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

Sandra A. Santos
Danny C. Sorensen

July 1995

TR95-20

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

Sandra A. Santos *

Danny C. Sorensen †

June 30, 1995

Abstract

The trust–region subproblem arises frequently in linear algebra and optimization applications. Recently, matrix–free methods have been introduced to solve large–scale trust–region subproblems. These methods only require a matrix–vector product and do not rely on matrix factorizations [4, 7]. These approaches recast the trust–region subproblem in terms of a parameterized eigenvalue problem and then adjust the parameter to find the optimal solution from the eigenvector corresponding to the smallest eigenvalue of the parameterized eigenvalue problem. This paper presents a new matrix–free algorithm for the large–scale trust–region subproblem. The new algorithm improves upon the previous algorithms by introducing a unified iteration that naturally includes the so called hard case. The new iteration is shown to be superlinearly convergent in all cases. Computational results are presented to illustrate convergence properties and robustness of the method.

AMS classification: Primary 65F15; Secondary 65G05

*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 (augusta@rice.edu). This author was supported by FAPESP (93/4907-5).

†Dept. of Computational and Applied Mathematics, Rice University, P.O. BOX 1892, Houston, TX 77251-1892, USA (sorensen@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).

Key words and phrases: Regularization, constrained quadratic optimization, trust region, Lanczos method.

1 Introduction

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

$$\begin{aligned} \min \quad & \frac{1}{2}x^T Ax + g^T x \\ \text{s.t.} \quad & \|Cx\| \leq \Delta \,, \end{aligned}$$

where $A \in I\!R^{n\times n}$, $A = A^T$, $x, g \in I\!R^n$, $C \in I\!R^{n\times n}$ is nonsingular, $\Delta \in I\!R_+$, and $\|\cdot\|$ is the Euclidean norm. Here $I\!R_+$ denotes the set of the nonnegative real numbers. There are several applications for this basic problem. Two significant examples are the regularization or smoothing of discrete forms of ill–posed problems and the trust–region mechanism used to force convergence in optimization methods.

A solution x_* to the problem must satisfy a relation of the form $(A+\mu C^T C)x_* = -g$ with $\mu \geq 0$. The parameter μ is the regularization parameter for ill–posed problems and the Levenberg–Marquardt parameter in optimization. The matrix 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. With a change of variables, we can assume that $C = I$, the identity matrix, and this will be the case in the following discussion.

If positive–definite matrices of the form $A + \mu I$ can be decomposed into a Cholesky factorization, then the method proposed by Moré and Sorensen (cf. [2]) can be used to solve the problem. In several important applications, factoring or even forming these matrices is out of the question. This has motivated the recent development of conjugate–gradient style matrix–free methods that only require matrix–vector products. The recent works of Sorensen [7] and Rendl & Wolkowicz [4] provide such algorithms. Both approaches recast the trust–region subproblem in terms of a parameterized eigenvalue problem and rely upon matrix–vector products. Sorensen's algorithm provides a superlinearly convergent scheme to adjust the parameter and find the optimal vector x_* from the eigenvector of the

parameterized problem, as long as the *hard case* does not occur. The so–called hard case is characterized whenever the vector g is orthogonal to the eigenspace of A corresponding to its smallest eigenvalue δ_1, and the solution p to the system $(A - \delta_1 I)p = -g$ is such that $\|p\| < \Delta$. For the hard case, Sorensen's algorithm is linearly convergent. The recommended technique used in [7] to find the smallest eigenvalue and corresponding eigenvector of the parameterized problem is the *Implicitly Restarted Lanczos Method* (cf. [6]), which meets the requirements of limited storage and reliance only on matrix–vector products. Rendl & Wolkowicz present a primal–dual semidefinite framework for the trust–region subproblem, where a dual simplex–type method is used in the general iteration and a primal simplex–type method provides steps for the hard–case iteration. In their work the calculation of the smallest eigenvalue and corresponding eigenvector of the parameterized problem, required at each iteration, is done using a block Lanczos routine.

The purpose of this work is to present a new matrix–free algorithm for solving the large–scale trust–region subproblem. Our algorithm is similar to those proposed in [4, 7] in the sense that the trust–region subproblem is solved through a parametric eigenvalue problem. However, our algorithm is able to cope with the hard case naturally in the same basic iteration. It is not necessary to switch between two fundamentally different schemes when a potential hard case is present. This improved scheme is based upon computing the two smallest eigenvalues and corresponding eigenvectors of the parameterized problem and the information concerning the second smallest eigenpair is incorporated whenever it is appropriate. This does not increase the work or storage required in any substantial way over the method proposed in [7]. The iteration we propose is based upon a two–point interpolating scheme that is different from [7]. We show this new iteration is also superlinearly convergent. Moreover, our convergence results include the hard case naturally, since no special iterations are performed. Such a unified approach is not achieved in either [4] or [7].

This work is organized as follows. In §2 we analyze the structure of the problem and related results. There we give a complete characterization of the hard case with respect to the parameterized bordered eigenproblems. The detailed algorithm is presented in §3.

The local convergence analysis is developed in §4. Preliminary numerical experiments are described in §5. Finally, in §6 we establish some conclusions and ideas for future research.

2 Structure of the Problem

The trust–region subproblem has a tremendous amount of structure which must be exploited to design an efficient algorithm. The problem we are interested in solving is

$$\begin{aligned} \min \quad & \tfrac{1}{2}x^T Ax + g^T x \\ \text{s.t.} \quad & \|x\| \leq \Delta \, . \end{aligned} \tag{1}$$

Due to the structure of (1), its optimality conditions are both necessary and sufficient, as stated in the next lemma, where we follow [7] in the non–standard but more convenient use of a non–positive multiplier.

Lemma 1: *A feasible vector x_* is a solution to (1) if and only if x_* is a solution to an equation of the form $(A - \lambda_* I)x_* = -g$ with $A - \lambda_* I$ positive semidefinite, $\lambda_* \leq 0$ and $\lambda_*(\Delta - \|x_*\|) = 0$.*

Proof: See [5]. □

The optimality conditions of (1) 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. One possibility is to define the function $\varphi(\lambda) = \|(A - \lambda I)^{-1} g\|$ and to solve $\varphi(\lambda) = \Delta$, monitoring λ to be no greater than the smallest eigenvalue of A, so that the Cholesky factorization of $A - \lambda I$ is well defined. Applying Newton's method to solve $\dfrac{1}{\varphi(\lambda)} - \dfrac{1}{\Delta} = 0$ has a number of computationally attractive features (cf. [2]) and this is the preferred approach when the Cholesky factorization of $A - \lambda I$ is tractable. When the cost or storage requirements of a Cholesky factorization become prohibitive, a new approach is required. The introduction of another 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. To begin, we observe that

$$\frac{\alpha}{2} + \psi(x) = \frac{1}{2} \begin{pmatrix} 1 & x^T \end{pmatrix} \begin{pmatrix} \alpha & g^T \\ g & A \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix} , \tag{2}$$

where $\psi(x) \equiv \frac{1}{2} x^T A x + g^T x$. As in [7], we denote the bordered matrix appearing on the right of (2) as B_α. Thus, using $y = (1 \; x^T)^T$, and denoting by e_1 the first canonical unit vector of $\mathbb{R}^{n+1}$, problem (1) can be rewritten as

$$\begin{aligned} \min \quad & \tfrac{1}{2} y^T B_\alpha y \\ \text{s.t.} \quad & y^T y \leq 1 + \Delta^2, \; e_1^T y = 1, \end{aligned}$$

which suggests that the desired solution might be found in terms of an eigenpair of B_α. If an eigenvector q corresponding to an eigenvalue λ of B_α can be normalized such that its first component is one, i.e. $q = (1 \; x^T)^T$, then

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

From this it follows that

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

and

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

Hence,

$$\alpha - \lambda = g^T (A - \lambda I)^{-1} g = \sum_{j=1}^{n} \frac{\gamma_j^2}{\delta_j - \lambda} , \tag{5}$$

where $\{\delta_j\}$ are the eigenvalues of A and $\{\gamma_j\}$ are the expansion coefficients of g in the eigenvector basis. The case where the eigenvector of B_α has a zero first component will be analyzed below. We shall denote the eigenvalues of A by δ_j and such that

$$\delta_1 = \cdots = \delta_\ell < \delta_{\ell+1} \leq \cdots \leq \delta_n \tag{6}$$

so that δ_1 is the smallest eigenvalue of A, with multiplicity ℓ. The eigenvalues of B_α shall be denoted by $\{\lambda_j(\alpha)\}$ with $\lambda_1(\alpha) \leq \lambda_2(\alpha) \leq \cdots \leq \lambda_{n+1}(\alpha)$.

As a consequence of Cauchy's interlace theorem (cf. [3], p. 186), the eigenvalues of the matrix A interlace the eigenvalues of the bordered matrix B_α. This can also be seen directly from equation (5). Therefore, the smallest eigenvalue $\lambda_1(\alpha)$ of B_α satisfies $\lambda_1(\alpha) \leq \delta_1$. This assures the matrix $A - \lambda_1(\alpha)I$ is positive semidefinite, regardless of the value of α. Furthermore, $\lambda_1(\alpha)$ is well separated from the rest of the spectrum of B_α for small values of Δ and it is expected that a Lanczos–type algorithm will be successful in computing this extreme eigenpair.

Equations (3)–(4) express λ and hence x implicitly in terms of α, suggesting the definition of a convenient function as follows:

$$\phi(\lambda) \equiv g^T(A - \lambda I)^{-1}g = -g^T x \ ,$$

and so,

$$\phi'(\lambda) = g^T(A - \lambda I)^{-2}g = x^T x \ ,$$

where differentiation is with respect to λ and $(A - \lambda I)x = -g$. Figure 1.a shows the typical behavior of ϕ. It is worthwhile noticing that both ϕ and ϕ' are readily available and contain valuable information with respect to problem (1).

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. If α can be adjusted so the corresponding x satisfies $\phi'(\lambda) = x^T x = \Delta^2$ 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 optimal and solves the trust–region subproblem.

In case problem (1) has an unconstrained minimizer, $\lambda > 0$ will be found with $\|x\| < \Delta$ during the course of adjusting α. As a consequence, A is positive definite and the desired interior solution can be computed using the conjugate–gradient method.

It remains to consider the possibility that the eigenvector of the bordered matrix B_α associated to $\lambda_1(\alpha)$ has first component zero and thus cannot be normalized to have

its first component equal to one. However, this is equivalent to the so–called hard case, analyzed in [2] for problems where the Cholesky factorization for $A - \lambda I$ is affordable and discussed in [4, 7] in the large–scale context. Figure 1.b shows ϕ in presence of the hard case, where one can see that δ_1 is not a pole of ϕ. Next, we state the precise result describing the hard case for problem (1), which can only occur if the vector g is orthogonal to the eigenspace $\mathcal{S}_1 \equiv \{q \mid Aq = q\delta_1\}$ associated to δ_1.

