated Lagrange multiplier. If $\lambda_e \leq 0$, then by Theorem 7.4, if $y_0(t_e) \neq 0$, $\bar{x} := \frac{y(2:n+1)(t_e)}{y_0(t_e)}$ is a solution to

$$\min\{x^T A x - 2a^T x : \|x\|^2 \leq \|\bar{x}\|^2\}$$

The idea now is to project $\bar{x}$ on the feasible set of TRS. Let x_{new} be the new feasible point, it is obtained by the following formula.

$$x_{new} := \frac{z(t_e)}{\|z(t_e)\|} \cdot s$$

This projection may not seem natural, as we might have thought of taking $\frac{\bar{x}}{\|\bar{x}\|} \cdot s$, but $\bar{x}$ is not defined in the hard case. On the contrary, the above projection is always well defined.
There are two reasons for taking such a step. First, it is always a good thing to obtain a new feasible point, because then we can check the objective value at this point and we may improve the duality gap. Second, by the optimality of $\bar{x}$, $q(\bar{x}) \leq q^*$. Therefore, if $\|\bar{x}\|$ is close to s, we can expect $q(x_{new})$ to be close to q^*.

8 Numerical Experiments

8.1 The Hard Case

We now provide numerical evidence that the Rendl-Wolkowicz Algorithm (our modified version) is better suited to handle the hard case (case 2) than the Moré-Sorensen Algorithm. It is stated in [20] 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 Moré-Sorensen Algorithm returns an approximate solution that satisfies

$$\tilde{q} \leq (1-\sigma)^2 q^*,$$

where σ is an input to the algorithm, and the approximate solution of the Rendl-Wolkowicz Algorithm satisfies

$$\tilde{q} \leq \frac{1}{1-2\text{dgaptol}} q^*,$$

where dgaptol is the desired relative duality gap tolerance. Hence, to get equivalent accuracy, we choose

$$\sigma = 1 - \sqrt{\frac{1}{1+2\text{dgaptol}}}.$$

We used randomly generated sparse hard case (case 2) trust region subproblems, where the density is of the order of $1/(20n\log n)$. The tolerance parameter *dgaptol* was set to 10^{-12}. Each row in the following table gives the average number of iterations and cpu time for 10 problems of size n. We did not go beyond $n = 640$ for the Moré-Sorensen Algorithm to avoid the large computation times. The results, given in Table 8.1 and plotted in Figure 8.1, illustrate the improved performance in both the number of iterations and the computation time.

We note that the GLTR Algorithm does not appear in the above comparison. The reason is obvious: the algorithm was not designed to handle this case. We also note that it is easy to construct examples where the GLTR algorithm fails: randomly choose a and A and multiply A by increasing values

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

Table 8.1: Iterations; Modified-RW and MS Algorithms on hard case (case 2(ii)) examples.

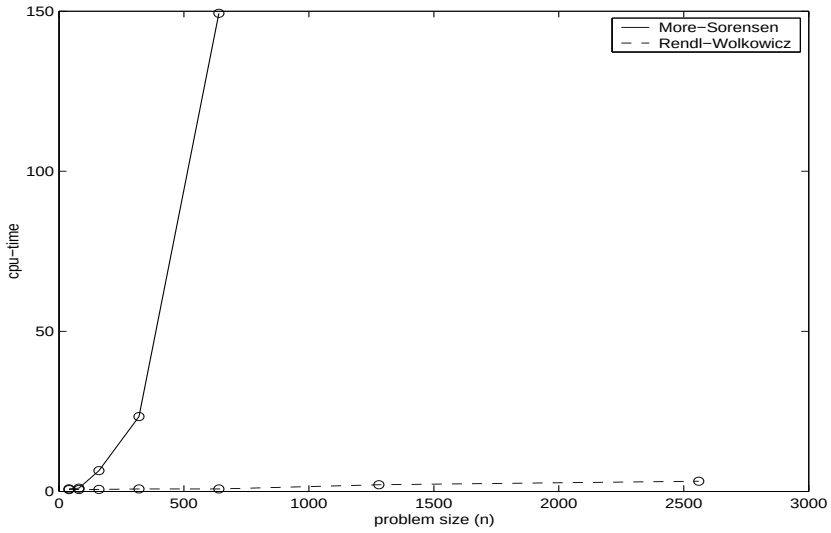

Figure 8.1: Cputime; Modified-RW and MS Algorithms on hard case (case 2(ii)) examples.

of $\alpha = 1, 2, 3, ...$ This procedures makes a relatively small compared to A and creates the near hard case. Generally, what one observes when α is large enough, is that solving the tridiagonal trust region subproblem fails because the algorithm cannot find a proper starting value of λ to satisfy (4.6). The algorithm uses an initial value of $\lambda = \lambda_1(T_k) - (1 - \lambda_1(T_k))\sqrt{\epsilon}$, where ϵ is the unit roundoff. However, in this case, this value is still too large to start the Newton iterations.

