1 Introduction

An important problem in linear algebra and optimization is the *Trust-Region Subroblem* : Minimize a quadratic function subject to an ellipsoidal constraint. A mathematical statement of the problem is

$$min\ \frac{1}{2}x^T Ax + g^T x \quad \text{subject to } \|Cx\| \leq \Delta$$

where A is an $n \times n$ symmetric matrix , g an n vector, x is the unknown n vector , C is a nonsingular matrix, Δ is given positive number. The norm is the standard 2-norm, T denotes transpose, and all quantities are real.

This basic problem has many applications. The regularization or smoothing of discrete forms of ill-posed problems such as those arising in seismic inversion and the trust-region mechanism used to force convergence in optimization methods are two examples of significant computational importance. Discussions of the problem of minimizing a quadratic function subject to a quadratic constraint may be found in [5], [6], [9]. Applications to unconstrained optimization algorithms are given in [8] , [9] , [13], and applications to constrained optimization algorithms are discussed in [1], [3], [4], [11]. For applications to seismic inversion, see [7] , [15].

A solution x to the problem must satisfy a relation of the form

$$(A + \mu C^T C)x = -g$$

with $\|Cx\| = \Delta$. The parameter μ is the regularization parameter for ill-posed problems, and it is the Levenberg-Marquardt parameter in optimization. C is often constructed to impose a smoothness condition on the solution x for ill-posed problems and it is used to incorporate scaling of the variables in optimization. With a change of variables one can assume $C = I$ and this will be the case considered 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 More' and Sorensen [9] can be used to solve the problem. In some important applications , e.g. seismic inversion and large scale constrained optimization, factoring or even forming these matrices is out of the question. A conjugate-gradient style method for the large scale trust-region subproblem requiring only matrix-vector products $w \leftarrow Av$ would be highly desirable.

The purpose of this paper is to present an algorithm for solving the large scale trust-region subproblem that requires a fixed-size limited storage proportional to n and relies only upon matrix-vector products. The algorithm recasts the trust-region subproblem in terms of a parameterized eigenvalue problem and adjusts the parameter with a superlinearly convergent iteration to find the optimal vector x from the eigenvector of the parameterized problem. Only the smallest eigenvalue and corresponding eigenvector of the parameterized problem needs to be computed. The Implicitly Restarted Lanczos Method as implemented in the ARPACK [14] software is one technique that meets the requirements of limited storage and reliance only on matrix-vector products. An algorithm that is designed to solve the related large scale quadratically constrained least-squares problem is presented in [6]. The author is not aware of another algorithm that is suitable for the general (indefinite) large scale case.

2 The Trust-Region Subproblem

The trust-region subproblem has a very interesting mathematical structure that lends itself to efficient computational techniques once the subtlety of the structure is exposed. In this section and throughout the remainder of the paper $C = I$ is assumed and the problem to be considered is

$$(2.1) \qquad min \frac{1}{2} x^T A x + g^T x \quad \text{subject} \quad \text{to } \|x\| \leq \Delta$$

The optimality conditions for this problem are interesting and computationally attractive since they are both necessary and sufficient and provide a means to reduce the given n dimensional constrained optimization problem to a zero finding problem in a single scalar variable. The conditions are given in the following lemma.

Lemma 2.1 *The vector x is a solution to (2.1) 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 $\lambda(\Delta - \|x\|) = 0$.

The statement of these conditions is slightly nonstandard in the use of a negative rather than a positive λ. The reason for this will be made clear shortly. A simple proof of this lemma is given in [12].

The method developed by More' and Sorensen [9] relies upon the ability to compute a Cholesky factorization

$$R_\lambda^T R_\lambda = A - \lambda I$$

whenever this matrix is positive definite. For any such λ one can solve

$$R_\lambda^T R_\lambda x_\lambda = -g \quad \text{and} \quad \text{then} \quad R_\lambda^T q_\lambda = x_\lambda$$

to evaluate the function

$$\phi(\lambda) \equiv \frac{1}{\Delta} - \frac{1}{\|x_\lambda\|}$$

and its derivative

$$\phi'(\lambda) = \frac{\|q_\lambda\|^2}{\|x_\lambda\|^3}$$

and thus apply Newton's method to find a solution to the equation

$$\phi(\lambda) = 0.$$

This method will rapidly find solutions that are on the boundary of the trust region but it is not appropriate for large-scale problems which do not afford a Cholesky decomposition.

