A New Matrix–Free Algorithm for the Large–Scale Trust–Region Subproblem

Marielba Rojas *   Sandra A. Santos †   Danny C. Sorensen ‡

September 8, 1999

Technical Report 175, Department of Informatics, University of Bergen, Norway.
Technical Report 99–19, Department of Computational and Applied Mathematics, Rice University, Houston.

Abstract

We present a matrix–free algorithm for the large–scale trust–region subproblem. Our algorithm relies on matrix–vector products only and does not require matrix factorizations. We recast the trust–region subproblem as a parameterized eigenvalue problem and compute an optimal value for the parameter. We then find the optimal solution of the trust–region subproblem from the eigenvectors associated with two of the smallest eigenvalues of the parameterized eigenvalue problem corresponding to the optimal parameter. The new algorithm uses a different interpolating scheme than existent methods and introduces a unified iteration that naturally includes the so-called hard case. We show that the new iteration is well defined and convergent at a superlinear rate. We present computational results to illustrate convergence properties and robustness of the method.

AMS classification: Primary 65F15; Secondary 65G05
Key words and phrases: Regularization, constrained quadratic optimization, trust region, Lanczos method.

---

*Department of Informatics, University of Bergen, 5020 Bergen, Norway (`mrojas@ii.uib.no`). On leave from Rice University, Houston. This author was supported in part by the Research Council of Norway, and by NSF cooperative agreement CCR-9120008.

†Department of Mathematics, State University of Campinas, CP 6065, 13081-970, Campinas, SP, Brazil (`sandra@ime.unicamp.br`) and visiting member of the Center for Research on Parallel Computation and the Department of Computational and Applied Mathematics, Rice University, P.O. BOX 1892, Houston, TX 77251-1892, USA. This author was supported by FAPESP (93/4907-5), CNPq, FINEP and FAEP-UNICAMP.

‡Dept. of Computational and Applied Mathematics, Rice University, 6100 Main St., Houston, TX 77005-1892, USA (`sorensen@caam.rice.edu`). This author was supported in part by NSF cooperative agreement CCR-9120008, and by ARPA contract number DAAL03-91-C-0047 (administered by the U.S. Army Research Office).

1 Introduction

An important problem in optimization and linear algebra is the *trust–region subproblem*: minimize a quadratic function subject to an ellipsoidal constraint,

$$\min \ \tfrac{1}{2}x^T Ax + g^T x \ \text{ subject to } \ \|x\|_2 \leq \Delta,$$

where $A \in \mathbb{R}^{n\times n}$, $A = A^T$; $x, g \in \mathbb{R}^n$ and $\Delta > 0$. Two significant applications of this basic problem are the regularization or smoothing of discrete forms of ill–posed problems and the trust–region globalization strategy used to force convergence in optimization methods.

A solution x_* to the problem must satisfy an equation of the form $(A + \mu I)x_* = -g$ with $\mu \geq 0$. The parameter μ is the Tikhonov regularization parameter for ill–posed problems and the Levenberg–Marquardt parameter in optimization. The constraint might also involve a matrix $C \neq I$ where C is often constructed to impose a smoothness condition on the solution x_* for ill–posed problems and to incorporate scaling of the variables in optimization. We will not treat this case explicitly here. However, in many applications the matrix C will be nonsingular and therefore with a change of variables we can reduce the problem to the case we are considering.

If we can afford to compute the Cholesky factorization of matrices of the form $A + \mu I$, then the method proposed by Moré and Sorensen (cf. [9]) is the method of choice to solve the problem. However, in many important applications, factoring or even forming these matrices is prohibitive. This has motivated the development of matrix–free methods that rely only on matrix–vector products. The first method in this class is the method of Steihaug [17] which computes the solution to the problem in a Krylov subspace. This method is very efficient in conjunction with optimization methods, however it does not compute an optimal solution and cannot handle a special situation known as the *hard case*, which we will describe later. New methods based on matrix–vector products are the ones by Golub and von Matt [3], Sorensen [16], Rendl and Wolkowicz [12] and Pham Dinh and Hoai An [10]. Recently, Lucidi, Palagi and Roma [7] presented new properties of the trust–region subproblem that provide useful tools for the development of new classes of algorithms for this problem in the large–scale context. As we were finishing this paper we became aware of a new method proposed by Hager [4] where an SQP approach is used to solve the trust–region subproblem.

Golub and von Matt [3] base their algorithm on the theory of Gauss quadrature and do not include in their analysis the possibility of the hard case. Pham Dinh and Hoai An [10] develop an algorithm based on difference of convex functions. Their strategy is very inexpensive due to its projective nature, but needs a restarting mechanism to ensure convergence to a global solution. The approaches of Sorensen [16] and Rendl and Wolkowicz [12] recast the trust–region subproblem as a parameterized eigenvalue problem and design an iteration to find an optimal value for the parameter. The idea of formulating the trust–region subproblem in terms of an eigenvalue problem is also exploited in Gander, Golub and von Matt [1]. Rendl and Wolkowicz present a primal–dual semidefinite framework for the trust–region subproblem, where a dual simplex–type method is used in the basic iteration and a primal simplex–type method provides steps for the hard–case iteration. At each iteration, the method computes the smallest eigenvalue and corresponding eigenvector of the parameterized problem using a block Lanczos routine. Sorensen's algorithm provides a superlinearly convergent scheme to adjust the parameter and finds the optimal vector x_* from the eigenvector of the parameterized problem, as long as the hard case does not occur. For the hard case, Sorensen's algorithm is linearly convergent. The algorithm uses the Implicitly Restarted Lanczos Method (IRLM) (cf. [15]) to compute the smallest eigenvalue and corresponding eigenvector of the parameterized problem. The IRLM is particularly suitable for large–scale applications since it has low and fixed storage requirements and relies only on matrix–vector products.

In this work we present a new matrix–free algorithm for the large–scale trust–region subproblem. Our algorithm is similar to those proposed in [12, 16] in the sense that we solve the trust–region subproblem through a parameterized eigenvalue problem, but it differs from those approaches in that we do not need two different schemes for the standard case and the hard case. Our algorithm can handle all the cases in the same basic iteration. We achieved this improvement over the methods in [12, 16], by computing two eigenpairs of the parameterized problem and incorporating information about the second eigenpair whenever it is appropriate. This strategy does not substantially increase the computational cost over the method in [16]. We introduce a two–point interpolating scheme that is different from the one in [16]. We show that this new iteration is also convergent and that the convergence rate is superlinear. Moreover, our convergence results naturally include the hard case, since no special iterations are necessary. Such a unified approach is not achieved in either [12] or [16].

The organization of this work is the following. In Section 2 we analyze the structure of the problem and motivate the algorithm. In Section 3 we give a complete characterization of the hard case with respect to the parameterized eigenproblems. We describe the algorithm in detail in Section 4. In Section 5 we present the convergence analysis. We describe preliminary numerical experiments in Section 6 and present some conclusions in Section 7.

2 Structure of the Problem

The problem we are interested in solving is

$$\begin{array}{ll} \min & \psi(x) \\ \text{s.t.} & \|x\| \leq \Delta \ , \end{array} \tag{1}$$

where $\psi(x) = \frac{1}{2}x^T Ax + g^T x$; A, g as before and $\|\cdot\| \equiv \|\cdot\|_2$ throughout the paper.

Due to the structure of (1), its optimality conditions are both necessary and sufficient, as stated in the next lemma, where we follow [16] in the nonstandard but notationally more convenient use of a nonpositive multiplier.

Lemma 2.1 ([14]) *A feasible vector x_* is a solution to (1) with corresponding Lagrange multiplier λ_* if and only if x_*, λ_* satisfy $(A - \lambda_* I)x_* = -g$ with $A - \lambda_* I$ positive semidefinite, $\lambda_* \leq 0$ and $\lambda_*(\Delta - \|x_*\|) = 0$.*

Proof. See [14]. □

In order to design efficient methods for solving problem (1) we must exploit the tremendous amount of structure of this problem. In particular, the optimality conditions are computationally attractive because they provide a means to reduce the given n–dimensional constrained optimization problem into a zero–finding problem in a single scalar variable. For example, we could define the function $\varphi(\lambda) = \|(A - \lambda I)^{-1} g\|$ and solve the *secular equation* $\varphi(\lambda) = \Delta$, monitoring λ to be no greater than the smallest eigenvalue of A, so that the Cholesky factorization of $A - \lambda I$ is well defined. Using Newton's method to solve $\dfrac{1}{\varphi(\lambda)} - \dfrac{1}{\Delta} = 0$ has a number of computationally attractive features (cf. [9]) and we should use this approach when we can afford to compute the Cholesky factorization of $A - \lambda I$. When computing a Cholesky factorization is too expensive, we need to use a different strategy. The introduction of a new parameter will make it possible to convert the original trust–region subproblem into a scalar problem that is suitable for the large–scale setting. The conversion amounts to embedding the given problem into a parameterized bordered matrix eigenvalue problem. Consider the *bordered* matrix

$$B_\alpha = \begin{pmatrix} \alpha & g^T \\ g & A \end{pmatrix}$$

and observe that

$$\frac{\alpha}{2} + \psi(x) = \frac{1}{2}(1, x^T) B_\alpha \begin{pmatrix} 1 \\ x \end{pmatrix}.$$

Therefore there exists a value of the parameter α such that we can rewrite problem (1) as

$$\begin{array}{ll} \min & \frac{1}{2} y^T B_\alpha y \\ \text{s.t.} & y^T y \leq 1 + \Delta^2, \;\; e_1^T y = 1, \end{array} \tag{2}$$

where e_1 is the first canonical unit vector in $\mathbb{R}^{n+1}$. This formulation suggests that we can find the desired solution in terms of an eigenpair of B_α in the following way. Suppose that $\{\lambda, (1, x^T)^T\}$ is an eigenpair of B_α. Then

$$\begin{pmatrix} \alpha & g^T \\ g & A \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix} = \begin{pmatrix} 1 \\ x \end{pmatrix} \lambda \, ,$$

which is equivalent to

$$\alpha - \lambda = -g^T x \tag{3}$$

and

$$(A - \lambda I)x = -g \, . \tag{4}$$

Now, let $\delta_1, \delta_2, \ldots, \delta_d$ be the *distinct* eigenvalues of A in nondecreasing order. Then

$$\alpha - \lambda = -g^T x = \sum_{j=1}^{d} \frac{\beta_j^2}{\delta_j - \lambda} \tag{5}$$

where β_j^2 is the sum of the squares of the expansion coefficients of g in the eigenvector basis, corresponding to all the eigenvectors associated with δ_j.

Observe that as a consequence of Cauchy's Interlace Theorem (cf. [11], p.186), and also from equation (5), the eigenvalues of A interlace the eigenvalues of B_α. In particular, if $\lambda_1(\alpha)$ is the smallest eigenvalue of B_α, then $\lambda_1(\alpha) \leq \delta_1$. This implies that the matrix $A - \lambda_1(\alpha)I$ is always positive semidefinite independently of the value of α. Moreover, $\lambda_1(\alpha)$ is usually well separated from the rest of the spectrum of B_α, particularly for small values of Δ. In these cases, we expect a Lanczos–type method to be very efficient in computing this eigenvalue and the corresponding eigenvector.

Equations (3)–(4) express λ and hence x implicitly in terms of α, suggesting the definition of a convenient function as follows. Let † denote the pseudoinverse of a matrix and let us define

$$\phi(\lambda) \;\; \equiv \;\; g^T (A - \lambda I)^\dagger g = -g^T x \, .$$

Therefore,

$$\phi'(\lambda) = g^T [(A - \lambda I)^\dagger]^2 g = x^T x \, ,$$

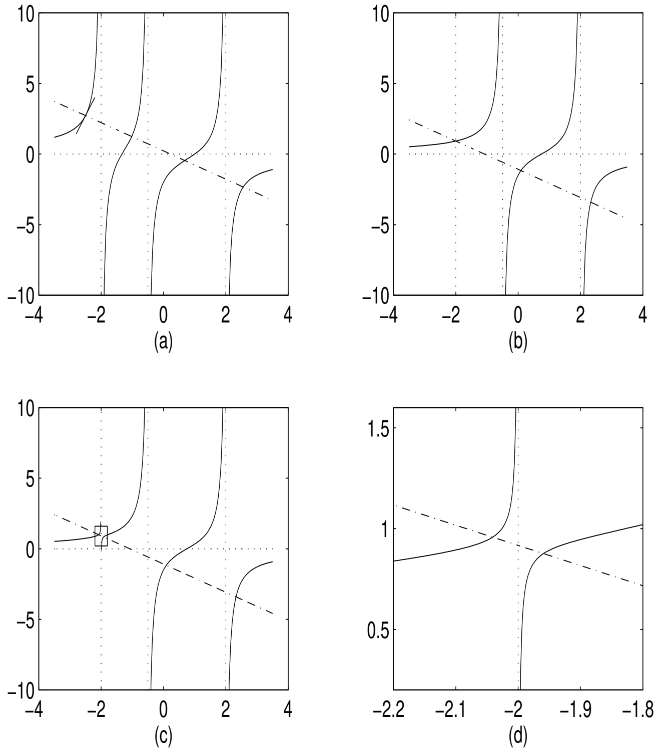

Figure 1: Example of the typical pattern of $\phi(\lambda)$ (solid) and the straight line $f(\lambda) = \alpha_* - \lambda$ (dashdotted). The three smallest eigenvalues of A are -2, -0.5 and 2. (a) general case with the slope at λ_* also plotted; (b) exact hard case; (c) near hard case; (d) detail of box in (c).

where differentiation is with respect to λ, and x satisfies $(A - \lambda I)x = -g$. The function ϕ appears in many contexts [2, 8, 18, 19] and Figure 1.a shows its typical behavior. It is worth noticing that the values of ϕ and ϕ' at an eigenvalue λ of B_α, are readily available and contain valuable information with respect to problem (1), as long as λ has a corresponding eigenvector with nonzero first component.

Finding the smallest eigenvalue and a corresponding eigenvector of B_α for a given value of α, and then normalizing the eigenvector to have its first component equal to one will provide a means to evaluate the rational function ϕ and its derivative at appropriate values of λ, namely, at $\lambda = \lambda_1(\alpha) \leq \delta_1$. If we can adjust α so that the corresponding x satisfies $x^T x = \phi'(\lambda) = \Delta^2$ with $\alpha - \lambda = \phi(\lambda)$, then

$$(A - \lambda I)x = -g \quad \text{and} \quad \lambda(\Delta - \|x\|) = 0$$

with $A - \lambda I$ positive semidefinite. If $\lambda \leq 0$ then x is a boundary solution for the trust–region subproblem. In case we find $\lambda > 0$ with $\|x\| < \Delta$ during the course of adjusting α, then this implies that the matrix A is positive definite and that

$\|A^{-1}g\| < \Delta$. As showed in [9], these two conditions imply that problem (1) has an interior solution that satisfies $Ax = -g$.

The availability of the values λ, $\phi(\lambda)$, $\phi'(\lambda)$ makes it possible to use rational interpolation to adjust the parameter using these values as interpolation points. The adjustment of α by means of rational interpolation, consists of constructing a rational interpolant $\widehat{\phi}$ and finding a special point $\widehat{\lambda}$ such that $\widehat{\phi}'(\widehat{\lambda}) = \Delta^2$. We then compute the new parameter as $\alpha_+ = \widehat{\lambda} + \widehat{\phi}(\widehat{\lambda})$. In this approach it is necessary to safeguard α_+ to ensure convergence of the iteration. This idea was discussed in [5, 14] and used in [16]. The algorithm in this paper follows this approach.

3 Characterization of the Hard Case

We assumed in the previous discussion that the smallest eigenvalue of B_α had a corresponding eigenvector with nonzero first component. It remains to consider the possibility that all the eigenvectors associated with $\lambda_1(\alpha)$ have first component zero so that we cannot normalize any of them to have its first component equal to one. In this case, the proposed strategy for solving problem (1) breaks down. However, this can happen only when g is orthogonal to $\mathcal{S}_1$, where $\mathcal{S}_j = \{q \mid Aq = \delta_j q\}$, $j = 1, 2, \ldots, d$.

The condition $g \perp \mathcal{S}_1$ is a necessary condition for the ocurrence of the so–called hard case. Therefore, we call this situation a potential hard case. Observe that in a potential hard case δ_1 is no longer a pole of ϕ as Figure 1.b illustrates. We discuss the hard case in detail at the end of this section. At this point we will concentrate on the potential hard case, which has intriguing consequences. We will show that in a potential hard case, for all values of α greater than certain critical value $\tilde{\alpha}_1$, all the eigenvectors corresponding to the smallest eigenvalue of B_α will have first component zero. We will also show that for any α, there is always a well defined eigenvector of B_α, depending continuously on α, that we can safely normalize to have first component one. If $g \not\perp \mathcal{S}_1$ or $g \perp \mathcal{S}_1$ and $\alpha \leq \tilde{\alpha}_1$, then this eigenvector corresponds to $\lambda_1(\alpha)$. If $g \perp \mathcal{S}_1$ and α exceeds the critical value $\tilde{\alpha}_1$ by a small amount, this parameterized vector corresponds to the second smallest eigenvalue of B_α. A complete understanding of this case leads to the main algorithm of this paper. The following results are the bases for this understanding.

