A Survey of the Trust Region Subproblem within a Semidefinite Framework

by

Charles Fortin

A thesis

presented to the University of Waterloo

in fulfilment of the

thesis requirement for the degree of

Master of Mathematics

in

Combinatorics and Optimization

Waterloo, Ontario, Canada, 2000

Abstract

Trust region subproblems arise within a class of unconstrained methods called trust region methods. The subproblems consist of minimizing a quadratic function subject to a norm constraint. This thesis is a survey of different methods developed to find an approximate solution to the subproblem. We study the well-known method of Moré and Sorensen [18] and two recent methods for large sparse subproblems: the so-called Lanczos method of Gould et al. [7] and the Rendl and Wolkowicz algorithm [31]. The common ground to explore these methods will be semidefinite programming. This approach has been used by Rendl and Wolkowicz [31] to explain their method and the Moré and Sorensen algorithm; we extend this work to the Lanczos method. The last chapter of this thesis is dedicated to some improvements done to the Rendl and Wolkowicz algorithm and to comparisons between the Lanczos method and the Rendl and Wolkowicz algorithm. In particular, we show some weakness of the Lanczos method and show that the Rendl and Wolkowicz algorithm is more robust.

Acknowledgements

Le temps que l'on prend, pour dire merci
Est le seul qui reste au bout de nos jours
Les voeux que l'on fait, l'effort que l'on sème
Chacun les récolte en soi-même
Aux beaux jardins du temps qui court

C'est sur cette adaption de *Gens du Pays* du chanteur et poète Gilles Vigneault que j'aimerais remercier tout ceux et celles qui ont rendu possible cette thèse de maîtrise. Car si sur la page couverture n'apparaît que mon nom, le lecteur attentif pourra y lire, dissimulés sous le papier, entre le *Charles* et le *Fortin* les noms des gens mentionnés ci-dessous.

À commencer par mes parents, Monique et Normand, qui m'ont toujours épaulé dans mes projets et mes rêves et qui se sont toujours inquiétés de mon sort en ce lointain Ontario.

Of course, I want to thank my supervisor Henry Wolkowicz, for his availability, his support and his devotion to help me in my research efforts. He has succeeded in making come true everything I hoped to achieve when coming to the University of Waterloo.

Thank you also to Philippe Toint and Nick Gould for answering my questions on the Lanczos method.

I will never forget all the friends I met here during these two years and who made my time here enjoyable. First of all, my two German roomates, Oliver and Jan Peter; it is very rare in ones life to find two great friends all at once. Aussi je remercie ma ≪ grande ≫ amie Mimi pour toute son attention et pour m'avoir fait connaître le trajet Waterloo-Toronto. Je ne saurais évidemment oublier Christian, Ève, Philippe et Kati avec qui j'ai partagé *l'expérience Ontarienne*. From my first year here, I also think of Matthias, Achim, Antonella, Jan and my school buddy James. Ma deuxième année ici a été tout aussi excitante, et je le dois en partie aux charmantes Laurence et Michelle; merci d'avoir réaffirmé par contraste mon identité Québecoise. More recent friends are Peter, Marco, Benjamin and Eric. I will also remember the school friends and colleagues: James, Debbie, Marina, Kerri, Phil, Lise, George, John, Hristo, Jason and Serge. A particular thanks to Brian for his encouragements and his everyday joyousness and to Miguel for his suggestions to improve this thesis and for his help with my math and computer problems. I also have special thoughts for the people in Aiki-Jujutsu: Charlotte, Malte, Connie, Mike, Paul and Josh.

Je dois également remercier les amis de Québec qui ont pris le temps de m'écrire ou de me téléphoner pour qu'on ne se perde pas de vue: Marie-Diane, Fabrice, Sophie, Marc-André, Marc-Hubert, Sébastien, mes frères Mario et Vincent, Isabelle, ma nièce Anne-Sophie (merci de ne pas avoir grandi trop vite!), Dubé, Mike, Dan, Labine et Annie.

Je ne serais rien sans vous tous and I owe it all to you.

Contents

Introduction

In unconstrained optimization, we deal with the standard problem of finding the minimum of a function $f : \mathbb{R}^n \to \mathbb{R}$. If we assume the function is twice continuously differentiable, many methods can be applied to find the minimum. Steepest descent and Newton's method are probably among the best known. Newton's method relies on a second order approximation of the function f at each iteration. Although it has proven to be very efficient, one of its disadvantages is that it does not possess global convergence and the performance of the method is very dependent on the initial estimate. In particular, one can converge to a saddle point, a local maximum, or a local minimum.

One way to get around these difficulties is to minimize at each iteration the same quadratic model as in Newton's method, but instead of considering the whole space for the minimization, we restrict ourselves to a ball which is referred to as the *trust region*. This is the idea behind *trust region methods*. The major difficulty is to efficiently solve at each iteration the *trust region subproblem* (TRS)

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

Here, A is an $n \times n$ symmetric matrix, a is an $n \times 1$ vector, s is a positive scalar and x is the $n \times 1$ vector of unknowns. All matrix and vector entries are real. It is this last

problem that is the main subject of this thesis.

Depending on the values of A, a and s, different ways of solving the trust region subproblem need to be considered. Two different cases may occur and are referred to in the literature as the *easy case* and the *hard case*. The hard case or near hard case is what causes numerical difficulties in solving the problem.

A major recent concern of TRS is exploiting the sparsity of the matrix A for large problems. This is of course linked with the increasing speed of computers and any modern trust region algorithm has to take this factor into account. Also, since the trust region subproblem needs to be approximately solved many times, fast convergence in very few iterations is needed, rather than just quadratic convergence for example. Indeed, although quadratic convergence is fast when one gets close to the solution, it does not guarantee that the overall number of iterations needed to obtain an approximation to the solution will be small.

In this thesis, we study three different methods that consider all of the above difficulties for TRS. The first one we study is the standard Moré and Sorensen algorithm [18] that was published in 1983 and is the first algorithm able to handle the hard case efficiently. The two other ones are more recent and were designed to solve large problems. We will look at the primal-dual algorithm of Rendl and Wolkowicz [31] and the so-called *Lanczos method* of Gould, Lucidi, Roma and Toint [7]. The first authors have shown that semidefinite programming could be use to describe the steps of the Moré and Sorensen algorithm and also to derive new algorithms like theirs. We use the same framework to show that the same can be done with the Lanczos method. Semidefinite programming will therefore be the link between all the above methods. Other recent algorithms for large and sparse trust region subproblems are the ones of Sorensen [27], Santos and Sorensen [25] and Hager [8]; though they will not be considered here.

This thesis has the following structure: in Chapter 1, we look at the evolution of

the ideas and methods behind the trust region subproblem. In Chapter 2, we study the motivation for solving this problem, i.e. trust region methods. We give an algorithm for trust region methods and outline their attractive properties. In Chapter 3, we present the general theory needed to approach the trust region subproblem. Necessary and sufficient conditions are derived and what differentiates the easy and the hard case is explained. In Chapter 4, we consider the Moré and Sorensen algorithm and the Lanczos method. In Chapter 5, we present the duality results associated with TRS. This section is the key to derive our semidefinite framework and explain the Rendl and Wolkowicz algorithm which is described in Chapter 6. In Chapter 7, we give numerical results. First, we study the performance of a new step to the boundary used in the Rendl and Wolkowicz algorithm. Second, we compare the performance of the Lanczos method and the Rendl and Wolkowicz algorithm when, respectively, used to solve the trust region subproblems within a trust region method.

The new contributions of this thesis are found in Chapters 5, 6 and 7. In Section 5.1, we polished the work of Rendl and Wolkowicz [31] concerning the different dual problems related to the trust region subproblem. Section 5.3 sets the Lanczos method within a semidefinite framework is entirely new. In Chapter 6, a new primal step to the boundary for the Rendl and Wolkowicz algorithm is suggested in Section 6.3.2. Finally, the results and conclusion of the numerical experiments of Chapter 7 are also new contributions to the field.

Chapter 1

Historical Background

Among the papers related to trust region methods and to the trust region subproblem, many of them cite the work by Levenberg [12] in 1944 who introduced the basic idea behind trust region methods. Levenberg was interested by the least squares problem

$$\min \sum_{1=1}^{n} f_i(x)^2 \tag{1.1}$$

encountered particularly in curve fitting, where the f_i's are nonlinear functions. The usual approach for this problem was to approximate f_i with a first order Taylor's approximation F_i about x_k and to set the derivative of the function $\sum_{i=1}^{n} F_i(x)^2$ to 0 to get x_{k+1}. Of course, this method is faced with the problem that x_{k+1} can be too far away from x_k and a decrease in the initial program (1.1) might not occur because the linear model does not hold so far away. Levenberg thought of a way to restrict the distance where the next iterate can be found, therefore setting the basic idea of trust region methods. His idea though was different to what is now used in trust region methods. Actually, he used a

quadratic penalty method and solved

$$\min\ \omega \sum_{1=1}^{n} f_i(x)^2 + \|x\|^2,$$

where ω was a well chosen scalar. He proved that with this method he could get decrease at each iterate for problem (1.1).

A more direct link to trust region methods is the work of Marquardt [14] in 1963. He solved the same problem as Levenberg, i.e. he was minimizing $\sum_{i=1}^{n} F_i(x)^2$, but at each iteration he would really find the solution to a trust region subproblem. Due to the structure of the problem, he would only face trust region subproblems for which the quadratic objective was a convex function. He would not directly find the minimum of a convex quadratic inside a given sphere, but rather within an unknown sphere (i.e. the radius of the sphere was not known a priori, but was instead implicitly determined in solving his subproblem). Similarly to what is done in trust region methods, he also had a scheme to adjust the radius of his trust region at each iteration. It is the work of Marquardt and Levenberg that would eventually inspire Powell, in 1970, to derive the first trust region algorithm for solving unconstrained minimization problems (see [20] and [21]).

In 1966, Goldfeld, Quandt and Trotter [5] wrote a paper called *Maximization by quadratic hill-climbing* which is the origin of trust region methods and they solved trust region subproblems for non-convex quadratic objectives. Although their paper did not go through all the subtleties of the trust region subproblem, as for example the distinction between the easy case and the hard case, they gave the necessary and sufficient optimality conditions. Their trust region method could solve unconstrained minimization problems, but, with Marquardt, their trust region subroutine would not directly minimize a quadratic inside a fixed sphere.

The disadvantage of the methods of Marquardt and Goldfeld, Quandt and Trotter was that the trust region subproblems they could solve had a small neighborhood when a special case occurred which is now referred to as the *hard case* in the literature. Moreover, being able to handle this so-called hard case is essential when the minimum of a quadratic over a fixed sphere is sought. In 1981, Gay [4] showed how to recognize the hard case numerically and stated a theorem that could be used to solve the problem when this case occurred. His theorem, as mentioned two years later in a paper by Moré and Sorensen [18], had some numerical disadvantages, since many iterations were needed to be able to recognize the hard case when it occurred.

As the theory of the trust region subproblem evolved, so too did the theory surrounding its reason for existence: trust regions methods. One example would be the paper by Sorensen [26] in 1982 where he derived strong convergence results for his algorithm, which is similar to the one found in recent nonlinear optimization books (see for example [1] and [3]). In his paper, Sorensen proved, under some mild assumptions, global convergence and quadratic convergence for his method.

In 1983, Moré and Sorensen [18] wrote *Computing a trust region step*, which now remains a classic for the trust region subproblem. They developed a way to solve efficiently the problem in the easy case using Newton's method on a nearly linear function and only basic linear algebra techniques needed to be applied. In the hard case, they stated a lemma that was not as restrive as what Gay [4] initially proposed. In fact, their algorithm is very effective in this case.

Since the Moré and Sorensen algorithm uses Cholesky's factorization, it is not designed to take advantage of the sparsity for large sparse problems. Since the publication of their paper, many authors considered attacking the problem in a different way that would not compromise sparsity (see for example [7],[25],[27] and [31]). This is where the research is now, as increasingly fast computers enable us to solve problems with a large number of

variables and handling sparsity is a key factor now in any algorithm.

Chapter 2

The Trust Region Method

In this chapter, we study how the trust region method (TRM) is a natural way to implement Newton's method in unconstrained optimization. The convergence properties of TRM are very appealing as we can expect to satisfy first and second order optimality conditions. Furthermore, under some assumptions we will specify, TRM achieves global convergence. This is a noticeable improvement to Newton's method.

Newton's method can be applied to the problem of finding the unconstrained minimum of a continuous function, say $f(\cdot)$, i.e. solving

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & x \in \mathbb{R}^n. \end{aligned}$$

Given x_k, the next iterate x_{k+1} is computed by finding a minimum of the quadratic model of $f(\cdot)$ about x_k, i.e.

$$\begin{aligned} x_{k+1} \in \quad \operatorname{argmin} \quad & \tilde{q}(x) := f(x_k) + \nabla f(x_k)^T(x - x_k) + \tfrac{1}{2}(x - x_k)^T \nabla^2 f(x_k)(x - x_k) \\ \text{s.t.} \quad & x \in \mathbb{R}^n. \end{aligned}$$

Assuming positive definiteness for $\nabla^2 f(x_k)$ yields

$$x_{k+1} = x_k - (\nabla^2 f(x_k))^{-1} \nabla f(x_k).$$

Usually, the restricted step Newton's method is written as

$$x_{k+1} = x_k - \alpha_k (\nabla^2 f(x_k))^{-1} \nabla f(x_k), \quad \alpha_k \in (0, 1], \tag{2.1}$$

where α_k is a carefully chosen step length. Two problems arise with this method. First, because we need to solve a linear system, if it is ill-conditioned we will have difficulties in computing x_{k+1}. Second, if $\nabla^2 f(x_k)$ is not positive definite, x_{k+1} might not yield a decrease for the objective function $f(\cdot)$ as $(\nabla^2 f(x_k))^{-1} \nabla f(x_k)$ might not be a descent direction and the step $\alpha_k (\nabla^2 f(x_k))^{-1} \nabla f(x_k)$ might increase $f(\cdot)$ for any positive α_k.

The idea behind TRM is to correct these difficulties by restricting the iterate x_{k+1} to belong to a closed neighborhood of x_k and to set x_{k+1} to be the minimum of the quadratic model about x_k within this neighborhood. More precisely, for $s_k \in \mathbb{R}_+$ a positive scalar, x_{k+1} is set to be

$$\begin{aligned} x_{k+1} \in \quad & \operatorname{argmin} \quad \tilde{q}(x) \\ & \text{s.t.} \qquad \|x - x_k\|^2 \leq s_k^2. \end{aligned}$$

Because the feasible set is compact, x_{k+1} is now well defined independently of $\nabla^2 f(x_k)$. Furthermore, for sufficiently small s_k, $f(x_{k+1}) < f(x_k)$ unless both $\nabla f(x_k) = 0$ and $\nabla^2 f(x_k) \succeq 0$, i.e. unless first and second order necessary optimality conditions hold at x_k. The above problem can be solved if, given A, an $n \times n$ real symmetric matrix, $a \in \mathbb{R}^n$, and $s \in \mathbb{R}_+$ the following problem, which we refer to as the *trust region subproblem* (TRS),

can be solved:

$$\text{(TRS)} \qquad \begin{array}{ll} \min & q(x) := x^T A x - 2a^T x \\ \text{s.t.} & \|x\|^2 \leq s^2. \end{array} \tag{2.2}$$

The next step now is to construct an efficient algorithm. The idea is to modify the size of the neighborhood at each iteration depending on how well the quadratic model approximates the function f. If the model is valid in a large neighborhood, then we wish to take a neighborhood as large as possible. On the other hand, if the model is a poor approximation of f, we may need to reduce the size of our neighborhood to ensure a decrease in the objective function. Let x_k be our current iterate and $x_k + \delta_k$ be the solution to the minimization of our quadratic model about x_k over the ball of radius s_k centered at x_k. Then

$$f(x_k) - f(x_k + \delta_k)$$

will be the *actual reduction* of our objective function and

$$\tilde{q}(x_k) - \tilde{q}(x_k + \delta_k)$$

will be the *predicted reduction* of our objective function made by our quadratic model. To measure how good our quadratic model is, we compute the ratio of the actual reduction to the predicted reduction, i.e.

$$r_k := \frac{f(x_k) - f(x_k + \delta_k)}{\tilde{q}(x_k) - \tilde{q}(x_k + \delta_k)}.$$

If the ratio is close to 1 or greater than 1, we might want to enlarge the neighborhood where we *trust* our model to allow a larger step. On the other hand, if the ratio is small,

or even negative, then we will reduce our neighborhood.

We can now outline the trust region method. Many variants exist. Our algorithm is taken from Fletcher [3].

Algorithm 2.1. (Trust Region Method)

1. *Given* x_k *and* s_k, *calculate* $\nabla f(x_k)$ *and* $\nabla^2 f(x_k)$.

2. *Solve for* δ_k

$$\begin{aligned} \delta_k \in \quad & \text{argmin} \quad \nabla f(x_k)^T \delta_k + \tfrac{1}{2}\delta_k^T \nabla^2 f(x_k)\delta_k \\ & \text{s.t.} \qquad \|\delta_k\|^2 \leq s_k^2. \end{aligned}$$

3. *Evaluate* r_k.

4. (a) *If* $r_k < 0.25$ *set* $s_{k+1} = \|\delta_k\|/4$.

   (b) *If* $r_k > 0.75$ *and* $\|\delta_k\| = s_k$ *set* $s_{k+1} = 2s_k$.

   (c) *Otherwise set* $s_{k+1} = s_k$.

5. *If* $r_k \leq 0$ *set* $x_{k+1} = x_k$ *else* $x_{k+1} = x_k + \delta_k$.

We end this section with two theorems about first and second order optimality conditions when minimizing function $f(\cdot)$. Proofs of these theorems can be found in Fletcher [3].

Theorem 2.1 (Global convergence). *For Algorithm 2.1, if* $x_k \in B \subset \mathbb{R}^n \; \forall k$, *where* B *is bounded, and if* $f \in \mathbb{C}^2$ *on* B, *then there exists an accumulation point* x^∞ *which satisfies first and second order necessary conditions.*

Theorem 2.2 (Quadratic Convergence). *If the accumulation point* x^∞ *of Theorem 2.1 also satisfies the second order sufficient conditions, then for the main sequence* $r_k \to 1$,

$x_k \to x^\infty$, $\inf s_k > 0$ *and the bound* $\|\delta\|^2 \leq s_k^2$ *is inactive for sufficiently large* k. *Also the convergence is quadratic.*

The first theorem tells us, if the sequence of iterates is bounded, that we can expect to have an accumulation point satisfying the first and second order optimality conditions. The second theorem tells us that eventually the method reduces to Newton's method if we converge to a local minimum where the second order sufficient optimality condition holds, hence quadratic convergence occurs.

An interesting fact to be aware of for a trust region algorithm like Algorithm 2.1 is that accumulation points for which stationarity does not hold may exist. This has recently been shown by Yuan [34]. He constructed an example where one stationary accumulation point exists (as expected according to Theorem 2.1), but where two non-stationary accumulation points also exist. Yuan [34] shows that trust region methods that feature a step similar to step 5 of Algorithm 2.1 may have non-stationary accumulation points, i.e. we may have

$$\limsup_{k\to\infty} \|\nabla f(x_k)\| \ > 0.$$

On the other hand, trust region algorithms that set $x_{k+1} = x_k$ if $r_k \leq \tau_0$, where τ_0 is a positive number greater than 0, all have stationary accumulation points, i.e.

$$\lim_{k\to\infty} \|\nabla f(x_k)\| \ = 0.$$

In the numerical tests of Chapter 7, we use a trust region method (Algorithm 7.1) that has this feature.

This chapter gave the motivation for studying TRS. The next chapter deals with how one might compute the desired minimum. Fortunately, the problem has a fair amount of

structure.