It is possible to re-parameterize the trust-region subproblem to obtain a scalar problem that is tractable in the large scale setting. A motivating observation is that for a given real number α

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

where $\psi(x) \equiv \frac{1}{2}x^T A x + g^T x$.

For a fixed α the goal is to minize a vertical translation of the function $\psi(x)$ over the set $\{x : 1 + x^T x = 1 + \Delta^2\}$. This suggests the solution may be found in terms of an eigen-pair of the bordered matrix. An eigenvalue λ and corresponding normalized eigenvector $(1, x^T)^T$ of the bordered matrix will satisfy

$$\begin{pmatrix} \alpha & g^T \\ g & A \end{pmatrix}\begin{pmatrix} 1 \\ x \end{pmatrix} = \begin{pmatrix} 1 \\ x \end{pmatrix}\lambda \tag{2.2}$$

and it follows that

$$\alpha - \lambda = -g^T x \quad \text{and} \quad (A - \lambda I)x = -g. \tag{2.3}$$

Hence,

$$\alpha - \lambda = g^T(A - \lambda I)^{-1}g = \sum_{j=1}^{n} \frac{\gamma_j^2}{\delta_j - \lambda} \tag{2.4}$$

where $\{\delta_j\}$ are the eigenvalues of A and $\{\gamma_j\}$ are the expansion coefficients of g in the eigenvector basis.

The bordered matrix appearing on the left in (2.2) will play a key role and for future reference this matrix will be denoted as B_α. A moments reflection on the consequences of (2.4) will reveal some very useful information. This equation shows the eigenvalues of the matrix A interlace the eigenvalues of the bordered matrix B_α (This is also a consequence of the Cauchy interlace theorem.) . Hence, the smallest eigenvalue λ of B_α satisfies $\lambda \leq \delta_1$ where δ_1 is the smallest eigenvalue of A. This assures the matrix $A - \lambda I$ is positive semi-definite *regardless* of the value of α. Moreover, this eigenvalue is well separated from the rest of the spectrum of B_α for smaller values of Δ and it is expected that a Lanzos type algorithm will be successful in computing this eigenvalue and the corresponding eigenvector.

Equations (2.3) define λ and hence x implicitly as a functions of α. Let the function ϕ be defined by

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

then

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

where differentiation is with respect to λ and $(A - \lambda I)x = -g$.

Finding the smallest eigenvalue and corresponding eigenvector of the bordered matrix B_α for a given value of α and then normalizing the eigenvector to have its first component equal to one provides a means to evaluate the rational function ϕ and its derivative at values of $\lambda < \delta_1$ the smallest eigenvalue of A. If α can be adjusted so the corresponding x satisfies $\phi'(\lambda) = x^T x = \Delta^2$ with $\alpha - \lambda = \phi(\lambda)$, then

$$(A - \lambda I)x = -g, \quad \lambda(\Delta - \|x\|) = 0$$

with $A - \lambda I$ positive semidefinite. If $\lambda \leq 0$ then x is optimal and solves the trust-region subproblem. If $\lambda > 0$ is found with $\|x\| < \Delta$ during the course of adjusting α , then A is positive definite and the solution to the trust-region subproblem is the unconstrained minimizer $-A^{-1}g$. The only other possibility is the eigenvector of the bordered matrix has first component *zero* and thus cannot be normalized to have its first component equal to one. This is equivalent to the so called *hard case* analyzed in [9]. The hard case is discussed at length here in Section 5 .

This development has lead to a reformulation of the trust-region subproblem in terms of a parameterized eigenvalue problem. In fact, a sequence of eigenvalue problems will have to be solved in order to iteratively adjust the parameter α to produce the optimal λ and x. Therefore, if this observation is to be helpful, a rapidly convergent method must be devised to adjust α to the optimal value and an efficient method for computing the smallest eigenvalue and corresponding eigenvector of the bordered matrix must be available. Keeping in mind the assumption that only matrix-vector products $w \leftarrow Av$ are available, a Lanczos method seems to be a natural choice for an eigenvalue method . A well-suited variant of the Lanczos method is presented in the next section. This will be followed with the development of a rapidly convergent iteration to adjust α.

3 The Implicitly Restarted Lanczos Process

Lanczos methods have been used extensively to solve large, sparse symmetric eigenvalue problems $Ax = \lambda x$. In exact arithmetic, the Lanczos process is a scheme to tridiagonalize a symmetric $A \in \mathcal{R}^{n \times n}$. After $j-$steps of the Lanczos process, an orthonormal $n \times j$ matrix V_j and a symmetric tridiagonal matrix T_j are produced such that

$$AV_j = V_j T_j \quad + f_j e_j^T \tag{3.1}$$

where f_j is a vector of length n with $V_j^T f_j = 0$ and e_j is the jth co-ordinate vector of length j . This is easily shown to be a truncation of the complete orthogonal reduction of A to tridiagonal form that typically precedes the implicitly shifted tridiagonal QR iteration.