Lemma 3.1 *For any $\alpha \in \mathbb{R}$ and any $q \in \mathcal{S}_j$, $1 \leq j \leq d$, $\{\delta_j, (0, q^T)^T\}$ is an eigenpair of B_α if and only if g is orthogonal to $\mathcal{S}_j$.*

Proof. The proof follows from the observation that $g \perp \mathcal{S}_j$ and $Aq = \delta_j q$ are equivalent to

$$\begin{pmatrix} \alpha & g^T \\ g & A \end{pmatrix} \begin{pmatrix} 0 \\ q \end{pmatrix} = \delta_j \begin{pmatrix} 0 \\ q \end{pmatrix}.$$

□

If $\mathcal{Z}_1(\alpha)$ is the eigenspace of B_α corresponding to δ_1, Lemma 3.1 establishes that the set $\{(0, q^T)^T \mid q \in \mathcal{S}_1\}$ is a subset of $\mathcal{Z}_1(\alpha)$. Note that while $\mathcal{S}_1$ corresponds to the smallest eigenvalue of A, $\mathcal{Z}_1(\alpha)$ does not necessarily correspond to the smallest eigenvalue of B_α. These subspaces have the same dimension for all but one exceptional value of α. The following result states that there is a unique value of α for which $\dim \mathcal{Z}_1(\alpha) = \dim \mathcal{S}_1 + 1$.

Lemma 3.2 *Suppose that g is orthogonal to $\mathcal{S}_j$, $1 \le j \le d$, and let $p_j = -(A - \delta_j I)^\dagger g$. The pair $\{\delta_j, (1, p_j^T)^T\}$ is an eigenpair of B_α if and only if $\alpha = \tilde{\alpha}_j$ where $\tilde{\alpha}_j = \delta_j - g^T p_j$.*

Proof. First we observe that $g \perp \mathcal{S}_j$ implies that $g \in \mathcal{R}(A - \delta_j I)$ and therefore

$$(A - \delta_j I)p_j = -(A - \delta_j I)(A - \delta_j I)^\dagger g = -g, \tag{6}$$

since $(A - \delta_j I)(A - \delta_j I)^\dagger$ is an orthogonal projector onto $\mathcal{R}(A - \delta_j I)$.
Now, let $\alpha = \tilde{\alpha}_j$. Then

$$\begin{pmatrix} \tilde{\alpha}_j & g^T \\ g & A \end{pmatrix} \begin{pmatrix} 1 \\ p_j \end{pmatrix} = \begin{pmatrix} \tilde{\alpha}_j + g^T p_j \\ g + A p_j \end{pmatrix} = \delta_j \begin{pmatrix} 1 \\ p_j \end{pmatrix},$$

since by definition of $\tilde{\alpha}_j$ we have $\tilde{\alpha}_j + g^T p_j = \delta_j$ and by (6), $g + A p_j = \delta_j p_j$.
Suppose now that $\{\delta_j, (1, p_j^T)^T\}$ is an eigenpair of B_α, i.e.

$$\begin{pmatrix} \alpha & g^T \\ g & A \end{pmatrix} \begin{pmatrix} 1 \\ p_j \end{pmatrix} = \delta_j \begin{pmatrix} 1 \\ p_j \end{pmatrix}.$$

It follows directly from this relationship that $\alpha = \tilde{\alpha}_j = \delta_j - g^T p_j$. □

The following corollary summarizes the main results from Lemmas 3.1 and 3.2.

Corollary 3.1 *Suppose that g is orthogonal to $\mathcal{S}_j$, $1 \le j \le d$, and let $\mathcal{Z}_j(\alpha) = \{z \in \mathbb{R}^{n+1} \mid B_\alpha z = \delta_j z\}$. If $\tilde{\alpha}_j = \delta_j + g^T p_j$ with $p_j = -(A - \delta_j I)^\dagger g$ then $\dim \mathcal{Z}_j(\tilde{\alpha}_j) = \dim \mathcal{S}_j + 1$ and for any other value of α, $\dim \mathcal{Z}_j(\alpha) = \dim \mathcal{S}_j$. Moreover, if m_j is the multiplicity of δ_j and $\{q_1, \ldots, q_{m_j}\}$ is an orthogonal basis for $\mathcal{S}_j$ then*

$$\left\{ \begin{pmatrix} 1 \\ p_j \end{pmatrix}, \begin{pmatrix} 0 \\ q_1 \end{pmatrix}, \ldots, \begin{pmatrix} 0 \\ q_{m_j} \end{pmatrix} \right\}$$

is an orthogonal basis for $\mathcal{Z}_j(\tilde{\alpha}_j)$ and

$$\left\{ \begin{pmatrix} 0 \\ q_1 \end{pmatrix}, \ldots, \begin{pmatrix} 0 \\ q_{m_j} \end{pmatrix} \right\}$$

is an orthogonal basis for $\mathcal{Z}_j(\alpha)$, for $\alpha \ne \tilde{\alpha}_j$.

The result in Lemma 3.1 was also stated in [16], the idea behind Lemma 3.2 was presented in [12]. We present here a general formulation of these results given in [13]. In the next results from [13], we establish that there always exists an eigenvector of B_α that we can normalize to have first component equal to one, and we characterize the eigenvalue to which this eigenvector corresponds.

Theorem 3.1 ([13]) *Let $\lambda(\alpha)$ be the smallest solution of the equation*

$$\phi(\lambda) = \alpha - \lambda\ .$$

Then, for any value of α, $\lambda(\alpha)$ is an eigenvalue of B_α with a corresponding eigenvector that can be normalized to have first component one.

Proof. Suppose first that g is orthogonal to $\mathcal{S}_i$, $i = 1, 2, \ldots, \ell$ with $1 \le \ell < d$. Then

$$\begin{aligned} \phi(\lambda) &= g^T (A - \lambda I)^\dagger g \\ &= \sum_{j=\ell+1}^{d} \frac{\beta_j^2}{\delta_j - \lambda}. \end{aligned}$$

Let $\lambda(\alpha)$ be the smallest solution of the equation $\phi(\lambda) = \alpha - \lambda$. Then $\lambda(\alpha) \in (-\infty, \delta_{\ell+1})$. Since $\phi(\lambda)$ is strictly increasing on its domain and $f(\lambda) = \alpha - \lambda$ is a decreasing straight line, we conclude that $\lambda(\alpha)$ is unique. Since $\lambda(\alpha)$ depends continuously on α, so does $p(\alpha) = -(A - \lambda(\alpha)I)^\dagger g$ and also $v(\alpha) = (1, p(\alpha)^T)^T$. Let us see now that $v(\alpha)$ is an eigenvector of B_α associated with $\lambda(\alpha)$. Consider

$$\begin{pmatrix} \alpha & g^T \\ g & A \end{pmatrix} \begin{pmatrix} 1 \\ p(\alpha) \end{pmatrix} = \begin{pmatrix} \alpha + g^T p(\alpha) \\ g + Ap(\alpha) \end{pmatrix}$$

and note that

$$\begin{aligned} \alpha + g^T p(\alpha) &= \alpha - \phi(\lambda(\alpha)) \\ &= \lambda(\alpha), \quad \text{by definition of } \lambda(\alpha). \end{aligned}$$

Now, $g \perp \mathcal{S}_i,\ i = 1, 2, \ldots, \ell$ implies that $g \in \mathcal{R}(A - \lambda I)$ for $\lambda \in (-\infty, \delta_{\ell+1})$. Thus, $g \in \mathcal{R}(A - \lambda(\alpha)I)$ and we have $(A - \lambda(\alpha)I)p(\alpha) = -g$. It follows that

$$g + Ap(\alpha) = \lambda(\alpha)p(\alpha)$$

and therefore, $B_\alpha v(\alpha) = \lambda(\alpha)v(\alpha)$.
Suppose now that g is not orthogonal to $\mathcal{S}_1$. Then $\lambda(\alpha) \in (-\infty, \delta_1)$ and this implies $A - \lambda(\alpha)I$ is nonsingular and the previous proof holds with $p(\alpha) = -(A - \lambda(\alpha)I)^{-1} g$.
□

The following result characterizes the smallest $\ell+1$ distinct eigenvalues of B_α if g is orthogonal to the eigenspaces corresponding to the smallest ℓ distinct eigenvalues of A. In case g is not orthogonal to $\mathcal{S}_1$ then the lemma characterizes the smallest eigenvalue of B_α. We will denote by $\lambda_j(\alpha)$, $j = 1, 2, \ldots, n+1$ the eigenvalues of B_α in nondecreasing order.

Lemma 3.3 ([13]) *Let $\{\lambda(\alpha),\ v(\alpha)\}$ be the eigenpair of B_α given by Theorem 3.1 and define $\tilde{\alpha}_i$ as in Lemma 3.2. Then, if $g \not\perp \mathcal{S}_1$ then $\lambda_1(\alpha) = \lambda(\alpha)$.*
If $g \perp \mathcal{S}_k$, for $k = 1, 2, \ldots, \ell$ and $1 \le \ell < d$, then

(i) If $\alpha = \tilde{\alpha}_i$, $1 \le i \le \ell$ then $\lambda_j(\alpha) = \delta_j$, $j = 1, 2, \ldots, \ell$. In this case, $\lambda_{\ell+1}(\alpha)$ is the second smallest root of equation $\phi(\lambda) = \alpha - \lambda$.

(ii) If $\alpha < \tilde{\alpha}_1$ then $\lambda_1(\alpha) = \lambda(\alpha)$ and $\lambda_j(\alpha) = \delta_{j-1}$, $j = 2, \ldots, \ell + 1$.

(iii) If $\tilde{\alpha}_{i-1} < \alpha < \tilde{\alpha}_i$, $2 \le i \le \ell$ then $\lambda_i(\alpha) = \lambda(\alpha)$, $\lambda_j(\alpha) = \delta_j$ for $j = 1, \ldots, i-1$, and $\lambda_j(\alpha) = \delta_{j-1}$ for $j = i+1, \ldots, \ell+1$.

(iv) If $\alpha > \tilde{\alpha}_\ell$ then $\lambda_j(\alpha) = \delta_j$, $j = 1, 2, \ldots, \ell$ and $\lambda_{\ell+1}(\alpha) = \lambda(\alpha)$.

Proof. These results are a direct consequence of Cauchy's Interlace Theorem, Lemmas 3.1 and 3.2, and the properties of the functions $\phi(\lambda)$ and $\alpha - \lambda$. □

We can expect difficulties in practice when the vector g is nearly orthogonal to the eigenspace $\mathcal{S}_1$. If this happens, there still exists $\lambda_* < \delta_1$ and x_* such that $(A - \lambda_* I)x_* = -g$, $\|x_*\| = \Delta$, with λ_* quite close to δ_1. We call this situation a *near hard case* and Figure 1.c illustrates it. In the detail shown in Figure 1.d, we can see that in this case, the derivative ϕ' changes rapidly for λ close to δ_1, so the problem of finding λ_* satisfying the correct slope $\phi'(\lambda_*) = \Delta^2$ is very ill-conditioned.

In the remainder of the section we discuss the hard case and present the results that allow us to compute a nearly optimal solution for the trust–region subproblem in this situation. The hard case can only occur when $g \perp \mathcal{S}_1$, the matrix A is indefinite or positive semidefinite and singular, and for certain values of Δ. This case was analyzed in [9] for medium–scale problems and discussed in [12, 16] in the large–scale context. The precise statement is the following.

Lemma 3.4 ([14]) Assume g is orthogonal to $\mathcal{S}_1$ and let $p = -(A - \delta_1 I)^\dagger g$. If $\delta_1 \le 0$ and $\|p\| < \Delta$, then the solutions of (1) consist of the set $\{x \mid x = p + z\ ,\ z \in \mathcal{S}_1\ ,\ \|x\| = \Delta\}$.

Proof. See [14]. □

As we can see, it is precisely in the hard case that in the process of adjusting α we will compute values such that $\alpha > \tilde{\alpha}_1$. As Lemma 3.3 establishes, in this case all the eigenvectors corresponding to the smallest eigenvalue of B_α have first component zero. Moreover, in a near hard case the eigenvectors will have very small first components and dividing by these values will introduce large roundoff errors. Theorem 3.1 and Lemma 3.3 suggest a strategy for handling this situation, namely using the eigenvector of B_α with the desired structure guaranteed by Theorem 3.1

and the corresponding eigenvalue to obtain the interpolation points, so we can proceed with the adjustment of the parameter α. We will need a safeguarding strategy to enforce convergence of this iteration. We will describe this strategy in the next section where we present the algorithm in detail.

The following results provide the theoretical bases for declaring convergence in the hard case. Results within the same philosophy are presented in [9, 16]. The idea behind these results is to exploit the information available at each iteration and, with practically no additional cost, detect a nearly optimal solution in the hard case or near hard case. Theorem 3.2, Lemma 3.5 and Lemma 3.6 contain these results. Theorem 3.2 establishes that, under certain conditions, the last n components of a special linear combination of eigenvectors of B_α, form a nearly optimal solution for problem (1). Lemma 3.5 establishes the conditions under which we can compute the special linear combination and Lemma 3.6 shows how to compute it. Theorem 3.2 follows from a more general result from [13] but we present a different proof here. Lemma 3.5 is a reformulation of a result from [13] and Lemma 3.6 is from [13].

Theorem 3.2 ([13]) *Let $\lambda_1(\alpha)$ be the smallest eigenvalue of B_α with a corresponding eigenvector $z_1 = (\nu_1, \tilde{z}_1^T)^T$. Let $\lambda_i(\alpha)$ be* any *of the remaining n eigenvalues of B_α with a corresponding eigenvector $z_i = (\nu_i, \tilde{z}_i^T)^T$. Define $Z = [z_1 \;\; z_i]$, $\widetilde{Z} = [\tilde{z}_1 \;\; \tilde{z}_i]$, and assume $Z^T Z = I$. Let $\eta > 0$.*

If there exists $t = (\tau_1, \tau_2)^T$, with $\|t\| = 1$ such that

(i) $(e_1^T Z t)^2 = \dfrac{1}{1+\Delta^2}$, *and*

(ii) $(\lambda_i(\alpha) - \lambda_1(\alpha))\; \tau_2^2\; (1+\Delta^2) \le -2\eta\psi(\tilde{x})$, *for* $\tilde{x} = \dfrac{\widetilde{Z}t}{e_1^T Z t}$

then

$$\psi(x_*) \le \psi(\tilde{x}) \le \frac{1}{1+\eta}\psi(x_*)$$

where x_* is a boundary solution for problem (1) with $\psi(x_*) \le 0$.

Proof. Since x_* is a boundary solution of (1), we have $\psi(x_*) \le \psi(x)$, $\forall\; x \in \mathbb{R}^n$ such that $\|x\| = \Delta$. Therefore, in order to prove that $\psi(x_*) \le \psi(\tilde{x})$, it will suffice to show that $\|\tilde{x}\| = \Delta$.

Note that $\dfrac{Zt}{e_1^T Z t} = (1, \tilde{x}^T)^T$ and therefore

$$\begin{aligned} \|(1, \tilde{x}^T)\|^2 = 1 + \|\tilde{x}\|^2 &= \left\| \frac{Zt}{e_1^T Z t} \right\|^2 \\ &= \frac{1}{(e_1^T Z t)^2} \end{aligned}$$

since $\|t\| = 1$ and $Z^T Z = I$ by hypothesis. Thus, by *(i)*

$$1 + \|\tilde{x}\|^2 = 1 + \Delta^2.$$

This implies $\|\tilde{x}\| = \Delta$ and therefore, $\psi(x_*) \leq \psi(\tilde{x})$.
Let us now prove the other part of the inequality.
Observe that $\alpha + 2\psi(x_*) = (1, x_*^T)\ B_\alpha\ (1, x_*^T)^T$. Thus, by Rayleigh quotient properties

$$\alpha + 2\psi(x_*) \quad \geq \quad \lambda_1(\alpha)\|(1, x_*^T)^T\|^2.$$

Since $\|x_*\| = \Delta$ it follows that $\|(1, x_*^T)^T\|^2 = 1 + \Delta^2$, and therefore

$$\alpha + 2\psi(x_*) \geq \lambda_1(\alpha)(1 + \Delta^2). \tag{7}$$

Now observe that $\alpha + 2\psi(\tilde{x}) = (1, \tilde{x}^T)\ B_\alpha\ (1, \tilde{x}^T)^T$, and since $(1, \tilde{x}^T)^T = \dfrac{1}{e_1^T Z t} Z t$, it follows that

$$\begin{aligned} \alpha + 2\psi(\tilde{x}) &= t^T Z^T B_\alpha Z t \frac{1}{(e_1^T Z t)^2} \\ &= [\lambda_1(\alpha)\tau_1^2 + \lambda_i(\alpha)\tau_2^2]\ (1 + \Delta^2), \end{aligned}$$

by *(i)* and the fact that $z_1,\ z_i$ are eigenvectors of B_α. Since $\tau_1^2 + \tau_2^2 = 1$, we have

$$\begin{aligned} \alpha + 2\psi(\tilde{x}) &= [\lambda_1(\alpha)(1 - \tau_2^2) + \lambda_i(\alpha)\tau_2^2]\ (1 + \Delta^2) \\ &= [\lambda_1(\alpha) + (\lambda_i(\alpha) - \lambda_1(\alpha))\tau_2^2]\ (1 + \Delta^2) \end{aligned}$$

and therefore

$$\begin{aligned} \alpha + 2\psi(\tilde{x}) - (\lambda_i(\alpha) - \lambda_1(\alpha))\tau_2^2(1 + \Delta^2) &= \lambda_1(\alpha)\ (1 + \Delta^2) \\ &\leq \alpha + 2\psi(x_*), \quad \text{by (7)}. \end{aligned}$$

If $(\lambda_i(\alpha) - \lambda_1(\alpha))\tau_2^2(1 + \Delta^2) \leq -2\eta\psi(\tilde{x})$, then

$$\alpha + 2\psi(\tilde{x}) + 2\eta\psi(\tilde{x}) \quad \leq \quad \alpha + 2\psi(x_*)$$

and we can conclude $\psi(\tilde{x}) \leq \dfrac{1}{1 + \eta}\psi(x_*)$.
Therefore $\psi(x_*) \leq \psi(\tilde{x}) \leq \frac{1}{1+\eta}\psi(x_*)$ as claimed. □

It follows directly from this result that

$$0 \leq \psi(\tilde{x}) - \psi(x_*) \leq -\frac{\eta}{1 + \eta}\psi(x_*)$$

$$|\psi(\tilde{x}) - \psi(x_*)| \leq \frac{\eta}{1 + \eta}|\psi(x_*)|\ . \tag{8}$$

The inequality (8) implies that under the conditions of Theorem 3.2, $\psi(\tilde{x})$ will be arbitrarily close to $\psi(x_*)$. We will call such $\tilde{x}$ a quasi–optimal solution for problem (1).
The next result establishes conditions for computing the vector t in Theorem 3.2.