Chapter 3

The Structure of the Trust Region Subproblem

In this chapter, we present the theory needed to understand TRS. We will from now on refer to TRS as it was stated in (2.2). Of course the first step is to derive the Karush-Kuhn-Tucker optimality conditions. It is shown that the second order optimality condition has a strengthened form for TRS. We also demonstrate that unless a special situation occurs, the optimal solution will always lie on the boundary. The two theorems needed to show this and their proofs are taken from Sorensen [26] (see also Gay [4]). Of course, knowing these optimality conditions is essential as any algorithm for TRS will aim at satisfying them.

We end this chapter with two different cases we need to be aware of to solve TRS efficiently. Most of the difficulties come from the case referred to as the hard case and we show by example why being able to solve this case is essential.

3.1 Optimality Conditions

We first derive the Karush-Kuhn-Tucker necessary optimality conditions and establish that the second order necessary conditions can be stated in a strengthened form to make them also sufficient. In particular, we show that the convexity of the Lagrangian function and the complementary slackness equation implies that the conditions are sufficient. We let in the following theorem and for the rest of this thesis the symbol $\succeq$ be the Löwner partial order on positive semidefinite matrices, i.e. $A \succeq B$ if $A-B$ is positive semidefinite. Similarly, $A \succ B$ if $A - B$ is positive definite.

Theorem 3.1 (Necessary and Sufficient Conditions). *x^* is a solution to (2.2) if and only if $\|x^*\|^2 \leq s^2$ and x^* is a solution to an equation of the form*

$$(A - \lambda^* I)x^* = a$$

with $A - \lambda^* I \succeq 0$, $\lambda^* \leq 0$, and $\lambda^*(s^2 - \|x^*\|) = 0$. Furthermore, if $A - \lambda^* I \succ 0$, x^* is unique.

Proof:

First suppose that x^* is a solution to TRS. Since $s > 0$, without loss of generality, we can assume that the trivial case $x^* = 0$ does not hold. Note that x^* is a regular point, since the only constraint has non zero gradient and therefore is linearly independent. Therefore, there exists a unique Lagrange multiplier λ^*. Define the Lagrangian function

$$L(x, \lambda) := x^T A x - 2a^T x - \lambda(\|x\|^2 - s^2).$$

The Karush-Kuhn-Tucker conditions for (2.2) yield

$$\|x^*\|^2 \leq s^2 \quad \text{(Feasibility)},$$
$$\nabla_x L(x^*, \lambda^*) = (A - \lambda^* I)x^* - a = 0 \quad \text{(Stationarity)},$$
$$\lambda^*(\|x^*\| - s^2) = 0 \quad \text{(Complementary Slackness)},$$
$$\lambda^* \leq 0,$$
$$\text{and if } \|x^*\|^2 = s^2, \quad y^T(A - \lambda^* I)y \geq 0 \quad \text{for all } y \ni y^T x^* = 0 \quad \text{(2nd order nec. cond.)}.$$

We now prove that the second order necessary condition can in fact be strengthened to $A - \lambda^* I \succeq 0$. Since x^* solves TRS, $q(x) \geq q(x^*)$ for all x satisfying $\|x\|^2 = \|x^*\|^2$. This with $(A - \lambda^* I)x^* = a$ yields

$$x^T Ax - 2x^{*T}(A - \lambda^* I)x \geq x^{*T} Ax^* - 2x^{*T}(A - \lambda^* I)x^*.$$

Using $\|x\|^2 = \|x^*\|^2$, we can write the previous equation in the following way:

$$x^T(A - \lambda^* I)x - 2x^{*T}(A - \lambda^* I)x \geq -x^{*T}(A - \lambda^* I)x^*$$

$$\Leftrightarrow x^T(A - \lambda^* I)x - 2x^{*T}(A - \lambda^* I)x + x^{*T}(A - \lambda^* I)x^* \geq 0$$

$$\Leftrightarrow (x^* - x)^T(A - \lambda^* I)(x^* - x) \geq 0.$$

This yields

$$y^T(A - \lambda^* I)y \geq 0 \quad \text{for all } y \ni y^T x^* \neq 0,$$

since for $y \ni y^T x^* \neq 0$, $y = \alpha(x^* - x)$ for a well chosen $\alpha \neq 0$ and x (choose $\alpha = \frac{\|y\|^2}{2y^T x^*}$ and $x = x^* - \frac{1}{\alpha}y$). This last inequality in addition to the second order necessary condition yields

$$A - \lambda^* I \succeq 0.$$

To prove the converse, let x^* and $\lambda^* \leq 0$ be a solution to

$$\begin{aligned} &\|x^*\|^2 \leq s^2, \\ &(A - \lambda^* I)x^* = a, \\ &\lambda^*(\|x^*\|^2 - s^2) = 0, \\ &A - \lambda^* I \succeq 0. \end{aligned}$$

Then for any $x \in \mathbb{R}^n$

$$(x^* - x)^T (A - \lambda^* I)(x^* - x) \geq 0 \tag{3.1}$$

$$\Leftrightarrow x^T (A - \lambda^* I)x - 2x^{*^T}(A - \lambda^* I)x \geq x^{*^T}(A - \lambda^* I)x^* - 2x^{*^T}(A - \lambda^* I)x^*$$

$$\Leftrightarrow x^T (A - \lambda^* I)x - 2a^T x \geq x^{*^T}(A - \lambda^* I)x^* - 2a^T x^*$$

$$\Leftrightarrow q(x) \geq q(x^*) - \lambda^*(\|x^*\|^2 - \|x\|^2).$$

Now since $\lambda^*(\|x^*\|^2 - s^2) = 0$, either $\lambda^* = 0$ or $\|x^*\|^2 = s^2$. If $\lambda^* = 0$, the last inequality establishes that x^* solves TRS and that it is an unconstrained minimizer for $q(x)$,

i.e a minimizer over $\mathbb{R}^n$. Alternatively, if $\|x^*\|^2 = s^2$, then $-\lambda^*(\|x^*\|^2 - \|x\|^2) \geq 0$ for all x such that $\|x\|^2 \leq s^2$ and again the last inequality establishes that x^* solves TRS. Note also that if $\|x^*\|^2 = s^2$ then the last inequality implies that x^* is a solution to $\min\{q(x) : \|x\|^2 = s^2\}$ independent of the sign of λ^*. Finally, uniqueness follows because (3.1) holds strictly for $x \neq x^*$ if $A - \lambda^* I \succ 0$. ■

We now show that if an unconstrained minimum exists for $q(\cdot)$, unless it is unique and lies in the interior of the trust region $\{x : \|x\|^2 \leq s^2\}$, there will exist a solution on the boundary, that is $\|x^*\|^2 = s^2$ for some optimal x^*.

Theorem 3.2. *The problem (2.2) has no solution on the boundary if and only if A is positive definite and $\|A^{-1}a\|^2 < s^2$.*

Proof:

If A is positive definite and $\|A^{-1}a\|^2 < s^2$, then it follows that the minimizer of $q(x)$ is unique and because it lies within the trust region, there is no solution on the boundary. If no solution lies on the boundary, then the optimal solution x^* satisfies $\|x^*\|^2 < s^2$ and by complementary slackness we must have $\lambda^* = 0$. Since $A - \lambda^* I \succeq 0$ by Theorem 3.1, then $A \succeq 0$. If A was singular, then choosing $z \ni Az = 0$ for some z such that $\|x^* + z\|^2 = s^2$ would imply, by Theorem 3.1, that $x^* + z$ is an optimal solution to TRS, contradicting the fact that no solution lies on the boundary. Therefore, A must be positive definite. By stationarity, $s^2 > \|x^*\|^2 = \|A^{-1}a\|^2$. ■

3.2 Different Cases to Consider

Looking back at the optimality conditions, if we assume a solution to TRS occurs on the boundary, it would seem natural to attempt solving the equation

$$\|(A - \lambda I)^{-1}a\|^2 = s^2. \tag{3.2}$$

This leads us to wonder when is $A - \lambda^* I$ singular. The conditions for this to occur are well know in the literature. Singularity of the matrix $A - \lambda^* I$ is referred to as case 2 of the *hard case*. Non-singularity can result in either the so-called *easy case* and also case 1 of the *hard case*. Some authors refer both to the easy case and case 1 of the hard case as the easy case.

If a is not perpendicular to the null space

$$\{z : (A - \lambda_1(A)I)z = 0\},$$

where $\lambda_1(A)$ is the smallest eigenvalue of A, then we have the easy case and $A - \lambda^* I \succ 0$. In particular $A - \lambda^* I$ is invertible. We can therefore solve (3.2). To see this, note that

$$a \not\perp \mathcal{N}(A - \lambda_1(A)I) \Rightarrow a \notin \mathcal{R}(A - \lambda_1(A)I) \Rightarrow \nexists x \ni (A - \lambda_1(A)I)x = a,$$

where the first implication follows from $\mathcal{N}(A - \lambda_1(A)I) \perp \mathcal{R}(A - \lambda_1(A)I)$ ($\mathcal{N}$ stands for the null space and $\mathcal{R}$ for the range space). The last result implies that for $\lambda = \lambda_1(A)$ stationarity is not satisfied, hence $\lambda^* \neq \lambda_1(A)$. Since $A - \lambda^* I \succeq 0 \Rightarrow \lambda^* \leq \lambda_1(A)$ and $A - \lambda^* I \succ 0 \Leftrightarrow \lambda^* < \lambda_1(A)$, then $A - \lambda^* I \succ 0$.

In a complementary manner, the hard case occurs when a is perpendicular to the eigenspace of the smallest eigenvalue of A. Two possibilities may occur: $\lambda^* < \lambda_1(A)$ and

this is referred to as the hard case (case 1), or $\lambda^* = \lambda_1(A)$ and we call this the hard case (case 2). We resume the three different cases in the following table:

Easy case	Hard case (case 1)	Hard case (case 2)
1) $a \not\perp \mathcal{N}(A - \lambda_1(A)I)$	1) $a \perp \mathcal{N}(A - \lambda_1(A)I)$	1) $a \perp \mathcal{N}(A - \lambda_1(A)I)$
2) $\lambda^* < \lambda_1(A)$	2) $\lambda^* < \lambda_1(A)$	2) $\lambda^* = \lambda_1(A)$
Note that 1) implies 2)		

Table 3.1: The three different cases for the trust region subproblem.

In the hard case (case 1) no difficulty occurs and we can still solve (3.2). On the other hand, in the hard case (case 2), it is possible that $\|(A - \lambda I)^{-1}a\|^2$ is considerably less than s^2 for all the λ that make $A - \lambda I$ positive definite. We show this with the following example taken from [18].

Example 3.1. *Let*

$$A = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad a = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Here $\lambda_1(A) = -1$ *and clearly this is the hard case. For a given* λ*, let* $x(\lambda)$ *be*

$$x(\lambda) := (A - \lambda I)^{-1}a.$$

A short computation gives

$$\|x(\lambda)\|^2 = \frac{1}{(1-\lambda)^2}.$$

For $A - \lambda I \succ 0$, $\lambda < -1$, *and the right hand term takes values between* $(0, 1/4)$. *Therefore,*

for $s < 1/2$, $\|x(\lambda)\|^2 = s^2$ has a solution $\lambda < -1$ and we find ourselves in the hard case (case 1). If $s \geq 1/2$, then λ^* has to be -1 and letting

$$x^* = \begin{bmatrix} \sqrt{s^2 - 1/4} \\ 1/2 \end{bmatrix}$$

shows, after a quick check, that λ^* and x^* satisfy the optimality conditions. This is then the hard case (case 2).

This example shows that if the hard case occurs, solutions $x(\lambda)$ for different values of $\lambda < \lambda_1(A)$ might have a norm much smaller than the radius of the trust region. Being able to handle such problems is important since in the trust region method we want to take steps as large as possible between each iterate. Restricting ourselves to the easy case implies that a large step might be impossible if we find ourselves in the hard case. This was a flaw in the algorithms of Marquardt [14] and Goldfeld and al. [5].

Numerically, when a has small components in the null space of $A - \lambda_1(A)I$, we have to consider the hard case. Not being able to handle the hard case leads to numerical difficulties due to the ill-conditioning of the matrix $A - \lambda I$ for λ close to $\lambda_1(A)$. Fortunately, the handling of the hard case has been solved by Gay [4] and perfected by Moré and Sorensen [18]. We study the Moré and Sorensen algorithm in the next chapter.

Chapter 4

Two Different Methods to Solve TRS

In this chapter, we explore two different methods that solve TRS. The first method is the one developed by Moré and Sorensen [18] in a paper they published in 1983. The paper is commonly cited in the field, since it is the first algorithm to have an efficient numerical method for handling the hard case and, as we mentioned previously, this seems like an important feature for a trust region algorithm.

The second method is due to Gould, Lucidi, Roma and Toint [7]. This method is quite new, as it was published in 1999, and presents a way to exploit sparsity of the matrix A. The motivation comes from the increasing speed of computers and the desire to handle larger problems. Since A is the Hessian of $f(\cdot)$, for a problem with a large number of variables, the Hessian is likely to have many zero entries.

A third method will be presented in this thesis, but in Chapter 6, since we first need to present the duality theory for TRS in Chapter 5. The duality theory will also be used to explain the first two methods in a different way than what will be done in this chapter.

4.1 The Moré and Sorensen Algorithm

This algorithm features efficient handling of the easy and the hard case, but it does not exploit sparsity. In the easy case and the hard case (case 1), Newton's method is used implicitly to solve (for λ) the equation $\|(A - \lambda I)^{-1}a\|^2 = s^2$, although this is not the equation Newton's method is applied to. The hard case (case 2) is handled by moving from a feasible solution to the boundary. The technique is meant to handle the hard case (case 2), but also proves to be of use in the other cases. It is executed every time a solution in the interior of the feasible set is encountered, and therefore few iterations are needed when the hard case (case 2) occurs. Also, if a unique unconstrained minimizer exists and is in the trust region, then in at most two iterations the algorithm will terminate and find this optimal solution.

4.1.1 Handling the Easy Case and the Hard Case (Case 1)

Assuming the solution of TRS lies on the boundary, the easy case is handled by finding a λ which satisfies the equation

$$\|(A - \lambda I)^{-1}a\|^2 = s^2, \quad A - \lambda I \succeq 0.$$

This is possible in the easy case since we know that $A - \lambda^* I$ is invertible. First, let $x(\lambda) := (A - \lambda I)^{-1}a$, and let $Q\Lambda Q^T = A$, where Q is an orthonormal matrix having eigenvectors of A as its columns and Λ is a diagonal matrix having the eigenvalues of A on its diagonal in nondecreasing order, that is $\Lambda_{11} = \lambda_1(A) \leq \ldots \leq \Lambda_{nn} = \lambda_n(A)$. This can be done since A is a symmetric matrix. Then we have

$$\|x(\lambda)\|^2 = \|(A - \lambda I)^{-1}a\|^2 = \|Q(\Lambda - \lambda I)^{-1}Q^T a\|^2 = \|(\Lambda - \lambda I)^{-1}Q^T a\|^2$$

$$\Rightarrow \|x(\lambda)\|^2 = \sum_{j=1}^{n} \frac{\gamma_j^2}{(\lambda_j(A) - \lambda)^2}, \tag{4.1}$$

where γ_j is the jth component of $Q^T a$. This expression will help in the analysis that follows.

Instead of applying Newton's method on the function $\|x(\lambda)\|^2 - s^2$ to find its zero, Moré and Sorensen consider the function

$$\phi(\lambda) := \frac{1}{s} - \frac{1}{\|x(\lambda)\|},$$

which shares the same zero (see Reinsch [22],[23] and Hebden [24]). It can be shown, using the rational structure of $\|x(\lambda)\|^2$, that this function is less nonlinear. In fact, if A is a multiple of the identity, then it is purely linear. Therefore, Newton's method applied to this function will be much more efficient. Now, by definition of Newton's method, and given λ_k, then λ_{k+1} is obtained in the following way:

$$\lambda_{k+1} = \lambda_k - \frac{\phi(\lambda_k)}{\phi'(\lambda_k)}.$$

In practice, the Moré and Sorensen algorithm uses the algorithm below to compute λ_{k+1}. In this algorithm λ_k is assumed to be nonpositive, not to be equal to $\lambda_1(A)$ and to strictly satisfy the strengthened second order optimality conditions (so that the Cholesky factorization can be used).

Algorithm 4.1. *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 is then $x(\lambda_k)$).*
3. *Solve, for y, $R^T y = x$.*

4. *Let* $\lambda_{k+1} = \lambda_k - \left[\frac{\|x\|}{\|y\|}\right]^2 \left[\frac{(\|x\|-s)}{s}\right]$.

We now show that these linear algebra computations are indeed executing Newton's method. First note that

$$Q(\Lambda - \lambda_k I)Q^T = R^T R \Rightarrow (R^T)^{-1}Q = RQ(\Lambda - \lambda_k I)^{-1} \tag{4.2}$$

and also

$$Q^T R^T RQ = \Lambda - \lambda_k I \Rightarrow (RQ(\Lambda - \lambda_k I)^{-1/2})^T (RQ(\Lambda - \lambda_k I)^{-1/2}) = I$$

$$\Rightarrow RQ(\Lambda - \lambda_k I)^{-1/2} \quad \text{is an orthonormal matrix.} \tag{4.3}$$

We have

$$y = (R^T)^{-1}x = (R^T)^{-1}(R^T R)^{-1}a = (R^T)^{-1}(A - \lambda_k I)^{-1}a = (R^T)^{-1}Q(\Lambda - \lambda_k I)^{-1}Q^T a$$

$$= (R^T)^{-1}Q(\Lambda - \lambda_k I)^{-1}\gamma = RQ(\Lambda - \lambda_k I)^{-2}\gamma,$$

where the last equality follows from (4.2). This gives

$$\|y\|^2 = \|RQ(\Lambda - \lambda_k I)^{-1/2}(\Lambda - \lambda_k I)^{-3/2}\gamma\|^2 = \|(\Lambda - \lambda_k I)^{-3/2}\gamma\|^2 = \sum_{j=1}^{n} \frac{\gamma_j^2}{(\lambda_j(A) - \lambda_k)^3},$$

where we have used (4.3) in the second equality. Now we have

$$\frac{d}{d\lambda}\|x(\lambda)\|^2 \Big|_{\lambda=\lambda_k} = 2\sum_{j=1}^{n} \frac{\gamma_j^2}{(\lambda_j(A) - \lambda_k)^3} = 2\|y\|^2.$$

This gives

$$\frac{d}{d\lambda}\|x(\lambda)\|\Big|_{\lambda=\lambda_k} = \frac{d}{d\lambda}(\|x(\lambda)\|^2)^{1/2}\Big|_{\lambda=\lambda_k} = \frac{\frac{d}{d\lambda}\|x(\lambda)\|^2\Big|_{\lambda=\lambda_k}}{2\|x(\lambda_k)\|} = \frac{\|y\|^2}{\|x(\lambda_k)\|} = \frac{\|y\|^2}{\|x\|}.$$

Therefore

$$\phi'(\lambda_k) = \frac{d}{d\lambda}\phi(\lambda)\Big|_{\lambda=\lambda_k} = \frac{d}{d\lambda}\left(\frac{1}{s} - \frac{1}{\|x(\lambda)\|}\right)\Big|_{\lambda=\lambda_k} = \frac{\frac{d}{d\lambda}\|x(\lambda)\|\Big|_{\lambda=\lambda_k}}{\|x(\lambda_k)\|^2} = \frac{\|y\|^2}{\|x\|^3}$$

and λ_{k+1} can then be expressed as

$$\lambda_k - \frac{\phi(\lambda_k)}{\phi'(\lambda_k)} = \lambda_k - \left(\frac{1}{s} - \frac{1}{\|x\|}\right)\frac{\|x\|^3}{\|y\|^2} = \lambda_k - \left(\frac{\|x\| - s}{s}\right)\frac{\|x\|^2}{\|y\|^2},$$

as given in the algorithm.