The eigenvalues of T_j approximate a subset of eigenvalues of A. If μ, y is an eigen-pair for T_j (i.e. $T_j y = y\mu$) then $\mu, x = V_j y$ is an approximate eigen-pair for A and the error of approximation is given by

$$\|Ax - x\mu\| = \|f_j\| |e_j^T y| \ . \tag{3.2}$$

In particular, the approximation is exact when $f_j = 0$. Eigenvalues and eigenvectors of the symmetric tridiagonal matrix T_j may be determined by symmetric QR method or some other suitable technique.

There are a number of numerical difficulties with the original Lanczos process and these difficulties have been addressed extensively in the literature [10]. The method developed in [12] provides an alternate approach to the classic numerical difficulties associated with the Lanczos process. The underlying idea in [12] is to recognize the residual vector f_j is a function of the initial starting vector (i.e. the first column of V_j) and to then adjust this starting vector to make the residual vector vanish. The total number of Lanczos steps is limited to a fixed prescribed value k and the starting vector is iteratively updated in a way that forces the norm of the residual vector f_k to converge to zero. This limits storage requirements and allows full numerical orthogonality of the Lanczos basis vectors to be enforced due to the limited computational costs.

The iteration involves repeated application of *polynomial filters* to the starting vector and an in-place updating of the k-step Lanczos factorization . The iteration repeatedly updates the starting vector: $v_1 \leftarrow \pi(A)v_1$ where the polynomial π is applied implicitly through a mechanism directly related to the implicitly shifted QR technique. The polynomial is constructed to damp undesirable eigenvector components from the starting vector forcing it into an invariant subspace. This leads to termination of the Lanczos sequence which begins with this starting vector in precisely k steps with $f_k = 0$. The k eigenvalues of the associated T_k will be the eigenvalues of interest. The construction and application of these polynomials, how to update in-place, and other related details are explained in [12]. The technique is analogous to the implicitly shifted QR iteration for dense matrices and shares a number of important numerical properties associated with that process.

With respect to the subject of this paper, the major advantage of this implicit restart approach is

- **Fixed space:** In this scheme, the number of Lanczos basis vectors never exceeds a pre-specified bound that is proportional to the number of eigenvalues sought. Moreover, as in the basic Lanczos process, only matrix-vectors products are required with A. Peripheral storage of basis vectors for eigenvector construction is not required.

By virtue of the fixed modest number of Lanczos basis vectors, it is computationally feasible to maintain full numerical orthogonality among the basis vectors. The maintenance of orthogonality ensures no spurious eigenvalues are computed.

The standard Lanczos process provides a partial solution to the trust-region subproblem. If V_k is the Lanczos basis obtained through k steps of

(3.1) and T_k is the resulting tridiagonal matrix then $T_k = V_k^T A V_k$ and the change of variables $x = V_k y$ in (2.1) gives

$$\psi_k(y) \equiv \psi(V_k y) = \frac{1}{2} y^T T_k y + g_k^T y$$

where $g_k = V_k^T g$. If k is relatively small then the method described in [9] can be used to solve

$$min \frac{1}{2} y^T T_k y + g_k^T y \quad \text{subject} \quad \text{to} \ \|y\| \leq \Delta.$$

If y_* is the solution to this problem then the invariance of the 2-norm under orthogonal transformation implies $x_* \equiv V_k y_*$ solves

$$min \frac{1}{2} x^T A x + g^T x \quad \text{subject} \quad \text{to} \ \|x\| \leq \Delta \quad \text{and} \quad x \in Span(V_k).$$

In fact, this approach is completely equivalent to the approach of Steihaug [16] (also see [18]) until the first index k for which the solution to the reduced problem is on the trust-region boundary instead of in the interior. This approach naturally extends that technique to include directions of negative curvature in the solution to the trust-region subproblem. However, unlike the Steihaug approach this will require storage of the Lanczos basis. Since there is no *apriori* limit on the number of Lanczos steps (basis vectors) required for the reduced problem to produce a near optimal solution to the original problem, this does not provide a practical means to iteratively solve general large-scale trust-region subproblems.