8.2 An Example where the GLTR Algorithm Fails

Consider minimizing

$$f(x) = \sum_{i=1}^{n-1} (x_i^2 - 1)^2 + (x_n - 1)^2,$$

using a trust region method, starting at the initial point

$$x_0 = [0, \ldots, 0, 3/2]^T.$$

We have

$$\nabla f(x_0) = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, \quad \nabla^2 f(x_0) = \begin{bmatrix} -4 & 0 & \ldots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & -4 & 0 \\ 0 & \ldots & 0 & 2 \end{bmatrix}.$$

We chose x_0 so that the hard case (case 2) appears eventually. Indeed, the gradient is perpendicular to the space spanned by the eigenvectors of the smallest eigenvalue of the Hessian. Let s_0 be the initial size of the trust region, then a short computation shows the hard case (case 2) occurs if the initial trust region radius $s_0 \geq 1/6$.

Let us choose $s_0 = 1$ so the hard case (case 2) occurs initially and use the GLTR Algorithm to solve the TRS for the unknown d (this would be the first TRS to be solved within a trust region method)

$$\begin{array}{ll} \min & d^T \nabla^2 f(x_0) d + 2 \nabla f(x_0)^T d \\ \text{s.t.} & \|d\| \leq s_0. \end{array} \tag{8.1}$$

Going through the computations of Algorithm 4.1 we obtain

$$\begin{array}{l} \alpha_0 = \|g_0\|^2 / (p_0^T A p_0) = 1/2, \\ \|d_1\| = \|d_0 + \alpha_0 p_0\| = \|[0, \ldots, 0, -1/2]^T\| = 1/2 \leq s_0 \\ g_1 = g_0 + \alpha_0 A p_0 = [0, \ldots, 0]^T. \end{array}$$

The algorithm would stop since

$$\|g_1\| = 0 < \max(10^{-8}, 10^{-5} \|\nabla f(x_0)\|).$$

Thus we obtain the approximate solution d_1 to the TRS (8.1). Although we may have expected the GLTR Algorithm to fail finding a good approximate solution because the hard case (case 2) occurs, the cause of the failure is different and our example illustrates a somewhat unexpected weakness of the algorithm: the conjugate gradient method may converge to a saddle point located within the trust region. Indeed, d_1 does satisfy $\nabla^2 f(x_0)d_1 + \nabla f(x_0) = 0$, but obviously $\nabla^2 f(x_0)$ is indefinite so that the strengthened second order optimality condition is not satisfied. Hence, we notice that whenever $s_0 \leq 1/2$, the GLTR Algorithm will return d_1 as an approximate solution to the TRS, independently of the tolerances used in the stopping criteria.

We may ask what happens if $s_0 \leq 1/2$. In this case α_0, d_1 and g_1 are as previously computed, except that $\|d_1\| \geq s_0$ and it is then known that $INTERIOR$ is false, i.e. the solution to (8.1) is on the boundary. Thus, the tridiagonal trust region subproblem (4.4)

$$\begin{array}{ll} \min & 2h^2 + 2h \\ \text{s.t.} & h^2 = s_0^2, \end{array}$$

which is one dimensional in this case, needs to be solved. Using our notation, its optimal solution, h_0 with Lagrange multiplier λ_0, is $h_0 = -s_0$ with $\lambda_0 = 2 - 1/s_0$. We then need γ_1 to test convergence and a quick computation yields $\gamma_1 = 0$. Hence the convergence test would tell us we are done, since

$$\gamma_1 |e_1^T h_0| = 0 < \max(10^{-8}, 10^{-5}\|\nabla f(x_0)\|).$$

Note that since the left hand side is zero, we stop independent of the tolerance we choose on the right hand side. We finally recover the approximate optimal solution d^* for (8.1) with corresponding Lagrange multiplier λ^* from (4.9):

$$d^* = \nabla f(x_0) * h_0 = [0, \ldots, 0, -s_0]^T \quad \text{and} \quad \lambda^* = 2 - 1/s_0.$$