Lemma 2: *Assume g is orthogonal to $\mathcal{S}_1$ and let $p = -(A - \delta_1 I)^{\dagger} g$, where $\dagger$ denotes the Moore–Penrose generalized inverse. If $\delta_1 \leq 0$ and $\|p\| < \Delta$, then the solutions to (1) consist of the set $\{x \mid x = p + z \ , \ z \in \mathcal{S}_1 \ , \ \|x\| = \Delta\}$.*

Proof: See [5]. □

When g is orthogonal to $\mathcal{S}_1$, there is a potential hard–case situation. This condition has an intriguing consequence. In this case it may be impossible to suitably normalize the eigenvector of interest. For all values of α greater than a certain critical value $\widetilde{\alpha}$, the eigenvectors corresponding to the smallest eigenvalue of B_α will have a zero first component. However, in the exact hard case, there is a well defined eigenvector depending continuously on α that can be safely normalized for all values of α. When α exceeds the critical value $\widetilde{\alpha}$, this parameterized vector corresponds to the second smallest eigenvalue of B_α. A complete understanding of this case has led to the main algorithm of this paper. This understanding is based upon the following results.

Lemma 3: *For any $\alpha \in I\!R$ and $q \in \mathcal{S}_1$, $\{\delta_1 \ , \ (0 \ q^T)^T\}$ is an eigenpair of B_α if and only if g is orthogonal to $\mathcal{S}_1$.*

Proof: The proof is straightforward since $g \perp \mathcal{S}_1$ and $Aq = q\delta_1$ are equivalent to

$$\begin{pmatrix} \alpha & g^T \\ g & A \end{pmatrix} \begin{pmatrix} 0 \\ q \end{pmatrix} = \begin{pmatrix} 0 \\ q \end{pmatrix} \delta_1 \ ,$$

independently of α. $\square$

The previous lemma establishes that $\dim \mathcal{S}_1 \leq \dim \mathcal{S}_1(\alpha) \equiv \dim \mathcal{N}(B_\alpha - \delta_1 I)$ and shows that $\{(0\ q^T)^T \mid Aq = q\delta_1\} \subset \mathcal{N}(B_\alpha - \delta_1 I)$ where $\mathcal{N}$ denotes the null space. These sets are equal for all but one exceptional value of α. The following result states that there is a unique value of α such that $\dim \mathcal{S}_1(\alpha) = \dim \mathcal{S}_1 + 1$.

Lemma 4: *Suppose that g is orthogonal to $\mathcal{S}_1$ and $p = -(A - \delta_1 I)^\dagger g$. If $\widetilde{\alpha} = \delta_1 - g^T p$ then $\{\delta_1\ ,\ (1\ p^T)^T\}$ is an eigenpair of $B_{\widetilde{\alpha}}$. Moreover, $(1\ p^T)^T$ is orthogonal to $(0\ q^T)^T$, for every $q \in \mathcal{S}_1$.*

Proof: Suppose that $\widetilde{\alpha} = \delta_1 - g^T p$ with $p = -(A - \delta_1 I)^\dagger g$. The assumption $g \perp \mathcal{S}_1$ implies

$$(A - \delta_1 I)p = -(A - \delta_1 I)(A - \delta_1 I)^\dagger g = -g$$

and thus the eigenvalue relation

$$\begin{pmatrix} \widetilde{\alpha} & g^T \\ g & A \end{pmatrix} \begin{pmatrix} 1 \\ p \end{pmatrix} = \begin{pmatrix} 1 \\ p \end{pmatrix} \delta_1$$

follows immediately. Now, if $q \in \mathcal{S}_1$ then

$$\begin{aligned} \begin{pmatrix} 0 & q^T \end{pmatrix} \begin{pmatrix} 1 \\ p \end{pmatrix} &= q^T p = -q^T (A - \delta_1 I)^\dagger g \\ &= q^T (A - \delta_1 I)^\dagger (A - \delta_1 I) p = q^T (A - \delta_1 I)(A - \delta_1 I)^\dagger p = 0\ , \end{aligned}$$

since $(A - \delta_1 I)q = 0$. $\square$

The result in Lemma 3 was also stated in [7] and the idea behind Lemma 4 was presented in [4]. These results are the heart of the algorithm developed in the next section since they provide the necessary tools for handling the hard case in the same iteration designed for the general case. The next corollary summarizes the main results from Lemmas 3 and 4.

Corollary: *Suppose that g is orthogonal to $\mathcal{S}_1$. If $\widetilde{\alpha} = \delta_1 + g^T(A-\delta_1 I)^\dagger g$ then* $\dim \mathcal{S}_1(\widetilde{\alpha}) = \dim \mathcal{S}_1 + 1$ *and for any other value of* α, $\dim \mathcal{S}_1(\alpha) = \dim \mathcal{S}_1$. *Moreover, if* $\{\ q_1,\ \cdots,\ q_\ell\ \}$ *is an orthogonal basis for* $\mathcal{S}_1$ *then*

$$\left\{ \begin{pmatrix} 1 \\ p \end{pmatrix}, \begin{pmatrix} 0 \\ q_1 \end{pmatrix}, \cdots, \begin{pmatrix} 0 \\ q_\ell \end{pmatrix} \right\}$$

is an orthogonal basis for $\mathcal{S}_1(\widetilde{\alpha})$ *and*

$$\left\{ \begin{pmatrix} 0 \\ q_1 \end{pmatrix}, \cdots, \begin{pmatrix} 0 \\ q_\ell \end{pmatrix} \right\}$$

is an orthogonal basis for $\mathcal{S}_1(\alpha)$, $\alpha \neq \widetilde{\alpha}$.

Numerical difficulties can be expected when the vector g is nearly orthogonal to the eigenspace $\mathcal{S}_1$. If this happens, there still exist $\lambda_* < \delta_1$ 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, one can see that 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, i.e. a small change in data produces a large change in the solution.

3 The Algorithm

Keeping in mind the availability of a well–suited variant of the Lanczos method, namely the *Implicitly Restarted Lanczos Method* (cf. [6]), we will develop a rapidly convergent iteration to adjust α based on this process. Our goal is to fit 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 [7] 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 $\widehat{\phi}'(\widehat{\lambda}) = \Delta^2$. Finally, we use $\widehat{\lambda}$ and $\widehat{\phi}(\widehat{\lambda})$ to update the parameter α and compute the next point $\{\lambda, x\}$. The new elements in our algorithm are the introduction of safeguards for the sequence in α, the usage of the information relative to the second smallest eigenvalue of matrix B_α and the introduction of a different interpolating scheme, where the currently available information is exploited to a greater extent.

3.1 Interpolating Scheme

To begin the iteration, we need a single–point interpolating scheme. As in [7], we use an interpolating function of the form

$$\widehat{\phi}(\lambda) = \frac{\gamma^2}{\delta - \lambda} .$$

Let $\{\lambda_0, x_0\}$ denote the point corresponding to the initial α_0 so that

$$\alpha - \lambda_0 = -g^T x_0 \quad \text{with} \quad (A - \lambda_0 I)x_0 = -g .$$

Requiring $\widehat{\phi}(\lambda_0) = \phi(\lambda_0) = -g^T x_0$ and $\widehat{\phi}'(\lambda_0) = \phi'(\lambda_0) = x_0^T x_0$ leads to a straightforward derivation of expressions for the coefficients δ and γ^2,

$$\delta = \lambda_0 - \frac{g^T x_0}{x_0^T x_0} \quad \text{and} \quad \gamma^2 = \frac{(g^T x_0)^2}{x_0^T x_0} .$$

It is easy to see that $\delta = \dfrac{x_0^T A x_0}{x_0^T x_0}$, which has the desirable feature $\delta_1 \leq \delta$. Computing $\widehat{\lambda}$ such that $\widehat{\phi}'(\widehat{\lambda}) = \Delta^2$ we have

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

and, after a little algebraic manipulation, the updating formula for α is shown to be

$$\alpha_1 = \widehat{\lambda} + \widehat{\phi}(\widehat{\lambda}) = \alpha_0 + \frac{\alpha_0 - \lambda_0}{\|x_0\|}\left(\frac{\Delta - \|x_0\|}{\Delta}\right)\left(\Delta + \frac{1}{\|x_0\|}\right) . \tag{7}$$

This method is linearly convergent and may be slow in some cases, so it will be used just to obtain a second iterate from an initial guess, to provide the starting values needed in a method using two points.

The two–point method is based on using the four pieces of available information at the k-th iteration, namely $\phi(\lambda_{k-1})$, $\phi'(\lambda_{k-1})$, $\phi(\lambda_k)$ and $\phi'(\lambda_k)$. We compute $\widehat{\lambda}$ such that

$$\frac{1}{\Delta} = \frac{1}{\sqrt{\phi'(\lambda_{k-1})}}\left(\frac{\lambda_k - \widehat{\lambda}}{\lambda_k - \lambda_{k-1}}\right) + \frac{1}{\sqrt{\phi'(\lambda_k)}}\left(\frac{\widehat{\lambda} - \lambda_{k-1}}{\lambda_k - \lambda_{k-1}}\right) , \tag{8}$$

obtaining

$$\widehat{\lambda} = \frac{\lambda_{k-1}\|x_{k-1}\|(\|x_k\| - \Delta) + \lambda_k\|x_k\|(\Delta - \|x_{k-1}\|)}{\Delta(\|x_k\| - \|x_{k-1}\|)} . \tag{9}$$

This is equivalent to defining

$$\widehat{\phi}(\lambda) = \frac{\gamma^2}{\delta - \lambda} + \eta \tag{10}$$

for any η and computing $\widehat{\lambda}$ such that $\dfrac{1}{\sqrt{\widehat{\phi}'(\widehat{\lambda})}} = \dfrac{1}{\Delta}$. One can easily verify using (8) that

$$\gamma^2 = \frac{(\lambda_k - \lambda_{k-1})^2\|x_{k-1}\|^2\|x_k\|^2}{(\|x_k\| - \|x_{k-1}\|)^2}$$

and

$$\delta = \frac{\lambda_k\|x_k\| - \lambda_{k-1}\|x_{k-1}\|}{\|x_k\| - \|x_{k-1}\|} .$$

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. Taking advantage of the available values $\phi(\lambda_{k-1})$ and $\phi(\lambda_k)$, we define $\eta_j = \phi(\lambda_j) - \dfrac{\gamma^2}{\delta - \lambda_j}$, $j = k-1$ and $j = k$. Applying the linear interpolation philosophy and defining the weights by means of the already computed value $\widehat{\lambda}$, we choose

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

It is worthwhile noticing that the pole δ in the interpolating function (10) satisfies $\delta > \max\{\lambda_{k-1}, \lambda_k\}$. Therefore, δ is not strongly attached to δ_1, as in [7], but can move to the right towards $\delta_{\ell+1}$ (see (6)), depending on the occurrence or not of a potential exact or near hard case.