4 Adjusting Alpha

Recasting the trust region problem as a parameterized eigenvalue problem together with the Implicitly Restarted Lanczos method provides a viable approach to large scale problems if the optimal parameter α can be computed rapidly. Recall that the goal is to adjust α so that

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

where

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

with $(A-\lambda I)x = -g$. One possibility would be to apply a standard iteration such as the Secant Method to the problem

$$\frac{1}{\Delta} - \frac{1}{\|x_{\lambda(\alpha)}\|} = 0.$$

The approach adopted here is to develop a special interpolation-based iteration that takes advantage of the structure of the problem. This interpolation-based iterative method will take the following form: Let $\hat{\phi}(\lambda)$ interpolate ϕ and ϕ' at some previous iterate(s).

Algorithm 1

While $\left(\left|\frac{\|x\|-\Delta}{\Delta}\right| > tol\right)$

1. Construct the interpolant $\hat{\phi}$ based on the the current and perhaps previous iterates;
2. Let $\hat{\lambda}$ satisfy $\hat{\phi}'(\hat{\lambda}) = \Delta^2$;
3. Put $\alpha_+ = \hat{\lambda} + \hat{\phi}(\hat{\lambda})$;
4. Compute the smallest eigenvalue and corresponding normalized eigenvector of B_{α_+} to get the new iterates x_+ and λ_+ ;

End

Two iterations of this type will be developed. One is based on just the previous iterate and the other on the previous two iterates. The first is linearly convergent and the second will prove to be superlinearly convergent.

To construct the single point method, consider an interpolant of the form

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

Let x_1 and λ_1 denote the current iterates corresponding to α so that

$$\alpha - \lambda_1 = -g^T x_1 \quad \text{with} \quad (A - \lambda_1 I)x_1 = -g.$$

The interpolant must satisfy

$$\frac{\gamma^2}{\delta - \lambda_1} = -g^T x_1 \quad \text{and} \quad \frac{\gamma^2}{(\delta - \lambda_1)^2} = x_1^T x_1,$$

and from this it is straightforward to derive

$$\delta = \lambda_1 - \frac{g^T x_1}{x_1^T x_1} \quad \text{and} \quad \gamma^2 = \frac{(g^T x_1)^2}{x_1^T x_1}.$$

It is easy to show $\delta = \frac{x_1^T A x_1}{x_1^T x_1}$, and this is a nice feature since it implies $\delta_1 \leq \delta$ where δ_1 is the smallest eigenvalue of A. The formula for $\hat{\lambda}$ in Step 2 of Algorithm 1 is given by

$$\hat{\lambda} = \delta + \frac{g^T x_1}{\|x_1\|\Delta}$$

and the updating formula to obtain α_+ at Step 3 is shown to be

$$\alpha_+ \quad = \quad \hat{\lambda} + \frac{\gamma^2}{\delta - \hat{\lambda}} = \alpha + \frac{(\alpha - \lambda_1)}{\|x_1\|}\left[\frac{\Delta - \|x_1\|}{\Delta}\right]\left[\Delta + \frac{1}{\|x_1\|}\right]$$

after a little algebraic manipulation. This method is linearly convergent and may be slow in some cases so it will not suffice to solve the entire problem. However, it may be used to obtain a second iterate from an initial guess to provide the starting values needed to initiate a method based upon interpolating two previous iterates at each step.

The two-point method is based upon an interpolant of the form

$$\hat{\phi}(\lambda) = \frac{\gamma^2}{\delta - \lambda} + \beta(\delta - \lambda) + \eta.$$

Let x_1 and λ_1 denote the current iterates and let x_2 and λ_2 denote the previous ones. The pole δ is defined by

$$\delta = min(\delta_{min}, \frac{x_1^T A x_1}{x_1^T x_1}) \text{ if } \|x_1\| < \Delta \text{ or } \|x_2\| < \Delta,$$

or

$$\delta = max(\frac{x_1^T A x_1}{x_1^T x_1}, \frac{x_2^T A x_2}{x_2^T x_2}) \text{ if } \|x_1\| > \Delta \text{ and } \|x_2\| > \Delta,$$

and then $\delta_{min} \leftarrow min(\delta_{min}, \delta)$. The remaining three coefficients are determined to satisfy

$$\hat{\phi}(\lambda_1) = -g^T x_1, \quad \hat{\phi}'(\lambda_1) = x_1^T x_1, \quad \hat{\phi}'(\lambda_2) = x_2^T x_2.$$