Lemma 3.5 ([13]) *Let $z_i = (\nu_i, \tilde{z}_i^T)^T$, with $\nu_i \in \mathbb{R}$, $\tilde{z}_i \in \mathbb{R}^n$ for $i = 1, 2$. Define the matrices $Z = [z_1 \;\; z_2]$ and $\widetilde{Z} = [\tilde{z}_1 \;\; \tilde{z}_2]$, and assume $Z^T Z = I$. If $\|Z^T e_1\|^2 \geq \dfrac{1}{\beta}$, for $\beta > 0$ then there exists $t \in \mathbb{R}^2$ with $t \neq 0$ that satisfies*

$$\|Zt\|^2 = \beta (e_1^T Z t)^2. \tag{9}$$

Proof. Observe that we can rewrite (9) as

$$\begin{aligned} t^T Z^T Z t &= \beta \, (e_1^T Z t)^2 \\ &= \beta \, (t^T Z^T e_1 e_1^T Z t) \end{aligned}$$

which is equivalent to

$$t^T \, [I - \beta \, Z^T e_1 e_1^T Z] \, t = 0 \tag{10}$$

since $Z^T Z = I$ by hypothesis. Equation (10) will have a nontrivial solution only if the matrix $M = I - \beta \, Z^T e_1 e_1^T Z$ is indefinite or positive semidefinite and singular. So, let us study the eigenvalues of M. The two eigenpairs of the matrix $M = I - \beta \, Z^T e_1 e_1^T Z$ are given by

$$\{1 - \beta \; e_1^T Z Z^T e_1, \; Z^T e_1\} \quad \text{and} \quad \{1, v\} \quad \text{with } v \perp Z^T e_1 \, .$$

Therefore, equation (10) will have nontrivial solutions if $\mu_1 = 1 - \beta \; e_1^T Z Z^T e_1 \leq 0$. In other words, if $\|Z^T e_1\|^2 = e_1^T Z Z^T e_1 \geq \dfrac{1}{\beta}$ then there exists $t \in \mathbb{R}^2$ with $t \neq 0$ such that t satisfies (10). □

Note that choosing $\beta = 1 + \Delta^2$ in Lemma 3.5 and normalizing t to have $\|t\| = 1$, will give a vector that satisfies the conditions of Theorem 3.2. The following lemma provides a way of computing such vector.

Lemma 3.6 ([13]) *Let $\beta \in \mathbb{R}$, $\beta > 0$ and let $z \in \mathbb{R}^n$. The equation*

$$t^T [I - \beta z z^T] t = 0 \tag{11}$$

in t with $t \in \mathbb{R}^n$, has $2(n-1)$ nontrivial solutions if the matrix $M = I - \beta z z^T$ is indefinite and one nontrivial solution if M is positive semidefinite and singular.

Proof. Let $P \in \mathbb{R}^{n \times n}$ be such that $P^T z = \|z\| e_1$ with $P^T P = I$ and apply this orthogonal transformation to the matrix M to obtain

$$P^T [I - \beta z z^T] P = I - \beta \|z\|^2 e_1 e_1^T.$$

Therefore the solutions of equation (11) in this new basis are the solutions of

$$y^T \begin{pmatrix} -\theta & 0 \\ 0 & I \end{pmatrix} y = 0$$

where $\theta = -1 + \beta \|z\|^2 e_1 e_1^T$.
The nontrivial solutions of (11) are then given by $t = Py$ where

(1) $y = (1, \sqrt{\theta} e_i^T)^T$ and $y = (-1, \sqrt{\theta} e_i^T)^T$ with e_i the i–th canonical vector in $\mathbb{R}^{n-1}$, $i = 1, 2, \ldots, n-1$, if M is indefinite, i.e. if $\theta > 0$, or

(2) $y = e_1$, if M is positive semidefinite and singular, i.e. if $\theta = 0$.

Therefore equation (11) has $2(n-1)$ nontrivial solutions if M is indefinite and one nontrivial solution if M is positive semidefinite and singular. □

Remark: Suppose $n = 2$ and $z = (\nu_1, \nu_i)^T$ in Lemma 3.6. Then if ${\nu_1}^2 + {\nu_i}^2 > \frac{1}{\beta}$, the vector $t = (\tau_1, \tau_2)^T$ is given by

$$\tau_1 = \frac{\nu_1 - \nu_i\sqrt{\beta({\nu_1}^2 + {\nu_i}^2) - 1}}{({\nu_1}^2 + {\nu_i}^2)\sqrt{\beta}}, \quad \tau_2 = \frac{\nu_1 + \nu_i\sqrt{\beta({\nu_1}^2 + {\nu_i}^2) - 1}}{({\nu_1}^2 + {\nu_i}^2)\sqrt{\beta}}$$

or

$$\tau_1 = \frac{\nu_i + \nu_1\sqrt{\beta({\nu_1}^2 + {\nu_i}^2) - 1}}{({\nu_1}^2 + {\nu_i}^2)\sqrt{\beta}}, \quad \tau_2 = \frac{\nu_1 - \nu_i\sqrt{\beta({\nu_1}^2 + {\nu_i}^2) - 1}}{({\nu_1}^2 + {\nu_i}^2)\sqrt{\beta}} \; .$$

If ${\nu_1}^2 + {\nu_i}^2 = \frac{1}{\beta}$ then t is given by

$$\tau_1 = \frac{\nu_1}{\sqrt{{\nu_1}^2 + {\nu_i}^2}}, \quad \tau_2 = \frac{\nu_i}{\sqrt{{\nu_1}^2 + {\nu_i}^2}} \; .$$

The previous results are the bases for the algorithm in next section, since they provide the necessary tools for handling the hard case in the same iteration designed for the standard case and for computing a solution in this case.

4 The Algorithm

Keeping in mind the availability of a well–suited variant of the Lanczos method, namely the *Implicitly Restarted Lanczos Method* (cf. [15]), we will develop a rapidly convergent iteration to adjust α based on this process. Our goal is to adjust α so that

$$\alpha - \lambda = \phi(\lambda) \quad , \quad \phi'(\lambda) = \Delta^2 \; ,$$

where

$$\phi(\lambda) = -g^T x \quad , \quad \phi'(\lambda) = x^T x \ ,$$

with $(A - \lambda I)x = -g$.

The approach of this work is similar to the one in [16] in the following sense. We compute a function $\widehat{\phi}$ which interpolates ϕ and ϕ' at two properly chosen points. Then, from the interpolating function $\widehat{\phi}$ we determine $\widehat{\lambda}$ satisfying

$$\widehat{\phi}'(\widehat{\lambda}) = \Delta^2 \ . \tag{12}$$

Finally, we use $\widehat{\lambda}$ and $\phi(\widehat{\lambda})$ to update the parameter α and compute the next iterates $\{\lambda, x\}$. The new elements in our algorithm are the introduction of safeguards for the sequence in α, the use of the information relative to the second smallest eigenvalue of the matrix B_α and the introduction of a different interpolating scheme, where the currently available information is exploited to a greater extent. Considering that the interpretation of the primal feasibility equations of [12] can be related to (12), the description of our algorithm has also some flavor of the approach in [12], where an inverse interpolation scheme is used to satisfy primal feasibility. However, in the presence of the hard case, we do not need to combine distinct interpolating functions, as in [12] nor switch to another algorithm as in [16]. In this section we will assume that the vector g is nonzero. If $g = 0$ then problem (1) reduces to solving an eigenvalue problem for the smallest eigenvalue of A. We shall first describe the components of the algorithm and then present the complete method.

4.1 Interpolating Schemes

To begin the iteration, we need a single–point interpolating scheme. We use the approach derived in [16] which gives the following expression for α_1.

$$\alpha_1 = \widehat{\lambda} + \widehat{\phi}(\widehat{\lambda}) = \alpha_0 + \frac{\alpha_0 - \lambda_0}{\|x_0\|} \left(\frac{\Delta - \|x_0\|}{\Delta} \right) \left(\Delta + \frac{1}{\|x_0\|} \right) \tag{13}$$

where

$$\widehat{\lambda} = \delta + \frac{g^T x_0}{\|x_0\| \Delta} \ .$$

This method is linearly convergent and may be slow in some cases, so we will use it just to obtain a second pair of iterates, which together with λ_0, x_0 will be the starting values for a two–point method. In the two–point method we use the four pieces of information available at the k-th iteration, namely $\phi(\lambda_{k-1})$, $\phi'(\lambda_{k-1})$, $\phi(\lambda_k)$ and $\phi'(\lambda_k)$ as follows. We compute $\widehat{\lambda}$ such that

$$\frac{1}{\Delta} = \frac{1}{\sqrt{\phi'(\lambda_{k-1})}}\left(\frac{\lambda_k - \widehat{\lambda}}{\lambda_k - \lambda_{k-1}}\right) + \frac{1}{\sqrt{\phi'(\lambda_k)}}\left(\frac{\widehat{\lambda} - \lambda_{k-1}}{\lambda_k - \lambda_{k-1}}\right) \; , \tag{14}$$

obtaining

$$\widehat{\lambda} = \frac{\lambda_{k-1}\|x_{k-1}\|(\|x_k\| - \Delta) + \lambda_k\|x_k\|(\Delta - \|x_{k-1}\|)}{\Delta(\|x_k\| - \|x_{k-1}\|)} \; . \tag{15}$$

This is equivalent to defining

$$\widehat{\phi}(\lambda) = \frac{\gamma^2}{\delta - \lambda} + \eta \tag{16}$$

for any η and computing $\widehat{\lambda}$ such that $\dfrac{1}{\sqrt{\widehat{\phi}'(\widehat{\lambda})}} = \dfrac{1}{\Delta}$. It is easy to verify using (14) that

$$\gamma^2 = \frac{(\lambda_k - \lambda_{k-1})^2\|x_{k-1}\|^2\|x_k\|^2}{(\|x_k\| - \|x_{k-1}\|)^2} \qquad \text{and} \qquad \delta = \frac{\lambda_k\|x_k\| - \lambda_{k-1}\|x_{k-1}\|}{\|x_k\| - \|x_{k-1}\|} \; .$$

Ideally, $\eta = \phi(\widehat{\lambda}) - \dfrac{\gamma^2}{\delta - \widehat{\lambda}}$, where $\phi(\widehat{\lambda})$ is the value we are going to estimate in order to update α. Using the values $\phi(\lambda_{k-1})$ and $\phi(\lambda_k)$, we first define $\eta_j = \phi(\lambda_j) - \dfrac{\gamma^2}{\delta - \lambda_j}$, for $j = k-1,\ k$. Then, applying the linear interpolation philosophy on λ_j, η_j, and defining the weights by means of the already computed value $\widehat{\lambda}$, we choose

$$\eta = \left(\frac{\lambda_k - \widehat{\lambda}}{\lambda_k - \lambda_{k-1}}\right)\eta_{k-1} + \left(\frac{\widehat{\lambda} - \lambda_{k-1}}{\lambda_k - \lambda_{k-1}}\right)\eta_k \; .$$

After some manipulation we can express the updating formula for α as

$$\begin{aligned}
\alpha_{k+1} &= \widehat{\lambda} + \omega\phi(\lambda_{k-1}) + (1-\omega)\phi(\lambda_k) \\
&+ \frac{\|x_{k-1}\|\|x_k\|(\|x_k\| - \|x_{k-1}\|)}{\omega\|x_k\| + (1-\omega)\|x_{k-1}\|}\frac{(\lambda_{k-1} - \widehat{\lambda})(\lambda_k - \widehat{\lambda})}{(\lambda_k - \lambda_{k-1})} \\
&= \omega\alpha_{k-1} + (1-\omega)\alpha_k \\
&+ \frac{\|x_{k-1}\|\|x_k\|(\|x_k\| - \|x_{k-1}\|)}{\omega\|x_k\| + (1-\omega)\|x_{k-1}\|}\frac{(\lambda_{k-1} - \widehat{\lambda})(\lambda_k - \widehat{\lambda})}{(\lambda_k - \lambda_{k-1})} \; ,
\end{aligned} \tag{17}$$

where $\omega = \dfrac{\lambda_k - \widehat{\lambda}}{\lambda_k - \lambda_{k-1}}$, $\alpha_{k-1} = \lambda_{k-1} + \phi(\lambda_{k-1})$ and $\alpha_k = \lambda_k + \phi(\lambda_k)$.

As we discussed in Section 3, we need a special strategy to obtain interpolation points in potential hard cases. We describe this strategy in §4.2.

4.2 Choice of Interpolation Points

According to Lemma 3.1, if the first component of the eigenvector corresponding to the smallest eigenvalue of B_{α_k} is zero, this will indicate a potential hard case and we will have $\lambda_1(\alpha_k) = \delta_1$. However, Lemma 3.3 establishes that for α_k slightly larger than $\tilde{\alpha}_1$ there is an eigenvector with significant first component that corresponds to the *second* smallest eigenvalue of B_α. Therefore, we propose to use an eigenpair corresponding to an eigenvalue that is close to the second smallest eigenvalue of the bordered matrix to obtain the interpolation point whenever we detect a potential hard case. As we shall explain, not only can we keep the size of the iterate x_k under control, but we can also ensure convergence of $\{\lambda_k, x_k\}$ to $\{\delta_1, p\}$ by driving the parameter α_k to the value $\tilde{\alpha}_1$ given by Lemma 3.2. Recall, Lemma 3.2 established that there will be an eigenvector with significant first component corresponding to $\lambda_1(\alpha_k)$, precisely when α_k assumes the special value $\tilde{\alpha}_1 = \delta_1 - g^T p$. Moreover, the use of this second eigenvector prevents numerical difficulties in a near hard–case situation.

There is an easy way to detect a potential hard case during an iteration. Let $(\nu_1, u_1^T)^T$ be a unitary eigenvector of B_{α_k} corresponding to $\lambda_1(\alpha_k)$. Then, we declare ν_1 to be "small", indicating a near hard case has been detected, if the condition $\|g\||\nu_1| \le \varepsilon\sqrt{1-\nu_1{}^2}$ holds for a given $\varepsilon \in (0,1)$. This is motivated as follows. Since $(A - \lambda_1(\alpha_k)I)u_1 = -g\nu_1$, we have

$$\frac{\|(A-\lambda_1(\alpha_k)I)u_1\|}{\|u_1\|} = \frac{\|g\||\nu_1|}{\sqrt{1-\nu_1{}^2}}$$

and hence $\|g\||\nu_1| \le \varepsilon\sqrt{1-\nu_1{}^2}$ assures that $\|(A-\lambda_1(\alpha_k)I)u_1\| \le \varepsilon\|u_1\|$. In other words, $\{\lambda_1(\alpha_k), u_1\}$ is an approximate eigenpair of A and the eigenvector $(\nu_1, u_1^T)^T$ from the bordered matrix is essentially impossible to normalize. This is approximately the situation described in Lemma 3.1. Of course, this test can be made scale independent by choosing $\varepsilon = \hat{\varepsilon}\|A\|$, for $\hat{\varepsilon} \in (0,1)$.