If one can find a λ_0 such that $\lambda_0 < \lambda_1(A)$ and $\phi(\lambda_0) > 0$, then Algorithm 4.1 converges quadratically, since $\phi(\lambda)$ is a convex function strictly increasing on $(-\infty, \lambda_1(A))$. Hence it has a unique zero of multiplicity one and Newton's method ensures quadratic convergence when initiated from a λ, inside the interval where the function is increasing and convex, that satisfies $\phi(\lambda) > 0$. In practice, it is always possible to find such a λ, because we always find ourselves in the easy case (equation (4.1) shows that in the easy case $\|x(\lambda)\|$ is a function that takes all values from 0 to ∞ when λ varies from $-\infty$ to $\lambda_1(A)$. Hence, for $\lambda < \lambda_1(A)$ and close to $\lambda_1(A)$, $\phi(\lambda)$ is positive.), since, generically, a is not exactly perpendicular to the null space of S_1. Yet, we have to be careful when a is almost perpendicular to the null space of S_1. This is called the *almost hard case*. In the almost hard case, $\phi(\cdot)$ has a sharp simple cusp at $\lambda_1(A)$ and this is due to the fact that in theory $\|x(\lambda)\|$ is bounded when $\lambda < \lambda_1(A)$. 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 very

small interval. Furthermore, Algorithm 4.1 may have computational difficulties, since eventually the matrices $A - \lambda_k I$ will be ill-conditioned. Figure (A.3) illustrates this case which is referred to as the almost hard case (case 2). On the other hand, in the easy case, or even in the almost hard case (case 1), the function is smooth near λ^* and also, because λ^* is not close to $\lambda_1(A)$, there is no ill-conditioning of the matrices $A - \lambda_k I$ and no difficulties are encountered when Algorithm 4.1 is applied. Figures (A.1) and (A.2) illustrate these two cases.

The Moré and Sorensen algorithm applies Algorithm 4.1 and uses a backtracking scheme, on the iterate λ_k that was obtained, to guarantee $\lambda_k \leq 0$ and $\lambda_k < \lambda_1(A)$. Quadratic convergence occurs once a λ that satisfies $\lambda < \lambda_1(A)$ and $\phi(\lambda) > 0$ is found. The difficulty of finding such λ in the almost hard case (case 2) and the ill-conditioning problems would slow down the algorithm. Fortunately, the Moré and Sorensen algorithm has a very efficient way to handle this case. This is what we explain in the next section.

4.1.2 Handling the Hard Case (Case 2)

In the almost hard case (case 1), i.e. when λ^* is far enough from $\lambda_1(A)$, then the above algorithm still gives good results and numerical difficulties don't occur. On the other hand, the almost hard case (case 2) requires more care, since λ^* is close to $\lambda_1(A)$. In theory, in the hard case (case 2), a solution to TRS can be obtained by finding a solution $x(\lambda_1(A))$ to the system

$$(A - \lambda_1(A)I)x = a$$

with a norm less than or equal to s and an eigenvector $z \in S_1$. Then, for some $\tau \in \mathbb{R}$, such that $\|x(\lambda_1(A)) + \tau z\| = s$,

$$x^* = x(\lambda_1(A)) + \tau z$$

satisfies the optimality conditions of Theorem 3.1. The following lemma is the key to implement this idea numerically and it elaborates on Lemma 3.4 in Moré and Sorensen [18].

Lemma 4.1 (Primal step to the boundary). *Let $0 < \sigma < 1$ be given and suppose that*

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

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

$$\|x + z\|^2 = s^2, \quad \|Rz\|^2 \leq \sigma(\|Rx\|^2 - \lambda s^2).$$

Then

$$-q(x + z) \geq (1 - \sigma)(\|Rx\|^2 - \lambda s^2) \geq -(1 - \sigma)q(x^*), \tag{4.4}$$

where x^* is optimal for TRS. Therefore

$$|q(x + z) - q(x^*)| \leq \sigma |q(x^*)|. \tag{4.5}$$

Proof:

For any $z \in \mathbb{R}^n$ we have

$$\begin{aligned} q(x+z) &= (x+z)^T(A-\lambda I)(x+z) - 2a^T(x+z) + \lambda\|x+z\|^2 \\ &= (x+z)^T R^T R(x+z) - 2x^T R^T R(x+z) + \lambda\|x+z\|^2 \\ &= x^T R^T Rx + 2x^T R^T Rz + z^T R^T Rz - 2x^T R^T Rx - 2x^T R^T Rz + \lambda\|x+z\|^2 \\ &= -(\|Rx\|^2 - \lambda\|x+z\|^2) + \|Rz\|^2. \end{aligned} \tag{4.6}$$

If $\|x+z\|^2 = s^2$ and $\|Rz\|^2 \leq \sigma(\|Rx\|^2 - \lambda s^2)$, then

$$-q(x+z) \geq (1-\sigma)(\|Rx\|^2 - \lambda s^2). \tag{4.7}$$

If $x + z^* = x^*$, where x^* is optimal for TRS, then

$$q(x+z^*) \geq -(\|Rx\|^2 - \lambda s^2) + \|Rz\|^2 \geq -(\|Rx\|^2 - \lambda s^2)$$

and

$$-q(x+z^*) \leq \|Rx\|^2 - \lambda s^2. \tag{4.8}$$

Inequalities (4.7) and (4.8) yield (4.4), which implies

$$\begin{aligned} - \quad & q(x+z) \geq (1-\sigma)(-q(x+z^*)) \\ \Rightarrow \quad & q(x+z) - q(x+z^*) \leq -\sigma q(x+z^*) \\ \Rightarrow \quad & \mid q(x+z) - q(x+z^*) \mid \leq \sigma \mid q(x+z^*) \mid . \end{aligned} \tag{4.9}$$

The last inequality is (4.5) and follows since $q(x+z) - q(x+z^*) \geq 0$. Therefore the left hand side of (4.9) is positive and $-q(x+z^*) = |q(x+z^*)|$. ■

A consequence of this lemma is that if we can choose a small σ for which there exists a z that satisfies $\|Rz\|^2 \leq \sigma(\|Rx\|^2 - \lambda s^2)$, $x+z$ is nearly optimal, i.e., according to (4.5), the relative distance between $q(x+z)$ and $q(x^*)$ is less than σ. 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. Moreover, given a feasible solution inside the trust region, taking such a step might still improve the objective if $\|Rz\|$ can be made small. Therefore, the lemma's application goes beyond the almost hard case (case 2). Because the goal of this step is to improve the objective of TRS, to which we will refer to as the *primal objective*, in opposition to the *dual objective* of the next chapter, we call this technique a *primal step to the boundary.*

We end this section by discussing two technical details of this step. First, given R, a normalized vector z is computed such that $\|Rz\|$ is as small as possible using a LINPACK (now LAPACK could be used) technique. Second, a scalar τ such that $\|x+\tau z\|^2 = s^2$ is obtained by the following formula:

$$\tau = \frac{s^2 - \|x\|^2}{x^T z + sgn(x^T z)^2((x^T z)^2 + (s^2 - \|x\|^2))^{1/2}}. \tag{4.10}$$

This expression comes from the fact that for such a z and a feasible solution x there are two values of τ such that $\|x+\tau z\|^2 - s^2 = 0$ (the left hand term is a quadratic polynomial in τ). Using equation (4.6), we have

$$\begin{aligned} q(x+\tau z) &= -(\|Rx\|^2 - \lambda\|x+\tau z\|^2) + \|R(\tau z)\|^2 \\ &= -(\|Rx\|^2 - \lambda s^2) + \tau^2\|Rz\|^2. \end{aligned}$$

This implies that we must pick the solution τ with smallest magnitude, i.e. among the two possible steps to the boundary, we choose the closest one to x. This is how the above expression for τ is obtained.

Lemma 4.1 is very handy in practice. Every time a solution is inside the trust region, we can try to obtain an improvement for the objective function using this lemma. Moreover, if the almost hard case (case 2) is encountered, the algorithm handles it immediately and very few iterations are needed in practice (2-3 iterations). This idea has been generalized to the sparse case in the Rendl and Wolkowicz algorithm presented in Chapter 6.

4.1.3 The Main Algorithm

Basically, the algorithm tries to solve TRS using Algorithm 4.1 and a safeguarding scheme is used to ensure the λ_k are such that $A - \lambda_k I \succ 0$ and $\lambda_k \leq 0$. This scheme also keeps upper and lower bounds on λ^* and makes sure the gap decreases after each iteration. Furthermore, the safeguarding scheme is such that if the solution to TRS lies inside the trust region (i.e. it is an unconstrained minimum), then after at most two iterations, $\lambda = 0$ is tried and an optimal solution is found. If the almost hard case occurs, as stated above, $x(\lambda_k)$ lies in the interior of the trust region and a primal step to the boundary is taken.

The Moré and Sorensen algorithm, at the time it was published, was a breakthrough, as it could handle any of the cases that occurred for TRS very efficiently. Especially, in the almost hard case (case 2), it takes only few iterations, as opposed to previous algorithms that were slowed down in this case. Since then, other algorithms have worked on exploiting the possible sparsity of A which occurs in large problems. The Moré and Sorensen algorithm fails to exploit sparsity because the Cholesky factorization is used

(though for some problems, a sparse Cholesky algorithm may work). For example, if A has non-zero components only on its diagonal and its first row and column, the matrix R obtained with the Cholesky factorization will be full (assuming symmetric permutations are not used). We present in the next section an algorithm that exploits sparsity.

4.2 The Lanczos Method

The method presented here is issued from a paper by Gould, Lucidi, Roma and Toint [7] and is quite recent since it was published in 1999. As mentioned before, current attempts to solve TRS now focus on how sparsity of the matrix A can by exploited. The algorithm developed by Gould et al. only requires matrix-vector multiplications and therefore exploits the sparse property of A. The main technique used in the algorithm involves a Lanzcos tridiagonalization of the matrix A. Hence, this TRS method is referred to as the *Lanczos method*.

The approach used here is to solve the relaxed problem

$$\begin{aligned} \min \quad & q(x) \\ \text{s.t.} \quad & \|x\|^2 \leq s^2 \\ & x \in S, \end{aligned} \tag{4.11}$$

where S is a specially chosen subspace of $\mathbb{R}^n$. The way S is chosen is inspired by the Steihaug-Toint algorithm [28], where the conjugate gradient method is used to find an approximation to the solution of TRS. Unless a global minimizer exists for $q(\cdot)$ and lies in the interior of the trust region (so the conjugate gradient method converges to this minimizer), this algorithm follows the piecewise linear path obtained from the conjugate gradient method. Once the path hits the boundary, the location where the path and the boundary meet is set to be the approximation.

When the boundary has not been attained after k iterations of the conjugate gradient method, x_{k+1} is the solution to (4.11) with the subspace S defined by

$$S := \text{span}\{a, Aa, A^2a, A^3a, \ldots, A^ka\} \tag{4.12}$$

(see Bertsekas [1] p.133). The Lanczos method uses the same kind of subspaces.

Lanczos tridiagonalization (see Golub [6]) can be used to build an orthonormal basis $\{q_0, q_1, \ldots, q_k\}$ for the subspace S. Moreover, if Q_k is the matrix $Q_k := (q_0, q_1, \ldots, q_k)$, then the following equations hold

$$\begin{aligned} AQ_k - Q_kT_k &= \gamma_{k+1}q_{k+1}e_{k+1}^T, & (4.13)\\ Q_k^TQ_k &= I_{k+1}, & (4.14)\\ Q_k^TAQ_k &= T_k, & (4.15)\\ -Q_k^Ta &= \gamma_0 e_1, & (4.16)\\ -a &= \gamma_0 q_0, & (4.17) \end{aligned}$$

where I_{k+1} is the identity matrix of dimension $k+1$, e_{k+1} is its $k+1-$th column, T_k is the tridiagonal matrix

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

$\delta_k = q_k^TAq_k$, $\gamma_0 = \|a\|$, $q_{-1} = 0$, $q_0 = a/\|a\|$, and $\gamma_k = \|(A - q_{k-1}^TAq_{k-1}I)q_{k-1} - \gamma_{k-1}q_{k-2}\|$ and $q_k = ((A - q_{k-1}^TAq_{k-1}I)q_{k-1} - \gamma_{k-1}q_{k-2})/\gamma_k$ for $k > 0$.

The T_k have the property that their extremal eigenvalues become better approximations, as k grows, of the extremal eigenvalues of A. Now, (4.11) is equivalent to

$$\begin{array}{lll} \min & q(x) & = \quad \min \quad h^T Q_k^T A Q_k h - 2(Q_k^T a)^T h \\ \text{s.t.} & \|x\|^2 \leq s^2 & \text{s.t.} \quad \|h^T Q_k^T Q_k h\|^2 \leq s^2. \\ & x = Q_k h & \\ & h \in \mathbb{R}^{k+1} & \end{array}$$

Using (4.14), (4.15) and (4.16) yields the equivalent problem

$$\begin{array}{ll} \min & h^T T_k h + 2\gamma_0 e_1^T h \\ \text{s.t.} & \|h\|^2 \leq s^2. \end{array} \tag{4.18}$$

The Lanczos method works initially as the Steihaug-Toint algorithm, i.e. the conjugate gradient method is applied until it converges or until it hits the boundary. If the boundary is attained, it starts solving problems of the type (4.11) using the subspaces of the form (4.12). When this step is reached, this means that there is no global minimizer in the interior of the trust region and the solution must be on the boundary. Therefore, instead of solving problem (4.18), the following problem is solved:

$$\begin{array}{ll} \min & h^T T_k h + 2\gamma_0 e_1^T h \\ \text{s.t.} & \|h\|^2 = s^2. \end{array} \tag{4.19}$$

The advantage of solving problem (4.19) is that the Moré and Sorensen algorithm may be used to find the optimum, even for large problems, since the Cholesky factorization can take advantage of the tridiagonal form of T_k to preserve sparsity. They do not use the full power of the Moré and Sorensen algorithm, since they handle the almost hard case (case 2) like the easy case and since they are able to find a $\bar{\lambda}$ to start Algorithm 4.1

such that

$$\phi(\bar{\lambda}) = \frac{1}{s} - \frac{1}{\|h(\bar{\lambda})\|} > 0, \quad T_{k+1} - \bar{\lambda} I \succ 0 \quad \text{and} \quad \lambda \leq 0.$$

Therefore, the generated sequence of λ immediately converges quadratically towards λ^* and no safeguarding is necessary. We will refer to this modified form of the Moré and Sorensen algorithm as the *simplified Moré and Sorensen algorithm*. Because this algorithm is also used in the almost hard case (case 2), ill-conditioning in this case will slow down Algorithm 4.1, as mentioned in the previous section.

We return to problem (4.19). By the proof of Theorem 3.1, the necessary and sufficient optimality conditions are that there exist an optimal solution h_k to (4.19) and a corresponding Lagrange multiplier λ_k such that

$$\begin{aligned} \|h_k\|^2 &= s^2, \\ (T_k - \lambda_k I)h_k &= \gamma_0 e_1, \\ T_k - \lambda_k I &\succeq 0. \end{aligned} \tag{4.20}$$

Now, $x_k := Q_k h_k$ is an optimal solution to the relaxed problem (4.11) and is in particular a feasible solution for (2.2), since $\|x_k\|^2 = s^2$. We then get, using (4.14), (4.15) and (4.16),

$$Q_k^T(A - \lambda_k I)x_k = Q_k^T(A - \lambda_k I)Q_k h_k = (T_k - \lambda_k I_{k+1})h_k = \gamma_0 e_1 = Q_k^T a$$

and

$$\lambda_k \leq 0 \quad \text{and} \quad \lambda_k(\|x_k\|^2 - s^2) = 0.$$

Hence, the stationarity of x_k for problem (2.2) is satisfied up to a matrix multiplication on the left hand side, complementary slackness is satisfied and the sign of λ_k is correct. Yet, the positive semidefiniteness of $A - \lambda_k I$ is not ensured, although for large k we can expect it since $\lambda_k \leq \lambda_1(T_k)$ by optimality and since Lanczos tridiagonalization implies that $\lambda_1(T_k) \to \lambda_1(A)$ (and reaches it eventually for some $k \leq n-1$). To decide how good the approximation x_k is, we have the following theorem, which is Theorem 5.1 in [7].

Theorem 4.1.

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

Proof:

$$\begin{aligned} Ax_k &= AQ_k h_k \\ &= Q_k T_k h_k + \gamma_{k+1} q_{k+1} e_{k+1}^T h_k \quad \text{(from (4.13))} \\ &= Q_k(\lambda_k h_k + \gamma_0 e_1) + \gamma_{k+1} e_{k+1}^T h_k q_{k+1} \quad \text{(from (4.20))} \\ &= \lambda_k Q_k h_k + \gamma_0 Q_k e_1 + \gamma_{k+1} e_{k+1}^T h_k q_{k+1} \\ &= \lambda_k x_k + \gamma_0 q_0 + \gamma_{k+1} e_{k+1}^T h_k q_{k+1} \\ &= \lambda_k x_k + a + \gamma_{k+1} e_{k+1}^T h_k q_{k+1} \quad \text{(from (4.17))}. \end{aligned}$$

The norm equality follows since $q_{k+1}^T q_{k+1} = 1$. ■

This theorem is used in the stopping criteria of the algorithm: 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 TRS.

We now state their algorithm.

Algorithm 4.2 (Lanczos 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,

$x_{k+1} = x_k + \alpha_k p_k$,

else

solve the tridiagonal trust region subproblem (4.19), using the simplified More and Sorensen algorithm,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 $x_k = Q_k h_k$.

A first note on the algorithm is that if INTERIOR is always true throughout the algorithm, then the algorithm is the conjugate gradient method and convergence is satisfied

when the norm of the gradient of the global minimizer sought is close to 0. A second note is that the conjugate gradient method is used to generate the vectors $q_0, q_1, \ldots, q_k$ of the Lanczos tridiagonalization algorithm, since we can obtain them from the relationship

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

and the tridiagonal matrix T_k can be obtained from

$$T_k = \begin{bmatrix} \frac{1}{\alpha_0} & \frac{\sqrt{\beta_0}}{|\alpha_0|} & & & & \\ \frac{\sqrt{\beta_0}}{|\alpha_0|} & \frac{1}{\alpha_1} + \frac{\beta_0}{\alpha_0} & \frac{\sqrt{\beta_1}}{|\alpha_1|} & & & \\ & \frac{\sqrt{\beta_1}}{|\alpha_1|} & \frac{1}{\alpha_2} + \frac{\beta_1}{\alpha_1} & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \frac{1}{\alpha_{k-1}} + \frac{\beta_{k-2}}{\alpha_{k-2}} & \frac{\sqrt{\beta_{k-1}}}{|\alpha_{k-1}|} \\ & & & & \frac{\sqrt{\beta_{k-1}}}{|\alpha_{k-1}|} & \frac{1}{\alpha_k} + \frac{\beta_{k-1}}{\alpha_{k-1}} \end{bmatrix}.$$

This also gives the relationship between T_{k-1} and T_k. A third note is that if the almost hard case (case 2) occurs for a problem of type (4.19), the method is slowed down. The paper of Gould et al. [7] describes a way of handling this case, but it remains ineffective in practice. A last note is that $\min(0.1, \|a\|^{0.1})\|a\|$ was used in the stopping criteria of the original Lanczos method. We use $\max(10^{-8}, 10^{-5}\|a\|)$. This expression was also suggested in [7]. In the Rendl and Wolkowicz algorithm, we use this tolerance for the duality gap. As we will show in the duality chapter, the stopping criteria of the Lanczos method partially measures the same duality gap. We use the same tolerance in both algorithms , i.e. $\max(10^{-8}, 10^{-5}\|a\|)$, so that we can compare both algorithms in the numerical section.

Concluding remarks on this algorithm would first be that convergence of the Lanczos method is based on the quadratic convergence of Newton's method applied to the function

$\phi(\cdot)$. Yet, quadratic convergence of a trust region algorithm is not what is most important. Rather, quick finite termination is what is needed and is what the Moré and Sorensen algorithm achieves. To obtain this, handling of the almost hard case (case 2) is essential. A second remark is that compared to the Moré and Sorensen algorithm, the Lanczos method is able to use the sparsity of A, but loses the handling of the almost hard case (case 2). It would be nice if both the almost hard case (case 2) and sparsity could be handled in an algorithm. We present such an algorithm in Chapter 6. To come to this, we first need to derive the duality theory behind TRS. This is the subject of the next chapter.