Satisfying the derivative conditions requires

$$(4.1) \qquad \frac{\gamma^2}{(\delta - \lambda_1)^2} - \beta = x_1^T x_1, \quad \frac{\gamma^2}{(\delta - \lambda_2)^2} - \beta = x_2^T x_2,$$

and it follows that

$$(4.2) \qquad \gamma^2 = \frac{[x_2^T x_2 - x_1^T x_1][(\delta - \lambda_1)(\delta - \lambda_2)]^2}{(\lambda_2 - \lambda_1)[2\delta - (\lambda_1 + \lambda_2)]},$$

$$(4.3)\qquad \beta = \frac{\gamma^2}{(\delta-\lambda_1)^2} - x_1^T x_1 = \frac{x_2^T x_2(\delta-\lambda_2)^2 - x_1^T x_1(\delta-\lambda_1)^2}{(\lambda_2-\lambda_1)[2\delta-(\lambda_1+\lambda_2)]},$$

and

$$\eta = -g^T x_1 - \beta(\delta-\lambda_1) - \frac{\gamma^2}{(\delta-\lambda_1)}.$$

The formula for $\hat{\lambda}$ in Step 2 is derived from the condition

$$\frac{\gamma^2}{(\delta-\hat{\lambda})^2} - \beta = \Delta^2$$

and yields

$$(4.4)\qquad \hat{\lambda} = \delta - \sqrt{\frac{\gamma^2}{\Delta^2+\beta}}.$$

Finally, the formula for α_+ is

$$(4.5)\qquad \alpha_+ = \hat{\lambda} + \eta + \beta(\delta-\hat{\lambda}) + \frac{\gamma^2}{\delta-\hat{\lambda}}.$$

The formula (4.5) is, unfortunately, plagued with numerical cancellation problems and computational experience has shown this will prevent superlinear convergence when the quantity $\left|\frac{\|x\|-\Delta}{\Delta}\right|$ falls below the square root of working precision (i.e. below 10^{-8} when working in double precision on a Sun workstation). After considerable manipulation one may arrive at a mathematically equivalent update formula that does achieve superlinear convergence to the level of working precision. This formula is

$$(4.6)\qquad \alpha_+ = \alpha + \frac{(\delta-\lambda_1)\theta}{(1+\sqrt{1+\theta})\sqrt{1+\theta}}\left[\frac{\Delta^2 - x_1^T x_1}{1+\sqrt{1+\theta}} + x_1^T x_1 + 1\right],$$

where

$$\theta = \left[\frac{\Delta^2 - x_1^T x_1}{x_2^T x_2 - x_1^T x_1}\right]\left[\left(\frac{\delta-\lambda_1}{\delta-\lambda_2}\right)^2 - 1\right].$$

Considering the branch of the function ϕ that is supposed to be approximated by these formulas, it is desirable that the formula (4.2) yields a positive number and that the number $\beta + \Delta^2$ appearing under the square root sign in (4.4) is also positive so that the iteration will be well defined. These conditions are indeed satisfied and this will be established in Section 6. In Section 6 it will also be established that the iteration based upon the two point formula is locally and superlinearly convergent. However, both iterations can break down when faced with the so-called *hard case*.

5 The Hard Case

There is one particularly difficult situation that may arise in trust region problems. This is referred to in [9] as the hard case. It can only occur when the vector g is orthogonal to the eigenspace $\mathcal{S}_1 \equiv \{q : Aq = q\delta_1\}$ corresponding to the smallest eigenvalue δ_1 of A. The precise statement is

Lemma 5.1 *Let $p = -(A - \delta_1 I)^\dagger g$. If $\delta_1 \leq 0$ and $\|p\| < \Delta$ then the solutions to (2.1) consist of the set*

$$\mathcal{S}_o \equiv \{x : x = p + z, \quad z \in \mathcal{S}_1, \quad \|x\| = \Delta\}.$$

In the statement of Lemma (5.3) the symbol $\dagger$ denotes the Moore-Penrose generalized inverse. This lemma is proved in [12] and its computational implications are discussed in [9]. The following lemma is a restatement of a result given in [9] that is useful in dealing with the hard case.

Lemma 5.2 *Let $0 < \sigma < 1$ be given and suppose*

$$(A - \lambda I)p = -g, \quad \lambda \leq 0,$$

with $(A - \lambda I)$ positive semidefinite. If

$$\|p + z\| = \Delta, \quad \textit{and} \quad z^T(A - \lambda I)z \leq -\sigma(g^T p + \lambda\Delta^2)$$

then

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

where $\psi_* \leq 0$ is the optimal value of (2.1) .