When a near hard case has been detected, we need an alternative way to define the pair $\{\lambda_k, x_k\}$. At each iteration, at essentially no extra cost, we compute an eigenpair corresponding to the smallest eigenvalue of B_{α_k}, which we denote by $\{\lambda_1(\alpha_k), (\nu_1, u_1^T)^T\}$, and also an eigenpair corresponding to an eigenvalue close to the second smallest eigenvalue of B_α, which we denote by $\{\lambda_i(\alpha_k), (\nu_2, u_2^T)^T\}$. If both $|\nu_1|$ and $|\nu_2|$ are small, that is, if $\|g\||\nu_1| \le \varepsilon\sqrt{1-\nu_1{}^2}$ and $\|g\||\nu_2| \le \varepsilon\sqrt{1-\nu_2{}^2}$ then we decrease the parameter α_k. According to Theorem 3.1 there always exists an eigenvector of the bordered matrix with significant first component for any value of α and as we mentioned before, according to Lemma 3.3, as α_k approaches the critical value, this normalizable eigenvector will correspond either to the first or to the second smallest eigenvalue of B_{α_k}. In other words, for values of α_k near the critical value, either $\|g\||\nu_1| > \varepsilon\sqrt{1-\nu_1{}^2}$ or $\|g\||\nu_2| > \varepsilon\sqrt{1-\nu_2{}^2}$ will hold. Hence, after a possible reduction of the parameter α_k, the pair $\{\lambda_k, x_k\}$ is well defined if we compute it by the following procedure:

If $\|g\||\nu_1| \leq \varepsilon\sqrt{1-{\nu_1}^2}$ then set $\lambda_k = \lambda_i(\alpha_k)$ and $x_k = \dfrac{u_2}{\nu_2}$ ·

Otherwise, set $\lambda_k = \lambda_1(\alpha_k)$ and $x_k = \dfrac{u_1}{\nu_1}$ ·

Since λ_{k-1} and λ_k are not constrained to $(-\infty, \delta_1]$ but might well belong to the interval $(\delta_1, \delta_{\ell+1})$, the value $\widehat{\lambda}$ given by (15) may be greater than δ_1. In this case, we set $\widehat{\lambda} = \delta_U$, where δ_U is an upper bound for δ_1. In §4.3 we will show how to obtain an initial value for δ_U and how to update this value. We will also show how to safeguard α computed by (17).

4.3 Safeguarding

We need to introduce safeguarding to assure global convergence of the iteration. Let λ_*, x_* be an optimal pair for problem (1), satisfying the conditions in Lemma 2.1, except when there is only an interior solution in which case we define $x_* = -(A-\lambda_* I)^\dagger g$ such that $\|x_*\| = \Delta$. Let $\alpha_* = \lambda_* - g^T x_*$. Rendl and Wolkowicz [12] presented the following bounds for the optimal parameter α_*:

$$\delta_1 - \frac{\|g\|}{\Delta} \leq \alpha_* \leq \delta_1 + \|g\|\Delta\,. \tag{18}$$

Computing a good approximation to δ_1 can be nearly as expensive as solving the given trust–region subproblem. For this reason, as observed in [12], we shall replace the above bounds by some simple alternatives. First, note that any Rayleigh quotient $\delta_U \equiv \dfrac{v^TAv}{v^Tv}$ gives an upper bound for δ_1. Therefore, if the diagonal of the matrix A is explicitly available, we take $\delta_U = \min\{a_{ii} \mid i = 1, \ldots, n\}$, otherwise we take $\delta_U \equiv \dfrac{v^TAv}{v^Tv}$ where v is a random vector. From (18) we see that $\alpha_* \leq \alpha_U$, for $\alpha_U = \delta_U + \|g\|\Delta$. Since $\alpha \leq 0$ implies B_α is not positive definite, we set $\alpha_0 = \min\{0, \alpha_U\}$ to assure that $\lambda_1(\alpha_0) \leq 0$. After solving for $\lambda_1(\alpha_0)$ and setting $\delta_L = \lambda_1(\alpha_0)$ and $\alpha_L = \delta_L - \dfrac{\|g\|}{\Delta}$, we immediately have that $\alpha_L \leq \alpha_*$, since the interlacing property implies $\delta_L \leq \delta_1$. Using this simple scheme to obtain δ_L and δ_U as initial lower and upper bounds for δ_1, we can start with

$$\alpha_L = \delta_L - \frac{\|g\|}{\Delta} \quad \text{and} \quad \alpha_U = \delta_U + \|g\|\Delta\,. \tag{19}$$

We update the upper bound δ_U at each iteration using information from the eigenpair corresponding to the smallest eigenvalue of the bordered matrix in the following way: $\delta_U = \min\left\{\delta_U, \frac{u_1^T A u_1}{u_1^T u_1}\right\}$, where $\frac{u_1^T A u_1}{u_1^T u_1} = \lambda_1(\alpha_k) - \nu_1 \frac{g^T u_1}{u_1^T u_1}$. As stated in §4.2, whenever we detect a potential hard case, $\{\lambda_1(\alpha_k), u_1\}$ approximates an eigenpair of A and $\lambda_1(\alpha_k)$ is a very good approximation to δ_1. Thus, δ_U becomes a sharp estimate of δ_1 in this case.

At every iteration, we update one of the safeguarding bounds α_L or α_U, so that we always reduce the length of the interval $[\alpha_L, \alpha_U]$. In case the value α_{k+1} predicted by the interpolating schemes (13) or (17) does not belong to the current safeguarding interval, we redefine α_{k+1} by means of a linear adjustment based on the upper bound δ_U. If this choice is not in the interval $[\alpha_L, \alpha_U]$, we simply set $\alpha_{k+1} = \dfrac{\alpha_L + \alpha_U}{2}$.

4.4 Initialization of α

As mentioned in §3.2, there is a simple choice for initializing α, given by $\alpha_0 = \min\{0, \alpha_U\}$, with α_U as in (19). This assures that $\lambda_1(\alpha_0) \leq 0$ but it has no additional properties. In an attempt to improve this initial guess, we have developed a more sophisticated *hot-start* strategy, based on the Lanczos process. To begin, we compute the following j–step Lanczos factorization, for the j smallest eigenvalues of A.

$$AV = VT + fe_j^T \tag{20}$$

where $V^T V = I_j$, with I_j the identity matrix of order j ($j \ll n$), $T \in \mathbb{R}^{j\times j}$ tridiagonal, $V^T f = 0$ and e_j denotes the jth canonical unit vector in $\mathbb{R}^j$.

The hot–start strategy consists of first changing variables in (1) using $x = Vy$ and solving the j-dimensional problem

$$\begin{array}{ll} \min & \frac{1}{2} y^T T y + g^T V y \\ \text{s.t.} & \|y\| \leq \Delta\ . \end{array}$$

Then, we compute a solution $\{\theta_*, y_*\}$ to this lower dimensional trust–region subproblem by using the algorithm in [9], based on the Cholesky factorization of the tridiagonal matrix $T - \theta I$, $\theta < \delta_1$. The initial value to be used is $\alpha = \theta_* - g^T V y_*$. In numerical experiments, the use of this hot start for α did not substantially improve the performance of the method.

We now show that we can use (20) to compute an eigenpair corresponding to the smallest eigenvalue of B_{α_0}. Observe

$$\begin{pmatrix} \alpha_0 & g^T \\ g & A \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & V \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & V \end{pmatrix} \begin{pmatrix} \alpha_0 & g^T V \\ V^T g & T \end{pmatrix} + \begin{pmatrix} 0 \\ f \end{pmatrix} e_{j+1}^T\ . \tag{21}$$

If we run the standard Lanczos process for A using $v_1 = g/\|g\|$ as initial vector then we obtain a tridiagonal matrix on the right–hand side of (21). This provides a way of computing the smallest eigenvalue of B_{α_0}.

4.5 Stopping Criteria

At each iteration we check for a boundary solution, an interior solution or a quasi–optimal solution according to Theorem 3.2. We can also stop if we reach a maximum number of iterations or if the length of the safeguarding interval is too small. Given the tolerances $\varepsilon_\Delta, \varepsilon_{HC}, \varepsilon_\alpha \in (0,1)$ and $\varepsilon_{Int} \in [0,1)$, we declare convergence of the algorithm according to the following criteria. Let $(\nu_1, u_1^T)^T$ be the eigenvector corresponding to $\lambda_1(\alpha_k)$ and let $\{\lambda_k, x_k\}$ be the current iterates, then we can write the stopping criteria in the following way.

1. Boundary Solution.

We detect a boundary solution if

$$(|\ \|x_k\| - \Delta\ |\ \le\ \varepsilon_\Delta * \Delta) \quad \text{and} \quad (\lambda_1(\alpha_k) \le 0).$$

If this condition is satisfied, the solution is

$$\lambda_* = \lambda_1(\alpha_k) \quad \text{and} \quad x_* = x_k\ .$$

2. Interior Solution.

We detect an interior solution if

$$(\|u_1\|\ <\ \Delta|\nu_1|) \quad \text{and} \quad (\lambda_1(\alpha_k) > -\varepsilon_{Int})\ .$$

In this case, the solution is $\lambda_*,\ x_*$ where $\lambda_* = 0$ and x_* satisfies the linear system $Ax = -g$, with A positive definite. The Conjugate Gradient Method is a natural choice for solving this system for most large–scale problems.

3. Quasi–optimal Solution.

To declare that we have found a quasi–optimal solution, we first compute t and $\tilde{x}$ as in Lemma 3.5, provided that the conditions of the lemma are satisfied. If $t = (\tau_1, \tau_2)^T$ and $\tilde{x}$ satisfy condition *(ii)* of Theorem 3.2 then $\tilde{x}$ is a quasi–optimal solution for problem (1) and we set $\lambda_* = \lambda_1(\alpha)\tau_1^2 + \lambda_i(\alpha)\tau_2^2$ and $x_* = \tilde{x}$.

4. The safeguarding interval is too small.

If $|\alpha_U - \alpha_L| \le \varepsilon_\alpha \max\{|\alpha_L|, |\alpha_U|\}$ then we stop the iteration and set $\lambda_* = \lambda_1(\alpha_k)$. If this criterion is satisfied and we do not have a boundary solution then we are in the hard case and α_* is within ε_α of $\tilde{\alpha}_1$. If ν_1 is large enough, we set $p = \dfrac{u_1}{\nu_1}$. Since $\|p\| < \Delta$ in this case, we compute x_* as $x_* = p + \tau z$ such that $\|x_*\| = \Delta$, where the vector z is an approximate eigenvector associated with the smallest eigenvalue of A. Of the two possible choices for τ, we choose the one with smaller magnitude since this value minimizes $\psi(p + \tau z)$ (see [9, p. 558]). This choice of τ is given by

$$\tau = \frac{\Delta^2 - \|p\|^2}{p^T z + sign(p^T z)\sqrt{(p^T z)^2 - (\Delta^2 - \|p\|^2)}}\ .$$

The vector z is usually available in potential hard cases since in those cases the eigenvectors corresponding to the smallest eigenvalue of B_{α_k} will often have a small first component. In the rather unlikely situation where this vector is not available, we increase the parameter and solve an eigenproblem for the smallest eigenvalue of the bordered matrix. This strategy will provide an approximate vector in $\mathcal{S}_1$ as Lemma 3.3 guarantees.

If ν_1 is too small or zero, we cannot compute a solution. This situation can arise in practice because the eigensolver might not provide the eigenvector with significant first component that the theory guarantees. We have not encountered this case in our experiments.

4.6 The Algorithm

Let us now put all these pieces together and present our algorithm for the large–scale trust–region subproblem (LSTRS). We describe Steps 2.1 and 2.5 of Algorithm 4.1 separately. In Step 2.1 we adjust the parameter α_k so that the eigenvector corresponding to the first or to an eigenvalue equal or close to the second smallest eigenvalue of B_{α_k}, has a significant first component. We might reduce the interval $[\alpha_L, \alpha_U]$ during this adjustment. In Step 2.5 we correct the parameter predicted by the interpolation schemes in case it does not belong to the current safeguarding interval $[\alpha_L, \alpha_U]$. We try a linear adjustment first and adopt the middle point of the current interval as a last resort. Figure 2 shows Algorithm 4.1, while Figures 3 and 4 show Steps 2.1 and 2.5, respectively.

Algorithm 4.1 *LSTRS.*

Input: $A \in \mathbb{R}^{n\times n}$, $g \in \mathbb{R}^n$, $\Delta > 0$, $\varepsilon_\Delta, \varepsilon_\nu, \varepsilon_{HC}, \varepsilon_\alpha \in (0,1)$, $\varepsilon_{Int} \in [0,1)$.
Output: λ_*, x_* *satisfying conditions of Lemma 2.1.*

1. `Initialization`
- **1.1** `Compute` $\delta_U \geq \delta_1$, `initialize` α_U `using (19), set` $\alpha_0 = \min\{0, \alpha_U\}$.
- **1.2** `Compute eigenpairs` $\{\lambda_1(\alpha_0), (\nu_1, u_1^T)^T\}$ `and` $\{\lambda_i(\alpha_0), (\nu_2, u_2^T)^T\}$, `corresponding to smallest eigenvalue and an eigenvalue close to the second smallest eigenvalue of` B_{α_0}
- **1.3** `Initialize` α_L `using (19).`
- **1.4** `Set` $k = 0$.

2. **repeat**
- **2.1** `Adjust` α_k.
- **2.2** `Update` $\delta_U = \min\left\{\delta_U, \dfrac{u_1^T A u_1}{u_1^T u_1}\right\}$.
- **2.3** **if** $\|g\||\nu_1| > \varepsilon_\nu \sqrt{1-{\nu_1}^2}$ **then**
  - `set` $\lambda_k = \lambda_1(\alpha_k)$ `and` $x_k = \dfrac{u_1}{\nu_1}$.
  - **if** $\|x_k\| < \Delta$ **then** $\alpha_L = \alpha_k$ **end if**
  - **if** $\|x_k\| > \Delta$ **then** $\alpha_U = \alpha_k$
  - **else set** $\lambda_k = \lambda_i(\alpha_k)$, $x_k = \dfrac{u_2}{\nu_2}$ `and` $\alpha_U = \alpha_k$ **end if**

  end if
- **2.4** `Compute` α_{k+1} `by interpolation scheme using (13) if` $k = 0$ `or (15) and (17) otherwise.`
- **2.5** `Safeguard` α_{k+1}.
- **2.6** `Set` $k = k + 1$.

until `convergence`

Figure 2: Method for the Large–Scale Trust–Region Subproblem.

Step 2.1 *Adjust* α_k.

Input: $\varepsilon_\nu, \varepsilon_\alpha \in (0,1)$, α_L, α_U, α_k with $\alpha_k \in [\alpha_L, \alpha_U]$.
Output: $\alpha_k, \{\lambda_1(\alpha_k), (\nu_1, u_1^T)^T\}$ and $\{\lambda_i(\alpha_k), (\nu_2, u_2^T)^T\}$.

· `Set` $\alpha = \alpha_k$
· **if** $k > 0$ **then**
  `compute eigenpairs` $\{\lambda_1(\alpha), (\nu_1, u_1^T)^T\}$ `and` $\{\lambda_i(\alpha), (\nu_2, u_2^T)^T\}$`,`
  `corresponding to smallest eigenvalue and an eigenvalue`
  `close to the second smallest eigenvalue of` B_α
  end if
· **while**
  $\|g\||\nu_1| \leq \varepsilon_\nu \sqrt{1-\nu_1{}^2}$ **and** $\|g\||\nu_2| \leq \varepsilon_\nu \sqrt{1-\nu_2{}^2}$
  and $|\alpha_U - \alpha_L| > \varepsilon_\alpha * \max\{|\alpha_L|, |\alpha_U|\}$ **do**
  $\alpha_U = \alpha$
  $\alpha = (\alpha_L + \alpha_U)/2$
  `Compute` $\{\lambda_1(\alpha), (\nu_1, u_1^T)^T\}$ `and` $\{\lambda_i(\alpha), (\nu_2, u_2^T)^T\}$
  end while
· `Set` $\alpha_k = \alpha$

Figure 3: Adjustment of α

Step 2.5 *Safeguard* α_{k+1}.

Input: α_{k+1} computed by Step 2.4 of Algorithm 4.1, $\delta_U \geq \delta_1$, α_L, α_U,
$\phi_i = -g^T x_i$ and $\phi_i' = \|x_i\|^2$, for $i = k-1, k$.
Output: Safeguarded value for α_{k+1}.

if $\alpha_{k+1} \notin [\alpha_L, \alpha_U]$
  if $k = 0$ **then** $\alpha_{k+1} = \delta_U + \phi_k + \phi_k'(\delta_U - \lambda_k)$ **end if**
  else if $\|x_k\| < \|x_{k-1}\|$ **then** $\alpha_{k+1} = \delta_U + \phi_k + \phi_k'(\delta_U - \lambda_k)$
    else $\alpha_{k+1} = \delta_U + \phi_{k-1} + \phi_{k-1}'(\delta_U - \lambda_{k-1})$.
    end if
  end if
  if $\alpha_{k+1} \notin [\alpha_L, \alpha_U]$`, set` $\alpha_{k+1} = (\alpha_L + \alpha_U)/2$ **end if**
end if

Figure 4: Safeguarding of α

5 Convergence Analysis

5.1 Iterates are well defined

Lemma 5.1 *The iterates generated by Algorithm 4.1 are well defined.*

Proof. In order to define the current iterate x_k in Algorithm 4.1, we must ensure that we can safely normalize an eigenvector corresponding to either the smallest eigenvalue or a value equal or close to the second smallest eigenvalue of B_{α_k}, to have first component one. This is accomplished in Step 2.1 where we adjust the parameter α_k until one of these two eigenvectors can be normalized to have first component one. Theorem 3.1 and Lemma 3.3 guarantee that the adjusting procedure in Step 2.1 yields a value of α such that for the smallest eigenvalue or a value equal or close to the second smallest eigenvalue of B_α, there exists a corresponding eigenvector with significant first component. $\square$

5.2 Local Convergence

5.2.1 Preliminary Results

Lemma 5.2 *Let λ_k, x_k be the iterates at iteration k of Algorithm 4.1. Then*

$$g \in \mathcal{R}(A - \lambda_k I).$$

Proof. If λ_k, x_k are the iterates at iteration k of Algorithm 4.1, then

$$\begin{pmatrix} \alpha_k & g^T \\ g & A \end{pmatrix} \begin{pmatrix} 1 \\ x_k \end{pmatrix} = \lambda_k \begin{pmatrix} 1 \\ x_k \end{pmatrix}.$$

Therefore, $(A - \lambda_k I)x_k = -g$ which implies that $g \in \mathcal{R}(A - \lambda_k I)$. $\square$

Lemma 5.3 *Let $\lambda_* \le \delta_1$ be the Lagrange multiplier corresponding to a boundary solution of problem (1). Then*

$$g \in \mathcal{R}(A - \lambda_* I).$$

Proof. If $\lambda_* < \delta_1$ then $A - \lambda_* I$ is nonsingular and $g \in \mathcal{R}(A - \lambda_* I)$. If $\lambda_* = \delta_1$ then $g \perp \mathcal{N}(A - \lambda_* I)$ must hold and therefore $g \in \mathcal{R}(A - \lambda_* I)$. $\square$

Remark: Since $(A-\lambda I)(A-\lambda I)^\dagger$ and $(A-\lambda I)^\dagger(A-\lambda I)$ are orthogonal projectors onto $\mathcal{R}(A - \lambda I)$, we have that

$$g = (A - \lambda I)(A - \lambda I)^\dagger g = (A - \lambda I)^\dagger (A - \lambda I) g \tag{22}$$

for any λ such that $g \in \mathcal{R}(A - \lambda I)$. In particular, Lemmas 5.2 and 5.3 imply that (22) holds for $\lambda = \lambda_k$ and $\lambda = \lambda_*$.