Chapter 5

Duality

In this chapter we study the duality theory behind TRS. One of the main theorems of this section is that strong Lagrangian duality holds for TRS. It enables us to build different dual problems whose properties give us a new look at the previous algorithms. In particular, we will show that the three algorithms we consider in this thesis can be set within a semidefinite framework. Rendl and Wolkowicz [31] showed that this was the case for their algorithm and the one of Moré and Sorensen. In this chapter we show that it is indeed also the case for the Lanczos method. We show that their stopping criteria is in fact measuring a duality gap.

5.1 Deriving the Duals

For the purpose of this section, we will deal with TRS where equality holds for the constraint ($q(\cdot)$ and s are defined in Chapter 2):

$$(\text{TRS}_{=}) \qquad \begin{array}{lll} q^* = & \min & q(x) \\ & \text{s.t.} & \|x\|^2 = s^2. \end{array} \tag{5.1}$$

We will refer to this problem as $\text{TRS}_{=}$. The necessary and sufficient optimality conditions for this problem are the same as those given by Theorem 3.1 for TRS, except that $\lambda^* \in \mathbb{R}$ (the proof can be derived from the proof of Theorem 3.1). Precisely, x^* is optimal for $\text{TRS}_{=}$ if and only if there exists a unique Lagrange multiplier $\lambda^* \in \mathbb{R}$ such that

$$\begin{aligned} (A - \lambda^* I)x^* &= a, \\ (A - \lambda^* I) &\succeq 0, \\ \|x^*\|^2 &= s^2. \end{aligned} \tag{5.2}$$

We first show that strong duality holds for $\text{TRS}_{=}$. The result is due to Stern and Wolkowicz [32] who showed the deeper result that strong duality holds for the minimization of a quadratic objective subject to the constraints $\beta \leq x^T C x \leq \alpha$, where C is a symmetric matrix (no definiteness is assumed) and where β and α are constants such that $\beta < \alpha$.

Theorem 5.1 (Strong Duality). *Strong Lagrangian duality holds for (5.1), i.e.*

$$q^* = \min_x \sup_\lambda \; L(x, \lambda) = \max_\lambda \inf_x \; L(x, \lambda),$$

where $L(x, \lambda) = q(x) - \lambda(\|x\|^2 - s^2)$ is the Lagrangian function, and the dual is attained.

Proof:

The left equality follows easily and we only need to prove the right equality. Define the

dual functional

$$\phi(\lambda) := \inf_x \; L(x, \lambda).$$

Then, by weak duality,

$$q^* \geq \max_\lambda \; \phi(\lambda). \tag{5.3}$$

One can show that $\lim_{\lambda \to -\infty} \phi(\lambda) = -\infty$ (to show this refer to equation (5.11) in the proof of our next theorem) and that $\phi(\lambda) = -\infty$ for $\lambda > \lambda_1(A)$, hence $\phi(\lambda)$ is a coercive function and the right expression is well defined. Now, if x^* is optimal for $\text{TRS}_=$ and λ^* is its Lagrange multiplier, we have

$$L(x^*, \lambda^*) = \inf_x \; L(x, \lambda^*).$$

This is true by the optimality conditions (5.2), i.e. $L(x, \lambda^*) = x^T(A - \lambda^* I)x - 2a^T x + \lambda^* s^2$ is a convex function (since $A - \lambda^* I \succeq 0$) which has a stationary point at x^* (since $(A - \lambda^* I)x^* - a = 0$). We then have the following:

$$\begin{aligned} q^* &= q(x^*) - \lambda^*(\|x^*\|^2 - s^2) \qquad \text{(by feasibility of } x^*) \\ &= L(x^*, \lambda^*) \; = \; \inf_x \; L(x, \lambda^*) \\ &= \phi(\lambda^*) \leq \max_\lambda \; \phi(\lambda). \end{aligned}$$

This yields

$$q^* \leq \max_\lambda \; \phi(\lambda). \tag{5.4}$$

Equations (5.3) and (5.4) imply that there is a zero duality gap. The attainment of the dual follows from the optimality conditions (5.2). ■

It is not too difficult to modify this proof to show that strong duality also holds for TRS. Theorem 5.1 modified to show strong duality for TRS would yield

$$\min_x \sup_{\lambda \leq 0} L(x,\lambda) = \max_{\lambda \leq 0} \inf_x L(x,\lambda) = \max_{\lambda \leq 0} \phi(\lambda). \tag{5.5}$$

We now use the last theorem to derive some dual problems to $\text{TRS}_=$. References to the work done in this section are Stern and Wolkowicz [32] and Rendl and Wolkowicz [31]. First, we consider the Lagrangian dual

$$\max_\lambda \inf_x L(x,\lambda), \tag{5.6}$$

for which we proved the optimum is q^*.

Since $L(x,\lambda) = x^T(A-\lambda I)x - 2a^Tx + \lambda s^2$, then the inner infimum in (5.6) goes to $-\infty$ if $A - \lambda I$ is not positive semidefinite. This yields the equivalent problem

$$q^* = \max_{A-\lambda I \succeq 0} \inf_x L(x,\lambda).$$

Define

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

where $\dagger$ stands for the Moore-Penrose generalized inverse. Now if $\lambda < \lambda_1(A)$, then

$A - \lambda I \succ 0$ and

$$\inf_x \; L(x, \lambda) = L((A - \lambda I)^{-1} a, \lambda) = h(\lambda). \tag{5.7}$$

When $\lambda = \lambda_1(A)$, if the system $(A - \lambda_1(A)I)x = a$ has a solution, then one of them is $x = (A - \lambda_1(A)I)^{\dagger} a$ and this vector is therefore a minimizer of $L(x, \lambda_1(A))$. Hence, when the system $(A - \lambda_1(A)I)x = a$ is consistent, then

$$\inf_x \; L(x, \lambda_1(A)) = L((A - \lambda_1(A)I)^{\dagger} a, \quad \lambda_1(A)) = h(\lambda_1(A)). \tag{5.8}$$

In the hard case, since a is perpendicular to $\mathcal{N}(A - \lambda_1(A)I)$, then $a \in \mathcal{R}(A - \lambda_1(A)I)$, i.e. the system $(A - \lambda_1(A)I)x = a$ has a solution. Using (5.7) and (5.8) we deduce the following result in the hard case:

$$q^* = \max_{A - \lambda I \succeq 0} \; h(\lambda). \tag{5.9}$$

In the easy case and the hard case (case 1), since $\lambda^* < \lambda_1(A)$, then λ^* can be found among the λ such that $A - \lambda I \succ 0$. Thus

$$q^* = L(x^*, \lambda^*) = \max_{\lambda} \inf_x \; L(x, \lambda) = \max_{A - \lambda I \succ 0} \; \min_x \; L(x, \lambda).$$

Again, (5.7) implies that in the easy case and the hard case (case 1)

$$q^* = \max_{A - \lambda I \succ 0} \; h(\lambda). \tag{5.10}$$

Problems (5.9) and (5.10) are quite similar and it would be nice to have a single problem that would include both. This is what the next theorem provides.

Theorem 5.2. *The following problem is the Lagrangian dual for* $TRS_{=}$

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

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 $A = Q\Lambda Q^T$ be defined as in section 4.1.1. Because $a \perp \mathcal{N}(A - \lambda_j(A)I)$ for $j = 1 \ldots l-1$, then 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 $w \in \mathbb{N}$, $\lambda_w \in (-\infty, \lambda_l)$ and $A - \lambda_w I$ is invertible. Then

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

$$= -\sum_{j=l}^{n} \frac{\gamma_j^2}{(\lambda_j - \lambda_w)} + \lambda_w s^2. \tag{5.11}$$

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 $\tilde{\lambda} = \lambda_l$, $h(\lambda_w) \to -\infty$. Therefore, $h(\cdot)$ has a vertical asymptote and is not continuous in λ_l.

When $\tilde{\lambda} \in (-\infty, \lambda_l)$, by equation (5.11) 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 (5.10) yields (D). Equation (5.10) also implies that in the easy case and the hard case (case 1) the sup can be replaced by a max. ■

This theorem, adapted to TRS, yields the following corollary:

Corollary 5.1. *The following problem is the Lagrangian dual for TRS*

$$q^* = \sup_{\lambda \leq 0, A-\lambda I \succ 0} h(\lambda). \tag{5.12}$$

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

We need to note three things from this theorem. First, $h(\cdot)$ is a concave function on $(-\infty, \lambda_l)$. To prove this, let $\{\lambda_w\}$ be a sequence, defined exactly as in the proof of the previous theorem, which converges to $\tilde{\lambda} \in (-\infty, \lambda_l)$. We have

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

(a note on the derivatives of $h(\cdot)$ can be found in Appendix B). Since $\lambda_j - \tilde{\lambda} > 0$, then the right term is nonpositive. Now note that

$$\sum_{j=l}^{n} \frac{\gamma_j^2}{(\lambda_j - \tilde{\lambda})^3} = a^T((A - \tilde{\lambda}I)^\dagger)^3 a.$$

Hence, for $\lambda \in (-\infty, \lambda_l)$,

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

and this proves the concavity of $h(\cdot)$ on $(-\infty, \lambda_l)$. This shows that $\text{TRS}_=$ is equivalent to finding the supremum of a concave function over an open interval (Appendix A provides the graph of $h(\cdot)$ for the different cases).

A second comment on Theorem 5.2 is that despite the fact that $A - \lambda I$ needs to be positive definite (since positive semidefiniteness is not enough as we will show below), (D) is analogous to a nonlinear semidefinite program.

A third comment is that in the easy case, (5.9) does not necessarily hold. We show this with the following example.

Example 5.1. *Let*

$$A = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \quad a = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad s = 1/2.$$

Obviously, a is not perpendicular to $\mathcal{N}(A - \lambda_1(A)I)$ and the easy case holds. A quick computation shows that

$$x^* = \begin{bmatrix} 1/2 \\ 0 \end{bmatrix} \quad \text{and} \quad \lambda^* = -3$$

satisfy the optimality conditions. We have

$$h(\lambda^*) = h(-3) = -\begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 1/2 & 0 \\ 0 & 1/4 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + (-3)\frac{1}{4} = \frac{-5}{4}$$

and $q(x^*) = -5/4$*, in agreement with (5.10). Now*

$$h(\lambda_1(A)) = h(-1) = -\begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} 0 & 0 \\ 0 & 1/2 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} + (-1)\frac{1}{4} = \frac{-1}{4}.$$

Therefore, for this example $h(\lambda_1(A)) > h(\lambda^*) = q^*$.

The next dual we consider is the one we obtain by taking the Lagrangian dual of (D) in the hard case, i.e. problem (5.9). We showed previously that $h(\cdot)$ is a concave function over $(-\infty, \lambda_l)$ and furthermore any $\lambda < \lambda_1(A)$ is a Slater point for (5.9). Therefore, there is no duality gap between (5.9) and its Lagrangian dual (see Bertsekas [1]) and strong duality holds, i.e

$$q^* = \max_{A-\lambda I \succeq 0} h(\lambda) = \max_{\lambda<\lambda_l} \inf_{X\succeq 0} h(\lambda) + \text{trace}\,(X(A-\lambda I))$$

$$= \min_{X\succeq 0} \sup_{\lambda<\lambda_l} h(\lambda) + \text{trace}\,(X(A-\lambda I)).$$

Now

$$h(\lambda) + \text{trace}\,(X(A-\lambda I)) = h(\lambda) - \lambda\text{trace}\,(X) + \text{trace}\,(XA)$$

is still a concave function in λ over the interval $(-\infty, \lambda_l)$ and goes to $-\infty$ as λ approaches λ_l from the left. When $\lambda \to -\infty$, different cases occur depending on the trace of X. If $\text{trace}\,(X) > s^2$, then the inner supremum goes to ∞ and we may wish to ignore these X. When $\text{trace}\,(X) = s^2$, it is not too hard to show that the function tends to $\text{trace}\,(XA)$ as $\lambda \to -\infty$, and since the function is concave, then the inner supremum is $\text{trace}\,(XA)$. When $\text{trace}\,(X) < s^2$, the function goes to $-\infty$ as $\lambda \to -\infty$ and therefore

the supremum is attained at a value of λ were the derivative is zero, i.e. for a λ such that $s^2 - a^T((A-\lambda I)^\dagger)^2 a - \text{trace}\,(X) = 0$. Furthermore, if X is such that $\text{trace}\,(X) = s^2$ and $\{X_n\}$ is a sequence of positive definite matrices such that $\text{trace}\,(X_n) < s^2$ and $X_n \to X$, then for n large enough,

$$\sup_{\lambda} \; h(\lambda) + \text{trace}\,(X_n(A - \lambda I))$$

is as close as we want to $\text{trace}\,(XA)$. Hence we may also ignore the X such that $\text{trace}\,(X) = s^2$ (yet to be rigorous, we need then to replace the outside min by an inf). This yields the following dual, which is also similar to a nonlinear semidefinite program, and to which we refer as (DD):

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

We end this section with a last dual problem which will be the key for the Rendl and Wolkowicz algorithm. (D) shows that $TRS_=$ is equivalent to finding the supremum of a concave function over an open interval. This last dual problem will show that $\text{TRS}_=$ is also equivalent to the maximization of a single variable concave function over $\mathbb{R}$.

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

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

To establish that this problem is equivalent to $\text{TRS}_=$, assume x^* and y_0^* are optimum for the homogenized problem. If $y_0^* = 1$, clearly there is nothing to show. If $y_0^* = -1$, then setting $x^* \leftarrow -x^*$ and $y_0^* = 1$ gives another optimal solution to the homogenized problem and the equivalence between $\text{TRS}_=$ and its homogenized form follows. Now, the homogenized problem is equal to

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

$$\geq \max_t \min_{\|x\|^2+y_0^2=s^2+1} x^T Ax - 2y_0 a^T x + t(y_0^2 - 1) \tag{5.15}$$

$$\geq \sup_{t,\lambda} \inf_{x,y_0} x^T Ax - 2y_0 a^T x + t(y_0^2 - 1) + \lambda(\|x\|^2 + y_0^2 - s^2 - 1).$$

$$= \sup_{r,\lambda} \inf_{x,y_0} x^T Ax - 2y_0 a^T x + r(y_0^2 - 1) + \lambda(\|x\|^2 - s^2)$$

$$= \sup_{\lambda} \left(\sup_{r} \inf_{x,y_0} x^T Ax - 2y_0 a^T x + r(y_0^2 - 1) + \lambda(\|x\|^2 - s^2) \right)$$

where $r = t + \lambda$.

Now because strong duality holds (here we need the full power of the Strong Duality Theorem of Stern and Wolkowicz [32]), this is equal to

$$\sup_{\lambda} \left(\inf_{x,y_0} \sup_{r} x^T Ax - 2y_0 a^T x + r(y_0^2 - 1) + \lambda(\|x\|^2 - s^2) \right)$$

$$= \sup_{\lambda} \inf_{x, y_0^2=1} x^T A x - 2y_0 a^T x + \lambda(\|x\|^2 - s^2).$$

Again, by strong duality this is equivalent to

$$\inf_{x, y_0^2=1} \sup_{\lambda} x^T A x - 2y_0 a^T x + \lambda(\|x\|^2 - s^2)$$

$$\begin{aligned} = \quad & \min \quad x^T A x - 2y_0 a^T x \quad = \quad q^* \\ & \text{s.t.} \quad \|x\|^2 = s^2 \\ & \qquad y_0^2 = 1. \end{aligned}$$

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

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

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

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

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

then we have the following unconstrained dual problem for TRS which we refer to as (UD):

$$\text{(UD)} \qquad \max_t k(t).$$

Since $\lambda_1(D(\cdot))$ is a concave function (see Appendix B), then $k(\cdot)$ is concave and this shows that $\text{TRS}_=$ is equivalent to an unconstrained concave maximization problem. We can also rewrite (UD) in the following way so that it becomes a linear semidefinite program:

$$\max_{D(t)\succeq\lambda I} (s^2+1)\lambda - t. \tag{5.16}$$

In the next two sections we show that (D) and (DD) can be used to explain the Moré and Sorensen algorithm and the Lanczos method. (UD) is used to solve efficiently TRS in the Rendl and Wolkowicz algorithm of the next chapter.

5.2 A Semidefinite Framework for the Moré and Sorensen Algorithm

In this section, we use the dual problems of the previous section to show that the Moré and Sorensen algorithm can be set within a semidefinite framework. We use the structure of the two dual programs (D) and (DD), which are similar to semidefinite programs, to show that the algorithm is in fact trying to solve those duals. Our analysis is restricted to the case where a solution lies on the boundary of the trust region. We show that, in the easy case and the hard case (case 1), the algorithm is trying solve a modified form of the stationarity condition for (D). In the hard case (case 2), (DD) is used to show that the primal step to the boundary is used to reduce the gap between (D) and (DD). The work outlined here is due to Rendl and Wolkowicz [31].

In the easy case and the hard case (case 1), Theorem 5.2 shows that

$$q^* = \max_{A-\lambda I\succ 0} h(\lambda). \tag{5.17}$$

Furthermore $\lambda_l = \lambda_1(A)$ and $h(\lambda)$ is concave on the open interval $(-\infty, \lambda_1(A))$. It goes to $-\infty$ as λ approaches $\lambda_1(A)$ from the left and also as $\lambda \to -\infty$. Therefore, to solve problem (5.17), we only need to find a $\lambda^* \in (-\infty, \lambda_1(A))$ such that

$$h'(\lambda^*) = -a^T((A - \lambda^* I)^\dagger)^2 a + s^2 = 0.$$

Since the λ we consider are less than $\lambda_1(A)$, $A - \lambda I$ is invertible for those λ and therefore we need to find a λ^* such that

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

As mentioned in section 4.1.1, the algorithm solves with Newton's method the modified equation

$$\frac{1}{s} - \frac{1}{\|(A - \lambda I)^{-1} a\|} = 0.$$

Although the function $h(\cdot)$ is not used explicitly, they really are trying to solve (5.17) using backtracking on the λ to insure feasibility.

In the hard case (case 2), as mentioned in section 4.1.2, given a feasible vector $x = x(\lambda) = (A - \lambda I)^\dagger a$, where $\lambda \leq \lambda_1(A)$, the idea to handle this case is to find a proper z to reduce the primal objective and move to the boundary (i.e. $\|x+z\|^2 = s^2$). Our framework suggests how such a z should be chosen and the result follows from the following equations:

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

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

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

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

This implies that z should be chosen to make $z^T(A - \lambda I)z$ small. For a fixed feasible λ for (D), the duality gap between (D) and (DD) is dependent of X and equals

$$\text{trace}\,(X(A - \lambda I)).$$

If we set $X = zz^T$, then

$$\text{trace}\,(X(A - \lambda I)) = \text{trace}\,(zz^T(A - \lambda I)) = \text{trace}\,(z^T(A - \lambda I))z = z^T(A - \lambda I))z.$$

Note that in the Moré and Sorensen 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 (D) and (DD), while maintaining feasibility for (DD).

5.3 A Semidefinite Framework for the Lanczos Method

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

us to wonder on how one might attempt to improve the algorithm or show where it fails using this framework.

Recall that the Lanczos method first looks for an unconstrained minimizer until it hits the trust region boundary of TRS, and solves problems of type (4.11) with larger and larger subspace S until an approximate solution is found. When problems of the type (4.11) are solved, solutions to TRS are known to be on the boundary and only TRS$_=$ needs to be solved. A solution x_k of (4.11) is said to be a good approximation if $\gamma_{k+1}|e_{k+1}^T h_k|$ is small and this is used as the stopping criteria for the algorithm. We now show the relationship between the duality gap and the stopping criteria.