More' and Sorensen used this lemma to detect near hard-case behavior and terminate the iterative solution to (2.1) early. In that setting , explicit eigen-information was not available and deemed too expensive to obtain. Instead, a suitable point z was obtained from the LINPACK condition estimator [2] applied to the Cholesky factor of $(A - \lambda I)$. In the present setting, the Cholesky factor is not computed but the necessary eigen-information will be readily available.

The reformulation leading to the key relation (2.3) depends upon the ability to normalize the selected eigenvector of the bordered matrix to have its first component set to one. This is of course impossible when the first component of this eigenvector vanishes. Interestingly enough, the hard case occurs precisely when this happens.

Lemma 5.3 *Every vector of the form* $(0, q^T)^T$ *with* $q \in \mathcal{S}_1$ *is an eigenvector of the bordered matrix*

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

if and only if g *is orthogonal to* $\mathcal{S}_1$ *.*

The proof of this lemma is very straightforward and will be omitted.

Generally, a near hard-case condition is painfully obvious in practice. If the search for the optimal α discussed in Section 4 is initiated with $\alpha = 0$ then the first iterate or its successor given by the one point interpolation formula typically will have an extremely small first component in the eigenvector corresponding to the smallest eigenvalue of the bordered matrix B_α. If the vector $(\nu, q^T)^T$ is an eigenvector of length one for the bordered matrix corresponding to the smallest eigenvalue λ then satisfying a test of the form

$$\sqrt{1-\nu^2} > \kappa\Delta|\nu|$$

with $\kappa >> 1$ detects the hard case. Moreover, since $(A - \lambda I)q = -g\nu$ it follows that

$$\frac{\|(A-\lambda I)q\|}{\|q\|} = \frac{\|g\||\nu|}{\sqrt{1-\nu^2}} \leq \frac{\|g\|}{\kappa\Delta} \tag{5.1}$$

and choosing $\kappa = \frac{\|g\|}{\epsilon\Delta}$ assures $\frac{\|(A-\lambda I)q\|}{\|q\|} \leq \epsilon$ and hence that λ , q are an approximate eigen-pair for A.

If a hard-case condition has been detected, set $\lambda_U = \lambda$, $z = q/\|q\|$. Put

$$\rho \equiv z^T A z = \lambda_U - \nu(g^T q)/(q^T q)$$

and enter the following iteration with x_1, λ_1 the most recent iterates obtained before detection of the hard case:

Algorithm 2

Let $\theta \in (0,1)$ and $\lambda_1 < \lambda_* < \lambda_U$.

Repeat :

1. $\alpha \leftarrow (1-\theta)\lambda_U + \theta\lambda_1 - g^T x_1 + (1-\theta)(\lambda_U - \lambda_1)(x_1^T x_1)$;
2. Compute λ and $(\nu, q^T)^T$ the smallest eigenvalue and corresponding vector of B_α;
3. Put $x_2 \leftarrow q/\nu$, $\lambda_2 \leftarrow \lambda$ and let τ satisfy $\|x_2 + z\tau\| = \Delta$;
4. *If* $\tau^2(\rho - \lambda_2) < -\sigma(g^T x_2 + \lambda_2 \Delta^2)$ *then* **stop** with $x \leftarrow x + z\tau$;
5. *If* $\|x_2\| > \Delta$ *then* $\lambda_U \leftarrow \lambda_2$, $\alpha \leftarrow min(2 * \lambda_U, \alpha - |\alpha|)$ *else* $x_1 \leftarrow x_2$, $\lambda_1 \leftarrow \lambda_2$;

End

Note that on entering this hard-case iteration λ_U will be a good underestimate to δ_1 the smallest eigenvalue of A. The update at Step 1 is derived from linear interpolation of ϕ and its first derivative at λ_1 and then solving for the α that would produce a new $\hat{\lambda} = (1-\theta)\lambda_U + \theta\lambda$ if ϕ were linear. In other words, α satisfies

$$\alpha - \hat{\lambda} = \phi(\lambda_1) + \phi'(\lambda_1)(\hat{\lambda} - \lambda_1) = -g^T x_1 + x_1^T x_1(\hat{\lambda} - \lambda_1)$$

with $\hat{\lambda} = \theta\lambda_1 + (1-\theta)\lambda_U$.

Since ϕ is convex on the interval $(-\infty, \delta_1)$ the new λ_2 obtained by solving the bordered problem with this α will satisfy $\lambda_1 < \lambda_2 < \hat{\lambda}$. Moreover, the length of the interval (λ_1, λ_U) will always shrink.