As indicated by Lemma 3, an exceedingly small first component ν_1 of the eigenvector $(\nu_1 \;\; u_1^T)^T$ of B_{α_k} corresponding to $\lambda_1(\alpha_k)$ will indicate a potential hard case. However, according to Lemma 4, there will be an eigenvector corresponding to the smallest eigenvalue, $\lambda_1(\alpha_k) = \delta_1$, with a substantial first component precisely when α_k assumes the

special value $\widetilde{\alpha} = \delta_1 - g^T p$, with $(A - \delta_1 I)p = -g$. We propose to use second smallest eigenpair of the bordered matrix whenever a potential hard case is detected. As we shall explain, not only can the size of the iterates $\|x_k\|$ be kept under control, but also convergence of $\{\lambda_k, x_k\}$ to $\{\delta_1, p\}$ will be attained by driving the parameter α_k to the value $\widetilde{\alpha}$ given by Lemma 4. Moreover, using the second smallest eigenpair of B_{α_k} prevents numerical difficulties in a near hard–case situation. Computationally, the first component ν_1 of the unitary eigenvector $(\nu_1 \; u_1^T)^T$ of B_{α_k} corresponding to the eigenvalue $\lambda_1(\alpha_k)$ is declared sufficiently small if the condition $\|g\||\nu_1| \leq \varepsilon\sqrt{1-\nu_1^2}$ 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 $\|g\||\nu_1| \leq \varepsilon\sqrt{1-\nu_1^2}$ assures that $\|(A-\lambda_1(\alpha_k)I)u_1\| \leq \varepsilon\|u_1\|$, which can be made scale independent by choosing $\varepsilon = \hat{\varepsilon}\|A\|$. In other words, $\{\lambda_1(\alpha_k), u_1\}$ is an approximate eigenpair of A, according to Lemma 3. For defining the point $\{\lambda_k, x_k\}$ at each iteration we compute the two smallest eigenpairs of B_{α_k}: $\{\lambda_1(\alpha_k), (\nu_1 \; u_1^T)^T\}$ and $\{\lambda_2(\alpha_k), (\nu_2 \; u_2^T)^T\}$. If $|\nu_1|$ is too small, that is, if $\|g\||\nu_1| \leq \varepsilon\sqrt{1-\nu_1^2}$ then $\lambda_k = \lambda_2(\alpha_k)$ and $x_k = \dfrac{u_2}{\nu_2}$. Otherwise, we 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 (9) may be greater than δ_1. Therefore, given δ_S, a current upper bound for δ_1, if $\widehat{\lambda} > \delta_S$ we simply use $\widehat{\lambda} = \delta_S$. After some manipulation and denoting by $\omega \equiv \dfrac{\lambda_k - \widehat{\lambda}}{\lambda_k - \lambda_{k-1}}$, $\phi_{k-1} \equiv \phi(\lambda_{k-1})$ and $\phi_k \equiv \phi(\lambda_k)$, the updating formula for α can be expressed by

$$\begin{aligned}
\alpha_{k+1} &= \widehat{\lambda} + \omega\phi_{k-1} + (1-\omega)\phi_k \\
&+ \frac{\|x_{k-1}\|\|x_k\|(\|x_k\| - \|x_{k-1}\|)}{\omega\|x_k\| + (1-\omega)\|x_{k-1}\|} \frac{(\lambda_{k-1} - \widehat{\lambda})(\lambda_k - \widehat{\lambda})}{(\lambda_k - \lambda_{k-1})} \\
&= \omega\alpha_{k-1} + (1-\omega)\alpha_k \\
&+ \frac{\|x_{k-1}\|\|x_k\|(\|x_k\| - \|x_{k-1}\|)}{\omega\|x_k\| + (1-\omega)\|x_{k-1}\|} \frac{(\lambda_{k-1} - \widehat{\lambda})(\lambda_k - \widehat{\lambda})}{(\lambda_k - \lambda_{k-1})},
\end{aligned} \tag{11}$$

where $\alpha_{k-1} = \lambda_{k-1} + \phi_{k-1}$ and $\alpha_k = \lambda_k + \phi_k$.

3.2 Safeguarding

Safeguarding must be introduced to assure global convergence of the iteration. In [4] bounds are presented for the optimal parameter $\alpha_* = \lambda_* - g^T x_*$:

$$\delta_1 - \frac{\|g\|}{\Delta} \leq \alpha_* \leq \delta_1 + \|g\|\Delta \,. \tag{12}$$

However, computing a good approximation to δ_1 can be as nearly as expensive as solving the given trust–region subproblem. For this reason, we shall replace the above bounds by some simple alternatives. First, note that any Rayleigh quotient $\delta_S \equiv \dfrac{v^T A v}{v^T v}$ gives an upper bound on δ_1. If the diagonal of the matrix A is explicitly available, we take $\delta_S = \min\{a_{ii} | i = 1, \ldots, n\}$, otherwise we take $\delta_S \equiv \dfrac{v^T A v}{v^T v}$ with v randomly chosen. From (12) we see that $\alpha_* \leq \alpha_U \equiv \delta_S + \|g\|\Delta$. Since $\alpha \leq 0$ implies B_α is not positive definite, we set $\alpha_0 = \min\{0, \alpha_U\}$ to assure that $\lambda_1(\alpha_0) \leq 0$. Upon solving for $\lambda_1(\alpha_0)$ and setting $\delta_I \equiv \lambda_1(\alpha_0)$, we immediately have a lower bound $\alpha_L \equiv \delta_I - \dfrac{\|g\|}{\Delta} \leq \alpha_*$, since the interlacing property implies $\delta_I \leq \delta_1$. Using this simple scheme to obtain δ_I and δ_S as an initial lower and upper bounds for δ_1, we can start with

$$\alpha_L = \delta_I - \frac{\|g\|}{\Delta} \quad \text{and} \quad \alpha_U = \delta_S + \|g\|\Delta \,. \tag{13}$$

From (13) we obtain initial bounds for the sequence in λ:

$$\lambda_L = \delta_I - \|g\| \left(\frac{1}{\Delta} + \Delta \right) \leq \lambda_* \leq \delta_S = \lambda_U \,. \tag{14}$$

The upper bound δ_S is updated every iteration using information from the smallest eigenpair of the bordered matrix: $\delta_S = \min \left\{ \delta_S, \frac{u_1^T A u_1}{u_1^T u_1} \right\}$, 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 §3.1, whenever a potential hard case is detected, $\{\lambda_1(\alpha_k), u_1\}$ approximates an eigenpair of A and $\lambda_1(\alpha_k)$ is a very good approximation to δ_1. Thus δ_S becomes a sharp estimate of δ_1.

It is worth mentioning that λ_{k-1} or λ_k might not belong to the current interval $[\lambda_L, \lambda_U]$. Nevertheless, such bounds are decisive for updating α_L and α_U. In case α_{k+1} computed by (11) does not belong to $[\alpha_L, \alpha_U]$, we use $\alpha_{k+1} = \dfrac{\alpha_L + \alpha_U}{2}$ instead. Observe that $[\alpha_L, \alpha_U] \subset [\lambda_L + \phi(\lambda_L), \lambda_U + \phi(\lambda_U)]$ and so $\alpha_{k+1} = \lambda_1(\alpha_{k+1}) + \phi(\lambda_1(\alpha_{k+1})) \in [\lambda_L +$

$\phi(\lambda_L), \lambda_U + \phi(\lambda_U)]$. Therefore, since ϕ is strictly increasing for $\lambda \leq \delta_1$ and $\lambda_1(\alpha_{k+1}) \leq \delta_1$ holds, we have $\lambda_1(\alpha_{k+1}) \in [\lambda_L, \lambda_U]$. Let $(\nu_1 \; u_1^T)^T$ be an eigenvector of $B_{\alpha_{k+1}}$ corresponding to $\lambda_1(\alpha_{k+1})$. If $\|u_1/\nu_1\| < \Delta$, we update $\lambda_L = \lambda_1(\alpha_{k+1})$ and $\alpha_L = \alpha_{k+1}$. Otherwise, $\lambda_U = \lambda_1(\alpha_{k+1})$ and $\alpha_U = \alpha_{k+1}$. In other words, the length of the interval $[\alpha_L, \alpha_U]$ is reduced at every iteration.

3.3 Initializing α

As mentioned in §3.2, there is a simple choice for initializing α: $\alpha_0 = \min\{0, \alpha_U\}$, where α_U is given by (13). 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 for A. To begin, we compute

$$AV = VT + fe_j^T \tag{15}$$

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

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

$$\begin{aligned} \min \quad & \tfrac{1}{2} y^T T y + g^T V y \\ \text{s.t.} \quad & \|y\| \leq \Delta \; . \end{aligned}$$

In other words, a solution $\{\theta_*, y_*\}$ satisfying $(T - \theta_* I)y_* = -V^Tg, \theta_*(\|y_*\| - \Delta) = 0, T - \theta_* I$ positive semidefinite, $\theta_* \leq 0$, $\|y_*\| \leq \Delta$ and $\theta_* \leq \delta_1$ is obtained applying the algorithm proposed in [2], based on Cholesky factorization of the tridiagonal matrix $T - \theta I$, $\theta < \delta_1$. Then, the initial value to be used is $\alpha = \theta_* - g^TVy_*$. The hot start for α may improve the convergence by reducing the number of iterations.

The Lanczos process for A, given by (15), can be used to compute the smallest eigenpair of B_{α_0}:

$$\begin{pmatrix} \alpha_0 & g^T \\ g & A \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & V \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & V \end{pmatrix} \begin{pmatrix} \alpha_0 & g^TV \\ V^Tg & T \end{pmatrix} + \begin{pmatrix} 0 \\ f \end{pmatrix} e_{j+1}^T \; . \tag{16}$$

Although in general $V^T g \neq \beta e_1$, if in the Lanczos process for A no restarts are used, the choice $Ve_1 = g/\|g\|$ provides a tridiagonal matrix on the right side of (16).

3.4 Stopping Criteria

Before introducing the criteria for declaring convergence in the algorithm, we include for completion the following lemma, which is a restatement of a result given in [2], was presented in [7] and provides a stopping criterion for the hard case.

Lemma 5: *Let $\varepsilon \in (0,1)$ be given and suppose $(A-\lambda I)p = -g$, $\lambda \leq 0$ with $A-\lambda I$ positive semidefinite. If $\|p+z\| = \Delta$ and $z^T(A-\lambda I)z \leq -\varepsilon(g^T p + \lambda\Delta^2)$ then*

$$\psi_* \leq \psi(p+z) \leq \frac{1}{2}(1-\varepsilon)(g^T p + \lambda\Delta^2) \leq (1-\varepsilon)\psi_* \,, \tag{17}$$

where $\psi_* \leq 0$ is the optimal value of (1).

It follows directly from (17) that $|\psi(p+z) - \psi_*| \leq \varepsilon|\psi_*|$, and so $p+z$ is nearly optimal for (1).

Given the tolerances $\varepsilon_\Delta, \varepsilon_\nu, \varepsilon_{hc} \in (0,1)$, convergence is stated as follows:

(i) If $|\,\|x_k\| - \Delta| \leq \varepsilon_\Delta \Delta$, then x_k is a binding solution, with corresponding multiplier λ_k.

(ii) If the condition $\|g\||\nu_1| \leq \varepsilon_\nu\sqrt{1-\nu_1^2}$ holds at least once, we have $z = \dfrac{u_1}{\|u_1\|}$, an approximation to an eigenvector of A corresponding to δ_1. If $\|x_k\| < \Delta$ then compute τ such that $\|x_k + \tau z\| = \Delta$. If $\tau^2(z^T A z - \lambda_k) \leq -\varepsilon_{hc}(g^T x_k + \lambda_k\Delta^2)$ and $\lambda_k \leq \delta_S$ then the hard case is declared and the solution is given by $\{\lambda_k, x_k + \tau z\}$.

(iii) If $\|x_k\| < \Delta$ and $0 < \lambda_k \leq \delta_S$ then the solution to (1) is inside the trust region and it is computed by solving the symmetric positive definite linear system $Ax = -g$, possibly using conjugate gradients. The corresponding multiplier is zero.

Finally, let us put all these pieces together and establish our algorithm.

Algorithm 1: Let $\varepsilon_\nu \in (0,1)$, $A \in I\!R^{n\times n}$, $A = A^T$, $g \in I\!R^n$, $\Delta \in I\!R_+$.