Since $\|x_k\|^2 = s^2$, we have

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

But Theorem 4.1 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 (D), i.e. $(A - \lambda_k I) \succeq 0$, then, the right hand term is the duality gap between $\text{TRS}_=$ and (D). 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 Lanczos 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

$$q(x_k) - h(\lambda_k) = \|(\gamma_{k+1}|e_{k+1}^T h_k|)^2 q_{k+1}\|_M^2 \,. \tag{5.18}$$

Two comments come from writing the duality gap in this form . First, 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 Lanczos method stops, but where $\|q_{k+1}\|_{A-\lambda_k I}^2$ is large enough so that the duality gap is also large.

Second, we may ask how (5.18) might be used to improve the Lanczos method. It could be used to replace the stopping criteria, but the need to compute the inverse of a large matrix makes this idea very costly. Unfortunately, so far no improvement to the method have been found using the information given by our framework.

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 measure of how good the approximation is, is directly linked to the duality gap between TRS and the dual problem (D). Furthermore, at each iteration, since feasibility is not insured for λ_k, the algorithm compares to a primal-dual infeasible algorithm.

We have seen in this chapter how much the duality theory is hidden behind the two algorithms considered so far. The duals also present some attractive structure like concavity and a simpler function to work with. The next chapter presents an algorithm that directly solves the dual (UD).

Chapter 6

The Rendl and Wolkowicz Algorithm

In this chapter, we present an algorithm that both exploits the sparsity of A and handles the hard case (case 2). The algorithm is due to Rendl and Wolkowicz [31] and is mainly based on the dual program (UD). Most of the theory behind the method is based on properties of the eigenvalues and eigenvectors of the parametric matrix $D(t)$. Our first section is dedicated to this subject. This will lead us to understand how $k(\cdot)$ behaves in the easy and the hard case and how the algorithm handles these two cases. Many tricks of the Moré and Sorensen [18] paper are being used in the algorithm, in particular, the primal step to the boundary. Furthermore, we show that a new way to take a step to the boundary may be used. We also outline many other tricks used in the algorithm that take advantage of the structure of $k(\cdot)$ and accelerate convergence. Finally, we end the chapter explaining how the algorithm solves TRS.

6.1 Eigenvalues and Eigenvectors of D(t)

Recall that

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

As we mentioned before, $\lambda_1(D(\cdot))$ is a concave function and therefore $k(\cdot)$ is a concave function too. The function also has special asymptotic structure. But, before examining this, we first need the following (which elaborates on Proposition 8, Lemma 9 and Lemma 15 in Rendl and Wolkowicz [31]):

Theorem 6.1. *Let $A = Q^T \Lambda Q$ be defined as in section 4.1.1. In the easy case, for $t \in \mathbb{R}$, $\lambda_1(D(t)) < \lambda_1(A)$ and has multiplicity 1. In the hard case, for $t < t_0$, $\lambda_1(D(t)) < \lambda_1(A)$ and has multiplicity 1, for $t = t_0$, $\lambda_1(D(t)) = \lambda_1(A)$ and has multiplicity $1 + i$ and for $t > t_0$, $\lambda_1(D(t)) = \lambda_1(A)$ and has multiplicity i, where i is the multiplicity of $\lambda_1(A)$ and t_0 is defined by*

$$t_0 := d(\lambda_1(A)) = \lambda_1(A) + \sum_{j \in \{i | (Q^T a)_i \neq 0\}} \frac{(Q^T a)_j^2}{\lambda_j(A) - \lambda_1(A)}.$$

Proof:

We assume here without loss of generality that A is a diagonal matrix with diagonal elements $\lambda_j(A)$, and that they are in nondecreasing order, i.e. $A_{jj} = \lambda_j(A)$. Note that in this case $Q = I$. There is no loss of generality, because we have

$$q(x) = (Q^T x)^T \Lambda (Q^T x) - 2(Q^T a)^T (Q^T x)$$

and

$$\begin{bmatrix} t & -(Q^Ta)^T \\ -(Q^Ta) & \Lambda \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & Q \end{bmatrix}^T \begin{bmatrix} t & -a^T \\ -a & A \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & Q \end{bmatrix}.$$

Therefore, the eigenvalues of $D(t)$ are the same as the above matrix on the left. So to simplify our analysis, we redefine $a \leftarrow (Q^Ta)$ and $A \leftarrow \Lambda$. We also assume that i is the multiplicity of $\lambda_1(A)$, i.e.

$$\lambda_1(A) = \lambda_2(A) = \ldots = \lambda_i(A) < \lambda_{i+1}(A) \leq \ldots \leq \lambda_n(A).$$

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, expanding with respect to the first column of $D(t)$,

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

Let $J = \{i | a_i \neq 0\}$ and, for $\lambda \notin \{\lambda_j(A) | j \in J\}$, define

$$d(\lambda) := \lambda + \sum_{j \in J} \frac{a_j^2}{\lambda_j(A) - \lambda}. \tag{6.1}$$

Then

$$\det(D(t) - \lambda I) = (t - d(\lambda)) \prod_{j=1}^{n} (\lambda_j(A) - \lambda) \quad \text{for } \lambda \notin \{\lambda_j(A) | j \in J\}. \tag{6.2}$$

Note that the eigenvalues of A are not necessarily eigenvalues for $D(t)$ since $d(\cdot)$ might not be defined for any of these values. Yet, if $\lambda_k \notin \{\lambda_j(A) | j \in J\}$ then $\lambda_k(A)$ is an eigenvalue for $D(t)$, since in this case $d(\cdot)$ is then well defined at $\lambda_k(A)$. In the easy

case, there exits $\lambda_h(A) \in \{\lambda_j(A)|j \in J\}$ with $h \in \{1, \ldots, i\}$. Without loss of generality, assume $\lambda_1(A) \in \{\lambda_j(A)|j \in J\}$. Therefore

$$\lim_{\lambda \to \lambda_1(A), \lambda < \lambda_1(A)} d(\lambda) = \infty \tag{6.3}$$

and we also have

$$\lim_{\lambda \to -\infty} d(\lambda) = -\infty. \tag{6.4}$$

Moreover

$$d'(\lambda) = 1 + \sum_{j \in J} \frac{a_j^2}{(\lambda_j(A) - \lambda)^2} > 0 \tag{6.5}$$

and

$$d''(\lambda) = \sum_{j \in J} \frac{2a_j^2}{(\lambda_j(A) - \lambda)^3} > 0 \quad if \lambda < \lambda_1(A).$$

Therefore, $d(\cdot)$ is strictly monotonically increasing and convex on $(-\infty, \lambda_1(A))$. In the hard case, $\lambda_h(A) \notin \{\lambda_j(A)|j \in J\}$ for $h \in \{1 \ldots i\}$ and $d(\lambda_1(A)) := t_0$ is well defined. If $\lambda_l(A) := \min(\lambda_j(A)|j \in J)$, then a similar analysis shows that $d(\cdot)$ is strictly monotonically increasing and convex on $(-\infty, \lambda_l(A))$.

We conclude from this analysis of $d(\cdot)$, that in the easy case, for a fixed $t \in \mathbb{R}$, the equation $t - d(\lambda) = 0$ always has a solution $\lambda < \lambda_1(A)$ and that this solution is unique. Because the eigenvalues of A and D(t) interlace (see [9]), in particular $\lambda_1(D(t)) \leq \lambda_1(A)$, if $\lambda < \lambda_1(A)$ and $t - d(\lambda) = 0$, then, by (6.2), $\lambda = \lambda_1(D(t))$. Since λ is the unique solution less than $\lambda_1(A)$, then $\lambda_1(D(t))$ has multiplicity one. This shows that in the easy case, for any t, $\lambda_1(D(t)) < \lambda_1(A)$ and has multiplicity 1.

In the hard case, for $t < t_0$, by an argument similar to the easy case, the equation $t - d(\lambda) = 0$ also has a unique solution strictly less than $\lambda_1(A)$ which is, by equation (6.2), $\lambda_1(D(t))$. So for $t < t_0$, $\lambda_1(D(t)) < \lambda_1(A)$ and has multiplicity 1. When $t = t_0$, because in the hard case $\lambda_l(A) > \lambda_1(A)$, then $d'(\lambda) > 0$ if $\lambda \leq \lambda_1(A)$. Therefore,

$$\lambda < \lambda_1(A) \Rightarrow d(\lambda) < d(\lambda_1(A)) \Rightarrow 0 < t_0 - d(\lambda).$$

Since

$$\det(D(t_0) - \lambda I) = (t_0 - d(\lambda)) \prod_{j=1}^{n} (\lambda_j(A) - \lambda), \tag{6.6}$$

then for $\lambda < \lambda_1(A)$, $\det(D(t_0) - \lambda I) > 0$. But $\det(D(t_0 - \lambda_1(A)I) = 0$, and therefore $\lambda_1(D(t)) = \lambda_1(A)$ and has multiplicity $i+1$ by (6.6). When $t > t_0$, $t - d(\lambda) = 0$ does not have a solution $\lambda \leq \lambda_1(A)$. Since $\lambda_1(A)$ is a solution to $\det(D(t) - \lambda I) = 0$, then again by equation (6.2) we get that $\lambda_1(D(t)) = \lambda_1(A)$ and has multiplicity i. ■

From this theorem we derive some basic properties of $k(\cdot)$. First, note from (6.4) that when $t \to -\infty$, the solutions in λ to $t - d(\lambda) = 0$ tends to $-\infty$ and by equation (6.1) it is asymptotic to t. Therefore

$$\lim_{t \to -\infty} \lambda_1(D(t)) = -\infty \quad \text{and} \quad \lambda_1(D(t)) \sim t \quad \text{as } t \to -\infty.$$

In the easy case (6.3) implies

$$\lim_{t \to \infty} \lambda_1(D(t)) = \lambda_1(A).$$

In the hard case, for t large enough, $t > t_0$ and $\lambda_1(D(t)) = \lambda_1(A)$.

These results for $\lambda_1(D(t))$ yield the following for $k(\cdot)$:

$$k(t) \sim s^2 t \quad \text{as } t \to -\infty, \tag{6.7}$$

$$k(t) \sim (s^2+1)\lambda_1(A) - t \quad \text{as } t \to \infty \text{ in the easy case,} \tag{6.8}$$

$$k(t) = (s^2+1)\lambda_1(A) - t \quad \text{for } t \geq t_0 \text{ in the hard case.} \tag{6.9}$$

We now look at the differentiability of the function $k(\cdot)$. For this we need to look at the eigenvectors of $\lambda_1(D(t))$. We have the following theorem derived from Lemma 12 and Lemma 15 in Rendl and Wolkowicz [31].

Theorem 6.2. *Let $y(t)$ be an eigenvector for $\lambda_1(D(t))$ and let $y_0(t)$ be its first component. Then in the easy case, for $t \in \mathbb{R}$, $y_0(t) \neq 0$. In the hard case, for $t < t_0$, $y_0(t) \neq 0$, for $t > t_0$, $y_0(t) = 0$, and for $t = t_0$, there exists a basis for the eigenspace of $\lambda_1(D(t_0))$ such that one eigenvector of this basis satisfies $y_0(t) \neq 0$ and the other eigenvectors satisfy $y_0(t) = 0$.*

Proof:

Consider the easy case and the case where $t < t_0$ in the hard case. By Theorem 6.1, $\lambda_1(D(t)) < \lambda_1(A))$. Assume $y_0(t) = 0$. Then a short computation shows that this implies $\lambda_1(D(t))$ is an eigenvalues of A, which is a contradiction.

In the hard case, when $t > t_0$, by Theorem 6.1 we know that $\lambda_1(D(t)) = \lambda_1(A)$ and has multiplicity i. Let $\{z_1, z_2, \ldots, z_i\}$ be a basis for the eigenspace of A corresponding to $\lambda_1(A)$ and let z_k be one of these vectors. Then

$$D(t) \begin{bmatrix} 0 \\ z_k \end{bmatrix} = \begin{bmatrix} t & -a^T \\ -a & A \end{bmatrix} \begin{bmatrix} 0 \\ z_k \end{bmatrix} = \begin{bmatrix} -a^T z_k \\ Az_k \end{bmatrix} = \lambda_1(A) \begin{bmatrix} 0 \\ z_k \end{bmatrix},$$

where last equality follows from the fact that in the hard case a is perpendicular to z_k.

This shows that

$$\left\{ \begin{bmatrix} 0 \\ z_1 \end{bmatrix}, \ldots, \begin{bmatrix} 0 \\ z_i \end{bmatrix} \right\} \tag{6.10}$$

is a basis for the eigenspace of $D(t)$ corresponding to $\lambda_1(D(t))$ and $y_0(t) = 0$ for any eigenvector of $\lambda_1(D(t))$.

Finally, when $t = t_0$ in the hard case, by Theorem 6.1 we have $\lambda_1(D(t)) = \lambda_1(A)$ with multiplicity $i + 1$. By the same argument as above, (6.10) is an independent set of eigenvectors for $\lambda_1(D(t))$ and since the multiplicity of this eigenvalue is $i+1$, there must exist an eigenvector in the orthogonal complement of the space spanned by the vectors of this set. Let ω be this eigenvector. Again, without loss of generality, assume A to be diagonal. Therefore $z_k = e_k$ for $k \in \{1, \ldots, i\}$ and $\omega = (\omega_0, 0, \ldots, 0, \omega_{i+1}, \ldots, \omega_n)^T$. If $\omega_0 = 0$, then

$$D(t) \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \omega_{i+1} \\ \vdots \\ \omega_n \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \lambda_{i+1}(A)\omega_{i+1} \\ \vdots \\ \lambda_n(A)\omega_n \end{bmatrix} = \lambda_1(A) \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \omega_{i+1} \\ \vdots \\ \omega_n \end{bmatrix},$$

where the first equality follows from multiplying $D(t)$ with w and the second equality follows since w is an eigenvector for $D(t)$. Now there exists $\omega_k \neq 0$ for $k \in \{i+1, \ldots, n\}$ and this implies $\lambda_1(A) = \lambda_k(A)$, which is a contradiction to $\lambda_1(A) < \lambda_j(A)$ for $j > i$. Hence $\omega_0 \neq 0$. Since the union of ω with the set (6.10) is a basis for the eigenspace of $\lambda_1(D(t))$, the result of the theorem follows. ■

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 [9]). We know from Theorem 6.1 that $\lambda_1(D(t))$ has multiplicity 1 in the easy case and when $t < t_0$ in the hard case. Hence, for these cases,

$$k'(t) = (s^2 + 1)y_0(t)^2 - 1. \tag{6.11}$$

In the hard case, when $t > t_0$, by equation (6.9), equation (6.11) 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; while the directional derivative from the right is -1 (see Appendix A for the graph of $k(\cdot)$ in the different cases).

Since $k(\cdot)$ is a concave and coercive (i.e. diverges to $-\infty$ as $|t| \to \infty$) function, to solve the dual (UD) 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.

6.2 Solving (UD)

In this section, we see how the Rendl and Wolkowicz algorithm solves $\text{TRS}_=$. The easy case is solved in a way similar to the Moré and Sorensen algorithm, except that the function used is not $h(\cdot)$, but $k(\cdot)$ in order to exploit sparsity. They handle the almost

hard case (case 2) similarly to what is done in the Moré and Sorensen algorithm, but no LAPACK routine is used to find the step direction. Instead, the direction is obtained from an eigenvector previously computed.

6.2.1 Solving the Easy Case and the Hard Case (Case 1)

We start this section with a theorem, derived from Theorem 14 in Rendl and Wolkowicz [31], showing that, in the easy case, to each t corresponds a solution to $q(\cdot)$ on a sphere of a certain radius.

Theorem 6.3. *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 $\lambda_1(D(t))$. If $y_0(t) \neq 0$, then*

$$x^* := \frac{1}{y_0(t)} z(t)$$

is an optimal solution of

$$\min\{q(x) : \|x\|^2 = \frac{1 - y_0(t)^2}{y_0(t)^2}\}$$

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

Proof:

By the definition of $y(t)$ we have

$$\begin{bmatrix} t & -a^T \\ -a & A \end{bmatrix} \begin{bmatrix} y_0(t) \\ z(t) \end{bmatrix} = \lambda_1(D(t)) \begin{bmatrix} y_0(t) \\ z(t) \end{bmatrix} \quad \text{and} \quad \|y(t)\|^2 = 1.$$

Expanding these equations gives

$$\begin{aligned} ty_0(t) - a^T z(t) &= \lambda_1(D(t))y_0(t), \\ (A - \lambda_1(D(t))I)z(t) &= y_0(t)a, \\ \text{and} \quad y_0(t)^2 + z(t)^T z(t) &= 1. \end{aligned}$$

Since $y_0(t) \neq 0$, let $x^* := \frac{1}{y_0(t)} z(t)$. Then

$$\begin{aligned} t - a^T x^* &= \lambda_1(D(t)), \\ (A - \lambda_1(D(t))I)x^* &= a, \qquad (6.12) \\ \text{and} \quad x^{*T} x^* &= \frac{1 - y_0(t)^2}{y_0(t)^2}. \qquad (6.13) \end{aligned}$$

Since the interlacing properties of $D(t)$ and A implies $\lambda_1(D(t)) \leq \lambda_1(A)$, we also have

$$A - \lambda_1(D(t))I \succeq 0. \qquad (6.14)$$

Let $\lambda^* = \lambda_1(D(t))$, by the optimality conditions (5.2), then (6.12), (6.13) and (6.14) imply that x^* and λ^* are optimal for TRS$_=$ with $s := \sqrt{\frac{1-y_0(t)^2}{y_0(t)^2}}$. ■

Lemma 13 in Rendl and Wolkowicz [31] shows that, in the easy case, if $y(t)$ is a normalized eigenvector with $y_0(t) > 0$, $y_0(t)$ is a function of $\mathbb{R} \to (0, 1)$ and is strictly monotonically decreasing. It is easy to show then that the function $\frac{1-y_0(t)^2}{y_0(t)^2}$ is a strictly decreasing function $(0, 1) \to (0, \infty)$. Hence for a given s, we can solve TRS$_=$ in the easy case by finding a t such that

$$s^2 = \frac{1 - y_0(t)^2}{y_0(t)^2}. \qquad (6.15)$$

Now note that $k'(t) = 0$ if and only if the previous equation is satisfied. Let

$$t^* := \text{argmax}\{k(t) : t \in \mathbb{R}\}.$$

Hence $\text{TRS}_=$ in the easy case can be solved by finding t^* that satisfies (6.15). 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 6.3.

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 Rendl and Wolkowicz algorithm does this by finding the zero of the function

$$\psi(t) := \sqrt{s^2 + 1} - \frac{1}{y_0(t)}. \tag{6.16}$$

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

6.3 Primal Steps to the Boundary

In this section, we first show that $\psi(\cdot)$ has no zero in the hard case (case 2). We then show how the Rendl and Wolkowicz algorithm handles this case. As in the Moré and Sorensen algorithm, a step to the boundary is taken. We end the section with a new way of stepping to the boundary and show that improvement of the objective function is guaranteed.

6.3.1 Equivalent Moré and Sorensen Primal Step to the Boundary

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 negative 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 6.1 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 (5.2) with $\lambda^* := \lambda_1(A)$ 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_g be such that $k'(t_g) < 0$, then t_g is defined to be on the *good side*. This expression comes from the fact that the good side is where we want to be in the Moré and Sorensen algorithm, that is when $\phi(\lambda) > 0$. Similarly, if for t_b we have $k'(t_b) > 0$, then t_b is defined to be on the *bad side*. If we have a point t_b from the bad side, then $t_b < t_0$ and this implies that $y_0(t_b) \neq 0$. Let $y(t) = [y_0(t), z(t)]^T$. We have

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

Theorem 6.2 implies that

$$x_b := \frac{1}{y_0(t_b)} z(t_b)$$

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 $\lambda_1(D(t_b)) > 0$ for a t_b on the bad 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 $\lambda_1(D(t_b)) \leq 0$. We can now apply Lemma 4.1 with the feasible solution x_b, since

$$(A - \lambda_1(D(t_b))I)x_b = a \quad \text{and} \quad \lambda_1(D(t_b)) \leq 0.$$

Yet, we need to find a vector z, with $\|z\| = 1$, such that $z^T(A - \lambda_1(D(t_b))I)z$ is small. Let t_g be a point from the good side. Then, by Theorem 6.2, in the hard case (case 2), for any eigenvector $y(t_g)$ of $\lambda_1(D(t_g))$, $z(t_g)$ is an eigenvector for $\lambda_1(A)$ and has a unit norm, since $y_0(t_g) = 0$ (notice that in the hard case (case 2), $t_g > t_0$). Hence

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

Therefore, for t_b close to t_0, $z(t_g)^T(A - \lambda(D(t_b))I)z(t_g)$ will be small. The new solution obtained on the boundary is then $x_b + \tau z(t_g)$, where τ is defined as in equation (4.10).