In agreement with (4.7) d^* satisfies stationarity, i.e.

$$(\nabla^2 f(x_0) - \lambda^* I)d^* + \nabla f(x_0) = 0.$$

But unless $s_0 \leq 1/6$, the strengthened second order optimality condition is not satisfied, i.e. $\nabla^2 f(x_0) - \lambda^* I$ is not positive semidefinite. If indeed $s_0 \leq 1/6$, then d^* in this case is the correct optimum for (8.1).

This example is interesting since it shows, when $s_0 \geq 1/6$, that the GLTR Algorithm cannot find an approximate solution to the TRS, independent of the tolerances. When $s_0 \geq 1/2$ the cause of failure is from the conjugate gradient converging to a saddle point. This does not seem to be a problem caused by the hard case (case 2), unlike the situation when $1/6 \leq s_0 \leq 1/2$.

8.3 A Comparison of the RW and GLTR Algorithms within a TR Framework

In [13], the authors suggest that the GLTR method may be stopped not only if convergence occurs, but also after a limited extra number of iterations, say N, once the solution is known to lie on the boundary of the trust region. More precisely, the algorithm stops if the subspace S (see (4.1)) is increased in dimension by N once the solution is known to be on the boundary of the trust region and problems of the type (4.2) are solved. The reason for limiting the size of the subspaces S once the boundary has been reached in the GLTR Algorithm is motivated by the fact that the authors in [13] question whether high accuracy is needed for the TRS within a trust region framework. 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

$$\min_{x \in \mathbb{R}^n} f(x) := \frac{x^T A x}{x^T B x}, \tag{8.2}$$

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

$$Ax = \lambda B x.$$

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

To solve each problem we used the trust region method described by Algorithm 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 a positive number - the radius of the trust region at iteration j.

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*

$$\frac{\|\nabla f(x_j)\|}{1+|f(x_j)|} < 10^{-2}. \tag{8.3}$$

2. *Solve for δ_j*

$$\begin{array}{rll} \delta_j \in & \arg\min & q_j(x_j) := \nabla f(x_j)^T\delta + \frac{1}{2}\delta^T\nabla^2 f(x_j)\delta \\ & \text{s.t.} & \|\delta\|^2 \le s_j^2. \end{array} \tag{8.4}$$

3. *Evaluate $r_j = \frac{f(x_j)-f(x_j+\delta_j)}{q_j(x_j)-q_j(x_j+\delta_j)}$.*

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 (8.3) which has been scaled here, this algorithm is Algorithm 6.1 in [13]. We chose x_0 randomly and have fixed $s_0 = 1$.

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. The problems were generated with the following MATLAB code:

```
density = 0.5;
rc = 0.1; %(rc is desired reciprocal condition number)
A = sprandsym(n,density);
B = sprandsym(n,density,rc,1);
x0 = (0.5-rand(n,1));.
```

For each problem, we used different subroutines for solving the TRS (8.4) within the trust region method. Either the RW or the GLTR Algorithms is used.

If the RW Algorithm solves (8.4) and the solution is on the boundary of the trust region, we stop if the duality gap, *dgaptol*, (between TRS and (5.17)) satisfies

$$\sqrt{\text{dgaptol}} \le \min(0.1, \max(10^{-8}, 10^{-5}\|\nabla f(x_j)\|). \tag{8.5}$$

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

$$\|\nabla f(x_j) + \nabla^2 f(x_j)\delta_j\| \leq \min(0.1, \max(10^{-8}, 10^{-5}\|\nabla f(x_j)\|). \tag{8.6}$$

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 trust region or if

$$\gamma_{k+1}|e_{k+1}^T h_k| < \min(0.1, \max(10^{-8}, 10^{-5}\|\nabla f(x_j)\|)) \tag{8.7}$$

or (8.6) is satisfied, depending if the solution is on the boundary of the trust region or not (see Algorithm 4.1).

The motivation behind the stopping criteria (8.5) and (8.7) is to ask approximately for the same accuracy in terms of the duality gap. Recall from Section 6.2 that $\gamma_{k+1}|e_{k+1}^T h_k|$ is an approximation for the square root of the duality gap between TRS and its dual (6.1). The stopping criteria (8.5) and (8.7) are set to reflect this relationship.