- Initialization
  - Compute $\delta_S \geq \delta_1$ and initialize α_0.
  - Compute the two smallest eigenpairs of B_{α_0}:
    $\{\lambda_1(\alpha_0), (\nu_1\ u_1^T)^T\}$ and $\{\lambda_2(\alpha_0), (\nu_2\ u_2^T)^T\}$.
  - Initialize safeguard bounds using (13)-(14).
  - If $\|g\||\nu_1| \leq \varepsilon_\nu\sqrt{1-\nu_1^2}$ then $\lambda_0 = \lambda_2(\alpha_0)$ and $x_0 = \dfrac{u_2}{\nu_2}$.
    Else $\lambda_0 = \lambda_1(\alpha_0)$ and $x_0 = \dfrac{u_1}{\nu_1}$.
  - One-point interpolation. Compute α_1 using (7) and safeguard it.
  - Compute the two smallest eigenpairs of B_{α_1}:
    $\{\lambda_1(\alpha_1), (\nu_1\ u_1^T)^T\}$ and $\{\lambda_2(\alpha_1), (\nu_2\ u_2^T)^T\}$.
  - $k \leftarrow 1$.
- Repeat
  - If $\|g\||\nu_1| \leq \varepsilon_\nu\sqrt{1-\nu_1^2}$ then $\lambda_k = \lambda_2(\alpha_k)$ and $x_k = \dfrac{u_2}{\nu_2}$.
    Else $\lambda_k = \lambda_1(\alpha_k)$ and $x_k = \dfrac{u_1}{\nu_1}$.
  - Two-point interpolation. Compute $\widehat{\lambda}$ using (9). If $\widehat{\lambda} > \delta_S$ then $\widehat{\lambda} = \delta_S$.
  - Update $\alpha_{k+1} = \widehat{\lambda} + \widehat{\phi}(\widehat{\lambda})$ using (11) and safeguard it.
  - Compute the two smallest eigenpairs of $B_{\alpha_{k+1}}$:
    $\{\lambda_1(\alpha_{k+1}), (\nu_1\ u_1^T)^T\}$ and $\{\lambda_2(\alpha_{k+1}), (\nu_2\ u_2^T)^T\}$.
  - $k \leftarrow k+1$.
  - Update safeguards.
- Until convergence

4 Local Convergence Analysis

In this section it is analyzed the local convergence rate of Algorithm 1. Before presenting the local convergence result, it is convenient to introduce some notation. The value of α that gives the optimal parameter λ_* and corresponding solution vector x_* will be denoted

by α_*. The notation $A_* \equiv A - \lambda_* I$, $A_\# \equiv A - \delta_1 I$ and $A_k \equiv A - \lambda_k I$, for any k, will be used. Also, $\mathcal{O}(1)$ will denote a quantity whose absolute value is bounded by a positive constant.

The next lemma provides useful tools for the convergence analysis. Throughout, we denote by $\dagger$ the Moore–Penrose generalized inverse.

Lemma 6: *Suppose* $A_i x_i = -g$ *and* $A_j x_j = -g$, *where* λ_i , $\lambda_j < \delta_{\ell+1}$. *Then*

$$(x_i - x_j)^T g = (\lambda_j - \lambda_i) x_i^T x_j \tag{18}$$

and

$$x_i^T x_i - x_j^T x_j = (\lambda_i - \lambda_j)\rho(i,j) \tag{19}$$

where, for $\lambda_i , \lambda_j \neq \delta_1$,

$$\rho(i,j) = x_i^T A_j^{-1} x_i + x_j^T A_i^{-1} x_j \tag{20}$$

and, for $\lambda_i = \delta_1$ *or* $\lambda_j = \delta_1$,

$$\rho(i,j) = x_i^T A_j^{\dagger} x_i + x_j^T A_i^{\dagger} x_j \ . \tag{21}$$

Moreover, if $(A - \delta_1 I)p = -g$, *then*

$$(x_i - x_j)^T p = (\lambda_i - \lambda_j) x_j^T A_i^{\dagger} p \tag{22}$$

and

$$(x_i - p)^T g = (\delta_1 - \lambda_i) p^T x_i \ . \tag{23}$$

Proof: If either $\lambda_i = \delta_1$ or $\lambda_j = \delta_1$, then $g \in \text{Range}(A - \delta_1 I)$ and thus

$$x_i = -A_i^{\dagger} g \quad , \quad x_j = -A_j^{\dagger} g \quad \text{and} \quad g = A_i A_i^{\dagger} g = A_j A_j^{\dagger} g.$$

Since $M^{\dagger} = M^{-1}$ for any nonsingular matrix M and since all of the above matrices

commute, we have in any case that

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

Moreover, in either case

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

Furthermore, if $(A - \delta_1 I)p = -g$, then in both cases we have

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

Since $p = -A_\#^\dagger g$, we also have

$$\begin{aligned} (x_i - p)^T g &= g^T(A_\#^\dagger - A_i^\dagger)g \\ &= g^T A_i^\dagger(A_i - A_\#)A_\#^\dagger g \\ &= (\delta_1 - \lambda_i)x_i^T p\,, \end{aligned}$$

what completes the proof. □

It should be noted that the formulas obtained in Lemma 6 are easily extended to any values of λ_i and λ_j where the pseudo–inverse is used whenever one of these numbers happens to be an eigenvalue of A. Moreover, formulas (18)–(23) also hold with either $\{\lambda_i, x_i\}$ or $\{\lambda_j, x_j\}$ replaced by $\{\lambda_*, x_*\}$.

Through the updating formula (11), we establish the following result relating $\lambda_{k+1} - \lambda_*$ with $\lambda_{k-1} - \lambda_*$ and $\lambda_k - \lambda_*$. The assumption that the trust region constraint binds at the

solution does not exclude the possibility of a hard case. In fact, in the hard case Lemma 2 implies that $(A - \delta_1 I)p = -g$, $\|p\| < \Delta$, and one can choose $x_* = p + z$, with $z \in \mathcal{S}_1$ such that $\|x_*\| = \Delta$.

Lemma 7: *Let $\{\lambda_*, x_*\}$ be a solution to problem (1) with $\|x_*\| = \Delta$. Then, there is a neighborhood $\mathcal{B}$ of λ_* such that if $\lambda_{k-1}, \lambda_k \in \mathcal{B}$ then the iterate λ_{k+1} produced by Algorithm 1 using formula (11) satisfies*

$$\lambda_{k+1} - \lambda_* = (\lambda_{k-1} - \lambda_*)(\lambda_k - \lambda_*)\mathcal{O}(1) \,. \tag{24}$$

Proof: We divide the analysis in two cases: $\lambda_* < \delta_1$ and $\lambda_* = \delta_1$. First, suppose that $\lambda_* < \delta_1$. Then A_* is nonsingular and there exists $r_0 \in (0, \delta_1 - \lambda_*)$ such that $A - \lambda I$ is nonsingular for every $\lambda \in \mathcal{B}_0 \equiv \{\lambda \,|\, |\lambda - \lambda_*| \leq r_0\}$. Let λ_{k-1} and $\lambda_k \in \mathcal{B}_0$. Then $\lambda_{k-1}, \lambda_k < \delta_1$ and from (9) it follows that

$$\begin{aligned} |\hat{\lambda} - \lambda_k| &= \left| \frac{\|x_{k-1}\|(\|x_{k-1}\| + \|x_k\|)(\Delta - \|x_k\|)}{\rho(k, k-1)\Delta} \right| \\ &= \frac{\|x_{k-1}\|(\|x_{k-1}\| + \|x_k\|)\rho(*, k)|\lambda_* - \lambda_k|}{\Delta(\Delta + \|x_k\|)\rho(k, k-1)} \,. \end{aligned} \tag{25}$$

Let $M = \max_{\lambda \in \mathcal{B}_0} \|(A - \lambda I)^{-1} g\|$, $m = \min_{\lambda \in \mathcal{B}_0} \|(A - \lambda I)^{-1} g\|$ and let ρ_L and ρ_U satisfy $0 < \rho_L \leq \dfrac{x^T (A - \lambda I)^{-1} x}{x^T x} \leq \rho_U$, for all $\lambda \in \mathcal{B}_0$ and $0 \neq x \in \mathbb{R}^n$. Then

$$\frac{\|x_{k-1}\|(\|x_{k-1}\| + \|x_k\|)}{\Delta(\Delta + \|x_k\|)} \leq \frac{M^2}{m^2} \,. \tag{26}$$

Moreover, due to (20),

$$\rho(*, k) = x_*^T x_* \frac{x_*^T A_k^{-1} x_*}{x_*^T x_*} + x_k^T x_k \frac{x_k^T A_*^{-1} x_k}{x_k^T x_k} \leq 2M^2 \rho_U \tag{27}$$

and

$$\rho(k, k-1) = x_k^T x_k \frac{x_k^T A_{k-1}^{-1} x_k}{x_k^T x_k} + x_{k-1}^T x_{k-1} \frac{x_{k-1}^T A_k^{-1} x_{k-1}}{x_{k-1}^T x_{k-1}} \geq 2m^2 \rho_L \,. \tag{28}$$

Therefore, by (26)–(28), it follows from (25) that

$$|\hat{\lambda} - \lambda_k| \leq \frac{M^4 \rho_U}{m^4 \rho_L} |\lambda_* - \lambda_k| \,.$$

Let $r_1 = \delta_1 - \lambda_* - r_0 > 0$ and $r_2 = \min\left\{\frac{r_1 m^4 \rho_L}{2M^4 \rho_U}, r_0\right\}$. For $\lambda_{k-1}, \lambda_k \in \mathcal{B}_2 \equiv \{\lambda \,|\, |\lambda - \lambda_*| \leq r_2\}$, we have $|\widehat{\lambda} - \lambda_k| \leq \frac{r_1}{2}$ and thus $\widehat{\lambda} < \delta_1 \leq \delta_S$. The neighborhood $\mathcal{B}$ is given by $\mathcal{B}_2$ in this case. Hence, by (9) and (11),

$$\alpha_{k+1} = \tau_1 + \tau_2 \,,$$

where

$$\tau_1 = \frac{\alpha_{k-1}\|x_{k-1}\|(\|x_k\| - \Delta) + \alpha_k \|x_k\|(\Delta - \|x_{k-1}\|)}{\Delta(\|x_k\| - \|x_{k-1}\|)}$$

and

$$\tau_2 = \frac{\|x_{k-1}\|\|x_k\|(\lambda_k - \lambda_{k-1}))(\Delta - \|x_{k-1}\|)(\Delta - \|x_k\|)}{\Delta(\|x_k\| - \|x_{k-1}\|)} \,.$$

Since $\|x_*\| = \Delta$, by (19)–(20) we have

$$\begin{aligned}
\tau_2 &= \frac{\|x_{k-1}\|\|x_k\|(\|x_{k-1}\| + \|x_k\|)}{\rho(k-1,k)} \frac{(\Delta - \|x_{k-1}\|)(\Delta - \|x_k\|)}{\Delta} \\
&= \frac{\|x_{k-1}\|\|x_k\|(\|x_{k-1}\| + \|x_k\|)}{\rho(k-1,k)} \frac{(\lambda_* - \lambda_{k-1})(\lambda_* - \lambda_k)\rho(k,*)\rho(k-1,*)}{\Delta(\Delta + \|x_{k-1}\|)(\Delta + \|x_k\|)} \\
&= (\lambda_{k-1} - \lambda_*)(\lambda_k - \lambda_*)\mathcal{O}(1) \,, \qquad (29)
\end{aligned}$$

where the $\mathcal{O}(1)$ constant in (29) follows from the hypothesis $\lambda_{k-1}, \lambda_k \in \mathcal{B}$.