5.2.2 Technical Lemmas

We present several technical lemmas that allow us to prove our local convergence result. We will use the following notation:

$$A_k \equiv A - \lambda_k I \quad \text{and} \quad A_* \equiv A - \lambda_* I. \tag{23}$$

The first lemma establishes a key relationship satisfied by the iterates computed by Algorithm 4.1.

Lemma 5.4 *Let λ_k, x_k be the iterates at iteration k of Algorithm 4.1. Then*

$$x_k = -(A - \lambda_k I)^{\dagger} g.$$

Proof. First note that if λ_k, x_k are the iterates at iteration k of Algorithm 4.1, then they satisfy

$$\begin{pmatrix} \alpha_k & g^T \\ g & A \end{pmatrix} \begin{pmatrix} 1 \\ x_k \end{pmatrix} = \lambda_k \begin{pmatrix} 1 \\ x_k \end{pmatrix}.$$

and therefore

$$(A - \lambda_k I) x_k = -g. \tag{24}$$

In order to prove the result we need to consider two cases.

Case 1: $\lambda_k \neq \delta_i,\ i = 1, 2, \ldots, d.$

In this case we have that $A - \lambda_k I$ is nonsingular, $(A - \lambda_k I)^{-1} = (A - \lambda_k I)^{\dagger}$ and from (24) we conclude

$$x_k = -(A - \lambda_k I)^{\dagger} g.$$

Case 2: $\lambda_k = \delta_i$, $1 \leq i \leq d.$

If $\lambda_k = \delta_i$ then (24) implies that $g \perp \mathcal{S}_i$. This follows from the observation that for any $q \in \mathcal{S}_i$, we have $0 = q^T (A - \delta_i I) x_k = -q^T g$. Corollary 3.1 now implies that $\alpha_k = \tilde{\alpha}_i$ and

$$\begin{aligned} x_k &= p_i \\ &= -(A - \delta_i I)^{\dagger} g, \end{aligned}$$

since $(1, x_k^T)^T$ is an eigenvector of B_{α_k}. This concludes the proof. $\square$

Before presenting the next lemma, which provides useful relationships for the convergence analysis, we introduce the following definition.

Definition 5.1 *Let λ_i, x_i and λ_j, x_j be the iterates computed by Algorithm 4.1 at iteration i and j, respectively. Then we define*

$$\rho(i,j) \equiv x_i^T A_j^\dagger x_i + x_j^T A_i^\dagger x_j. \tag{25}$$

We can substitute any of the iterates by λ_*, y, with $y = -A_*^\dagger g$. We denote this by $\rho(*,j)$ and $\rho(i,*)$, respectively.

Assuming that $A = QDQ^T$ is an eigendecomposition of A, i.e. Q is an orthogonal matrix and D is a diagonal matrix with the eigenvalues of A on the diagonal, we can write $\rho(i,j)$ in the following way.

$$\rho(i,j) = g^T Q D_i^\dagger (D_i^\dagger + D_j^\dagger) D_j^\dagger Q^T g$$

where $D_i = D - \lambda_i I$ and $D_j = D - \lambda_j I$. From this expression we obtain

$$\rho(i,j) = \sum_{k=1}^{d} \frac{\beta_k^2 (2\delta_k - \lambda_i - \lambda_j)}{(\delta_k - \lambda_i)^2 (\delta_k - \lambda_j)^2} \tag{26}$$

where β_k^2 is the sum of the expansion coefficients of g in the eigenvector basis, corresponding to all the eigenvectors associated with δ_k. As before, we are assuming that $\delta_1, \delta_2, \ldots, \delta_d$ are the distinct eigenvalues of A in nondecreasing order.

Lemma 5.5 *Let λ_i, x_i, λ_j, x_j and λ_k, x_k be the iterates computed by Algorithm 4.1 at iteration i, j and k, respectively. Then*

(i) $(x_i - x_j)^T g = (\lambda_j - \lambda_i) x_i^T x_j$.

(ii) $(x_i - x_j)^T x_k = (\lambda_i - \lambda_j) x_j^T A_i^\dagger x_k$.

(iii) $x_i^T x_i - x_j^T x_j = (\lambda_i - \lambda_j)\rho(i,j)$, *with $\rho(i,j)$ given by (25).*

Moreover, (i)–(iii) *also hold if we substitute any of the pairs above by λ_*, y, where λ_* is the Lagrange multiplier corresponding to a boundary solution of problem (1) and $y = -A_*^\dagger g$.*

Proof. Let us first prove *(i)*. Observe that by Lemma 5.4

$$(x_i - x_j)^T g \;=\; (A_j^\dagger g - A_i^\dagger g)^T g.$$

Therefore, using (22) and the fact that $A_i, A_i^\dagger, A_j, A_j^\dagger$ commute, we have

$$\begin{aligned} (x_i - x_j)^T g &= g^T (A_j^\dagger - A_i^\dagger) g \\ &= g^T A_i^\dagger (A_i - A_j) A_j^\dagger g \\ &= (\lambda_j - \lambda_i) x_i^T x_j. \end{aligned}$$

To prove *(ii)*, we use (22), Lemma 5.4 and the fact that $A_i, A_i^\dagger, A_j, A_j^\dagger$ commute, obtaining

$$\begin{aligned}(x_i - x_j)^T x_k &= g^T(A_i^\dagger - A_j^\dagger)x_k \\ &= g^T A_j^\dagger (A_j - A_i) A_i^\dagger x_k \\ &= (\lambda_i - \lambda_j) x_j^T A_i^\dagger x_k.\end{aligned}$$

Finally, let us prove *(iii)*. By (22), Lemma 5.4 and the fact that $A_i, A_i^\dagger, A_j, A_j^\dagger$ commute, we have

$$\begin{aligned}x_i^T x_i - x_j^T x_j &= g^T[(A_i^\dagger)^2 - (A_j^\dagger)^2]g \\ &= g^T[(A_i^\dagger)^2 A_j^2 - A_i^2 (A_j^\dagger)^2]g \\ &= g^T (A_i^\dagger)^2 (A_j - A_i)(A_j + A_i)(A_j^\dagger)^2 g \\ &= (\lambda_i - \lambda_j) x_i^T (A_i^\dagger + A_j^\dagger) x_j \\ &= (\lambda_i - \lambda_j)\rho(i,j).\end{aligned}$$

Observe that *(i)–(iii)* hold for λ_*, y, since (22) holds for λ_*, $y = -A_*^\dagger g$, and A_* commutes with the matrices above. This observation concludes the proof. □

Using the updating formula (17), we obtained the following result relating $\lambda_{k+1} - \lambda_*$ with $\lambda_{k-1} - \lambda_*$ and $\lambda_k - \lambda_*$ This lemma provides a key relationship for establishing the local convergence properties of Algorithm 4.1.

Lemma 5.6 *Let $\lambda_* \leq \delta_1$ be the Lagrange multiplier corresponding to a boundary solution of problem (1), with $g \neq 0$. Let λ_{k+1}, x_{k+1} be the* (k+1)*st iterates computed by Algorithm 4.1 using the two–point interpolating scheme given by (17) to update α. Then, there exists a neighborhood $\mathcal{B}$ of λ_* such that if λ_{k-1}, $\lambda_k \in \mathcal{B}$ then λ_{k+1} satisfies*

$$|\lambda_{k+1} - \lambda_*| \leq \mathcal{C}\ |\lambda_{k-1} - \lambda_*||\lambda_k - \lambda_*| \tag{27}$$

with $\mathcal{C}$ independent of k.

Proof. Let $y = -A_*^\dagger g$ and $\alpha_* = \lambda_* - g^T y$. We divide the proof in two cases: $\|y\| = \Delta$ and $\|y\| < \Delta$. In each case, we first find an appropriate neighborhood of λ_*, and then prove (27) for λ_{k-1}, λ_k in that neighborhood.

Case 1: $\|y\| = \Delta$.

We will first find a neighborhood $\mathcal{B}$ of λ_* such that if λ_{k-1}, $\lambda_k \in \mathcal{B}$ then $\widehat{\lambda} \in \mathcal{B}$, with $\widehat{\lambda}$ given by (15). In this part of the proof we define the numbers ℓ and m in the following way.

Let $0 \leq \ell < n$ and assume that $g \perp \mathcal{S}_i$, $i = 1, 2, \ldots, \ell$, where $\ell = 0$ indicates that $g \not\perp \mathcal{S}_1$. Let $m = 0$ if $\lambda_* < \delta_1$ and $m = \ell$ if $\lambda_* = \delta_1$. Define

$$r_1 = \frac{\delta_{m+1} - \lambda_*}{2} \quad \text{and} \quad \mathcal{B}_1 = \{\lambda \mid |\lambda - \lambda_*| \leq r_1\}$$

and suppose that $\lambda_{k-1}, \lambda_k \in \mathcal{B}_1$. Then by (15), Lemma 5.5 *(iii)* and the fact that $\|y\| = \Delta$, we have

$$\begin{aligned} \widehat{\lambda} - \lambda_* &= \lambda_k - \lambda_* + \frac{\|x_{k-1}\|(\|x_{k-1}\| + \|x_k\|)(\Delta - \|x_k\|)}{\rho(k-1,k)\Delta} \\ &= (\lambda_k - \lambda_*)\left[1 - \frac{\rho(*,k)\|x_{k-1}\|(\|x_{k-1}\| + \|x_k\|)}{\rho(k-1,k)\Delta(\Delta + \|x_k\|)}\right]. \end{aligned} \tag{28}$$

We will prove now that $|\widehat{\lambda} - \lambda_*| \leq |\lambda_k - \lambda_*|\vartheta$, with $\vartheta \in (0,1)$.
Let $\Delta_{max} = \max\limits_{\lambda \in \mathcal{B}_1} \|(A - \lambda I)^\dagger g\|$ and $\Delta_{min} = \min\limits_{\lambda \in \mathcal{B}_1} \|(A - \lambda I)^\dagger g\|$. Therefore

$$\frac{\|x_{k-1}\|(\|x_{k-1}\| + \|x_k\|)}{\Delta(\Delta + \|x_k\|)} \geq \left(\frac{\Delta_{min}}{\Delta_{max}}\right)^2. \tag{29}$$

In view of (26) we have that for $\lambda_{k-1},\ \lambda_k \in \mathcal{B}_1$

$$\rho(*,k) \;\; \geq \;\; \frac{(2\delta_{\ell+1} - \lambda_* - \lambda_k)}{(\delta_d - \lambda_*)^2(\delta_d - \lambda_k)^2}\|g\|^2 .$$

Since $\delta_{m+1} \leq \delta_{\ell+1}$ and since $\dfrac{-\delta_{m+1} + \lambda_*}{2} \leq \lambda_* - \lambda_k \leq \dfrac{\delta_{m+1} - \lambda_*}{2}$, we have

$$\begin{aligned} \rho(*,k) &\geq \frac{(2\delta_{m+1} - \lambda_* - \lambda_k)}{(\delta_d - \lambda_*)^2(\delta_d - \lambda_k)^2}\|g\|^2 \\ &= \frac{2(\delta_{m+1} - \lambda_*) + (\lambda_* - \lambda_k)}{(\delta_d - \lambda_*)^2[(\delta_d - \lambda_*) + (\lambda_* - \lambda_k)]^2}\|g\|^2 \\ &\geq \frac{2}{3}\|g\|^2\frac{(\delta_{m+1} - \lambda_*)}{(\delta_d - \lambda_*)^4} . \end{aligned} \tag{30}$$

Using similar manipulations we obtain

$$\begin{aligned} \rho(k-1,k) &\leq \frac{(2\delta_d - \lambda_k - \lambda_{k-1})\|g\|^2}{(\delta_{\ell+1} - \lambda_k)^2(\delta_{\ell+1} - \lambda_{k-1})^2} \\ &\leq \frac{(2\delta_d - \lambda_k - \lambda_{k-1})\|g\|^2}{(\delta_{m+1} - \lambda_k)^2(\delta_{m+1} - \lambda_{k-1})^2} \\ &\leq \frac{3 \cdot 2^4\|g\|^2(\delta_d - \lambda_*)}{(\delta_{m+1} - \lambda_*)^4}. \end{aligned} \tag{31}$$

It follows from (28), (29), (30) and (31) that

$$|\widehat{\lambda} - \lambda_*| \leq |\lambda_k - \lambda_*| \left| 1 - \left(\frac{\Delta_{min}}{\Delta_{max}}\right)^2 \frac{1}{72} \frac{(\delta_{m+1} - \lambda_*)^5}{(\delta_d - \lambda_*)^5} \right| \equiv |\lambda_k - \lambda_*| \vartheta$$

with $\vartheta \in (0, 1)$. Therefore $\widehat{\lambda} \in \mathcal{B}_1$ whenever $\lambda_{k-1},\ \lambda_k\ \in\ \mathcal{B}_1$.

Now, we use these results to establish (27). Let the neighborhood $\mathcal{B}$ be given by $\mathcal{B}_1$ and let $\lambda_{k-1},\ \lambda_k\ \in\ \mathcal{B}$, therefore $\widehat{\lambda} \in \mathcal{B}$ and to prove (27) we need to consider two possibilities: $\widehat{\lambda} < \delta_1$ and $\widehat{\lambda} \geq \delta_1$.