For each problem, if the RW algorithm solves the TRS (8.4), we give the number of iterations taken by the trust region method 8.1 to converge. If the GLTR Algorithm is used, we give the number of iterations (*iter* in our tables) taken by the trust region method 8.1 to converge, the average number of conjugate-gradient steps (*cg* in our tables) - recall the GLTR Algorithm does pure conjugate-gradient steps until it converge or knows the solution to the TRS lies on the boundary of the trust region - and the number of cases where Algorithm 3.1 fails to solve the TRS (4.4) (see Algorithm 4.1). This last output (*hc2* in our tables) is an indicator of the (almost) hard case (case 2). The results are given in Tables 8.2, 8.3, 8.4.

To understand the results, we first need to specify the approximate solution to (8.4) we took when Algorithm 3.1 failed to solve the TRS (4.4) in the GLTR Algorithm. Say this occurred at the first iteration after INTERIOR has been set to *false*. In this case, for an approximation to the optimum of (8.4), we used the *Cauchy point* (see [13]), which is the minimizer of the quadratic objective function of (8.4), within the trust region, along the steepest-descent direction $-\nabla f(x_j)$. If at least one iteration of the GLTR Algorithm had been done once INTERIOR had been set to *false*, we used the latest h_k computed to recover an approximate optimum $Q_k h_k$ to the optimum of (8.4).

	Algorithm used for solving the TRS (8.4)									
	RW	GLTR with N=2			GLTR with N=6			GLTR with N=n		
problem	iter	iter	hc2	cg	iter	hc2	cg	iter	hc2	cg
1	16	36	5	13.6	25	0	12.5	16	0	9.6
2	33	22	0	11.6	55	19	13.5	48	16	13.6
3	50	21	0	9.9	15	0	9.7	52	16	15.1
4	41	39	8	13.3	45	16	13.6	32	3	12.0
5	25	45	10	14.7	19	0	11.1	25	0	11.7

Table 8.2: RW and GLTR Algorithms; TR framework; size n=20.

	Algorithm used for solving the TRS (8.4)									
	RW	GLTR with N=2			GLTR with N=6			GLTR with N=n		
problem	it	it	hc2	cg	it	hc2	cg	it	hc2	cg
1	31	48	17	17	34	5	11.4	32	2	10.1
2	38	26	0	13.5	27	1	11.9	32	2	13.5
3	22	25	0	13.1	22	0	9.5	22	0	9.4
4	20	39	4	12.1	31	1	11.1	20	0	7.9
5	25	26	0	12.0	22	0	12.0	22	0	11.8

Table 8.3: RW and GLTR Algorithms; TR framework; size n=25.

From the results in Tables 8.2, 8.3 and 8.4, we first observe that the (almost) hard case (case 2) occurs in many problems. Algorithm 8.1, using the RW Algorithm for (8.4), takes fewer iterations to converge compared to using the GLTR Algorithm. This is independently of N. This suggests that handling the hard case (case 2) should be en essential feature for a robust trust region method, contrary to statements in [13]. However, we reach the same conclusions mentioned in [13] when the hard case (case 2) does not occur. We observe the surprising fact that inexact solutions may indeed lead to less iterations in some cases. This clearly requires more study.

As we may expect, when the hard case (case 2) does not occur and N=n, a trust region method, using either the RW Algorithm or the GLTR Algorithm

	Algorithm used for solving the TRS (8.4)									
	RW	GLTR with N=2			GLTR with N=6			GLTR with N=n		
problem	it	it	hc2	cg	it	hc2	cg	it	hc2	cg
1	61	14	0	9.6	36	7	16.2	57	24	19.1
2	38	50	17	21.0	35	2	16.6	37	6	16.5
3	34	22	0	15.7	27	1	15.3	45	17	20.2
4	34	19	0	14	25	0	13.2	36	8	16.6
5	36	38	7	15.2	26	0	11.8	32	3	13.7

Table 8.4: RW and GLTR Algorithms; TR framework; size n=30.

to solve the TRS (8.4), takes more or less the same number of iterations. This should be the case since we are asking for the same accuracy in (8.5) and (8.7).