As in the Moré and Sorensen algorithm, every time a feasible solution to TRS is obtained (each new point t_b from the bad side gives us a new feasible solution), if we have a point t_g from the good 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.

6.3.2 A New Primal Step to the Boundary

We show here a new way to take a primal step to the boundary which may give an improvement in the easy case and the hard case (case 1). It is justified by the fact that in these cases the n last components of $y(t_g)$, where t_g is a point from the good side,

might not be an eigenvector for $\lambda_1(A)$. This is because $y_0(t_g) \neq 0$ in the easy case and $y_0(t_g) \neq 0$ might occur in the hard case (case 1) if $t^* < t_0$. Hence the theoretical reasons behind the primal step direction $z(t_g)$ of the previous section do not hold.

If we have a point t_b from the bad side (hence $y_0(t_b) \neq 0$) with $\lambda_1(D(t_b)) \leq 0$, then we showed in the previous section that

$$x_b := \frac{1}{y_0(t_b)} z(t_b) \tag{6.17}$$

minimizes $q(\cdot)$ on the boundary of the trust region of radius $\sqrt{\frac{1-y_0(t_b)^2}{y_0(t_b)^2}}$, which is less than s. According to Theorem 3.1, the sign of the Lagrange multiplier for this solution implies that x_b also minimizes $q(\cdot)$ within the trust region of the same radius. Similarly, we can show, if we have a point t_g from the good side, with $y_0(t_g) \neq 0$ (i.e $t_g < t_0$) and $\lambda_1(D(t_g)) \leq 0$, that

$$x_g := \frac{1}{y_0(t_g)} z(t_g) \tag{6.18}$$

will minimize $q(\cdot)$ within a trust region of radius $\sqrt{\frac{1-y_0(t_g)^2}{y_0(t_g)^2}}$ which is larger than s. A natural way to approximate the solution of TRS and take a step to the boundary would be to pick a point on the boundary of the trust region of radius s which is on the segment linking x_b with x_g. This is somehow a linear interpolation of the solution to TRS using x_b and x_g. With the use of the 2 following lemmas, we show that this primal step to the boundary yields a decrease in the objective function, i.e.

$$q(x_b) \geq q(x_b + \tau(x_g - x_b)), \tag{6.19}$$

for $0 \leq \tau \leq 1$ such that $\|x_b + \tau(x_g - x_b)\|^2 = s^2$.

Lemma 6.1. *Let* $0 < s_1 < s_2$ *and let*

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

If $\|x_b\|^2 = s_1^2$ *and* $\|x_g\|^2 = s_2^2$ *and* $x_b^T(x_g - x_b) \neq 0$, *then* $x_b^T(x_g - x_b) > 0$.

Proof:

By the optimality of x_b and using (5.2), there exits $\lambda \leq 0$ such that $(A - \lambda I)x_b = a$ and $A - \lambda I \succeq 0$. By equation (4.6) we have

$$\begin{aligned} q(x_b + \alpha(x_g - x_b)) &= -(x_b^T(A - \lambda I)x_b - \lambda\|x_b + \alpha(x_g - x_b)\|^2) \\ &\quad +\alpha^2(x_g - x_b)^T(A - \lambda I)(x_g - x_b). \end{aligned} \tag{6.20}$$

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

$$\|x_b + \alpha_i(x_g - x_b)\|^2 = s_2^2 \quad i = 1, 2.$$

A short computation shows that

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

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

Now suppose $x_b^T(x_g - x_b) < 0$, then $\alpha_1 > 0$, $\alpha_2 < 0$ and $|\alpha_2| < |\alpha_1|$. By (6.20) then

$$q(x_b + \alpha_1(x_g - x_b)) = -(x_b^T(A - \lambda I)x_b - \lambda s_2^2) + \alpha_1^2(x_g - x_b)^T(A - \lambda I)(x_g - x_b) \quad (6.21)$$

$$q(x_b + \alpha_2(x_g - x_b)) = -(x_b^T(A - \lambda I)x_b - \lambda s_2^2) + \alpha_2^2(x_g - x_b)^T(A - \lambda I)(x_g - x_b). \quad (6.22)$$

Since $\|x_g\|^2 = s_2^2$, then $\alpha = 1$ solves $\|x_b + \alpha(x_g - x_b)\|^2 = s_2^2$ and therefore $\alpha_1 = 1$. By the optimality of x_g and by equations (6.21) and (6.22) we must have $|\alpha_1| \leq |\alpha_2|$. This is a contradiction, hence $x_b^T(x_g - x_b) > 0$. ■

Lemma 6.2. *Let x_b and x_g be defined as in Lemma 6.1. Suppose $x_b^T(x_g - x_b) \neq 0$. Let*

$$m(\alpha) := q(x_b + \alpha(x_g - x_b)).$$

Then

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

and therefore

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

Proof:

By lemma (6.20), $x_b^T(x_g - x_b) > 0$. Let

$$f(\epsilon) := \|x_b - \epsilon(x_g - x_b)\|^2.$$

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

$$f'(\epsilon) = 2\epsilon\|x_g - x_b\|^2 - 2x_b^T(x_g - x_b).$$

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

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

if $m'(0) > 0$, then $q(x_b - \epsilon(x_g - x_b)) < q(x_b)$ for ϵ small enough. Since for ϵ small enough $\|x_b - \epsilon(x_g - x_b)\|^2 < s_1^2$, then this contradicts the optimality of x_b. Hence

$$m'(0) \leq 0. \tag{6.23}$$

Let

$$w(\epsilon) := \|x_b + (1-\epsilon)(x_g - x_b)\|^2.$$

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

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

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

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

again, by the optimality of x_g, we have

$$m'(1) \leq 0. \tag{6.24}$$

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

$$m'(\alpha) = 2a\alpha + b.$$

From (6.23) and (6.24) we have

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

Let $0 < \alpha < 1$, then if $a > 0$

$$\alpha < 1 \Rightarrow m'(\alpha) = 2a\alpha + b < 2a + b \leq 0.$$

If $a < 0$,

$$0 < \alpha \Rightarrow m'(\alpha) = 2a\alpha + b < b \leq 0.$$

If $a = 0$

$$m'(\alpha) = b \leq 0.$$

Therefore, these inequalities with (6.23) and (6.24) show that

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

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

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

and the lemma follows. ■

If x_b and x_g, defined by equations (6.17) and (6.18), satisfy $x_b^T(x_g - x_b) \neq 0$, then a consequence of the previous lemma is equation (6.19). This equation yields a primal step to the boundary in the easy case and hard case (case 1) which insures a decrease of the primal objective.

6.3.3 Techniques

We briefly describe some techniques that are used in the algorithm to take advantage of the structure of the different functions used. A first technique that is used is *triangle interpolation*. Given a point from the bad side t_b and a point from the good side t_g, since $k'(t_b) > 0$, $k'(t_g) < 0$ and $k(\cdot)$ is a concave function, then an upper bound to q^* can be found at the intersection of the two tangent lines to $k(\cdot)$ in t_b and t_g. The t-value at the point of intersection also updates the approximation for t^*.

A second technique is also used when as above we have points t_b and t_g from the bad and good side. If $k(t_g) > k(t_b)$, then the intersection of the tangent line to $k(\cdot)$ at t_b and the constant function going through $(t_g, k(t_g))$ gives an approximation to t^* that is a lower bound for t^*. This technique is called *vertical cut*. A similar trick can be done if

$k(t_g) < k(t_b)$ and we obtain an upper bound for t^* in this case.

A third technique involves the use of the function $\psi(\cdot)$. In the easy case and the hard case (case 1) it is strictly decreasing and its domain is $\mathbb{R}$. Therefore we can let $t(\psi)$ be its inverse function. Because $\lim_{t\to-\infty} \psi(t) = \sqrt{s^2+1} - 1$, $t(\psi)$ will have an asymptote in $\psi = \sqrt{s^2+1} - 1$ and will be strictly decreasing on the interval $(-\infty, \sqrt{s^2+1} - 1)$. Hence, in the easy case and the hard case (case 1), using $t(\psi)$, we can use values of t to interpolate $t^* = t(0)$.

6.4 Solving TRS

In the previous section we showed how the Rendl and Wolkowicz algorithm is able, when the optimal solution is on the boundary, to solve TRS by maximizing the function $k(\cdot)$. We now show how it handles the case where the minimum is inside the trust region and outline how it solves TRS in general.

In Theorem 3.2, we showed that there does not exist a solution on the boundary of the trust region if and only if A is positive definite and the unconstrained minimizer for $q(\cdot)$ lies in the interior of the trust region. The following theorem is the key to recognizing this case and the proof follows easily from Theorem 3.1.

Theorem 6.4. *Let $\bar{x}$ be a solution to $(A - \lambda I)x = a$ with $A - \lambda I \succeq 0$. If $\lambda \leq 0$, then $\bar{x}$ is a solution to $\min\{x^T Ax - 2a^T x : \|x\|^2 \leq \|\bar{x}\|^2\}$. If $\lambda \geq 0$, then $\bar{x}$ is a solution to $\min\{x^T Ax - 2a^T x : \|x\|^2 \geq \|\bar{x}\|^2\}$.*

Now suppose that x_b satisfies $(A - \lambda I)x_b = a$ with $A - \lambda I \succeq 0$, that $\|x_b\|^2 \leq s^2$ and that $\lambda > 0$, then, by Theorem 6.4, x_b is a solution to $\min\{x^T Ax - 2a^T x : \|x\|^2 \geq \|x_b\|^2\}$. Since λ is positive and since $A - \lambda I$ is positive semidefinite, then A is positive definite. Moreover, by the optimality of x_b, we know that the unconstrained minimum lies inside the region $\{x : \|x\|^2 \leq \|x_b\|^2\}$ which is included in the trust region $\{x : \|x\|^2 \leq s^2\}$.

Therefore, the unconstrained minimizer lies in the trust region and we can apply an unconstrained method to find it, for example a conjugate gradient method.

In the algorithm, we get successive solutions x_b from the bad side and associated multipliers λ_b that satisfy the equation $(A-\lambda_b I)x_b = a$ with $(A-\lambda_b I) \succeq 0$. Therefore each x_b is a solution to $\min\{x^T Ax - 2a^T x : \|x\|^2 = \|x_b\|^2\}$. Checking the sign of the multiplier λ_b tells us if x_b is a solution to $\min\{x^T Ax - 2a^T x : \|x\|^2 \leq \|x_b\|^2\}$ or $\min\{x^T Ax - 2a^T x : \|x\|^2 \geq \|x_b\|^2\}$. If the latter case holds, since $\|x_b\|^2 \leq s^2$, then we know the unconstrained minimum lies in the trust region.

The algorithm solves TRS by trying to solve $k'(t) = 0$ using the function $\psi(\cdot)$. Doing so, points on the good and the bad side are obtained and their respective eigenvalues and eigenvectors $\lambda_1(D(t))$ and $y(t)$ are computed. The algorithm uses inverse interpolation, triangle interpolation and vertical cut to approximate t^*. When a point t_b on the bad side is obtained, a feasible solution x_b is obtained from equation (6.17). If $\lambda_1(D(t_b)) > 0$, then we know that an unconstrained minimizer lies in the trust region and we can apply the conjugate gradient method to obtain a solution to TRS. If the sign of the multiplier is nonpositive and a point from the good side has already been obtained, we can take a primal step to the boundary to obtain a new feasible solution with possibly a smaller objective value. Doing this primal step to the boundary handles the almost hard case (case 2), but note that the triangle interpolation in this case is also very effective, since the function $k(\cdot)$ is linear for $t > t_0$ and asymptotically linear for $t < t_0$. Each iteration gives us a new value of t for which $\lambda_1(D(t))$ and $y(t)$ are computed. This is done until we find a suitable approximation to the maximum of $k(\cdot)$. Bounds on t^* are always available and, if no better approximations to t^* is known, we take the middle point of the interval we know contains t^* to get a new t-value. Note that to speed up the algorithm, a line search such as the ones used in [33] could have been applied.

To conclude this chapter, note that the Rendl and Wolkowicz algorithm has the advantage over the Moré and Sorensen algorithm and the Lanczos method to handle sparsity and the almost hard case (case 2). The algorithm is mainly a dual algorithm, yet it uses the primal objective when taking primal steps to the boundary and in the stopping criteria when the duality gap is computed. Finally, the way of rewriting the dual (UD) in the form of problem (5.16) shows again that a semidefinite program is related to the algorithm.

Chapter 7

Numerical Experiments

This chapter deals with the numerical implementations of some of the ideas and methods discussed in the previous chapters. In the first section, we study the practical use of the new primal step to the boundary we suggested in section 6.3.2 for the Rendl and Wolkowicz algorithm. The second section is a comparison of the Rendl and Wolkowicz algorithm and the Lanczos method within a trust region method. The reader will find in Appendix D the different programs used to generate the results. All programming was done using Matlab 5.2.1.1420 and testing were done on a SUN SPARC station 4.

7.1 Testing the New Primal Step to the Boundary

As proposed in Section 6.3.2, in the Rendl and Wolkowicz algorithm, a primal step to the boundary may be taken when points from the good and the bad side are available. The new step we suggested can be applied when the first component of an eigenvector $y(t_g)$ for the smallest eigenvalue of the matrix $D(t_g)$ of a point t_g from the good side is non-zero. Hence, it may be applicable in the easy case and the hard case (case 1).

We included this step in the Rendl and Wolkowicz algorithm. More precisely, assume

points from the good and the bad side, say t_b and t_g, are available at a stage of the algorithm and that t_b and t_g are the most recent points found from both sides. Then we apply our new step if $y_0(t_g) \geq 10^{-3}/n$, where n is the size of the problem. Otherwise, the primal step of Section 6.3.1 is taken. We refer to this modification of the method as the *modified Rendl and Wolkowicz algorithm.*

The following tables show how this change improves the Rendl and Wolkowicz algorithm. For the first two tables, we considered dense trust region subproblems, i.e. subproblems where the Hessians of the quadratic objective were not sparse (we use the word *sparse* when less than half of the entries of a matrix are non-zero). In Table 7.1, we ran, for different dimensions of subproblems (n=20, 40, 60, 80, 100), 100 random subproblems (see Appendix D to see how those subproblems were constructed) and considered the ones where the easy case or the hard case (case 1) occurred and where a primal step to the boundary was taken to obtain an approximate solution using the Rendl and Wolkowicz algorithm. This allowed us to test the performance of our new primal step in the conditions it was meant to be applied. Our criteria to distinguish the easy case and the hard case (case 1) from the hard case (case 2) or almost hard case (case 2) is

$$\frac{\lambda_1(A) - \lambda^*}{1 + |\lambda_1(A)|} > 0.01. \tag{7.1}$$

If (7.1) holds, we say the easy case or the hard case (case 1) occurs. We considered the improvement for the number of iterations and for the number of matrix-vector multiplications. The percentages given in the tables represent on average the reduction in iterations (matrix-vector multiplications) of the Rendl and Wolkowicz algorithm when we apply the modified Rendl and Wolkowicz algorithm to the same subproblems. We see in Table 7.1 that on average the modified Rendl and Wolkowicz algorithm will take nearly 19% fewer iterations then the original Rendl and Wolkowicz algorithm and that it

seems to be independent of the size of the problem. Similarly, the improvement for the number of matrix-vector multiplications is in the range of 16% and again it seems to be independent of the size of the problem.

In Table 7.2, we considered again 100 subproblems for each dimension, but in this case we considered all the subproblems to compute our average percentages of improvement, i.e. we did not reject the subproblems where the hard case (case 2) or almost hard case (case 2) occurred and the subproblems where no primal step to the boundary were taken to find approximate solutions using the Rendl and Wolkowicz algorithm. In the latter case, the two algorithms perform exactly the same operations and the percentages of improvement are negligible. Hence, we can expect the average percentages of improvement to be lower than in Table 7.1. The idea is to see how helpful the new primal step is in general. We see that we improve the Rendl and Wolkowicz algorithm in general by about 14% for the number of iterations required and by about 11% for the number of matrix-vector multiplications.

	n=20 (%)	n=40 (%)	n=60 (%)	n=80 (%)	n=100 (%)
Percentage of improvement for the number of iterations	19.24	17.70	19.33	20.05	20.40
Percentage of improvement for the number of matrix-vector multiplications	17.23	14.70	15.28	15.74	17.86

Table 7.1: Improvements obtained using the new primal step to the boundary in the easy case or the hard case (case 1) when steps to the boundary need to be taken.

We see that using this new step improves the performance of the Rendl and Wolkowicz algorithm significantly, and the results outlined here indicate that this will be the case for problems of all sizes. Furthermore, the modified Rendl and Wolkowicz algorithm never needed more iterations to solve any trust region subproblem than the Rendl and Wolkowicz algorithm and the optimal values found when the algorithm stopped were

	n=20 (%)	n=40 (%)	n=60 (%)	n=80 (%)	n=100 (%)
Percentage of improvement for the number of iterations	15.38	15.67	11.85	13.61	13.53
Percentage of improvement for the number of matrix-vector multiplications	13.76	12.93	9.48	10.76	10.49

Table 7.2: Improvements obtained using the new primal step to the boundary on general subproblems.

always at least as good.

So far we left out the performance of the modified algorithm on sparse subproblems. Tables 7.3, 7.4 and 7.5 illustrate how the algorithm performs when the density of the Hessian of the quadratic objective function is varied. For example, if the density is 0.1, then approximately $0.1\ n^2$ entries of the Hessian are non-zero. We considered again, for each dimension, 100 random subproblems. The percentages of improvement given are averages obtained when considering all of the subproblems (as in Table 7.2). We added a new row to give the proportion of problems that would have been accepted if a criteria similar to (7.1) would have been used.

	n=20 (%)	n=40 (%)	n=60 (%)	n=80 (%)	n=100 (%)
Percentage of improvement for the number of iterations	14.38	17.50	14.17	22.62	15.52
Percentage of improvement for the number of matrix-vector multiplications	12.79	14.99	11.54	17.55	12.36
Percentage of problems accepted	65	67	73	75	73

Table 7.3: Improvements obtained using the new primal step to the boundary when the density of the Hessian is 0.3.

	n=20 (%)	n=40 (%)	n=60 (%)	n=80 (%)	n=100 (%)
Percentage of improvement for the number of iterations	17.18	17.67	14.56	15.58	14.52
Percentage of improvement for the number of matrix-vector multiplications	15.32	15.10	11.65	12.25	10.91
Percentage of problems accepted	67	69	70	71	63

Table 7.4: Improvements obtained using the new primal step to the boundary when the density of the Hessian is 0.2.

	n=20 (%)	n=40 (%)	n=60 (%)	n=80 (%)	n=100 (%)
Percentage of improvement for the number of iterations	11.30	15.37	11.06	11.85	11.68
Percentage of improvement for the number of matrix-vector multiplications	9.77	13.20	13.61	9.60	9.12
Percentage of problems accepted	62	66	65	70	66

Table 7.5: Improvements obtained using the new primal step to the boundary when the density of the Hessian is 0.1.

The results confirm again that the improvements we get with the modified Rendl and Wolkowicz algorithm are independent of the size of the problem, yet as the density of the Hessian decreases, the improvements are slightly lower. More precisely, as the density takes the values 0.3, 0.2 and 0.1, the percentages of improvement for the number of iteration are approximately 17%, 16% and 12% respectively. Similarly, we get approximately 14%, 13% and 11% improvement in the number of matrix-vector multiplications as the density varies. Again, this shows that the modified Rendl and Wolkowicz algorithm is an improvement to the original method.