Now, observe that

$$\alpha_{k+1} - \alpha_* = \frac{(\alpha_{k-1} - \alpha_*)\|x_{k-1}\|(\|x_k\| - \Delta) + (\alpha_k - \alpha_*)\|x_k\|(\Delta - \|x_{k-1}\|)}{\Delta(\|x_k\| - \|x_{k-1}\|)} + \tau_2 \,. \qquad (30)$$

However, from equations (3)–(4) and (18), for any j we have

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

Using this along with (19)–(20), (30) becomes

$$\begin{aligned}
(\lambda_{k+1} - \lambda_*)(1 + x_{k+1}^T x_*) - \tau_2 &= \frac{(\lambda_{k-1} - \lambda_*)(1 + x_{k-1}^T x_*)\|x_{k-1}\|(\lambda_k - \lambda_*)\rho(k,*)}{\Delta(\|x_k\| - \|x_{k-1}\|)(\|x_k\| + \Delta)} \\
&\quad - \frac{(\lambda_k - \lambda_*)(1 + x_k^T x_*)\|x_k\|(\lambda_{k-1} - \lambda_*)\rho(k-1,*)}{\Delta(\|x_k\| - \|x_{k-1}\|)(\|x_{k-1}\| + \Delta)} \\
&= \frac{(\lambda_{k-1} - \lambda_*)(\lambda_k - \lambda_*)\tau_3}{\Delta(\|x_k\| - \|x_{k-1}\|)(\|x_{k-1}\| + \Delta)(\|x_k\| + \Delta)} \,,
\end{aligned}$$

where

$$\begin{aligned}\tau_3 &= (\|x_{k-1}\| - \|x_k\|)(\|x_{k-1}\| + \|x_k\| + \Delta)(1 + x_k^T x_*)\rho(k-1,*) \\ &+ x_*^T(x_{k-1} - x_k)\|x_{k-1}\|(\|x_{k-1}\| + \Delta)\rho(k,*) \\ &+ (\rho(k,*) - \rho(k-1,*))\|x_{k-1}\|(\|x_{k-1}\| + \Delta)(1 + x_k^T x_*)\,.\end{aligned}$$

Since

$$\begin{aligned}x_*^T(x_{k-1} - x_k) &= x_*^T(A_k^{-1} - A_{k-1}^{-1})g \\ &= x_*^T A_k^{-1}(A_{k-1} - A_k)A_{k-1}^{-1}g \\ &= (\lambda_{k-1} - \lambda_k)x_*^T A_k^{-1} x_{k-1}\end{aligned}$$

and

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

it follows that

$$(\lambda_{k+1} - \lambda_*)(1 + x_{k+1}^T x_*) - \tau_2 = \frac{(\lambda_{k-1} - \lambda_*)(\lambda_k - \lambda_*)(\lambda_k - \lambda_{k-1})\mathcal{O}(1)}{\Delta(\|x_k\| - \|x_{k-1}\|)(\|x_{k-1}\| + \Delta)(\|x_k\| + \Delta)}\,.$$

Therefore, by (19)–(20), (29) and because $\lambda_{k-1}, \lambda_k \in \mathcal{B}$ we have

$$\lambda_{k+1} - \lambda_* = (\lambda_{k-1} - \lambda_*)(\lambda_k - \lambda_*)\mathcal{O}(1)$$

whenever $\lambda_* < \delta_1$.

It remains to analyze the possibility of $\lambda_* = \delta_1$. This corresponds to the hard case, so there exists $p \in \mathbb{R}^n$ such that $(A - \delta_1 I)p = -g$, $\|p\| \equiv \Delta_p < \Delta$. Since $g \perp \mathcal{S}_1$, the function $\varphi(\lambda) \equiv \|(A - \lambda I)^\dagger g\|$ is strictly increasing for $\lambda \leq \delta_{\ell+1}$. Moreover, if g is not orthogonal to $\mathcal{S}_{\ell+1} \equiv \{q \in \mathbb{R}^n \mid Aq = q\delta_{\ell+1}\}$, then $\varphi(\lambda)$ tends to infinity as λ approaches $\delta_{\ell+1}$ from the left. So, we can define λ_a and λ_b such that $\varphi(\lambda_a) = \dfrac{\Delta_p}{2}$ and $\varphi(\lambda_b) = \dfrac{\Delta + \Delta_p}{2}$. In case $g \perp \mathcal{S}_{\ell+1}$ and λ_b such that $\varphi(\lambda_b) = \dfrac{\Delta + \Delta_p}{2}$ is greater than $\delta_{\ell+1}$, we define $\lambda_b = \dfrac{\delta_1 + \delta_{\ell+1}}{2}$. Now, let $r_3 = \min\left\{\delta_1 - \lambda_a, \lambda_b - \delta_1, \dfrac{\delta_{\ell+1} - \delta_1}{2}\right\}$, so that for every λ_{k-1} and $\lambda_k \in \mathcal{B}_3 \equiv \{\lambda \,|\, |\lambda - \delta_1| \leq r_3\}$ we have

$$\frac{\Delta_p}{2} \leq \Delta_{k-1}, \Delta_k \leq \frac{\Delta + \Delta_p}{2} < \Delta \tag{31}$$

and

$$\delta_1 - r_3 \leq \lambda_{k-1}, \lambda_k \leq \delta_1 + r_3 < \delta_{\ell+1} \ . \tag{32}$$

Thus, by (19) and (31)

$$\begin{aligned}
\widehat{\lambda} - \delta_1 &= \frac{(\lambda_{k-1} - \delta_1)\|x_{k-1}\|(\|x_k\| - \Delta) + (\lambda_k - \delta_1)\|x_k\|(\Delta - \|x_{k-1}\|)}{\Delta(\|x_k\| - \|x_{k-1}\|)} \\
&= \lambda_k - \delta_1 + \frac{\|x_{k-1}\|(\|x_k\| + \|x_{k-1}\|)(\Delta - \|x_k\|)}{\rho(k, k-1)\Delta} \\
&\geq \lambda_k - \delta_1 + \frac{\Delta_p^2(\Delta - \Delta_p)}{4\Delta\rho(k, k-1)}.
\end{aligned} \tag{33}$$

Now, by (21) and (32)

$$\begin{aligned}
\rho(k, k-1) &= x_{k-1}^T A_k^\dagger x_{k-1} + x_k^T A_{k-1}^\dagger x_k \\
&= g^T A_{k-1}^\dagger (A_{k-1}^\dagger + A_k^\dagger) A_k^\dagger g \\
&\leq \frac{\|g\|(2\delta_n - \lambda_{k-1} - \lambda_k)}{(\delta_{\ell+1} - \lambda_{k-1})^2(\delta_{\ell+1} - \lambda_k)^2} \\
&\leq \frac{2\|g\|(\delta_n - \delta_1 + r_3)}{(\delta_{\ell+1} - \delta_1 - r_3)^4} \ .
\end{aligned}$$

Again by (32) it follows that

$$\frac{1}{\delta_{\ell+1} - \delta_1 - r_3} \leq \frac{2}{\delta_{\ell+1} - \delta_1} \quad \text{and} \quad \delta_n - \delta_1 + r_3 \leq \frac{3}{2}(\delta_n - \delta_1) \ ,$$

and so

$$\rho(k, k-1) \leq \frac{48\|g\|(\delta_n - \delta_1)}{(\delta_{\ell+1} - \delta_1)^4} \ . \tag{34}$$

Therefore, from (33) and (34)

$$\widehat{\lambda} - \delta_1 \geq \lambda_k - \delta_1 + \frac{\Delta_p^2(\Delta - \Delta_p)(\delta_{\ell+1} - \delta_1)^4}{192\Delta\|g\|(\delta_n - \delta_1)} \ .$$

Let $\sigma = \dfrac{\Delta_p^2(\Delta - \Delta_p)(\delta_{\ell+1} - \delta_1)^4}{192\Delta\|g\|(\delta_n - \delta_1)} > 0$. For $\lambda_k \geq \delta_1 - \sigma$, we have $\widehat{\lambda} \geq \delta_1$. So, let $r_4 = \min\{r_3, \sigma\}$. For λ_{k-1} and $\lambda_k \in \mathcal{B}_4 \equiv \{\lambda \,|\, |\lambda - \delta_1| \leq r_4\}$ it follows that, at iteration k,

$\widehat{\lambda} \geq \delta_1$ holds and the ideal safeguard $\widehat{\lambda} = \delta_1$ should be used. In this case the neighborhood $\mathcal{B}$ is given by $\mathcal{B}_4$. From (11) we have

$$\alpha_{k+1} = \tau_4 + \tau_5 \ ,$$

where

$$\tau_4 = \omega\alpha_{k-1} + (1-\omega)\alpha_k$$

and

$$\tau_5 = \frac{\|x_{k-1}\|\|x_k\|(\|x_k\| - \|x_{k-1}\|)}{\omega\|x_k\| + (1-\omega)\|x_{k-1}\|} \frac{(\lambda_{k-1} - \delta_1)(\lambda_k - \delta_1)}{(\lambda_k - \lambda_{k-1})} \ ,$$

with $\omega = \dfrac{\lambda_k - \delta_1}{\lambda_k - \lambda_{k-1}}$. Since

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

by (19), (21) and the hypothesis $\lambda_{k-1}, \lambda_k \in \mathcal{B}$ we have

$$\begin{aligned} \tau_5 &= \frac{\|x_{k-1}\|\|x_k\|\rho(k, k-1)(\lambda_k - \delta_1)(\lambda_{k-1} - \delta_1)}{\|x_{k-1}\|(\|x_k\| + \|x_{k-1}\|) + (\lambda_k - \delta_1)\rho(k, k-1)} \\ &= (\lambda_k - \delta_1)(\lambda_{k-1} - \delta_1)\mathcal{O}(1) \ . \end{aligned} \tag{35}$$

Now,

$$\alpha_{k+1} - \widetilde{\alpha} = \omega(\alpha_{k-1} - \widetilde{\alpha}) + (1-\omega)(\alpha_k - \widetilde{\alpha}) + \tau_5 \ . \tag{36}$$

However, by formulas (3)–(4) and (23) we have

$$\begin{aligned} \alpha_j - \widetilde{\alpha} &= \lambda_j - \delta_1 - g^T(x_j - p) \\ &= (\lambda_j - \delta_1)(1 + p^T x_j) \ . \end{aligned}$$

Thus, using (22), equation (36) can be rewritten as

$$\begin{aligned} (\lambda_{k+1} - \delta_1)(1 + p^T x_{k+1}) - \tau_5 &= \frac{(\lambda_k - \delta_1)(\lambda_{k-1} - \delta_1)(1 + p^T x_k)}{\lambda_k - \lambda_{k-1}} \\ &+ \frac{(\delta_1 - \lambda_{k-1})(\lambda_k - \delta_1)(1 + p^T x_{k-1})}{\lambda_k - \lambda_{k-1}} \\ &= \frac{(\lambda_k - \delta_1)(\lambda_{k-1} - \delta_1)p^T(x_{k-1} - x_k)}{\lambda_k - \lambda_{k-1}} \\ &= -(\lambda_k - \delta_1)(\lambda_{k-1} - \delta_1)g^T A_k^{\dagger} A_{k-1}^{\dagger} p \ . \end{aligned}$$

Finally, from (35) and since $\lambda_{k-1}, \lambda_k \in \mathcal{B}$ we have

$$\lambda_{k+1} - \delta_1 = (\lambda_{k-1} - \delta_1)(\lambda_k - \delta_1)\mathcal{O}(1)$$

and the proof is complete. □

Remark: The ideal safeguard $\widehat{\lambda} = \delta_1$ used in the second part of the proof of Lemma 7, when $\widehat{\lambda}$ given by (9) satisfies $\widehat{\lambda} \geq \delta_1$, is in practice replaced by $\widehat{\lambda} = \delta_S$. As pointed out in §3.2, it is precisely when the hard case occurs that the upper bound δ_S is a sharp estimate for δ_1. Using δ_S in place of δ_1 greatly reduces the cost of the iteration by avoiding Lanczos iterations devoted solely to the approximation of δ_1. Moreover, using δ_S appears to be just as effective as using δ_1 with respect to obtaining rapid convergence.