Lemma 5.4 *Assume $\theta < \frac{1}{4}$. Let λ_1^+ and λ_U^+ be the updated values of λ_1 and λ_U obtained from one pass through the hard-case iteration. Then*

$$|\lambda_U^+ - \lambda_1^+| \leq (1-\theta)|\lambda_U - \lambda_1|.$$

Proof: By its construction, λ_2 will satisfy $\phi(\lambda_2) = \alpha - \lambda_2$. Substituting the defined value of α gives

$$\phi(\lambda_2) = (1-\theta)\lambda_U + \theta\lambda_1 + \phi(\lambda_1) + (1-\theta)(\lambda_U - \lambda_1)(x_1^T x_1) - \lambda_2.$$

Rearranging terms will give

$$\phi(\lambda_2) - \phi(\lambda_1) = \lambda_1 - \lambda_2 + (1-\theta)[1 + x_1^T x_1](\lambda_U - \lambda_1). \tag{5.2}$$

It is straightforward to show

$$\phi(\lambda_2) - \phi(\lambda_1) = (\lambda_2 - \lambda_1)x_2^T x_1,$$

and substituting this into (5.2) and rearranging terms will give

$$(\lambda_2 - \lambda_1)(1 + x_2^T x_1) = (1-\theta)(1 + x_1^T x_1)(\lambda_U - \lambda_1).$$

If $\|x_2\| < \Delta$ then $\lambda_1^+ = \lambda_2$ and $\lambda_U^+ = \lambda_U$. Hence,

$$\begin{aligned}
\lambda_U^+ - \lambda_1^+ &= \lambda_U - \lambda_2 \\
&= \lambda_U - \lambda_1 - (\lambda_2 - \lambda_1) \\
&= \left[1 - (1-\theta)\frac{(1 + x_1^T x_1)}{(1 + x_2^T x_1)}\right](\lambda_U - \lambda_1) \\
&= \left[\frac{(x_2 - x_1)^T x_1}{(1 + x_2^T x_1)} + \theta\frac{(1 + x_1^T x_1)}{(1 + x_2^T x_1)}\right](\lambda_U - \lambda_1). \qquad (5.3)
\end{aligned}$$

Now, if $\lambda_2 - \lambda_1 < \frac{1}{4}(\lambda_U - \lambda_1)$ then

$$\begin{aligned}
\frac{(x_2 - x_1)^T x_1}{(1 + x_2^T x_1)} &= \frac{(\lambda_2 - \lambda_1) x_1^T A_2^{-1} x_1}{(1 + x_2^T x_1)} \\
&\leq \frac{(\lambda_2 - \lambda_1)}{(\delta_1 - \lambda_2)} \frac{x_1^T x_1}{(1 + x_2^T x_1)} \\
&\leq \left[\frac{\frac{1}{4}(\lambda_U - \lambda_1)}{(\delta_1 - \lambda_1) - (\lambda_2 - \lambda_1)}\right] \frac{x_1^T x_1}{(1 + x_1^T x_1)} \\
&\leq \frac{\frac{1}{4}(\lambda_U - \lambda_1)}{\frac{3}{4}(\lambda_U - \lambda_1)} \\
&= \frac{1}{3}
\end{aligned}$$

where $A_2 \equiv A - \lambda_2 I$. Thus

$$\lambda_U^+ - \lambda_1^+ \leq (\frac{1}{3} + \theta)(\lambda_U - \lambda_1) < \frac{3}{4}(\lambda_U - \lambda_1)$$

follows from (5.3). If $\lambda_2 - \lambda_1 \geq \frac{1}{4}(\lambda_U - \lambda_1)$ then

$$\lambda_U^+ - \lambda_1^+ = \lambda_U - \lambda_2 = (\lambda_U - \lambda_1) - (\lambda_2 - \lambda_1) \leq \frac{3}{4}(\lambda_U - \lambda_1)$$

and in both cases the desired result holds since $\frac{1}{4} < (1-\theta)$. Now suppose $\|x_2\| \geq \Delta$. Then $\lambda_U^+ = \lambda_2$ and $\lambda_1^+ = \lambda_1$ and it follows that

$$\lambda_U^+ - \lambda_1^+ = (1-\theta)\frac{(1 + x_1^T x_1)}{(1 + x_2^T x_1)}(\lambda_U - \lambda_1) < (1-\theta)(\lambda_U - \lambda_1).$$

This establishes the result. $\Box$

This result establishes convergence but is far from indicative of what will occur in practice. A value $\theta = .001$ works well in practice even though this Lemma would indicate a potentially slow rate of convergence with this value. This is because the point λ_2 almost always satisfies $\|x_2\| < \Delta$.