7.2 Comparison of the Rendl and Wolkowicz Algorithm and the Lanczos Method Within a Trust Region Method

In this section, we compare the performance of the Rendl and Wolkowicz algorithm and Lanczos method when used respectively within a trust region method. In the article of Gould et al. [7], the authors suggest that the Lanczos method may be stopped if convergence occurs or if a limited extra number of iterations, say k, have been done once the boundary has been encountered (i.e. in Algorithm 4.2, once the variable INTERIOR is set to false, at most k extra iterations are completed). The reason for doing so and obtaining a rather cheap approximation to the subproblem is motivated by the following, which can be found in the conclusion of the paper:

> "We must admit to being slightly disappointed that the new method" (Lanczos method) "did not perform uniformly better than the Steihaug-Toint scheme" (Lanczos method when $k = 0$) ", and we were genuinely surprised that a more accurate approximation does not appear to significantly reduce the number of function evaluations within a standard trust-region method, at least in the tests we performed. While this may limit the use of the methods developed here," (the Lanczos methods) "it also calls into question a number of other recent eigensolution-based proposals for solving the trust-region subproblem" (in particular, the Rendl and Wolkowicz algorithm)."While these authors demonstrate that their methods provide an effective means of solving the subproblem, they make no efforts to evaluate whether this is actually useful within a trust-region method. The results given in this paper suggest that this may not in fact be the case. (...) We believe that further testing is needed to confirm the trends we have observed here."

Gould et al. [7] question if the trust region subproblem needs to be solved accurately if we want a trust region method to be efficient. From the results obtained in their paper, they suggest that, indeed, the high accuracy is unnecessary. This is why they propose that, using the Lanczos method, at most a few iterations should be done once the INTERIOR variable is false. For the problems we tested here, we have set this limit to 10, i.e. $k = 10$. The previous quote also questions if being able to handle the hard case (case 2) or near hard case (case 2) is a desirable feature for a trust region subproblem algorithm.

The goal of this section is to answer all of the above questions. The first tests of this section are done on the following eleven problems (more details on the problems can be found in Appendix C):

1. BRYBND: Broyden banded function [10],
2. GENROSE: Generalized Rosenbrock [19],
3. EXPWSF: Extended Powell singular function [10],
4. TRIDIA: [13],
5. EXTROS: [13],
6. DBNDVF: Discrete boundary value function [10],
7. BTRDIA: Broyden tridiagonal function [10],
8. BNALIN: Brown almost-linear function [10],
9. LINFRK: Linear function full rank [10],
10. SENSORS: Optimal sensor placement [30],
11. WATSON: Watson function [10].

Except for the last four problems, these problems have a sparse Hessian.

Throughout this section, we will minimize a function $f(\cdot)$ of n real variables x. To do so, we use the trust region method described by the following algorithm. In this algorithm, $f(\cdot)$ represents the function to be minimized, x_k is an approximation of a minimizer after k iterations and s_k is a positive number - the radius of the trust region at iteration k.

Algorithm 7.1. (Trust Region Method)

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

$$\frac{\|\nabla f(x_k)\|}{1+|f(x_k)|} < 10^{-5}. \tag{7.2}$$

2. *Solve for δ_k*

$$\begin{aligned} \delta_k \in \quad & \text{argmin} \quad q_k(x_k) := \nabla f(x_k)^T\delta + \tfrac{1}{2}\delta^T\nabla^2 f(x_k)\delta \\ & \text{s.t.} \qquad \|\delta\|^2 \le s_k^2. \end{aligned}$$

3. *Evaluate $r_k = \frac{f(x_k)-f(x_k+\delta_k)}{q_k(x_k)-q_k(x_k+\delta_k)}$.*

4. *(a) If $r_k > 0.95$, set $s_{k+1} = 2s_k$ and $x_{k+1} = x_k + \delta_k$.*

   (b) If $0.01 \le r_k < 0.95$, set $s_{k+1} = s_k$ and $x_{k+1} = x_k + \delta_k$.

   (c) If $r_k < 0.01$, set $s_{k+1} = 0.5s_k$ and $x_{k+1} = x_k$.

Except for the stopping criteria which has been scaled here, this algorithm is Algorithm 6.1 in Gould et al. [7]. The initial approximation x_0 for a minimizer is problem dependent and the initial size of the trust region, s_0, is chosen to be 1. We use either the Rendl and Wolkowicz algorithm (in the modified form of Section 7.1) or the Lanczos method to solve the second step of the algorithm. When the Lanczos method is used, we

stop when convergence is reached (see the convergence criteria of Algorithm 4.2) or when at most 10 extra iterations have been done once the INTERIOR variable is set to be false. When the Rendl and Wolkowicz algorithm is used, if the solution is in the interior of the trust region, then we stop when we have a δ_k such that

$$\|\nabla f(x_k) + \nabla^2 f(x_k)\delta_k\| < \max(10^{-8}, 10^{-5}\|\nabla f(x_k)\|).$$

When the solution is on the boundary of the trust region, we first need some notation to define the stopping criteria. At some stage of the algorithm, let t_{up} and t_{low} be upper and lower bound for t^*, let q_{up} and q_{low} be upper and lower bounds on q^* and let x_{best} be the current approximation to a minimizer of the trust region subproblem. Define the following scalars

$$\begin{aligned} \text{dgaptol} &= \max(10^{-8}, 10^{-5}\|\nabla f(x_k)\|), \\ \text{normtol} &= \text{dgaptol}, \\ \text{zerotol} &= \text{dgaptol}/\log(n^{10}). \end{aligned}$$

Let $w_1 = 1$, if $\|x_{best}\| < (1 + \text{normtol})s$. Let $w_2 = 1$, if $\frac{q_k(x_{best}) - q_{low}}{1+|q_k(x_{best})|)} < 2\text{dgaptol}$. Let $w_3 = 1$, if $\frac{q_{up} - q_{low}}{|q_{up}|+1} <$ dgaptol . Let $w_4 = 1$, if the number of iteration is greater than 30. Let $w_5 = 1$, if $\frac{t_{up} - t_{low}}{1+|t_{up}|} <$ zerotol . Let these last five variables be 0 otherwise. When the solution is on the boundary, we stop when:

$$((w_1 \text{ or } w_2) \text{ and } w_3) \text{ or } w_4 \text{ or } w_5.$$

For each problem solved, we are interested in the number of function evaluations, the number of iterations, the number of matrix-vector multiplications and the computation time taken to obtain an approximation to the solution of each problem using Algorithm 7.1. In Table 7.6, under each variable of interest, there are two columns that show each the results obtained when using the Rendl and Wolkowicz algorithm (RW) or the Lanczos method (LM) was used to solve the trust region subproblems. For the Rendl and Wolkowicz algorithm, the Matlab function *eigs* was used to compute the eigenvalues when the Hessian is sparse. For the last four problems, the Hessians are dense and *eig* was used instead (this function however does not allow us to keep track of the number of matrix-vector multiplications).

	n	function evaluations		iterations		matrix-vector multiplications		cpu time (seconds)	
		RW	LM	RW	LM	RW	LM	RW	LM
BRYBND	1000	26	26	25	25	41941	505	612.94	125.63
GENROSE	100	68	69	79	85	37218	5857	508.66	255.38
EXPWSF	100	18	18	17	17	1250	297	13.88	2.43
TRIDIA	100	5	6	4	5	1496	577	23.09	1.78
EXTROS	500	14	14	15	15	1644	147	93.14	46.57
DBNDVF	25	3	11	2	10	429	1050	6.78	0.94
BTRDIA	200	7	7	6	6	3662	338	200.44	3.11
BNALIN	30	13	6	13	5	-	37	2.91	0.88
LINFRK	100	8	8	9	9	-	45	9.07	4.94
SENSORS	100	18	18	18	18	-	826	50.00	67.93
WATSON	31	13	13	14	12	-	700	28.88	23.81

Table 7.6: Comparing the performance of the Rendl and Wolkowicz algorithm and the Lanczos method within a trust region algorithm on different problems.

In the next table, we make a similar kind of comparison, but the Rendl and Wolkowicz algorithm is slightly changed based on the following idea: since we initially compute the smallest eigenvalue of A in the algorithm, if the latter is positive we may compute the Newton direction which will be a descent direction in this case. We take the Newton

step if the ratio of the actual improvement over the predicted improvement is above 0.95, otherwise we backtrack by 0.5 along the Newton direction until the ratio becomes higher than 0.95. If we take a step along the Newton direction, then the trust region radius is set to be the length of that step, unless it is less than $0.5s$, s or $2s$ depending if the performance ratio when moving along the Newton step is respectively less than 0.01, between 0.01 and 0.95 or greater than 0.95. This modification of the Rendl and Wolkowicz algorithm is referred to as RW2 in the following table.

	n	function evaluations		iterations		matrix-vector multiplications		cpu time (seconds)	
		RW2	LM	RW2	LM	RW2	LM	RW2	LM
BRYBND	1000	23	26	22	25	2507	505	316.74	125.63
GENROSE	100	100	69	78	85	30808	5857	343.69	255.38
EXPWSF	100	17	18	16	17	5805	297	56.07	2.43
TRIDIA	100	3	6	2	5	552	577	5.83	1.78
EXTROS	500	20	14	14	15	1708	147	69.64	46.57
DBNDVF	25	3	11	2	10	970	1050	3.01	0.94
BTRDIA	200	6	7	5	6	3345	338	191.22	3.11
BNALIN	30	10	6	9	5	316	37	1.73	0.88
LINFRK	100	3	8	1	9	37	45	1.41	4.94
SENSORS	100	18	18	18	18	-	826	52.57	67.93
WATSON	31	13	13	15	12	-	700	30.98	23.81

Table 7.7: Comparing the performance of the Rendl and Wolkowicz algorithm using the Newton step and the Lanczos method within a trust region algorithm on different problems.

From this set of problems, we derive from the last two tables a conclusion similar to the one in Gould et al. [7], i.e. it is not clear, looking at the number of function evaluations or iterations, which method performs the best. The version of the Rendl and Wolkowicz algorithm that moves along the Newton direction when it is a descent direction seems to do slightly better then the first version on these problems in terms of the number of iterations and as well for the number of function evaluations. Also,

because backtracking may be necessary when using the Newton direction, more functions evaluations are needed on some problems. When the Lanczos method is used, because the solutions to the trust region subproblems obtained are found on a subspace rather then on the whole space as in the Rendl and Wolkowicz algorithm, the number of matrix-vector multiplications and the computation times are on average higher when the latter is used. Thus, the previous results suggest that on the previous set of problems, most of the time one should choose the Lanczos method to solve the trust region subproblems, since this translates into lower computation costs. Hence, should we conclude as well that high accuracy solutions for the subproblems are not useful? Before answering, consider the following function:

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

The gradient of this function is zero when $|x_i| = 1$ or 0 for $i = 1 \ldots n-1$ and $x_n = 1$. For the Hessian to be positive semidefinite when the gradient is zero, we need $|x_i| = 1$ for $i = 1 \ldots n-1$, otherwise the Hessian is indefinite. Hence, when $|x_i| = 1$ for $i = 1 \ldots n-1$ and $x_n = 1$, then we have a local minimum and looking back at the function we see that it is also a global minimum.

In the following, we will minimize this function using a trust region method. Let

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

be the initial point to start the method. We have

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

x_0 in this problem is chosen to create the hard case (case 2) initially. Indeed, the gradient is perpendicular to the space spanned by the eigenvectors of the smallest eigenvalue of the Hessian. Easy computations as in Example 3.1 show that the hard case (case 2) occurs if the initial size of the trust region (i.e. s_0) is greater or equal to 1/6. Therefore, choosing $s_0 = 1$ makes the hard case (case 2) occur initially.

When the trust region method 7.1 is used to minimize this function, with the Rendl and Wolkowicz algorithm used to solve the trust region subproblems, it converges to the point

$$\tilde{x} = [1, -1, 1, -1, 1, 1, 1, 1, -1, 1]^T,$$

which is a global minimum of the function. On the other hand, when the Lanczos method is used to solve the subproblems, the method converges to the point

$$\hat{x} = [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]^T.$$

The trust region algorithm stops because the gradient at the previous point is zero. However, the Hessian at the same point is indefinite and we stop at a point which is neither a local or a global minimum. Since, the function is coercive, the iterates obtained by a trust region method will be bounded and according to Theorem 2.1, there will be

an accumulation point that satisfies first and second order optimality conditions. Hence, a trust region method should be able to escape from points where the gradient is zero and where the Hessian is indefinite. Indeed, this is possible by moving in the direction of an eigenvector of the smallest eigenvalue of the Hessian. Therefore, if we modify the stopping criteria and stop when (7.2) is satisfied, as before, and when the Hessian is positive semidefinite, we can prevent the algorithm to stop when the gradient is zero and when the Hessian is indefinite.

In the above problem, this requires that we are able to solve the hard case (case 2), since the gradient at $\hat{x}$ being zero, it is perpendicular to the space spanned by the eigenvectors of the smallest eigenvalue of $\nabla^2 f(\hat{x})$ and since for any positive radius s -the size of the trust region- the hard case (case 2) occurs. Because, the Lanczos method fails when the gradient is zero, even if we change the stopping criteria, Algorithm 7.1, with the Lanczos method used to solve the trust region subproblem, fails to find the optimum of the function and dies at $\hat{x}$. On the other hand, if the Rendl and Wolkowicz algorithm is used to minimize the function with the starting point $\hat{x}$, then it converges to a global minimum of the function.

This example shows that being able to handle the hard case (case 2) is an important feature for a trust region subproblem algorithm, since it leads to a robust trust region method. An analysis of the iterates obtained by Algorithm 7.1, when the Rendl and Wolkowicz algorithm is used to solve the subproblems and when x_0 is the starting point, shows that the algorithm does not encounter points where the gradient is zero and the Hessian is indefinite. In particular, $\hat{x}$ is not part of the iterates. This is caused by the fact that the solutions to the initial trust region subproblem are different depending on the algorithm used. Because the hard case (case 2) occurs at x_0, the Lanczos method cannot find an accurate solution. On the other hand, the Rendl and Wolkowicz algorithm does solve it up to the required accuracy and this is what explains the two different behaviors

of Algorithm 7.1. In fact, the Lanczos method in the first iteration gives an answer to the initial trust region subproblem which is in the interior of the trust region. More precisely, the conjugate gradient method, which is applied until the boundary is encountered, stops because it finds a point where the gradient of the quadratic objective is zero. Yet, since $\nabla^2 f(x_0)$ is indefinite, the problem is not convex and the solution is on the boundary. This shows another weakness of the Lanczos method: the conjugate gradient method, which is used before the boundary is encountered, may converge to a point where the gradient of the quadratic objective is zero, although the solution is not in the interior.

Hence, this example contradicts the assertions found in the paragraph quoted at the beginning of this section. As it may not come essential when minimizing some problems, beings able to solve the subproblems accurately within a trust region method is important. For example, one never knows when the hard case (case 2) or near hard case (case 2) may occur and, as we have seen, the behavior of the algorithm depending on the accuracy of the methods used to solve the subproblems may be very different. It is true that the Rendl and Wolkowicz algorithm is more expensive than the Lanczos method, when computation time and matrix-vector multiplications are considered, but this is the price one has to pay for the robustness of the trust region method. In practice, one may wish to attempt minimizing a function using the Lanczos method to approximate the solutions of the subproblems of the trust region method. Yet, if the Hessian of the approximate solution is not positive semidefinite, the Rendl and Wolkowicz algorithm may be necessary to solve the subproblems.

Chapter 8

Conclusion

We have considered in this thesis, three different algorithms for solving the trust region subproblem: the Moré and Sorensen algorithm, the Lanczos method and the Rendl and Wolkowicz algorithm. We have put the main focus on the last two, since they are modern algorithms that can exploit the sparsity of the Hessian. Yet, the Moré and Sorensen algorithm, although it does not exploit sparsity, solves the easy case and the case 1 of the hard case using Newton's method, it handles the case 2 of the hard case and many of the tricks used in this algorithm have been kept in the two other algorithms we considered. Semidefinite programs were the link to explain these three algorithms. This idea came first in the paper of Rendl and Wolkowicz [31] to explain the Moré and Sorensen algorithm and their algorithm. They showed through two different type of dual problems, that could be stated in the form of semidefinite programs, that the trust region subproblem was equivalent to the maximization of a concave function. Their work appeared in the *Duality* Chapter. Furthermore, we showed that one of these duals could be used to analyze the Lanczos method. In particular, this showed that measuring the norm of the gradient of the Lagrangian at an approximate solution was linked to measuring a duality gap.

We also proposed in the chapter reviewing the Rendl and Wolkowicz algorithm a new primal step to the boundary. A step that was equivalent to the Moré and Sorensen primal step to the boundary was used in the algorithm, but the new step presented was based on the fact that a decrease in the objective function could be obtained if one moves from a bad side point in the direction of a good side point when the sign of the multipliers are nonpositive.

It is this idea that was first tested in our numerical section and it seemed to improve the original Rendl and Wolkowicz algorithm. The second part of our numerical section compared the Lanczos method and the Rendl and Wolkowicz algorithm when used within a trust region method. Because the Rendl and Wolkowicz algorithm is able to handle the case 2 of the hard case and always give an accurate solution to the subproblem, unlike the Lanczos method that does not handle this case, the goal of this section was to see if we would obtain a similar conclusion to the one in the Gould et al. [7] paper. These authors suggested that solving the subproblem more accurately does not appear to reduce the number of function evaluations in a trust region method. We compared the Rendl and Wolkowicz algorithm and the Lanczos method when used within a trust region method to solve different test problems. On the first eleven problems we considered, our conclusions were similar to the ones of Gould et al. Yet, we constructed an example showing that on some problems, a trust region method that uses the Lanczos method to approximate the subproblems may get stuck at points where the gradient is zero and the Hessian indefinite. On the other hand, the Rendl and Wolkowicz algorithm, although more expensive, proved to solve all the problems considered. Hence, the extra computations of the Rendl and Wolkowicz algorithm compared to the Lanczos method is the price one has to pay for a more robust trust region method.

It appears that in some cases low accuracy solutions to the subproblems are as good as high accuracy solutions when the overall number of function evaluations is considered,

yet since one never knows when the case 2 of the hard case may occur, a robust trust region algorithm needs a subroutine that can solve the subproblems accurately, especially if the case 2 of the hard case occurs.

Appendix A

Figures

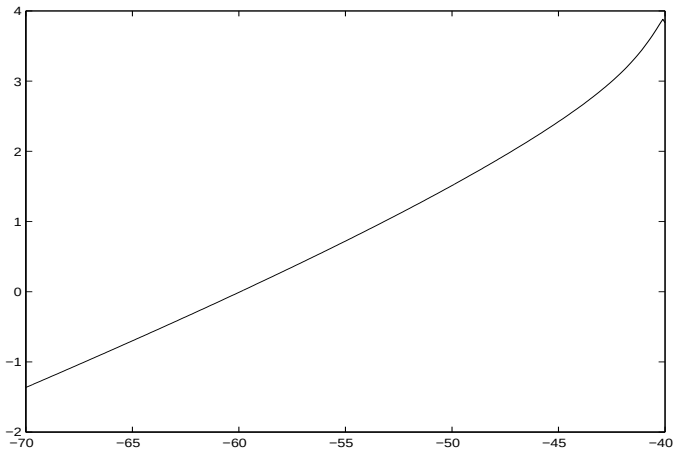

Figure A.1: $\phi(\lambda)$ in the easy case

In this figure $\lambda_1(A) = -40.1048$ and $\lambda^* = -59.8596$.