Case 1.1: $\|y\| = \Delta$ and $\widehat{\lambda} < \delta_1$.
In this case, we use formulas (15) and (17) obtaining

$$\alpha_{k+1} = T_1 + T_2$$

where

$$T_1 = \frac{\alpha_{k-1}\|x_{k-1}\|(\|x_k\| - \Delta) + \|x_k\|(\Delta - \|x_{k-1}\|)}{\Delta(\|x_k\| - \|x_{k-1}\|)}$$

and

$$T_2 = \frac{\|x_k\|\|x_{k-1}\|(\Delta - \|x_k\|)(\Delta - \|x_{k-1}\|)(\lambda_k - \lambda_{k-1})}{\Delta(\|x_k\| - \|x_{k-1}\|)}$$

We will now find an upper bound for $|T_2|$. From Lemma 5.5 *(iii)*, we have

$$T_2 = \frac{\|x_k\|\|x_{k-1}\|(\Delta - \|x_k\|)(\Delta - \|x_{k-1}\|)(\|x_k\| + \|x_{k-1}\|)}{\Delta\rho(k-1, k)}$$

and since $\|y\| = \Delta$ we can write

$$T_2 = \frac{\|x_k\|\|x_{k-1}\|(\|x_k\| + \|x_{k-1}\|)}{\rho(k-1, k)} \frac{(\|y\| - \|x_k\|)(\|y\| - \|x_{k-1}\|)}{\Delta}$$

Using Lemma 5.5 *(iii)* we obtain

$$T_2 = \frac{\|x_k\|\|x_{k-1}\|(\|x_k\| + \|x_{k-1}\|)}{\rho(k-1, k)} \frac{(\lambda_* - \lambda_{k-1})(\lambda_* - \lambda_k)\rho(k-1, *)\rho(k, *)}{\Delta(\Delta + \|x_k\|)(\Delta + \|x_{k-1}\|)}.$$

Now, since $\lambda_{k-1},\ \lambda_k\ \in\ \mathcal{B}$ and since $\delta_{m+1} \leq \delta_{\ell+1}$,

$$\rho(k-1, k) \;\; \geq \;\; \frac{2}{3}\|g\|^2 \, \frac{(\delta_{m+1} - \lambda_*)}{(\delta_d - \lambda_*)^4}, \tag{32}$$

$$\rho(*, k),\ \rho(*, k-1) \;\; \leq \;\; 10 \, \frac{\delta_d - \lambda_*}{(\delta_{m+1} - \lambda_*)^4} \cdot \tag{33}$$

Since for $\lambda_{k-1},\ \lambda_k\ \in\ \mathcal{B}$ we also have $\Delta_{min} \le \|x_{k-1}\|, \|x_k\| \le \Delta_{max}$, we obtain

$$|T_2| \ \le\ \mathcal{C}_2\ |\lambda_* - \lambda_{k-1}|\ |\lambda_* - \lambda_k|. \tag{34}$$

We will use this estimate in a moment. First, we need to relate it to $\lambda_{k+1} - \lambda_*$. To do this, consider

$$\alpha_{k+1} - \alpha_* \ = \ \frac{(\alpha_{k-1} - \alpha_*)\|x_{k-1}\|(\|x_k\| - \Delta) + (\alpha_k - \alpha_*)\|x_k\|(\Delta - \|x_{k-1}\|)}{\Delta(\|x_k\| - \|x_{k-1}\|)} + T_2\ . \tag{35}$$

From Lemma 5.5*(i)*, the definition of α_*, and since $\alpha_j - \lambda_j = -g^T x_j$ for $j > 0$, we have

$$\begin{aligned} \alpha_j - \alpha_* &= \lambda_j - \lambda_* - g^T(x_j - y) \\ &= (\lambda_j - \lambda_*)(1 + x_j^T y). \end{aligned} \tag{36}$$

Using (36) along with Lemma 5.5, (35) becomes

$$\begin{aligned} (\lambda_{k+1} - \lambda_*)(1 + x_{k+1}^T y) \ &= \ \frac{(\lambda_{k-1} - \lambda_*)\|x_{k-1}\|(\|x_k\| - \Delta)(1 + x_{k-1}^T y)}{\Delta(\|x_k\| - \|x_{k-1}\|)} \\ &- \ \frac{(\lambda_k - \lambda_*)\|x_k\|(\|x_{k-1}\| - \Delta)(1 + x_k^T y)}{\Delta(\|x_k\| - \|x_{k-1}\|)} + T_2 \\ &= \ \frac{(\lambda_{k-1} - \lambda_*)(\lambda_k - \lambda_*)T_3}{\Delta(\|x_k\| - \|x_{k-1}\|)(\|x_k\| + \Delta)(\|x_{k-1}\| + \Delta)} + T_2, \end{aligned}$$

where

$$\begin{aligned} T_3 \ &= \ (\|x_{k-1}\| - \|x_k\|)(\|x_{k-1}\| + \|x_k\| + \Delta)(1 + x_k^T y)\rho(k-1,*) \\ &+ \ y^T(x_{k-1} - x_k)\|x_{k-1}\|(\|x_{k-1}\| + \Delta)\rho(k,*) \\ &+ \ (\rho(k,*) - \rho(k-1,*))\|x_{k-1}\|(\|x_{k-1}\| + \Delta)(1 + x_k^T y)\ . \end{aligned}$$

Now, by Lemma 5.5 *(ii)* we have

$$y^T(x_{k-1} - x_k) = (\lambda_{k-1} - \lambda_k) y^T A_k^\dagger x_{k-1} \tag{37}$$

and by Lemma 5.5 *(iii)*

$$\begin{aligned} \rho(k,*) - \rho(k-1,*) \ &= \ x_k^T A_*^\dagger x_k + y^T A_k^\dagger y - x_{k-1}^T A_*^\dagger x_{k-1} - y A_{k-1}^\dagger y \\ &= \ g^T A_k^\dagger A_*^\dagger A_k^\dagger g - g^T A_{k-1}^\dagger A_*^\dagger A_{k-1}^\dagger g + y^T(A_k^\dagger - A_{k-1}^\dagger) y \\ &= \ g^T((A_k^\dagger)^2 - (A_{k-1}^\dagger)^2) y + (\lambda_k - \lambda_{k-1}) y^T A_k^\dagger A_{k-1}^\dagger y \\ &= \ (\lambda_k - \lambda_{k-1})(y^T A_{k-1}^\dagger A_k^\dagger y + x_{k-1}^T(A_k^\dagger + A_{k-1}^\dagger) A_k^\dagger y). \end{aligned} \tag{38}$$

Therefore by (33), (37), (38) and since $\lambda_{k-1},\ \lambda_k\ \in\ \mathcal{B}$, we have

$$|T_3| \le \mathcal{C}_3|\lambda_k - \lambda_{k-1}|\ .$$

We may now combine the estimates we have established for $T_1,\ T_2$ and T_3 to give

$$\begin{aligned}|\lambda_{k+1} - \lambda_*||1 + x_{k+1}^T y| &\le \mathcal{C}_3 \frac{|\lambda_{k-1} - \lambda_*|\ |\lambda_k - \lambda_*|\ |\lambda_k - \lambda_{k-1}|}{\Delta|(\|x_k\| - \|x_{k-1}\|)(\|x_k\| + \Delta)(\|x_{k-1}\| + \Delta)|} + |T_2| \\ &\le (\mathcal{C}_4 + \mathcal{C}_2)|\lambda_{k-1} - \lambda_*|\ |\lambda_k - \lambda_*|\end{aligned}$$

since $\lambda_{k-1},\ \lambda_k\ \in\ \mathcal{B}$ and (34) holds. Let us see now that $\dfrac{1}{1 + x_{k+1}^T y} < 1$. Note that

$$\begin{aligned}x_{k+1}^T y &= g^T A_{k+1}^\dagger A_*^\dagger g \\ &= \sum_{j=\ell+1}^{d} \frac{\beta_j^2}{(\delta_j - \lambda_{k+1})(\delta_j - \lambda_*)} \ge \frac{\|g\|^2}{(\delta_d - \lambda_{k+1})(\delta_d - \lambda_*)}\ .\end{aligned} \tag{39}$$

From this expression we can conclude $x_{k+1}^T y > 0$ since $\lambda_* < \delta_{m+1} \le \delta_d$ and also $\lambda_{k+1} < \delta_{m+1} \le \delta_d$, by the way we compute the iterates in Algorithm 4.1.
We can now claim (27) when $\|y\| = \Delta$ and $\widehat{\lambda} < \delta_1$.
Case 1.2: $\|y\| = \Delta$ and $\widehat{\lambda} \ge \delta_1$.
In this case we must use the ideal safeguard, setting $\widehat{\lambda} = \delta_1$. Before proceeding with the proof, we point out that this can only occur when $\lambda_* = \delta_1$. To see this recall that if $\lambda_* < \delta_1$ we have $\mathcal{B} = \{\lambda \mid |\lambda - \lambda_*| \le \frac{\delta_1 - \lambda_*}{2}\}$, and we proved that $\lambda_{k-1},\ \lambda_k\ \in\ \mathcal{B}$ implies $\widehat{\lambda} \in \mathcal{B}$. Therefore

$$\begin{aligned}\widehat{\lambda} - \lambda_* &\le \frac{\delta_1 - \lambda_*}{2} \\ \widehat{\lambda} &\le \frac{\delta_1 + \lambda_*}{2} < \delta_1\end{aligned}$$

if $\lambda_* < \delta_1$.
To continue with the proof we write the formula for α_{k+1} in this case as

$$\alpha_{k+1} = T_4 + T_5$$

where

$$T_4 = \omega\alpha_{k-1} + (1 - \omega)\alpha_k$$

and

$$T_5 = \frac{\|x_k\|\|x_{k-1}\|(\|x_k\| - \|x_{k-1}\|)}{\omega\|x_k\| + (1 - \omega)\|x_{k-1}\|} \frac{(\lambda_{k-1} - \delta_1)(\lambda_k - \delta_1)}{(\lambda_k - \lambda_{k-1})}.$$

Since $\widehat{\lambda} = \delta_1$ we have $\omega = \dfrac{\lambda_k - \delta_1}{\lambda_k - \lambda_{k-1}}$ and therefore

$$\begin{aligned} \omega\|x_k\| + (1-\omega)\|x_{k-1}\| &= \frac{(\lambda_k - \delta_1)\|x_k\| + (\delta_1 - \lambda_{k-1})\|x_{k-1}\|}{\lambda_k - \lambda_{k-1}} \\ &= \frac{(\lambda_k - \delta_1)\rho(k-1,k) + \|x_{k-1}\|(\|x_{k-1}\| + \|x_k\|)}{\|x_{k-1}\| + \|x_k\|} . \end{aligned} \tag{40}$$

by Lemma 5.5 *(iii)*.
Using (40), (25) and Lemma 5.5 *(iii)*, we obtain

$$T_5 = \frac{\|x_k\|\|x_{k-1}\|\rho(k,k-1)(\lambda_{k-1} - \delta_1)(\lambda_k - \delta_1)}{(\lambda_k - \delta_1)\rho(k-1,k) + \|x_{k-1}\|(\|x_{k-1}\| + \|x_k\|)} .$$

By (32) and the hypothesis that $\lambda_{k-1},\ \lambda_k \in \mathcal{B}$, we have

$$|T_5| \leq \mathcal{C}_5 |\lambda_{k-1} - \delta_1||\lambda_k - \delta_1|. \tag{41}$$

We now write

$$\begin{aligned} \alpha_{k+1} - \alpha_* &= \omega\alpha_{k-1} + (1-\omega)\alpha_k - \alpha_* + T_5 \\ &= \omega(\alpha_{k-1} - \alpha_*) + (1-\omega)(\alpha_k - \alpha_*) + T_5 . \end{aligned}$$

Equation (35) and the fact that $\lambda_* = \delta_1$ yield

$$|\lambda_{k+1} - \lambda_*||1 + y^T x_{k+1}| \leq \frac{|\lambda_{k-1} - \lambda_*||\lambda_k - \lambda_*|}{|\lambda_k - \lambda_{k-1}|} |y^T(x_k - x_{k-1})| + |T_5| .$$

Observe that $|y^T(x_k - x_{k-1})| \leq |g^T(A - \lambda_k I)^\dagger (A - \lambda_{k-1} I)^\dagger (A - \lambda_* I)^\dagger g|$, and we can compute and upper bound for this term using Cauchy–Schwartz inequality, continuity of $\|\cdot\|$ and that $\lambda_{k-1},\ \lambda_k \in \mathcal{B}$. Therefore, by (41) and since $\lambda_{k-1},\ \lambda_k \in \mathcal{B}$

$$|\lambda_{k+1} - \lambda_*||1 + y^T x_{k+1}| \leq (\mathcal{C}_6 + \mathcal{C}_5)|\lambda_{k-1} - \lambda_*||\lambda_k - \lambda_*|.$$

Using (39), we can now establish (27) when $\|y\| = \Delta$ and $\widehat{\lambda} \geq \delta_1$.

Case 2: $\|y\| < \Delta$.

In this situation, we are in the hard case and therefore $\lambda_* = \delta_1$ and $g \perp \mathcal{S}_i$, $i = 1, 2, \ldots, \ell$ with $1 \leq \ell < d$. For this case we will find a neighborhood $\mathcal{B}$ of λ_* such that $\lambda_{k-1},\ \lambda_k \in \mathcal{B}$ implies $\widehat{\lambda} > \delta_1$.

Let the function $\varphi(\lambda) \equiv \|(A - \lambda I)^\dagger g\|$. Then $\varphi(\lambda)$ is strictly increasing in $(-\infty, \delta_{\ell+1})$, and there exist $\lambda_a,\ \lambda_b$ such that $\varphi(\lambda_a) = \dfrac{\Delta_y}{2}$ and $\varphi(\lambda_b) = \dfrac{\Delta + \Delta_y}{2}$, with $\Delta_y = \|y\|$.

Let

$$r_2 = \min\left\{\frac{\delta_1-\lambda_a}{2}, \frac{\delta_1-\lambda_b}{2}, \frac{\delta_{\ell+1}-\delta_1}{2}\right\} \quad \text{and} \quad \mathcal{B}_2 = \{\lambda \mid |\lambda-\lambda_*| \le r_2\}.$$

Then for $\lambda_{k-1},\ \lambda_k\ \in \mathcal{B}_2$

$$\frac{\Delta_y}{2} \le \ \|x_{k-1}\|,\ \|x_k\|\ \le\ \frac{\Delta+\Delta_y}{2}$$
$$\text{and} \quad \delta_1 - r_2 \le \lambda_{k-1},\ \lambda_k\ \le \delta_1 + r_2 < \delta_{\ell+1}\ .$$

Now observe that using (15) and Lemma 5.5 *(iii)* we can write

$$\begin{aligned}
\widehat{\lambda}-\lambda_* &= \widehat{\lambda}-\delta_1 \\
&= \frac{(\lambda_{k-1}-\delta_1)\|x_{k-1}\|(\|x_k\|-\Delta)+(\lambda_k-\delta_1)\|x_k\|(\Delta-\|x_{k-1}\|)}{\Delta(\|x_k\|-\|x_{k-1}\|)} - \delta_1 \\
&= \lambda_k - \delta_1 + \frac{\|x_{k-1}\|(\Delta-\|x_k\|)(\|x_{k-1}\|+\|x_k\|)}{\Delta\rho(k,k-1)} \\
&\ge \lambda_k - \delta_1 + \frac{{\Delta_y}^2(\Delta-\Delta_y)}{4\Delta\rho(k,k-1)}\ . \qquad (42)
\end{aligned}$$

Observe now that for $\lambda_{k-1},\ \lambda_k\ \in\ \mathcal{B}_2$

$$\rho(k,k-1) \le \frac{3\cdot 2^4\|g\|^2(\delta_d-\delta_1)}{(\delta_{\ell+1}-\delta_1)^4}\ . \qquad (43)$$

Using (42) and (43) we obtain

$$\widehat{\lambda}-\delta_1 \ge \lambda_k - \delta_1 + \frac{{\Delta_y}^2(\Delta-\Delta_y)(\delta_{\ell+1}-\delta_1)^4}{3\cdot 2^6\Delta\|g\|^2(\delta_d-\delta_1)}\ .$$

Let $\zeta \equiv \dfrac{{\Delta_y}^2(\Delta-\Delta_y)(\delta_{\ell+1}-\delta_1)^4}{3\cdot 2^6\Delta\|g\|^2(\delta_d-\delta_1)}$. Observe that ζ is well defined since $\delta_d = \delta_1$ would imply $g \in {\rm I\!R}^{n\ \perp} = \{0\}$. Observe also that $\zeta > 0$, since $\Delta > \Delta_y$ and $\delta_{\ell+1} > \delta_1$. Then, for $\lambda_k \ge \delta_1 - \zeta$ we have $\widehat{\lambda} \ge \delta_1$. So, let

$$r_3 = \min\{r_2, \zeta\} \quad \text{and} \quad \mathcal{B}_3 = \{\lambda \mid |\lambda-\delta_1| \le r_3\}.$$

It follows that for $\lambda_{k-1},\ \lambda_k\ \in\ \mathcal{B}_3$, we have $\widehat{\lambda}\ \ge\ \delta_1$ and we must use the ideal safeguard, setting $\widehat{\lambda} = \delta_1$. The proof now proceeds as in Case 1.2 where the neighborhood $\mathcal{B}$ is given by $\mathcal{B}_3$, and we use (43) instead of (31).
The analysis of the two cases concludes the proof. □

Note that the assumption in Lemma 5.6 that the trust–region constraint is binding at the solution includes the possibility of the hard case, since in this case $x_* = -A_*^{\dagger}g + z$, with $z \in \mathcal{S}_1$ and $\|x_*\| = \Delta$.

5.2.3 Local Convergence Result

Theorem 5.1 *Let $\lambda_* \leq \delta_1$ be the Lagrange multiplier corresponding to a boundary solution of problem (1), with $g \neq 0$. Let $\{\lambda_k\}$, $\{x_k\}$ be the sequences of iterates generated by Algorithm 4.1 using the two–point interpolating scheme given by (17) to update α. There exists a neighborhood $\mathcal{B}$ of λ_* such that if $\lambda_{i-1},\ \lambda_i \in \mathcal{B}$ then for $k \geq i-1$*

(i) $\{\lambda_k\}$ remains in $\mathcal{B}$ and converges q–superlinearly to λ_*.

(ii) $\{x_k\}$ converges q–superlinearly to $y = -(A-\lambda_* I)^\dagger g$.

Proof. First we show that $\{\lambda_k\}$ converges to λ_* and that the rate of convergence is superlinear.
Let $r \in \mathbb{R}, r > 0$ and $\mathcal{B} = \{\lambda \mid |\lambda - \lambda_*| < r\}$ be the neighborhood of λ_* stated in Lemma 5.6 and suppose that $\lambda_{i-1},\ \lambda_i \ \in \ \mathcal{B}$, for $i \geq 1$. Then, Lemma 5.6 implies that there exists $\mathcal{C}$ such that

$$|\lambda_{i+1} - \lambda_*| \leq \mathcal{C}\ |\lambda_i - \lambda_*|\ |\lambda_{i-1} - \lambda_*|. \tag{44}$$

Let $\hat{r} = \min\{r, \frac{1}{2\mathcal{C}}\}$, define $\widehat{\mathcal{B}} = \{\lambda \mid |\lambda - \lambda_*| < \hat{r}\}$ and observe that $\widehat{\mathcal{B}} \subset \mathcal{B}$. Suppose $\lambda_{i-1},\ \lambda_i \ \in \ \widehat{\mathcal{B}}$, then $\lambda_{i-1},\ \lambda_i \ \in \ \mathcal{B}$ and (44) holds.
Observe now that for $\lambda_{i-1},\ \lambda_i \ \in \ \widehat{\mathcal{B}}$ we have $\mathcal{C}|\lambda_{i-1} - \lambda_*| \leq \frac{1}{2}$ and therefore

$$|\lambda_{i+1} - \lambda_*| \leq \frac{1}{2}|\lambda_i - \lambda_*|$$

which implies $\lambda_{i+1} \ \in \ \widehat{\mathcal{B}} \subset \mathcal{B}$.
It follows inductively that if $\lambda_{i-1},\ \lambda_i \ \in \ \mathcal{B}$ then $\lambda_k \in \mathcal{B}$ for $k \geq i-1$ and this implies

$$|\lambda_k - \lambda_*| \leq \frac{1}{2^{k-i+1}}|\lambda_{i-1} - \lambda_*|$$

and therefore $\lambda_k \to \lambda_*$ as $k \to \infty$.
To see that the rate of convergence is q–superlinear, observe that by (44), for $k \geq i$ we have

$$\frac{|\lambda_{k+1} - \lambda_*|}{|\lambda_k - \lambda_*|} = \mathcal{C}|\lambda_{k-1} - \lambda_*|,$$

which goes to zero as k goes to infinity.
In the second part of the proof we show that the sequence $\{x_k\}$ converges superlinearly to $y = -(A - \lambda_* I)^\dagger g$.