Based on Lemma 7, the local convergence result of Algorithm 1 is stated as follows.

Theorem 1: *Assume that problem (1) has a solution $\{\lambda_*, x_*\}$ with x_* on the boundary of the trust region. Then there exists a neighborhood $\mathcal{B}$ of λ_* such that the sequence of iterates produced by Algorithm 1 using the two-point scheme beginning with $\lambda_0, \lambda_1 \in \mathcal{B}$ is well defined, remains in $\mathcal{B}$ and converges superlinearly to λ_*. Moreover, if $\lambda_* < \delta_1$, the sequence $\{x_k\}$ converges superlinearly to x_* and, if $\lambda_* = \delta_1$, $\{x_k\}$ converges superlinearly to a vector p such that $(A - \delta_1 I)p = -g$, $\|p\| < \Delta$.*

Proof: Let $\varepsilon_k = \lambda_k - \lambda_*$ and λ_0, λ_1 in the neighborhood of λ_*, with radius r, that is stated in Lemma 7. From (24) there exists a constant $c > 0$ such that $\varepsilon_{k+1} \leq c\varepsilon_k\varepsilon_{k-1}$. Thus,

$$\varepsilon_{k+1} \leq c^{F_{k+1}-1}\varepsilon_1^{F_k}\varepsilon_0^{F_{k-1}} = \frac{(c\varepsilon_1)^{F_k}(c\varepsilon_0)^{F_{k-1}}}{c},$$

where F_j is the Fibonacci number of order j: $F_0 = F_1 = 1, F_{j+1} = F_j + F_{j-1}, j \geq 1$. Let $\widehat{r} = \min\{r, 1/c\}$. If $\lambda_0, \lambda_1 \in \mathcal{B} \equiv \{\lambda \,|\, |\lambda - \lambda_*| < \widehat{r}\}$ then $\varepsilon_{k+1} \to 0$ and the sequence $\{\lambda_k\}$ is well defined, remains in $\mathcal{B}$ and converges to λ_*. Again from (24) it follows that $\lambda_k \to \lambda_*$ superlinearly. If $\lambda_* < \delta_1$, we have $x_* = -A_*^{-1}g$, for every k sufficiently large $x_k = -A_k^{-1}g$

holds and $x_k - x_* = (\lambda_k - \lambda_*)A_*^{-1}x_k$. So, if $\lambda_0, \lambda_1 \in \mathcal{B}$ then $x_k \to x_*$ superlinearly because $\frac{\|x_{k+1} - x_*\|}{\|x_k - x_*\|} = |\lambda_{k-1} - \lambda_*|\mathcal{O}(1)$. If $\lambda_* = \delta_1$, then $p = -A_{\#}^{\dagger}g$ and $x_k = -A_k^{\dagger}g$ for any k. Since $g = A_{\#}A_{\#}^{\dagger}g$ and $g = A_kA_k^{\dagger}g$, it follows that $x_k - p = (\lambda_k - \delta_1)A_{\#}^{\dagger}x_k$. Thus, if $\lambda_0, \lambda_1 \in \mathcal{B}$ then $x_k \to p$ superlinearly since $\frac{\|x_{k+1} - p\|}{\|x_k - p\|} = |\lambda_{k-1} - \delta_1|\mathcal{O}(1)$. This completes the proof. □

The next lemma provides a relationship between the function ϕ and the interpolating function (10), to be used in the analysis of the near hard case below.

Lemma 8: *At any iteration k, the intermediate point given by (9) and the interpolating function (10) satisfy*

$$\widehat{\phi}(\widehat{\lambda}) - \phi(\lambda_k) = (\widehat{\lambda} - \lambda_k)\left[x_k^T x_{k-1} + \frac{\|x_{k-1}\|\|x_k\|\rho(k,k-1)(\widehat{\lambda} - \lambda_{k-1})}{(\lambda_k - \widehat{\lambda})\rho(k,k-1) + \|x_{k-1}\|(\|x_k\| + \|x_{k-1}\|)}\right], \quad (37)$$

whith $\rho(k,k-1)$ as defined in Lemma 6.

Proof: By (10) and (19) we have

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

where $\omega = \frac{\lambda_k - \widehat{\lambda}}{\lambda_k - \lambda_{k-1}}$. Thus, (37) follows from (19) and the proof is complete. □

A few comments are in order concerning the near hard case. As mentioned in the last paragraph of §2, in a near hard case, finding $\lambda_* < \delta_1$ 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 when compared to the other poles. That is, if $\{\gamma_j\}$ are the coefficients of g in the basis of eigenvectors of A and (6) holds, then $\gamma_1, \cdots, \gamma_\ell$ are practically zero. The strategy of Algorithm 1 for dealing with this case consists of building an interpolating function that ignores the pole δ_1 at early stages. Information concerning the second smallest eigenpair of B_{α_k} is combined with the movable pole $\delta > \max\{\lambda_{k-1}, \lambda_k\}$. The use of $\lambda_k > \delta_1$ happens because the first component ν_1 of the eigenvector $(\nu_1 \; u_1^T)^T$ associated to $\lambda_1(\alpha_k)$ is too small, so that $\|u_1/\nu_1\| = \|x_k\|$ is excessively large. Therefore $\{\lambda_k, x_k\} = \{\lambda_1(\alpha_k), u_1/\nu_1\}$ is redefined as $\{\lambda_2(\alpha_k), u_2/\nu_2\}$. Intuitively, this is a good strategy since in the exact hard case this would continuously select the correct eigenvector that will approach $(1 \; p^T)^T$ when α tends to $\widetilde{\alpha}$ from either side. Now, at iteration k the parameter α_k is updated as $\alpha_{k+1} = \widehat{\lambda} + \widehat{\phi}(\widehat{\lambda})$ with $\widehat{\lambda} \leq \delta_S$, where either $\widehat{\lambda} < \delta_S$, $\widehat{\phi}'(\widehat{\lambda}) = \Delta^2$ or $\widehat{\lambda} = \delta_S$, $\widehat{\phi}'(\delta_S) < \Delta^2$. Since $\lambda_* < \delta_1 \leq \delta_S$, the matrix A_* is nonsingular and, by the same arguments of the first part of the proof of Lemma 7, there exists a neighborhood $\mathcal{B}$ of λ_* such that $\lambda_{k-1}, \lambda_k \in \mathcal{B}$ and $\widehat{\lambda} - \lambda_k = (\lambda_* - \lambda_k)\mathcal{O}(1)$. In other words, the safeguarding $\widehat{\lambda} = \delta_S$ is eventually no longer necessary. By Lemma 8, the neighborhood $\mathcal{B}$ and (37) imply that $\widehat{\phi}(\widehat{\lambda}) - \phi(\lambda_k) = (\widehat{\lambda} - \lambda_k)\mathcal{O}(1) = (\lambda_* - \lambda_k)\mathcal{O}(1)$. 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 there is no need of using information of the second eigenpair of B_{α_k}. At early stages, however, it might be that $\widehat{\lambda} = \delta_S$. Although $\phi(\delta_1)$ is infinite, the interpolating function value $\widehat{\phi}(\delta_S)$ is finite. Using $\alpha_{k+1} = \delta_S + \widehat{\phi}(\delta_S)$ is essential in keeping the process under control.

The regularization of least squares problems arising from the discretization of ill–posed continuous problems provides an important class of trust-region subproblems. It is worth mentioning that, for these problems, Algorithm 1 converges either to the constrained

solution, if the trust region radius is small enough, or to the mininum norm interior solution. In fact, these problems are of the form $\min\{\|Cx+b\| \mid \|x\| \leq \Delta\}$ and for ill–posed problems the matrix $A = C^TC$ will be singular or nearly singular, and the vector $g = C^Tb$ will be orthogonal or nearly orthogonal to $\mathcal{S}_1$. Therefore, in the process of adjusting the parameter α, either λ_k converges to $\lambda_* < 0$ such that $(C^TC - \lambda_* I)x_* = -C^Tb$, with $\|x_*\| = \Delta$ and $C^TC - \lambda_* I$ positive definite or λ_k gets arbitrarily close to zero. In this last case, as α_k is driven to $\widetilde{\alpha} = g^TA^\dagger g$, the vector x_k tends to $p = -A^\dagger g$, with $\|p\| < \Delta$, that is, p is the minimum–norm interior solution.

5 Numerical Experiments

In this section we present some numerical experiments to demonstrate the viability of our approach. We coded Algorithm 1 in **MATLAB** (version 4.1) and through an interface between **MATLAB** and **FORTRAN** we used the Implicitly Restarted Lanczos Method (IRLM) implemented in the package **ARPACK** [6, 1]. All the six sets of tests were run in a SUN SPARC station IPX. The floating point arithmetic is IEEE standard double precision with machine precision $2^{-52} \approx 2.2204 \cdot 10^{-16}$. The first four set of tests are quite similar to the experiments presented in [7]. To put the performance of our algorithm in a context, we include the number of matrix–vector products required by conjugate gradients to solve the systems $(A - \lambda I)x = -g$, for known λ. In the first and second sets we study the sensitivity of the algorithm, respectively, to different tolerances and several sizes of the trust region radius, for problems without the hard case. The third set illustrates the local superlinear rate of convergence. In the fourth set of tests we analyze the behavior of Algorithm 1 for problems where the hard case occurs. In the fifth set we compare the usage of δ_1, the smallest eigenvalue of A, versus δ_S, an upper bound to δ_1, as well as performance for several competing choices for α_0. Finally, in the sixth set we provide a comparison between our algorithm and the approach proposed by Rendl & Wolkowicz [4].