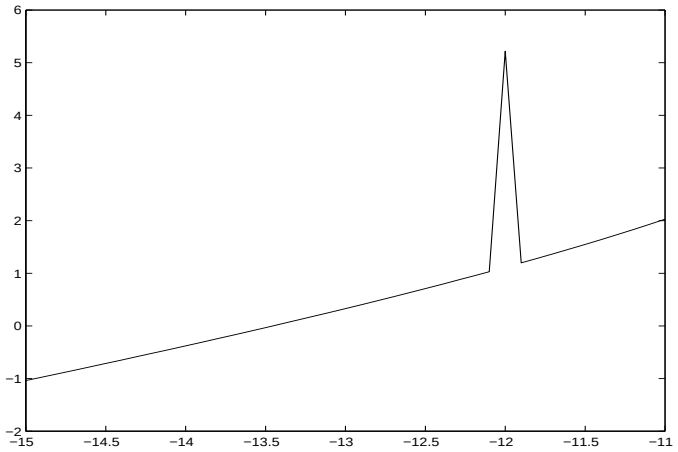

Figure A.2: $\phi(\lambda)$ in the almost hard case (case 1)

In this figure $\lambda_1(A) = -12.0000$ and $\lambda^* = -13.4545$.

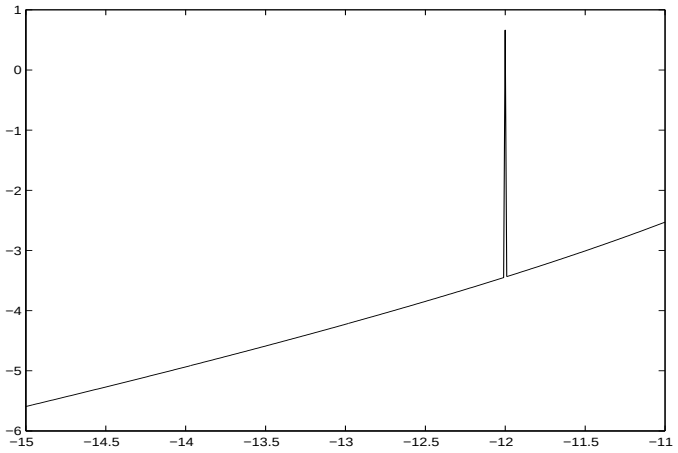

Figure A.3: $\phi(\lambda)$ in the almost hard case (case 2)

In this figure $\lambda_1(A) = -12$ and λ^* is almost -12.

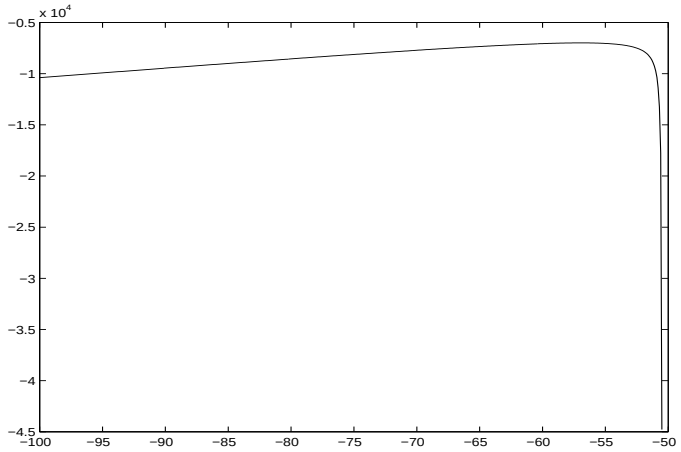

Figure A.4: $h(\lambda)$ in the easy case

In this figure $\lambda_1(A) = -50.4601$ and $\lambda^* = -58.8700$. Note that $h'(\lambda^*) = 0$.

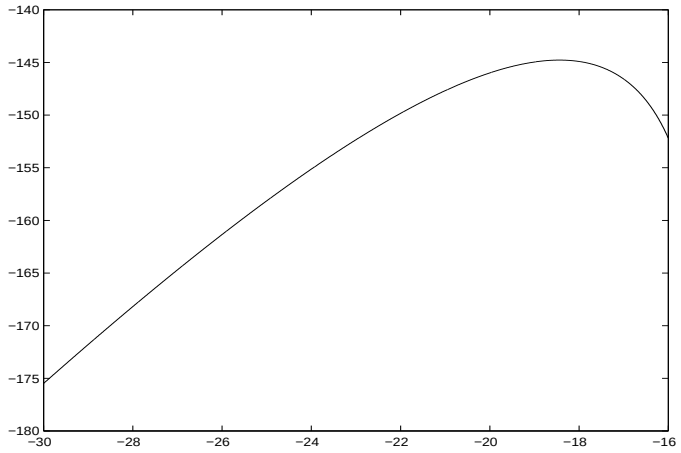

Figure A.5: $h(\lambda)$ in the hard case (case 1)

In this figure $\lambda_1(A) = -17.8085$ and $\lambda^* = -18.4342$. Note that $h'(\lambda^*) = 0$ and that the function is continuous in $\lambda_1(A)$.

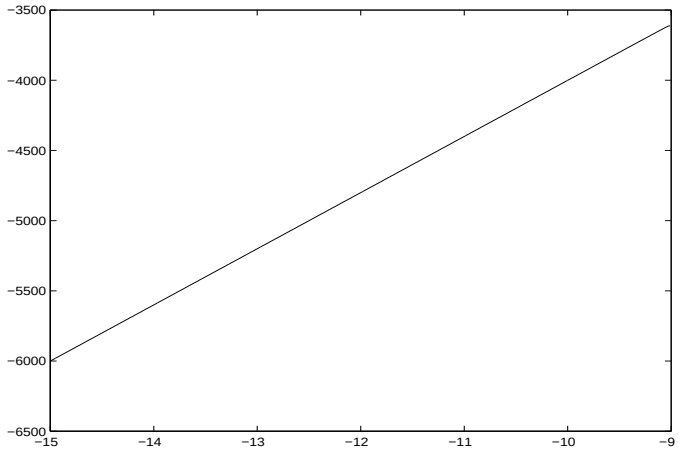

Figure A.6: $h(\lambda)$ in the hard case (case 2)

In this figure $\lambda_1(A) = \lambda^* = -12$. Again, the function is continuous in $\lambda_1(A)$ and striclty increasing on $(-\infty, \lambda_1(A)]$. This is why $\lambda^* = \lambda_1(A)$.

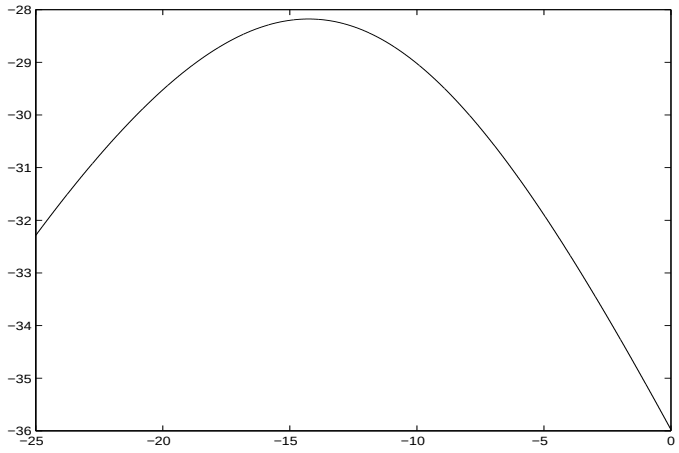

Figure A.7: $k(t)$ in the easy case

In this figure, $t^* = -14.2617$. Note that the function is differentiable everywhere and $k'(t^*) = 0$. We have also in this case $\lambda^* = -21.2191$ and $\lambda_1(A) = -17.4435$.

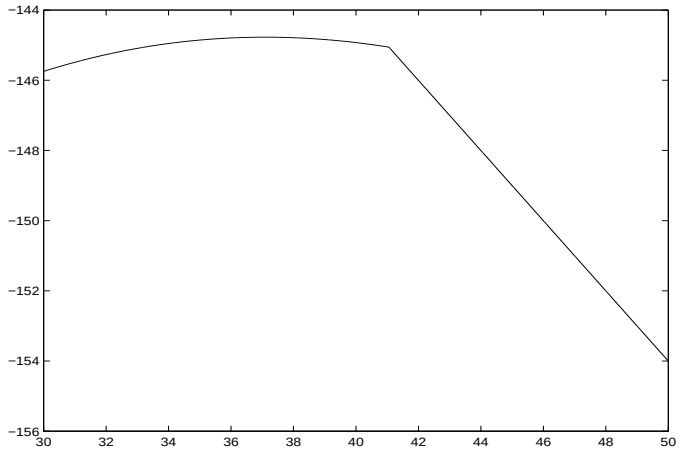

Figure A.8: $k(t)$ in the hard case (case 1)

In this figure, $t^* = 37.0878$ and $t_0 = 41.0549$. Note three things: the function is differentiable everywhere except in t_0, this is the hard case (case 1) since $k'(t^*) = 0$ and the function is purely linear for $t > t_0$. We also have $\lambda^* = -18.4391$ and $\lambda_1(A) = -17.8085$.

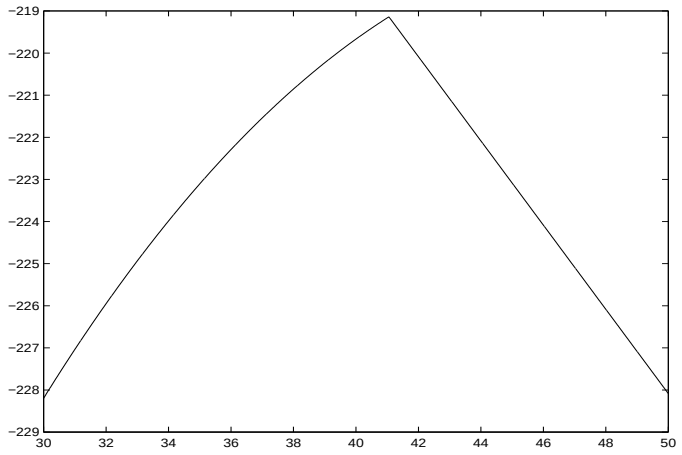

Figure A.9: $k(t)$ in the hard case (case 2)

In this figure, $t^* = t_0 = 41.0549$. Note three things: the function is differentiable everywhere except in t_0, this is the hard case (case 2) since $k(\cdot)$ is not differentiable in $t^* = t_0$ and the function is purely linear for $t > t_0$. We also have $\lambda^* = \lambda_1(A) = -17.8085$.

Appendix B

Mathematical Background

B.1 The derivatives of $h(\cdot)$

Recall that

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

Let λ_l and the sequence $\{\lambda_w\}$ be defined as in section 5.1 and assume that the sequence converges to $\tilde{\lambda} \in (-\infty, \lambda_l)$. Then for $w \in \mathbb{N}$

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

Since

$$\frac{d}{dZ}(Z^{-1})(\cdot) = -Z^{-1} \cdot Z^{-1},$$

then

$$h(\lambda_w + h) = -a^T((A - \lambda_w I)^{-1} - (A - \lambda_w I)^{-1}(-hI)(A - \lambda_w I)^{-1})a + o(h)a^T a + \lambda_w s^2 + hs^2$$

$$= -a^T(A - \lambda_w I)^{-1}a + \lambda_w s^2 - ha^T(A - \lambda_w I)^{-2}a + hs^2 + o(h)a^T a.$$

Therefore

$$\lim_{h\to 0} \frac{h(\lambda_w + h) - h(\lambda_w)}{h} = \lim_{h\to 0} \frac{-ha^T(A - \lambda_w I)^{-2}a + hs^2 + o(h)a^T a}{h}$$

$$= -a^T(A - \lambda_w I)^{-2}a + s^2 + \lim_{h\to 0} a^T a \frac{o(h)}{h} = -a^T(A - \lambda_w I)^{-2}a + s^2.$$

Hence

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

where we have used the fact that $\gamma_j = 0$ for $j \in \{l, \ldots, l-1\}$. Using the last equation and the fact that $\lambda_j - \tilde{\lambda} > 0$ for $j \in \{1, \ldots, n\}$, we get that

$$h'(\lambda_w) \to -\sum_{j=l}^{n} \frac{\gamma_j^2}{\lambda_j - \tilde{\lambda}} + s^2 = -a^T((A - \tilde{\lambda}I)^{\dagger})^2 a + s^2.$$

Since $h(\cdot)$ is a continuous function over $(-\infty, \lambda_l)$, then we get for λ in that interval

$$h'(\lambda) = -a^T((A - \tilde{\lambda}I)^{\dagger})^2 a + s^2.$$

To compute the second derivative of $h(\cdot)$ over $(-\infty, \lambda_l)$, similar computations would yield

$$h''(\lambda) = -2a^T((A - \tilde{\lambda}I)^\dagger)^3 a.$$

B.2 The concavity of $\lambda_1(D(\cdot))$

Here we show that $\lambda_1(D(\cdot))$ is a concave function. The result is based on the fact that if A and B are two symmetric matrices, then

$$\begin{array}{lcccccccc} \lambda_1(A+B) & = & \min & x^T(A+B)x & \geq & \min & x^TAx & + & \min \quad x^TBx = \lambda_1(A) + \lambda_1(B) \\ & & \text{s.t.} & \|x\| = 1 & & \text{s.t.} & \|x\| = 1 & & \text{s.t.} \quad \|x\| = 1 \end{array}.$$

Now let $\sigma \in [0,1]$, $t_1 \in \mathbb{R}$ and $t_2 \in \mathbb{R}$. Then by the inequalities above we have

$$\lambda_1(D(\sigma t_1 + (1-\sigma)t_2)) = \lambda_1(\sigma D(t_1) + (1-\sigma)D(t_2)) \geq \sigma\lambda_1(D(t_1)) + (1-\sigma)\lambda_1(D(t_2)).$$

This proves $\lambda_1(D(\cdot))$ is a concave function.

Appendix C

Details on the Test Problems

We briefly outline the parameter values and starting points we have chosen for each of the 10 problems we used for testing. For most of the problems, a standard starting point exists and for some of them we needed to provide one. We denote the starting point by x_0. We use the variable n for the number of variables in each problems. These problems are of the form

$$\sum_{i=1}^{m} f_i(x).$$

We now give the corresponding informations for each problem:

1. BRYBND: Broyden banded function [10],
   (a) $n = 1000$, $m = n$,
   (b) $x_0 = [-1, \ldots, -1]$,
2. GENROSE: Generalized Rosenbrock [19],
   (a) $n = 100$, $m = n$,

(b) $x_0 = [\frac{1}{n+1}, \frac{2}{n+1}, \ldots, \frac{n}{n+1}]$,

3. EXPWSF: Extended Powell singular function [10],

   (a) $n = 52$, $m = n$,

   (b) $(x_0)_{4j-3} = 3$, $(x_0)_{4j-2} = -1$, $(x_0)_{4j-1} = 0$, $(x_0)_{4j} = 1$,

4. TRIDIA: [13],

   (a) $n = 100$, $m = n$,

   (b) $x_0 = [1, \ldots, 1]$,

5. EXTROS: [13],

   (a) $n = 500$, $m = n/2$,

   (b) $x_0 = [1, \ldots, 1]$,

6. DBNDVF: Discrete boundary value function [10],

   (a) $n = 25$, $m = n$,

   (b) $(x_0)_i = \frac{i(i-n-1)}{(n+1)^2}$,

7. BTRDIA: Broyden tridiagonal function [10],

   (a) $n = 200$, $m = n$,

   (b) $x_0 = [-1, \ldots, -1]$,

8. BNALIN: Brown almost-linear function [10],

   (a) $n = 30$, $m = n$,

   (b) $x_0 = [\frac{1}{2}, \ldots, \frac{1}{2}]$,

9. LINFRK: Linear function full rank [10],

(a) $n = 100$, $m = n$,

(b) $x_0 = [1, \ldots, 1]$,

10. SENSORS: Optimal sensor placement [30],

(a) $n = 100$,

(b) $x_0 = [\frac{1}{n}, \frac{2}{n} \ldots, 1]$,

11. WATSON: Watson function [10],

(a) $n = 31$, $m = n$,

(b) $x_0 = [0, \ldots, 0]$.

Appendix D

Matlab Programs

D.1 Generating Random Trust Region Subproblems

In Section 7.1, we tested our new primal step to the boundary on hundreds of random trust region subproblems. This is the Matlab file that was used to generate these problems:

```
% INPUT: n order of matrix
s=rand/rand
tt=1/rand
A=tt*randn(n)
A=A+A'
tt=1/rand
a=tt*randn(n,1)
% The objective is x'Ax - 2a^T x and the trust region radius is s.
```

D.2 Files on the Trust Region Methods

The files that were used for testing in Section 7.2 can be found on the World Wide Web at the following address: http://orion.uwaterloo.ca/ hwolkowi. The files used to enter the information for each problem are **initialize.m**, **objective.m** and **objgradhess.m**. The files **newtrust.m** and **lanczoslim.m** respectively are the Rendl and Wolkowicz algorithm and the Lanczos method. **trmgould.m** and **trmbgouldlanczoslim.m** implement the trust region method of Algorithm 7.1 depending respectively if the Rendl and Wolkowicz algorithm or the Lanczos method is used to solve the trust region subproblems.

Bibliography

[1] D.P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1995.

[2] P.G. Ciarlet. *Introduction à l'analyse numérique matricielle et à l'optimisation*. Collection mathématiques apppliquées pour la maîtrise. Masson, fifth edition, 1990.

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

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

[5] S.M. Goldfeld, R.E. Quandt, and H.F. Trotter. Maximization by quadratic hill-climbing. *Econometrica*, 34(2):541–551, 1966.

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

[7] 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.

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

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

[10] B.S. Garbow J.J. Moré and K.E. Hillstrom. Testing unconstrained optimization software. *ACM Trans. Math. Software*, 7(1):17–41, 1981.

[11] C. Lemaréchal and F. Oustry. Semidefinite relaxations and lagrangian duality with application to combinatorial optimization. Technical Report 3710, INRIA, 655, avenue de l'Europe,38330 Montbonnot St-Martin (France), 1999.

[12] K. Levenberg. A method for the solution of certain nonlinear problems. *Quarterly of Applied Mathematics*, 2:164–168, 1944.

[13] G. Liu and J. Nocedal. Test results of two limited memory methods for large scale optimization. Technical Report NAM 04, Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, Illinois, 1985.

[14] D.W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963.

[15] J.M. Martínez. Local minimizers of quadratic functions on euclidian balls and spheres. *SIAM Journal on Optimization*, 4(1):159–176, 1994.

[16] J.M. Martínez and S.A. Santos. A trust-region strategy for minimization on arbitrary domains. *Mathematical Programming*, 68(3):267–301, 1995.

[17] J.J. Moré. Generalizations of the trust region problem. Technical report, Argonne National Laboratory, 1993. Preprint MCS-P349-0193.

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

[19] S. Nash. Newton-type minimization via the lanczos process. *SIAM journal on numerarical analysis*, 21:770–788, 1984.

[20] M.J.D. Powell. A new algorithm for unconstrained optimization. In J.B. Rosen, O.L. Mangasarian, and K. Ritter, editors, *Nonlinear Programming*, pages 31–65. Academic Press, New York, NY, 1970.

[21] M.J.D. Powell. Convergence properties of a class of minimization algorithms. In O.L. Mangasarian, R.R. Meyer, and S.M. Robinson, editors, *Nonlinear Programming 2*, pages 1–27. Academic Press, New York, NY, 1975.

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

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

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

[25] M. Rojas, S.A. Santos, and D.C. Sorensen. A new matrix-free algorithm for the large-scale trust-region subproblem. Technical Report TR99-19, Rice University, Houston, TX, 1999.

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

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

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

[29] 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.

[30] X. Wang and H. Zhang. Optimal sensor placement. *SIAM Review*, 35:641, 1993.

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

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

[33] 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.

[34] Y.X. Yuan. An example of non-convergence of trust region algorithms. In *Applied Optimization*, volume 14 of *Advances in nonlinear programming (Beijing, 1996)*, pages 205–215. Kluwer Acad. Publ., Dordrecht, 1998.

[35] Y. Zhang. Computing a celis-dennis-tapia trust-region step for equality constrained optimization. *Mathematical Programming*, 55(1):109–124, 1992.