Recall from Lemma 5.4 that $x_k = -A_k^\dagger g$ and let us study $x_k - y$ which is given by

$$\begin{aligned} x_k - y &= (A-\lambda_* I)^\dagger g - (A-\lambda_k I)^\dagger g \\ &= (A-\lambda_* I)^\dagger((A-\lambda_k I)-(A-\lambda_* I))(A-\lambda_k I)^\dagger g \\ &= (\lambda_* - \lambda_k)(A-\lambda_* I)^\dagger (A-\lambda_k I)^\dagger g. \end{aligned}$$

using (22) and rearranging terms. Taking norms on both sides we have

$$\begin{aligned} \|x_k - y\| &= |\lambda_k - \lambda_*| \; \|(A-\lambda_* I)^\dagger\| \|(A-\lambda_k I)^\dagger\| \; \|g\| \\ &\le \widehat{\mathcal{C}}|\lambda_k - \lambda_*| \end{aligned} \tag{45}$$

for a positive constant $\widehat{\mathcal{C}}$, since $\lambda_k \in \mathcal{B}$, $\lambda_* \le \delta_1$ and $\|g\|$ is constant. Therefore, since $\lambda_k \to \lambda_*$ as $k \to \infty$, we have that $x_k \to y$ as $k \to \infty$.
To see that the rate of convergence is q–superlinear, observe that (44) and (45) imply

$$\frac{\|x_{k+1} - y\|}{\|x_k - y\|} \le |\lambda_{k-1} - \lambda_*|$$

which goes to zero as k goes to infinity. This completes the proof. □

5.2.4 Near Hard Case

The next lemma provides a relationship between the function ϕ and the interpolating function (16). We will use this relationship in the analysis of the near hard case.

Lemma 5.7 *At iteration k of Algorithm 4.1 the interpolating function (16) satisfy*

$$\widehat{\phi}(\widehat{\lambda}) - \phi(\lambda_k) = (\widehat{\lambda} - \lambda_k)\left[x_k^T x_{k-1} + \frac{\|x_{k-1}\|\|x_k\|\rho(k,k-1)(\widehat{\lambda}-\lambda_{k-1})}{(\lambda_k - \widehat{\lambda})\rho(k,k-1) + \|x_{k-1}\|(\|x_k\| + \|x_{k-1}\|)}\right],$$

with $\rho(k,k-1)$ as in (25) and $\widehat{\lambda}$ given by (15).

Proof. By (16) and Lemma 5.5 *(iii)*, we have

$$\begin{aligned} \widehat{\phi}(\widehat{\lambda}) &= \frac{\gamma^2}{\delta - \widehat{\lambda}} + \omega\phi(\lambda_{k-1}) - \omega\frac{\gamma^2}{\delta - \lambda_{k-1}} \\ &+ (1-\omega)\phi(\lambda_k) - (1-\omega)\frac{\gamma^2}{\delta - \lambda_k} \\ &= \phi(\lambda_k) + \omega g^T(x_k - x_{k-1}) + \frac{\omega\gamma^2(\widehat{\lambda} - \lambda_{k-1})}{(\delta - \widehat{\lambda})(\delta - \lambda_{k-1})} \\ &+ \frac{(1-\omega)\gamma^2(\widehat{\lambda} - \lambda_k)}{(\delta - \widehat{\lambda})(\delta - \lambda_k)} \end{aligned}$$

$$\begin{aligned}
&= \phi(\lambda_k) + (\widehat{\lambda} - \lambda_k) x_k^T x_{k-1} + \frac{\gamma^2(\widehat{\lambda} - \lambda_k)(\widehat{\lambda} - \lambda_{k-1})}{(\delta - \widehat{\lambda})(\delta - \lambda_k)(\delta - \lambda_{k-1})} \\
&= \phi(\lambda_k) + (\widehat{\lambda} - \lambda_k) x_k^T x_{k-1} \\
&+ \frac{\|x_{k-1}\|\|x_k\|(\|x_k\| - \|x_{k-1}\|)(\widehat{\lambda} - \lambda_k)(\widehat{\lambda} - \lambda_{k-1})}{(\omega\|x_k\| + (1-\omega)\|x_{k-1}\|)(\lambda_k - \lambda_{k-1})},
\end{aligned}$$

where $\omega = \dfrac{\lambda_{k-1} - \widehat{\lambda}}{\lambda_k - \lambda_{k-1}}$. Thus, the result follows from Lemma 5.5. □

A few comments are in order concerning the near hard case. As mentioned in Section 3, finding $\lambda_* < \delta_1$ in a near hard case is a very ill–conditioned process. The difference $\delta_1 - \lambda_*$ can be very small to the extent of being undetectable within the given tolerances. The smaller the value $\delta_1 - \lambda_*$, the harder it is to determine $\{\lambda_*, x_*\}$. Furthermore, rounding errors generally will convert an exact into a near hard case. Although δ_1 is still a pole of ϕ when g is not exactly orthogonal to $\mathcal{S}_1$, the weight of such pole is very small in comparison to the other poles because the expansion coefficients of g in the basis of eigenvectors of A are practically zero for those eigenvectors associated with δ_1. The strategy that we follow in Algorithm 4.1 for dealing with this case consists in building an interpolating function that *ignores* the pole δ_1 at early stages, using the eigenpair corresponding to the second smallest eigenvalue of B_{α_k} to obtain the interpolation points. In addition, we use the second eigenpair to compute a vector that might be a quasi–optimal solution for the trust–region subproblem as established in Theorem 3.2. Moreover, as that theorem and related results established, it is not necessary to compute an eigenpair corresponding to the second smallest eigenvalue. This is especially useful when the vector g is orthogonal or nearly orthogonal to several eigenspaces corresponding to the smallest eigenvalues of A and those eigenvalues are clustered.

If we use information concerning a second eigenpair then we will have $\lambda_k > \delta_1$. This occurs because the first component ν_1 of the eigenvector $(\nu_1, u_2^T)^T$ associated with $\lambda_1(\alpha_k)$ is too small, so that $\|u_1/\nu_1\| = \|x_k\|$ becomes excessively large. Therefore $\{\lambda_k, x_k\}$ is defined as $\{\lambda_i(\alpha_k), u_2/\nu_2\}$. Intuitively, this is a good strategy since in the exact hard case this would continuously select the correct eigenvector that will approach $(1, p_1^T)^T$ when α tends to the value $\tilde{\alpha}_1$ stated in Lemma 3.2 from either side.

Now, at iteration k the parameter α_k is updated as $\alpha_{k+1} = \widehat{\lambda} + \widehat{\phi}(\widehat{\lambda})$ with $\widehat{\lambda} \le \delta_U$, where either $\widehat{\lambda} < \delta_U$, $\widehat{\phi}'(\widehat{\lambda}) = \Delta^2$ or $\widehat{\lambda} = \delta_U$, $\widehat{\phi}'(\delta_U) < \Delta^2$. By the same arguments of the proof of case 1 in Lemma 5.6, there exists a neighborhood $\mathcal{B}$ of λ_* such that if $\lambda_{k-1}, \lambda_k \in \mathcal{B}$ then $\widehat{\lambda} \in \mathcal{B}$, with $|\widehat{\lambda} - \lambda_k| = \vartheta|\lambda_* - \lambda_k|$, for $\vartheta \in (0,1)$. In other words, eventually the safeguarding $\widehat{\lambda} = \delta_U$ is no longer necessary. If $\lambda_{k-1}, \lambda_k \in \mathcal{B}$ then Lemma 5.7 implies that $|\widehat{\phi}(\widehat{\lambda}) - \phi(\lambda_k)| = \vartheta|\widehat{\lambda} - \lambda_k||\lambda_* - \lambda_k|$. The agreement between $\widehat{\lambda}$ and λ_k and between $\widehat{\phi}(\widehat{\lambda})$ and $\phi(\lambda_k)$ drive α_k towards $\alpha_* = \lambda_* + \phi(\lambda_*)$. As α_k approaches α_*, the reduction of the safeguarding interval $[\alpha_L, \alpha_U]$ at every iteration provides a means to avoid the numerical difficulties associated with a near hard case and eventually there is no need to use a second eigenpair of B_{α_k}. At early stages, however, it might be that $\widehat{\lambda} = \delta_U$. Although $\phi(\delta_1)$ is infinite, the interpolating function value $\widehat{\phi}(\delta_U)$ is finite. Using $\alpha_{k+1} = \delta_U + \widehat{\phi}(\delta_U)$ is essential in keeping the process under control.

5.3 Global Convergence

Theorem 5.2 *Algorithm 4.1 is globally convergent.*

Proof. The goal of Algorithm 4.1 is to solve the trust–region subproblem by either determining the existence of an interior solution, or by computing an optimal value α_* for the parameter α, such that the solution to the parameterized eigenvalue problem for B_{α_*} can be used to compute a boundary solution for the trust–region subproblem. The global convergence of Algorithm 4.1 is achieved by keeping α_k in an interval that contains the optimal parameter α_*.
We first recall that the initial safeguarding interval $[\alpha_L, \alpha_U]$ contains the optimal value α_*. Starting with that interval, the updating procedure for α_L and α_U, guarantees that α_* remains in the interval and that the safeguarding interval is reduced at each iteration. Therefore, since $\alpha_k = \lambda_k - g^T x_k$, after a finite number of iterations either the sequence $\{\lambda_k\}$ reaches the neighborhood of λ_* of Theorem 5.1 that guarantees convergence, or the length of the safeguarding interval $|\alpha_U - \alpha_L|$ goes to zero with $\alpha_L \leq \alpha_* \leq \alpha_U$. □

6 Numerical Experiments

In this section we present numerical experiments to demonstrate the viability of our approach and to illustrate different aspects of our method. We implemented Algorithm 4.1 (LSTRS) in `MATLAB 5.3` using a Mexfile interface to access the Implicitly Restarted Lanczos Method (IRLM) [15] implemented in `ARPACK` [6]. We ran our experiments on a SUN Ultrasparc 10 with a 300 MHZ processor and 256 Megabytes of RAM, running Solaris 5.6. The floating point arithmetic was IEEE standard double precision with machine precision $2^{-52} \approx 2.2204 \cdot 10^{-16}$. We present five sets of experiments. In the first and second sets we study the sensitivity of LSTRS to different tolerances for the trust–region radius and to different sizes of the trust–region radius, respectively, for problems where the hard case is not present. In order to put our method in context, we include the number of matrix–vector products required by the Conjugate Gradient Method to solve systems of the form $(A - \lambda I)x = -g$. The third set of experiments illustrates the local superlinear rate of convergence. The fourth set shows the behavior of LSTRS in the hard case. In the fifth set we provide a comparison with the semidefinite programming approach presented in [12].

The following tolerances are fixed in all the experiments. $\varepsilon_\nu = 10^{-2}$, $\varepsilon_\alpha = 10^{-8}$, $\varepsilon_{Int} = 10^{-8}$. We will indicate the values for the rest of the parameters when we describe each particular set of experiments.

6.1 Different Tolerances

In the first experiment, we show the behavior of the method when different levels of accuracies of the norm of the trust–region solution are required. The matrix A in (1) was $A = L - 5I$, where L is the standard 2–D discrete Laplacian on the unit square based upon a 5–point stencil with equally–spaced mesh points. The shift of $-5I$ was introduced to make A indefinite. The order of A was $n = 1024$. We solved a sequence of twenty related problems, differing only by the vector g, randomly generated with entries uniformly distributed on $(0, 1)$. We solved each of these problems for a fixed trust–region radius $\Delta = 100$ and for $\varepsilon_\Delta = 10^{-4}, 10^{-6}, 10^{-8}$, where ε_Δ is the relative accuracy of the norm of the computed solution with respect to Δ. The initial δ_U was the minimum of the diagonal of A and $\alpha_0 = \delta_U$. The tolerance for a quasi–optimal solution was set to $\varepsilon_{HC} = 10^{-16}$ in order to allow the method to compute a boundary solution, otherwise the quasi–optimal stopping criterion would be satisfied first.

For $\varepsilon_\Delta = 10^{-4}, 10^{-6}$ the number of Lanczos basis vectors was limited to nine and six shifts (i.e. six matrix-vector products) were applied on each implicit restart, while for $\varepsilon_\Delta = 10^{-8}$, the number of vectors was twenty with fourteen shifts on each implicit restart. The maximum number of restarts allowed was forty five for $\varepsilon_\Delta = 10^{-4}, 10^{-6}$ and one hundred for $\varepsilon_\Delta = 10^{-8}$. More basis vectors were needed for $\varepsilon_\Delta = 10^{-8}$, since in this case the eigenvalues were computed to a higher accuracy. We chose v_1, the initial vector for the IRLM, in the following way. In the first iteration of LSTRS, $v_1 = (1, 1, \ldots, 1)/\sqrt{n+1}$ and subsequently, v_1 was the first column of the matrix V containing the Lanczos vectors computed by the IRLM for the previous bordered matrix. This choice standardized the initial vector along the set of tests and performed better than a randomly generated vector, or the eigenvector corresponding to the smallest eigenvalue of B_{α_k}, or the vector $(0, g^T)^T$. Note that the last two options have the additional disadvantage of preventing the IRLM from finding the eigenspace of B_α corresponding to δ_1 whenever a potential hard case is present. As in [16] we relaxed the accuracy required in the eigenvalue solution and made it proportional to the relative accuracy in the computed solution. Specifically, $\|B_\alpha q - q\lambda\| < \varepsilon_{Lan}$, where $\varepsilon_{Lan} = \max\{\min\{\varepsilon_{Lan}, |\frac{\Delta - \|x\|}{\Delta}|\}, \varepsilon_{max}\}$ and $\varepsilon_{max} = 0.125, 0.1, 0.075$ for $\varepsilon_\Delta = 10^{-4},\ 10^{-6},\ 10^{-8}$, respectively.

In Table 1 we report the average number of iterations of LSTRS (LSTRS IT), the average number of matrix–vector products required by LSTRS (LSTRS MV) and the average number of matrix–vector products required by the Conjugate Gradient Method (CG MV) to solve the system $(A - \lambda_* I)x = -g$ to the same accuracy ε_Δ in the norm of the computed solution of LSTRS. The value of λ_* was the optimal value computed by LSTRS.

ε_Δ	LSTRS IT	LSTRS MV	CG MV	$\frac{LSTRS\ MV}{CG\ MV}$
10^{-4}	6.00	64.00	43.45	1.47
10^{-6}	7.50	83.00	57.00	1.46
10^{-8}	6.55	183.40	71.55	2.56

Table 1: Average behavior for different tolerances

We observe that for $\varepsilon_\Delta = 10^{-4}, 10^{-6}$ the behavior in [16] is reproduced: a trust–region solution requires fewer than twice as many matrix–vectors products on average than the number needed to solve a *single* linear system to the same accuracy using conjugate gradients. For $\varepsilon_\Delta = 10^{-8}$, even though LSTRS requires more matrix–vector products, the cost of LSTRS is less than three times the cost of solving one system by conjugate gradients.

If we repeat the experiment setting the tolerance for a quasi–optimal solution to $\varepsilon_{HC} = 10^{-6}$, we obtain the results in Table 2, where we observe the low number of matrix–vector products required by LSTRS. In this experiment we used nine Lanczos basis vectors for all cases and allowed a maximum of forty five restarts.

ε_Δ	LSTRS IT	LSTRS MV	CG MV	$\frac{LSTRS\ MV}{CG\ MV}$
10^{-4}	5.00	54.00	48.80	1.11
10^{-6}	5.40	62.00	62.90	0.99
10^{-8}	5.40	64.75	76.95	0.84

Table 2: Average behavior for different tolerances allowing quasi–optimal solutions

6.2 Different Trust–Region Radii

The second experiment illustrates the behavior of LSTRS for different sizes of the trust–region radius. The matrix A in (1) was of the form $A = UDU^T$ with D diagonal and $U = I - 2uu^T$, $u^T u = 1$. The elements of D were randomly selected from a uniform distribution on $(-5, 5)$. Both vectors u and g were randomly generated with entries uniformly distributed on $(-0.5, 0.5)$ and then u was normalized to have unit length. The order of A was $n = 1000$. We solved a sequence of ten problems generated with different seeds, for a fixed tolerance $\varepsilon_\Delta = 10^{-6}$ and Δ varying from 100 to 0.0001 by a factor of 10, for a total of seventy problems. The initial δ_U was set to -4.5 and $\alpha_0 = \min\{0, \alpha_U\}$. The tolerance for a quasi–optimal solution was set to $\varepsilon_{HC} = 10^{-6}$.