5.1 Different Tolerances

In the first experiment the matrix A on problem (1) is $A = L - 5I$, 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 $n = 1024$ and the trust–region radius was fixed at $\Delta = 100$. We solved a sequence of twenty related problems, differing only by the vector g, randomly generated with entries uniformly distributed on $(0, 1)$. Each of these problems was solved three times, with the tolerances $\varepsilon_\Delta = 10^{-4}$, 10^{-6} and 10^{-8}. As in [7], we relaxed the accuracy requirement of the eigenvalue solution computed by the IRLM. 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. The stopping criterion to be accomplished by **ARPACK** is the following: by the IRLM we have $B_\alpha V = VT + fe_j^T$, where $V^TV = I_j$, $T \in I\!R^{j\times j}$ is tridiagonal and $V^Tf = 0$. If $\{\mu, y\}$ is an eigenpair of T then $\{\mu, x\}$ is an approximate eigenpair of B_α, where $x = Vy$ and the error in the approximation is given by $\|B_\alpha x - x\mu\| = \|f\||e_j^Ty|$. Thus, for a fixed tolerance $\varepsilon_R \in (0, 1)$, the stopping condition in **ARPACK** is $(\|f\||e_j^Ty|)^2 \leq \varepsilon_R|\mu|G(j)$, which has to hold for the j smallest Ritz pairs $\{\mu, y\}$ and where $G(j)$ is the usual gap separation (cf. [3] pp.206, 222). For this set of experiments, $j = 9 - 6 = 3$. We used initially $\varepsilon_R = 0.5$ and subsequently $\varepsilon_R = \max\{\min\{\varepsilon_R, \left|\frac{\Delta - \|x\|}{\Delta}\right|\}, \varepsilon_{max}\}$, where $\varepsilon_{max} = 0.125, 0.100, 0.075$ for $\varepsilon_\Delta = 10^{-4}, 10^{-6}, 10^{-8}$, respectively. The IRLM was always started with $v = (1, \cdots, 1)^T/\sqrt{n+1}$ and subsequently the previously calculated eigenvector corresponding to the smallest eigenvalue was used as the initial vector. This choice not only standardized the starting vector across the battery of tests, it also performed slightly better than using a randomly generated normalized vector. We did not use $(0\ g^T)^T$ to start the IRLM because in a potential hard–case occurrence this might prevent the algorithm from finding the eigenspace of B_α corresponding to δ_1. The initial value of α was $\min\{0, \alpha_U\}$ and the initial upper bound for δ_1 was chosen as the minimum of the diagonal of A.

In Table 1 we report the average number of trust–region iterations (IT), the average number of matrix–vector products required by the trust–region algorithm (TRMV) and the

average number of matrix–vector products required to solve the system $(A - \lambda_* I)x = -g$ using the conjugate–gradient method (CGMV), where λ_* is the optimal value obtained by the trust–region algorithm.

ε_Δ	IT	TRMV	CGMV	$\frac{\text{TRMV}}{\text{CGMV}}$
10^{-4}	5.00	56.70	43.35	1.31
10^{-6}	7.30	94.20	57.00	1.65
10^{-8}	8.35	124.95	71.55	1.75

Table 1: Average behavior for different tolerances.

Here the behavior of the results presented in [7] is reproduced: a trust region solution requires fewer than twice as many matrix–vector products on average than the number needed to solve a single linear system to the same accuracy, using conjugate gradients.

5.2 Different Sizes for the Trust–Region Radius

In the second experiment the matrix A on problem (1) is 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 matrix A was of order $n = 1000$. The trust region radius varied by a factor of 10 through the values $100, 10, \ldots, 0.0001$ and each problem was solved within the tolerance $\varepsilon_\Delta = 10^{-6}$. The parameters for the IRLM were the following: nine Lanczos basis vectors, six shifts on each implicit restart, initial tolerance for **ARPACK** $\varepsilon_R = 0.03$ for $\Delta = 100$, $\varepsilon_R = 0.1$ for $\Delta = 10$ and $\varepsilon_R = 0.25$ for $\Delta < 10$. Afterwards, the value of ε_R was kept the same until $\left|\frac{\Delta - \|x_k\|}{\Delta}\right| < 0.1$, when $\varepsilon_R = 0.015$ for $\Delta = 100$, $\varepsilon_R = 0.05$ for $\Delta = 10$ and $\varepsilon_R = 0.125$ for $\Delta < 10$. The initial value of α and δ_S were $\min\{0, \alpha_U\}$ and -4.5, respectively.

At each iteration, once λ_k was determined as the smallest eigenvalue of B_{α_k}, we applied the conjugate–gradient method to solve the linear system $(A - \lambda_k I)x = -g$, for the sake of comparison. Each system was solved by conjugate gradients to the same level

of accuracy as the solution provided from the eigenvalue problem. The total number of matrix–vector products required by the eigenvalue method (TRMV) is to be compared to the number required by the conjugate–gradient method (CGMV). These results are presented in Table 2, where IT is the number of trust–region iterations. We selected these results from a set of ten problems generated with different seeds, most of them with practically the same behavior.

Δ	100	10	1	.1	.01	.001	.0001
IT	12	10	4	4	4	4	3
TRMV	636	216	48	48	48	48	33
CGMV	1034	329	47	34	25	20	13
$\frac{\|g+(A-\lambda_* I)x_*\|}{\|g\|}$	10^{-8}	10^{-7}	10^{-11}	10^{-16}	10^{-16}	10^{-16}	10^{-15}
$\left\|\frac{\Delta-\|x_*\|}{\Delta}\right\|$	10^{-7}	10^{-7}	10^{-7}	10^{-9}	10^{-11}	10^{-14}	10^{-7}

Table 2: Behavior for different sizes for the trust–region radius.

The same comments made in [7] are in order: the conjugate–gradient method has a much easier time for smaller values of Δ than for larger ones.

5.3 Superlinear Convergence

In the third experiment the matrix A is 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 in $(-0.5, 0.5)$. For problems without hard case, the trust–region radius and the tolerance were set respectively at $\Delta = 10$ and $\varepsilon_\Delta = 10^{-11}$. For problems with hard case, we used $\Delta = 100$, $\varepsilon_{hc} = 10^{-11}$ and $\varepsilon_\nu = 10^{-2}$. To generate the hard case, we performed the operation $g \leftarrow g - q(q^T g)$ to orthogonalize the vector g, randomly generated as before, against q, the eigenvector of A corresponding to the smallest eigenvalue δ_1. The eigenproblems were solved by the internal solver of **MATLAB**. The initial values for α and δ_S were the same as in §5.1.

k	$\left\|\frac{\Delta-\|x_k\|}{\Delta}\right\|$
1	8.4717E−01
2	1.1806E+01
3	4.0576E+00
4	2.0839E−01
5	5.3531E−02
6	2.0518E−03
7	2.4828E−05
8	1.2216E−08
9	8.2245E−14

(a)

k	$\left\|\frac{\tau^2(z^TAz-\lambda_k)}{g^Tx_k+\lambda_k\Delta^2}\right\|$
1	——
2	4.3891E+01
3	1.0486E+01
4	1.9241E−02
5	1.6997E−02
6	9.9393E−03
7	8.6270E−05
8	8.8998E−07
9	2.4546E−13

(b)

Table 3: Verifying superlinear convergence: (a) easy and (b) hard case.

In Tables 3.a and 3.b we monitor the progressive decrease in the magnitude of $\left|\frac{\Delta-\|x_k\|}{\Delta}\right|$ and $\left|\frac{\tau^2(z^TAz-\lambda_k)}{g^Tx_k+\lambda_k\Delta^2}\right|$ as iteration k proceeds. The optimal pair $\{\lambda_*, x_*\}$ satisfied $\frac{\|g+(A-\lambda_* I)x_*\|}{\|g\|} = 10^{-14}$ for problem (a) and $\frac{\|g+(A-\lambda_* I)x_*\|}{\|g\|} = 10^{-11}$ for problem (b).

5.4 The Hard Case

In the fourth experiment the matrix A is of the same form used in the second set of tests: $A = UDU^T$, D diagonal and $U = I-2uu^T$, $u^Tu = 1$. The elements of D =diag$(d_1, \ldots, d_n)$ were randomly generated and uniformly distributed on $(-5, 5)$. Then we sorted and set $d_1 \equiv -5$ so that $d_1 = \cdots = d_\ell < d_{l+1} \leq \cdots \leq d_n$, allowing multiplicity ℓ for the smalest eigenvalue of A. The order of matrix A was $n = 1000$. 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. In this case we know the corresponding eigenvectors of matrix A: $q_i = e_i - 2uu_i, i = 1, \ldots, n$, where e_i is the i-th unit canonical vector of $\mathbb{R}^n$ and u_i is the i-th component of vector u. Therefore, we have complete control in the generation of a hard case. In fact, if $\ell = 1$ we orthogonalized g against q_1. Otherwise we computed an orthonormal basis Z for the null space of $Q = [q_1 \cdots q_l]^T$, set $g = \sum_{i=1}^{n-\ell} z_i$, where z_i is the

i-th column of Z. Finally, we added a noise vector ϵ of norm 10^{-8} to g and normalized it: $g \leftarrow \frac{g+\epsilon}{\|g+\epsilon\|}$. To ensure that the hard case really occurred, we computed

$$\Delta_{min} = \|(A - d_1 I)^{\dagger} g\| = \|(D - d_1 I)^{\dagger} U^T g\| = \sqrt{\sum_{i=\ell+1}^{n} \frac{\gamma_i^2}{(d_i - d_1)^2)}}\,, \tag{38}$$

where $\gamma = U^T g$ and set $\Delta = 2\Delta_{min}$.

The problems were solved to the level $\varepsilon_{hc} = 10^{-6}$, with $\varepsilon_{\nu} = 10^{-2}$. The parameters for the IRLM were chosen as follows: ten Lanczos basis vectors, six shifts on each implicit restart, the initial tolerance for ARPACK was $\varepsilon_R = 10^{-4}$ and then $\varepsilon_R = \min\left\{\varepsilon_R, \left|\frac{\Delta - \|x_k\|}{1000\Delta}\right|, \left|\frac{\tau^2(z^T A z - \lambda_k)}{1000(g^T x_k + \lambda_k \Delta^2)}\right|\right\}$. The initial values of α and δ_S were respectively $\min\{0, \alpha_U\}$ and -4.5. We compared the performance of Algorithm 1 with the the algorithms proposed in [7], using the same parameters specified above in the code.

In Table 4.a we summarize the average results of a sequence of ten problems, generated with different seeds. We also generated problems with near hard case by adding a noise vector ϵ to g of norm 10^{-2}. The comparative results are reported in Table 4.b.

ℓ	IT (Alg.1, [7])	TRMV (Alg.1, [7])	$\|g + (A - \lambda_* I)x_*\|$ (Alg.1, [7])
1	(9.0, 7.1)	(2340.6, 2232.8)	(10^{-4}, 10^{-3})
5	(11.0, 7.2)	(2940.2, 2221.8)	(10^{-4}, 10^{-3})
10	(10.8, 7.2)	(2830.8, 2193.6)	(10^{-4}, 10^{-3})

(a)

ℓ	IT (Alg.1, [7])	TRMV (Alg.1, [7])	$\|g + (A - \lambda_* I)x_*\|$ (Alg.1, [7])
1	(12.0, 15.2)	(3243.6, 4377.2)	(10^{-4}, 10^{-3})
5	(12.3, 31.8)	(3257.4, 9550.8)	(10^{-4}, 10^{-4})
10	(12.4, 37.6)	(3271.0, 11552.0)	(10^{-4}, 10^{-4})

(b)

Table 4: Behavior for ℓ–dimensional eigenspace $\mathcal{S}_1$: (a) hard case and (b) near hard case

5.5 Initializing α

In the fifth experiment the matrix A on problem (1) is again $A = UDU^T$, generated as in §5.4, except that $\ell = 1$ was used. The order of A was $n = 1000$ and the vectors u and g had their entries randomly selected from a uniform distribution on $(-0.5, 0.5)$. The vector u was normalized to have unit length. The vector g was orthogonalized against $q_1 = e_1 - 2uu_1$, a noise vector of norm 10^{-8} was added to it and then it was normalized so that g also had unit length. Computing Δ_{min} as in (38), we generated two set of tests: with and without hard case, respectively using $\Delta = 5\Delta_{min}$ and $\Delta = 0.2\Delta_{min}$. The problems were solved to the level $\varepsilon_\Delta = 10^{-6}$, $\varepsilon_{hc} = 10^{-6}$, with $\varepsilon_\nu = 10^{-2}$. For the IRLM, we used nine or ten Lanczos basis vectors, respectively when $\Delta = 0.2\Delta_{min}$ or $\Delta = 5\Delta_{min}$. In both cases six shifts were applied on each implicit restart. Initially, the tolerance for `ARPACK` was $\varepsilon_R = 10^{-2}$ when $\Delta = 0.2\Delta_{min}$ and $\varepsilon_R = 10^{-4}$ when $\Delta = 5\Delta_{min}$. Subsequently, $\varepsilon_R = \min\left\{\varepsilon_R, \left|\frac{\Delta - \|x_k\|}{1000\Delta}\right|, \left|\frac{\tau^2(z^T A z - \lambda_k)}{1000(g^T x_k + \lambda_k \Delta^2)}\right|, 10^{-3}\right\}$.