Note that all the TRS (8.4) solved in all our problems were nonconvex, i.e. the minimum eigenvalue of $\nabla^2 f(x_j)$ was always negative. This is surprising since we note that on average, many conjugate-gradient steps are taken before the GLTR Algorithm notices that the solution to (8.4) lies on the boundary of the trust region.

8.4 Solving Large and Sparse Trust Region Subproblems

The goal of the previous numerical subsections was to illustrate the different behaviors of the algorithms we are surveying and the size of the problems we were solving was not so important for what we were investigating. So far, the problems we have solved are considered to be small or medium size. The purpose of this last subsection is to show we can solve much larger problems. The following result show how we used the RW Algorithm to solve problems of size $n = 100000$ of different density.

Precisely, in Table 8.5 below, each row 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 non-zero) and each column to the value of the parameter *dgaptol*. In each entry of the table is given the average taken over 5 random trust region subproblems of the computation time (cpu), the

number of matrix-vector multiplications (mvm) and the number of iterations (it) taken by the RW Algorithm to find an approximate solution.

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

Table 8.5: RW Algorithms; TRS framework; size n=100 000.

As we may expect, the computation time and the matrix-vector multiplications increase as the density increases and the duality gap tolerance decreases. Furthermore, considering the reasonable length of the computation time taken to solve such TRS, we conclude that it is now within our reach to use trust region methods to minimize functions with hundreds of thousands of variables assuming the Hessian has a sparse structure.

9 Conclusion

In this paper we have presented a survey of the TRS. Our emphasis was on robustness and on solving large sparse problems. We focused on three algorithms using the modern primal-dual approaches: the classical MS algorithm and two recent algorithms, the RW and GLTR Algorithms, designed to solved large and sparse TRS. We have also studied many duals to TRS which can be formulated as semidefinite programs. These were used to link all three algorithms within a semidefinite framework. In addition to providing a clear and simple unifying analysis between the different algorithms, this framework, as we have seen, may be used to explain the strengths/weaknesses of

the algorithms. For example, we have shown that the stopping criteria of the GLTR algorithm is based on the approximation of a duality gap. The framework may also be used to derive new methods as it was done for the RW Algorithm.

We also presented a modified/enhanced RW algorithm, where new heuristics and techniques were introduced, in particular when taking a primal step to the boundary. However, the main improvement came from a new way of treating the 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. Precisely, an equivalent problem to TRS is first considered, where the eigenvalues of A are shifted so they become non-negative. Second, eigenvectors corresponding to $\lambda_1(A)$ are used to perturb the matrix A. If the hard case occurs initially, after i consecutive perturbation, where i is the multiplicity of $\lambda_1(A)$, we obtain an equivalent TRS where the hard case (case 2) cannot occur.

Our final section included numerics which showed the advantage of using the RW Algorithms over the MS Algorithms 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 trust region method may have an impact on the total number of iterations if the hard case occurs frequently enough. Thus, the robustness of a TRS Algorithm is indeed an important feature, even though we should mention again that inexact solutions may lead to less iterations in a trust region method when the hard case does not occur in the subproblems. We finally showed it is possible to solve large sparse TRS with hundreds of thousands of variables in a reasonable number of iterations.

A Notation

- TRS, the trust region subproblem
- $q(x) := x^T Ax - 2a^T x$, TRS objective function
- $\lambda_i(A)$, i-th smallest eigenvalue of A
- λ^*, optimal Lagrange multiplier for TRS
- $x(\lambda) = (A - \lambda I)^\dagger a$, solution of $A - \lambda I = a$ of minimum norm
- $h(\lambda) = -a^T (A - \lambda I)^\dagger a + \lambda s^2$, dual functional

- $\phi(\lambda) := \|x(\lambda)\| - s$, primal feasibility
- $\psi(\lambda) := \frac{1}{s} - \frac{1}{\|x(\lambda)\|} = 0$, secular equation
- $D(t)$, parameterized matrix using A, a
- $k(t) = (s^2+1)\lambda_1(D(t)) - t$, unconstrained dual maximization function
- $y_0(t)$, first component of normalized eigenvector of $\lambda_1(D(t)$
- $k'(t) = (s^2+1)y_0(t)^2 - 1$, equivalent primal feasibility
- $\psi(t) = \sqrt{s^2+1} - \frac{1}{y_0(t)}$, equivalent secular function

References

[1] S. J. Benson, Y. YE, and X. Zhang. Solving large-scale sparse semidefinite programs for combinatorial optimization. *SIAM J. Optim.*, 10(2):443–461 (electronic), 2000.