The parameters for the IRLM were the following. For $\Delta = 100, 10$ the number of Lanczos basis vectors was thirty and twenty shifts were applied on each implicit restart, while for $\Delta \leq 1$, the number of vectors was nine with six shifts on each implicit restart. The maximum number of restarts was one hundred and fifty and forty five, respectively. The difference in the number of basis vectors is due to the fact that for larger radii the hard case and near hard case are more likely to occur and therefore the smallest eigenvalues of the bordered matrix become more clustered and the IRLM needs more space and iterations to compute the desired eigenpairs to the required accuracy. The initial vector for the IRLM was chosen as in §6.1. We relaxed the accuracy required in the eigenvalue solution in the following way. The initial values for ε_{Lan} was $0.03, 0.1$ and 0.25 for $\Delta = 100, 10$ and $\Delta < 10$, respectively. The value of ε_{Lan} was kept the same until $\left|\frac{\Delta - \|x_k\|}{\Delta}\right| < 0.1$, when $\varepsilon_{Lan} = 0.015, 0.05$ and 0.125 for $\Delta = 100, 10$ and $\Delta < 10$, respectively. The results of the experiment are shown in Table 3.

Δ	100	10	1	0.1	0.01	0.001	0.0001
IT	9.9	7.7	4.6	4	3.8	3	3
LSTR MV	1032.4	354	46.9	38.8	37.2	29.4	29.4
CG MV	921.4	410	29.3	28.7	27.2	12	12.5
$\frac{\|g+(A-\lambda_* I)x_*\|}{\|g\|}$	10^{-3}	10^{-3}	10^{-2}	10^{-11}	10^{-2}	10^{-2}	10^{-13}
$\left\|\frac{\Delta-\|x_*\|}{\Delta}\right\|$	10^{-16}	10^{-16}	10^{-8}	10^{-9}	10^{-12}	10^{-10}	10^{-7}

Table 3: Average behavior for different trust–region radii

As observed in [16], the Conjugate Gradient Method has a much easier time for smaller values of Δ.

6.3 Superlinear Convergence

The purpose of the third experiment was to verify superlinear convergence. The matrix A was again set to $A = L - 5I$ with L the 2–D discrete Laplacian on the unit square, but now $n = 256$. The vector g was randomly generated with entries uniformly distributed on $(-0.5, 0.5)$. We studied problems with and without hard case. To generate the hard case, we orthogonalized the vector g randomly generated as before against the eigenvector q corresponding to the smallest eigenvalue of A. We accomplished this by setting $g \leftarrow g - q(q^T g)$. For the problem without hard case the trust–region radius was $\Delta = 10$ and $\varepsilon_\Delta = 10^{-11}$. For the problem with hard case the radius was $\Delta = 100$ and $\varepsilon_{HC} = 10^{-11}$. The eigenproblems were solved with the `MATLAB` routine `eig`. The results are shown in Table 4, were we report the quantity $\left|\frac{\Delta-\|x_k\|}{\Delta}\right|$ for the problem without hard case, and the quantity $\frac{(\lambda_i(\alpha)-\lambda_1(\alpha))\ \tau_2^2\ (1+\Delta^2)}{-2\eta\psi(\widetilde{x})}$ from Theorem 3.2, for the problem with hard case.

k	$\frac{\Delta - \|x_k\|}{\Delta}$
0	8.739485e-01
1	1.101152e+01
2	7.790406e-01
3	5.987336e-01
4	1.247129e-01
5	2.593978e-02
6	3.410990e-04
7	1.038581e-06
8	4.049703e-11
9	8.704149e-15

(a)

k	$\frac{(\lambda_i(\alpha_k)-\lambda_1(\alpha_k))\ \tau_2^2\ (1+\Delta^2)}{-2\eta\psi(\widetilde{x})}$
0	1.694375e-01
1	*
2	4.112269e-02
3	9.276102e-03
4	6.306448e-04
5	5.851597e-06
6	4.159997e-09
7	2.116485e-09
8	7.976599e-10
9	1.267130e-12

(b)

Table 4: Verification of superlinear convergence for problems without hard case (a) and with hard case (b)

The quantity $\|(A - \lambda_* I)x_* + g\|/\|g\|$ was of order 10^{-14} for problem (a) and 10^{-7} for problem (b). A $*$ in the hard case means that we could not check for a quasi–optimal solution since the conditions of Lemma 3.5 were not satisfied.

6.4 The Hard Case

The fourth experiment illustrates the behavior of the method in the hard case. The matrix A was of the form $A = UDU^T$, with $D = diag(d_1, \ldots, d_n)$ and $U = I - 2uu^T$, $u^T u = 1$. The elements of D were randomly generated with a uniform distribution on $(-5, 5)$, then sorted in nondecreasing order and set $d_i \equiv -5$ for $i = 1, 2, \ldots, \ell$, allowing multiplicity ℓ for the smallest eigenvalue of A. Both vectors u and g were randomly generated with entries selected from a uniform distribution on $(-0.5, 0.5)$ and then u was normalized to have unit length. The order of A was $n = 1000$.

In this case, the eigenvectors of the matrix A are of the form $q_i = e_i - 2uu_i$, $i = 1, \ldots, n$ with e_i the i–th canonical vector in $\mathbb{R}^n$ and u_i the i–th component of the vector u. This provides complete control in the generation of the hard case. In fact, if $\ell = 1$ the vector g was orthogonalized against q_1 computed by the formula given above. For $\ell > 1$, g was computed as the sum of the vectors in an orthonormal basis for the orthogonal complement of $\mathcal{S}_1$. After this, a noise vector s was added to g and $g \leftarrow \frac{g+s}{\|g+s\|}$. Both hard case and near hard case were generated by adding noise vectors of norms 10^{-8} and 10^{-2}, respectively. To ensure that the hard case really occurred, we computed $\Delta_{min} = \|(A - d_1 I)^\dagger g\|$ and set $\Delta = 2\Delta_{min}$.

The problems were solved to the level $\varepsilon_{HC} = 10^{-6}$. The initial δ_U was set to -4.5 and $\alpha_0 = \min\{0, \alpha_U\}$.

The parameters for the IRLM were chosen as follows. For the hard case, nine Lanczos basis vectors with six shifts on each implicit restart and a maximum of forty five restarts. For the near hard case, eighteen Lanczos basis vectors with twelve shifts on each implicit restart and a maximum of ninety restarts. The different number of basis vectors is due to the fact that in the near hard case the smallest eigenvalues of the bordered matrix become more clustered and the IRLM needs more space in order to compute the desired eigenpairs. The tolerance ε_{Lan} was fixed at 10^{-2}.

In Table 5 (a) and (b) we summarize the average results for a sequence of ten problems generated with different seeds, for problems with hard case and near hard case, respectively.

ℓ	MV	IT	$\frac{\|(A-\lambda_* I)x_*+g\|}{\|g\|}$
1	683.9	9.6	10^{-2}
5	790.4	12.1	10^{-2}
10	1301.9	22.6	10^{-2}

(a)

ℓ	MV	IT	$\frac{\|(A-\lambda_* I)x_*+g\|}{\|g\|}$
1	1153.5	10	10^{-3}
5	1039.9	9.7	10^{-3}
10	1063.5	9.7	10^{-3}

(b)

Table 5: (a) The Hard Case and (b) Near Hard Case, when $\mathcal{S}_1$ has dimension $\ell \geq 1$.

6.5 Comparison with the semidefinite programming approach

Finally, we compared LSTRS with the semidefinite programming approach of [12]. In this experiment, we solved two different families of problems. For each family, we generated ten problems of each type (easy and hard case) with different seeds and solved them with Algorithm 4.1 (LSTRS) and the semidefinite programming approach (SDP) of [12]. In both implementations the eigenproblems were solved by the function `eig` of `MATLAB` so that the methods worked with eigenpairs with the same level of accuracy, and also to avoid the inconsistencies associated with having two different eigensolvers. We report average number of iterations (IT), average magnitude of the residual $\|(A - \lambda_* I)x_* + g\|/\|g\|$ and average relative accuracy in the norm of the trust–region solution, $|\Delta - \|x_*\||/\Delta$. Since we were using the function `eig` as eigensolver, we are also reporting the average number of calls to the eigensolver (SOLVES) to provide a means of comparing the amount of work needed by each method. It is important to point out that in large–scale applications the computational effort will concentrate on solving the eigenvalue problems and therefore in such situations we should also compare the cost of solving each eigenvalue problem.

In the first family of problems, the matrix A was $A = L - 5I$ of order $n = 256$ and the vector g was randomly generated with entries uniformly distributed on $(0, 1)$. As in §6.3 we orthogonalized g against the eigenvector of A corresponding to δ_1 to generate the hard case. For both easy and hard case we added a noise vector to g, of norm 10^{-8}. The trust–region radius was $\Delta = 100$. We used $\varepsilon_\Delta = 10^{-6}$ and we ran the experiments with $\varepsilon_{HC} = 10^{-8}$ and $\varepsilon_{HC} = 10^{-6}$. We report these results in Tables 6 and 7, respectively.

			IT	SOLVES	$\frac{\|g+(A-\lambda_* I)x_*\|}{\|g\|}$	$\frac{\Delta-\|x_*\|}{\Delta}$
$A = L - 5I$	easy case	LSTRS	5.0	5.0	10^{-13}	10^{-7}
		SDP	4.8	5.8	10^{-3}	10^{-3}
	hard case	LSTRS	8.0	9.7	10^{-9}	10^{-16}
		SDP	9.1	10.1	10^{-7}	10^{-7}

Table 6: Comparison with Semidefinite Programming approach. First set of problems, $\varepsilon_{HC} = 10^{-8}$.

			IT	SOLVES	$\frac{\|g+(A-\lambda_* I)x_*\|}{\|g\|}$	$\frac{\Delta-\|x_*\|}{\Delta}$
$A = L - 5I$	easy case	LSTRS	4.5	4.5	10^{-2}	10^{-8}
		SDP	4.8	5.8	10^{-3}	10^{-3}
	hard case	LSTRS	7.0	8.7	10^{-7}	10^{-16}
		SDP	9.1	10.1	10^{-7}	10^{-7}

Table 7: Comparison with Semidefinite Programming approach. First set of problems, $\varepsilon_{HC} = 10^{-6}$.

In the second family of problems A, g and Δ_{min} were generated exactly as in §6.4, where $A = UDU^T$ of order $n = 256$. For the easy case, $\Delta = 0.1\Delta_{min}$ and for the hard case $\Delta = 5\Delta_{min}$. The tolerances used for Algorithm 4.1 were $\varepsilon_\Delta = 10^{-6}$ and $\varepsilon_{HC} = 10^{-6}$. The results are reported in Table 8.

The previous tests indicate a marginal advantage to our algorithm in most cases. We believe this is partially due to the fact that in the SDP approach it is necessary to compute the smallest eigenvalue of A in order to begin the major iteration, while our approach avoids this extra calculation. From the comparative results, we can see that LSTRS obtained solutions with improved feasibility over the ones computed by the semidefinite programming approach. Moreover, LSTRS required slightly less computational effort overall to compute the solutions, especially in the hard case.

			IT	SOLVES	$\frac{\|g+(A-\lambda_* I)x_*\|}{\|g\|}$	$\frac{\Delta-\|x_*\|}{\Delta}$
$A = UDU^T$	easy	LSTRS	7.8	8.7	10^{-3}	10^{-14}
	case	SDP	4.4	5.4	10^{-4}	10^{-4}
	hard	LSTRS	6.4	12.5	10^{-3}	10^{-8}
	case	SDP	13.8	14.8	10^{-5}	10^{-5}

Table 8: Comparison with Semidefinite Programming approach. Second set of problems.

7 Conclusions

We have presented a new algorithm for the large–scale trust–region subproblem. The algorithm is based upon embedding the trust–region problem into a family of parameterized eigenvalue problems as developed in [16]. The main contribution of this paper has been to give a better understanding of the hard–case condition and to utilize this understanding to develop a better treatment of this case. As a result, we have designed a unified algorithm that naturally incorporates both the standard and hard cases.

We have proved that the iterates for this new algorithm converge and that the rate of convergence is superlinear, and we have demonstrated this computationally for both the standard and hard cases. This result represents a major improvement over the performance of the method originally presented in [16]. That approach used a different iteration for the hard case that was linearly convergent. In practice this behavior seemed to occur often and greatly detracted from the performance. It is worthwhile stressing that in the hard case, the algorithm presented here produces a sequence $\{\lambda_k\}$ which converges superlinearly to the optimal multiplier $\lambda_* = \delta_1$ and a sequence $\{x_k\}$ which converges superlinearly to the vector $p_1 = -(A - \delta_1 I)^{\dagger} g$. Note that, although p_1 is not the solution of the trust–region subproblem, it can be used to compute a nearly optimal solution by means of Theorem 3.2. We also compared our method to the semidefinite programming approach presented in [12], obtaining better results in terms of feasibility.

Our motivation for developing the LSTRS method came from some important large–scale applications. In particular, the regularization of ill–posed problems such as those arising in seismic inversion [20] provides an important class of trust–region subproblems. It was shown in [13] that near hard cases are the common situation for this class of problems where the vector g is nearly orthogonal to eigenspaces corresponding to several of the smallest eigenvalues of A. The work in [13] also reports the successful application of LSTRS to the regularization of discrete forms of ill–posed problems from inverse problems, including problems with real data. Further work in this area should include the use of LSTRS within a trust–region method for the solution of large–scale optimization problems.

Acknowledgements: We would like to thank Trond Steihaug for insightful discussions about this material. M. Rojas in particular would like to thank Professor Steihaug for arranging an extended visit to the University of Bergen and for the support received there.

References

[1] W. Gander, G.H. Golub and U. von Matt. A Constrained Eigenvalue Problem, *NATO ASI Series Vol. F70 Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms*, eds. G.H. Golub and P. Van Dooren, Springer–Verlag, Berlin Heidelberg, 1991.

[2] G.H. Golub. Some modified matrix eigenvalue problems, *SIAM Rev.*, 15(2):318–334, 1973.

[3] G.H. Golub and U. von Matt. Quadratically constrained least squares and quadratic problems, *Numer. Math.*, 59:561–580, 1991.

[4] W.W. Hager. Minimizing a quadratic over a sphere. Draft. 1999

[5] M.D. Hebden. An algorithm for minimization using exact second derivatives. Technical Report T.P. 515, Atomic Energy Research Establishment, Harwell, England, 1973.

[6] R.B. Lehoucq, D.C. Sorensen and C. Yang. *ARPACK User's Guide: Solution of Large Scale Eigenvalue Problems by Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, 1998.

[7] S. Lucidi, L. Palagi and M. Roma. On some properties of quadratic programs with a convex quadratic constraint, *SIAM J. Optim.*, 8:105–122, 1998.

[8] A. Melman. Numerical solution of a secular equation, *Numer. Math.*, 69:483–493, 1995.

[9] J.J. Moré and D.C. Sorensen. Computing a Trust Region Step, *SIAM J. Sci. Stat. Comput.*, 4(3):553–572, 1983.

[10] T. Pham Dinh and L.T. Hoai An. A D.C. Optimization Algorithm for Solving the Trust–Region Subproblem, *SIAM J. Optim.*, 8(2):476–505, 1998.

[11] B.N. Parlett. *The Symmetric Eigenvalue Problem*, Prentice–Hall, Englewood Cliffs NY, 1980.

[12] F. Rendl and H. Wolkowicz. A Semidefinite Framework for Trust Region Subproblems with Applications to Large Scale Minimization, *Math. Programming*, 77(2):273–299, 1997.

[13] M. Rojas. A Large–Scale Trust–Region Approach to the Regularization of Discrete Ill–Posed Problems. Ph.D. Thesis, Technical Report TR98–19, Department of Computational and Applied Mathematics, Rice University, Houston. May 1998.

[14] D.C. Sorensen. Newton's Method with a Model Trust Region Modification, *SIAM J. Numer. Anal.*, 19(2):409–426, 1982.

[15] D.C. Sorensen. Implicit Application of Polynomial Filters in a K–Step Arnoldi Method, *SIAM J. Matrix Anal. Appl.*, 13(1):357–385, 1992.

[16] D.C. Sorensen. Minimization of a Large–Scale Quadratic Function Subject to a Spherical Constraint, *SIAM J. Optim.*, 7(1):141–161, 1997.

[17] T. Steihaug. The Conjugate Gradient Method and Trust Regions in Large Scale Optimization, *SIAM J. Numer. Anal.*, 20(3):626–637, 1983.

[18] R. Stern and H. Wolkowicz. Indefinite trust region subproblems and non-symmetric eigenvalue perturbations, *SIAM J. Optim.*, 5(2):286–316, 1995.

[19] R.J. Stern and J.J. Ye. Variational analysis of an extended eigenvalue problem, *Lin. Alg. Appl.*, 220:391–418,1995.

[20] W.W. Symes. A Differential Semblance Criterion for Inversion of Multioffset Seismic Reflection Data, *Journal of Geophysical Research*, 98:2061–2073, 1993.