By applying the IRLM initially to matrix A, in this test we were able not only to replace the upper bound δ_S in Algorithm 1 by δ_1 but also to adopt the hot–start strategy for initializing α (see §3.3). By way of comparison, besides using the hot start (HS), we included $\alpha_0 = \min\{0, \alpha_U\}$, $\alpha_0 = \delta_1$ and $\alpha_0 = \delta_S$ for both family of problems ($\Delta = 0.2\Delta_{min}$, $\Delta = 5\Delta_{min}$). Initially, $\delta_S = -4.5$. The average results corresponding to ten problems generated by different seeds are reported in Table 5.

		$\Delta = 0.2\Delta_{min}$		$\Delta = 5\Delta_{min}$	
	α_0	IT	TRMV	IT	TRMV
Using	$\min\{0, \alpha_U\}$	8.2	2122.8	12.0	3303.8
δ_1	δ_1	7.0	1858.2	14.1	3697.4
	HS	7.2	1922.0	13.4	3562.6
Using	$\min\{0, \alpha_U\}$	7.6	2122.0	10.5	2973.6
δ_S	δ_S	6.6	1835.4	10.0	2790.4

Table 5: Average behavior for different α_0.

The hot start for α improves the convergence by reducing the number of iterations, especially when compared with $\alpha_0 = \min\{0, \alpha_U\}$. However, it does not outperform $\alpha_0 = \delta_S$.

Moreover, the usage of δ_1 in the algorithm instead of δ_S does not seem worthwhile, as can be seen by comparing $\alpha_0 = \min\{0, \alpha_U\}$ in both cases.

5.6 Comparison with Rendl & Wolkowicz

In the sixth experiment we solved two different family of problems. First the matrix A is $A = L - 5I$, with order $n = 256$. The trust–region radius was set to $\Delta = 100$. We solved ten related problems, differing by the vector g, randomly generated with entries uniformly distributed on $(0, 1)$. As in §5.3, an orthogonalization of g against the eigenvector of A corresponding to δ_1 generated a hard case. We also added a noise vector to g, with norm 10^{-8}. The second family of problems has $A = UDU^T$, generated exactly as in §5.5, with order $n = 256$. The tolerances used for Algorithm 1 were $\varepsilon_\Delta = 10^{-6}$, $\varepsilon_{hc} = 10^{-6}$ and $\varepsilon_\nu = 10^{-2}$. Ten problems of each type (easy and hard case) were generated with different seeds for each family and solved by both our algorithm and Rendl & Wolkowicz's code (RW). In both codes the eigenproblems were solved by the internal solver of **MATLAB**. The average results are reported in Table 6.

			IT	$\frac{\|g+(A-\lambda_* I)x_*\|}{\|g\|}$	$\left\|\frac{\Delta-\|x_*\|}{\Delta}\right\|$
$A = L - 5I$	easy case	Alg. 1	5.0	10^{-13}	10^{-7}
		RW	4.8	10^{-2}	10^{-13}
	hard case	Alg. 1	8.9	10^{-7}	10^{-16}
		RW	9.1	10^{-7}	10^{-15}
$A = UDU^T$	easy case	Alg. 1	6.4	10^{-13}	10^{-7}
		RW	7.9	10^{-4}	10^{-11}
	hard case	Alg. 1	7.1	10^{-5}	10^{-15}
		RW	11.5	10^{-7}	10^{-15}

Table 6: Comparison between Algorithm 1 and Rendl & Wolkowicz's code.

6 Conclusions

We have presented a new variant of an 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 [7]. 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. The result has been a unified algorithm that naturally incorporates both the standard and hard–case problems.

Superlinear convergence has been proved for this new algorithm and demonstrated computationally for both the standard and hard cases. This represents a major improvement over the performance of the method originally presented in [7]. In that approach, a different iteration was devised for the hard case that was not superlinearly convergent. Moreover, in practice this seemed to occur often and greatly detracted from the performance. Our computational results show this new approach overcomes these difficulties while retaining the good performance of the original algorithm for the standard case. We also compared our approach to the one of Rendl & Wolkowicz [4]. In that comparison we used the `EIG` function from `MATLAB` to supply the eigenvalues so that both methods were getting the same level of accuracy in the eigenvalue calculation. Thus, only the performance of the basic algorithms were compared and the inconsistencies introduced by having two different eigensolvers and different stopping criteria were avoided. These tests indicate a marginal advantage for our algorithm in terms of the number of eigenvalue problems that need to be solved in order to solve the given trust–region subproblem. We believe this is partially due to the need for the Rendl & Wolkowicz to determine the smallest eigenvalue δ_1 of A in order to begin the major iteration. Our approach avoids this extra calculation. Finally, our approach seems to be better suited to obtaining accuracy in the final solution to $(A - \lambda I)x = -g$.

Future work in this area should include a study of this approach for the regularization of ill–posed problems such as those arising in seismic inversion [8]. We feel that a further refinement of this approach is likely to be needed for this class of problems. In particular, near hard–case conditions seem to be associated with these problems.

Acknowledgements: We are indebted to Marielba Rojas for creating the `MEX` files needed to use `ARPACK` with `MATLAB`. We also wish to thank Dr. Rendl and Dr. Wolkowicz for making their `MATLAB` code available to us for experimentation.

References

[1] R. Lehoucq, D.C. Sorensen & P.A. Vu, ARPACK: An implementation of the Implicitly Restarted Arnoldi Iteration that computes some of the eigenvalues and eigenvectors of a large sparse matrix. Available from netlib@ornl.gov under the directory scalapack.

[2] J.J. Moré & D.C. Sorensen, Computing a Trust Region Step, *SIAM Journal on Scientific and Statistical Computing*, **4**, 553–572, 1983.

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

[4] F. Rendl & H. Wolkowicz, A Semidefinite Framework to Trust Region Subproblems with Applications to Large Scale Minimization, **CORR Report 94-32**, Department of Combinatorics and Optimization, University of Waterloo, 1994.

[5] D.C. Sorensen, Newton's Method with a Model Trust Region Modification, *SIAM Journal on Numerical Analysis*, **16**, 409–426, 1982.

[6] D.C. Sorensen, Implicit Application of Polynomial Filters in a K–Step Arnoldi Method, *SIAM Journal on Matrix Analysis and Applications*, **13**, 357–385, 1992.

[7] D.C. Sorensen, Minimization of a Large Scale Quadratic Function Subject to an Ellipsoidal Constraint, **TR94–27**, Department of Computational & Applied Mathematics, Rice University, 1994.

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

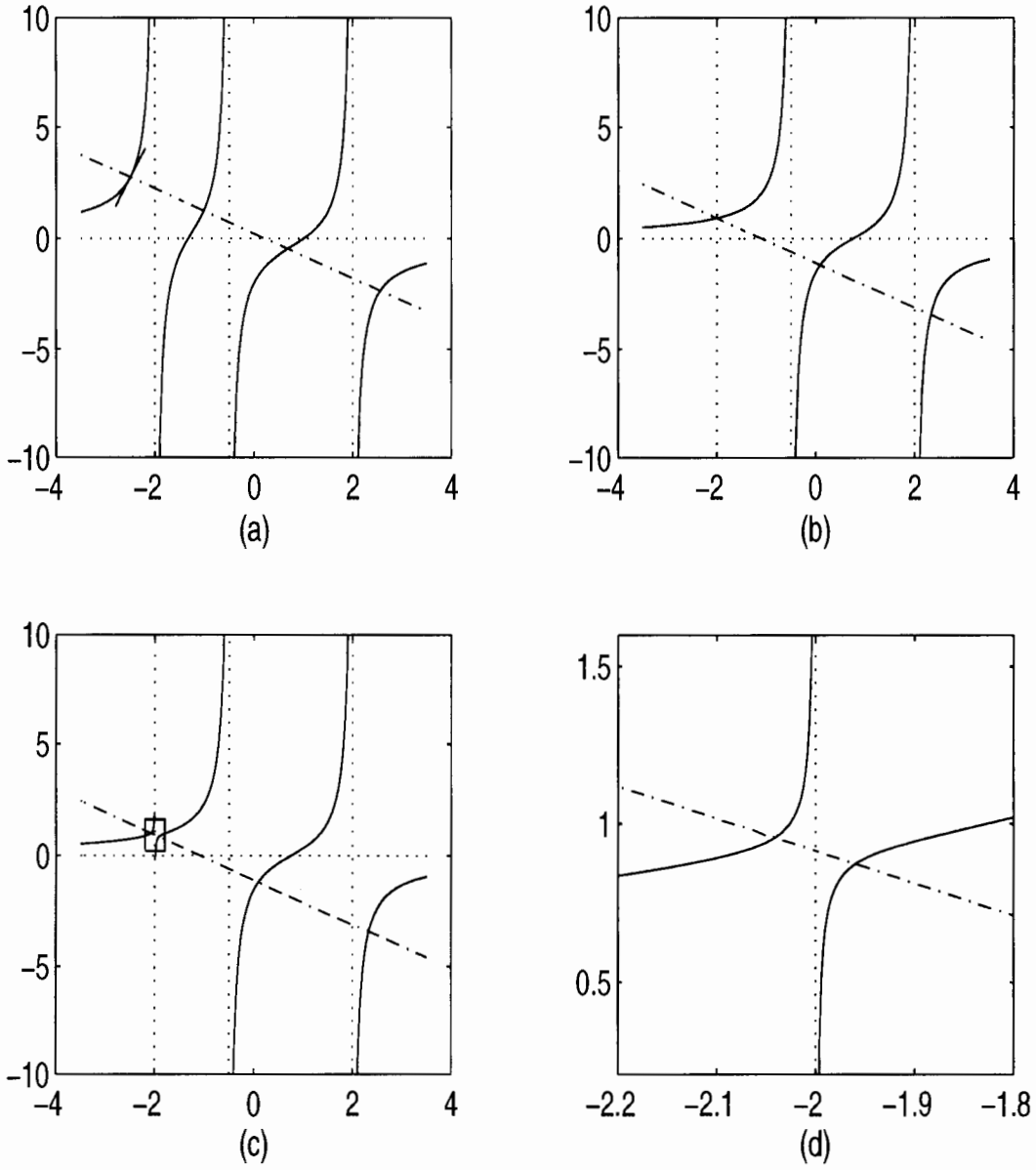

Figure 1: Example of the typical pattern of $\phi(\lambda)$ (solid) and the straight line $f(\lambda) = \alpha_* - \lambda$ (dashdotted). The three smallest eigenvalues of A are -2, -0.5 and 2. (a) general case with the slope at λ_* also plotted; (b) exact hard case; (c) near hard case; (d) detail of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