[2] A. Ben-Tal and M. Teboulle. Hidden convexity in some nonconvex quadratically constrained quadratic programming. *Math. Programming*, 72(1, Ser. A):51–63, 1996.

[3] Å. Björck. *Numerical methods for least squares problems.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1996.

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

[5] D.C. SORENSEN. Minimization of a large-scale quadratic function subject to a spherical constraint. *SIAM Journal on Optimization*, 7(1):141–161, 1997.

[6] J.W. Demmel. *Applied numerical linear algebra.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[7] A.V. Fiacco. *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming*, volume 165 of *Mathematics in Science and Engineering.* Academic Press, 1983.

[8] R. Fletcher. *Practical methods on optimization.* John Wiley and Sons, second edition, 1987.

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

[10] D.M. Gay. Computing optimal locally constrained steps. *SIAM Journal on Scientific and Statistical Computing*, 2(2):186–197, 1981.

[11] E.G. Gol'stein. *Theory of Convex Programming.* American Mathematical Society, Providence , RI, 1972.

[12] G.H. Golub and C.F. Van Loan. *Matrix Computation.* The Johns Hopkins University Press, third edition, 1996.

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

[14] W.W. Hager. Minimizing a quadratic over a sphere. Technical report, University of Florida, Gainsville, Fa, 2000.

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

[16] R.A. Horn and C.R. Johnson. *Matrix Analysis.* Cambridge University Press, 1987.

[17] Y. Levin and A. Ben-Israel. The newton bracketing method for convex minimization. *Comput. Optimiz. Appl.*, 21:213–229, 2002.

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

[19] J. J. Moré. Generalizations of the trust region problem. *Optim. Methods Software*, 2:189–209, 1993.

[20] J.J. Moré and D.C. Sorensen. Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing*, 4(3):553–572, 1983.

[21] S.G. Nash and J. Nocedal. 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, 1991.

[22] J. Nocedal and S.J. Wright. *Numerical optimization.* Springer-Verlag, New York, 1999.

[23] S. Poljak, F. Rendl, and H. Wolkowicz. A recipe for semidefinite relaxation for $(0,1)$-quadratic programming. *J. Global Optim.*, 7(1):51–73, 1995.

[24] C. Reinsch. Smoothing by spline functions. *Numerische Mathematik*, 10:177–183, 1967.

[25] C. Reinsch. Smoothing by spline functions ii. *Numerische Mathematik*, 16:451–454, 1971.

[26] M.D. Reinsch. An algorithm for minimization using exact second derivatives. Technical Report 515, Harwell Laboratory, Harwell, Oxfordshire,England, 1973.

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

[28] D.C. Sorensen. Newton's method with a model trust region modification. *SIAM Journal on Numerical Analysis*, 19(2):409–426, 1982.

[29] T. Steihaug. The conjuguate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, 20(3), 1983.

[30] R.J. Stern and H. Wolkowicz. Indefinite trust region subproblems and nonsymmetric eigenvalue perturbations. *SIAM Journal on Optimization*, 5(2):286–313, 1995.

[31] P.D. Tao and L.T.H. An. 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, 1996.

[32] A.N. TIKHONOV and V.Y. ARSENIN. *Solutions of Ill-Posed Problems.* V.H. Winston & Sons, John Wiley & Sons, Washington D.C., 1977. Translation editor Fritz John.

[33] P.L. TOINT. *Towards an efficient sparsity exploiting Newton method for minimization*, volume Sparse Matrices and Their Uses. I.S.Duff, academic press edition, 1981. pp.57-88.

[34] R.J. VANDERBEI. *Linear Programming: Foundations and Extensions.* Kluwer Acad. Publ., Dordrecht, 1998.

[35] H. WOLKOWICZ, R. SAIGAL, and L. VANDENBERGHE, editors. *HANDBOOK OF SEMIDEFINITE PROGRAMMING: Theory, Algorithms, and Applications.* Kluwer Academic Publishers, Boston, MA, 2000. xxvi+654 pages.

[36] S. WRIGHT. *Primal-Dual Interior-Point Methods.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, Pa, 1996.

[37] Y. YE. Combining binary search and Newton's method to compute real roots for a class of real functions. *Journal of Complexity*, 10:271–280, 1994.