Satisfaction of the stopping rule at Step 4 assures the conditions of Lemma (5.2) are satisfied so the accepted point x_1 satisfies

$$\psi(x_*) \leq \psi(x_1) \leq (1 - \sigma)\psi(x_*)$$

In many applications including the two mentioned previously, a value of $\sigma = .01$ is used and this is generally satisfied very rapidly indeed.

6 Safeguarding and Convergence

In this section the issues of forcing convergence and determining the rate of local convergence will be discussed. It will be shown that the iterates based upon the two point rational interpolation formulas are well defined and are locally convergent at a superlinear rate. This may be of considerable interest computationally since evaluating the function ϕ and its derivative requires the computation of the smallest eigenvalue and corresponding eigenvector of the bordered matrix B_α and this is potentially very expensive. Note, however, in practice one is often interested in just a few digits of accuracy and then superlinear convergence is of little consequence. Nevertheless, it is reassuring to know this rapid convergence can be expected when higher accuracy is needed.

There is very little to say about safeguarding. Perhaps in the future with more computational experience this will become an important issue. In the computational results presented here a fairly standard simple safeguard was used to obtain an *interval of uncertainty* and then to assure that this interval is updated on each iteration and required to decrease . This safeguard rarely forced a modification of the step given by the two point formula in Algorithm 1.

In order to present the local convergence result as simply as possible, it shall be useful to introduce some notation. The subscript 1 shall indicate the most recent iterate, and the subscript 2 shall denote the previous iterate. Thus λ_1 and λ_2 are the current and previous approximations to the optimal λ_*, and λ_1 is the smallest eigenvalue of the borderd matrix B_α. The updated λ_+ is the smallest eigenvalue of the updated B_{α_+}, and α_* will denote the value of α that gives the optimal parameter λ_* and corresponding solution vector x_*. The notation $A_j \equiv A - \lambda_j I$ for $j = 1, 2$ and $A_* \equiv A - \lambda_* I$ will be used. Thus $x_j = -A_j^{-1} g$ for $j = 1, 2$ and $x_* = -A_*^{-1} g$. At a general point λ the notation $A_\lambda \equiv A - \lambda I$ and $x_\lambda \equiv A_\lambda^{-1}$ will be used. Finally, the notation $\mathcal{O}((\lambda_* - \lambda_1)^j)$ will be used to denote a quantity whose absolute value is bounded by a fixed positive constant times the quantity $|\lambda_* - \lambda_1|^j$ for any value of λ_1 in a sufficiently small neighborhood of λ_* ($j = 0, 1, 2$).

First, the fact that the iterates are well defined shall be established. In this development it is useful to note

$$\begin{aligned} x_2^T x_2 - x_1^T x_1 &= g^T (A_2^{-2} - A_1^{-2}) g \\ &= g^T A_2^{-2} (A_1 - A_2)(A_1 + A_2) A_1^{-2} g \\ &= (\lambda_2 - \lambda_1)[x_2^T A_1^{-1} x_2 + x_1^T A_2^{-1} x_1]. \end{aligned} \tag{6.1}$$

From this it follows that

$$\begin{aligned} \gamma^2 &= \frac{[x_2^T x_2 - x_1^T x_1][(\delta - \lambda_1)(\delta - \lambda_2)]^2}{(\lambda_2 - \lambda_1)[2\delta - (\lambda_1 + \lambda_2)]} \\ &= \frac{[x_2^T A_1^{-1} x_2 + x_1^T A_2^{-1} x_1][(\delta - \lambda_1)(\delta - \lambda_2)]^2}{2\delta - (\lambda_1 + \lambda_2)}. \end{aligned} \tag{6.2}$$

Now, with the exception of the hard case, the smallest eigenvalue of the bordered matrix B_α is always less than the smallest eigenvalue δ_1 of A and $\delta > \delta_1$. Hence, $x_2^T A_1^{-1} x_2 > 0$, $x_1^T A_2^{-1} x_1 > 0$ and $2\delta - (\lambda_1 + \lambda_2) > 0$. Therefore, the formula (4.2) for γ^2 does indeed yield a positive number.

Moreover, the number $\Delta^2 + \beta$ appearing under the square root sign in (4.4) is always nonnegative:

Lemma 6.1 *The quantity $\Delta^2 + \beta$ in (4.4) is always nonnegative.*

Proof: If either $x_1^T x_1 \le \Delta^2$ or $x_2^T x_2 \le \Delta^2$ then $\Delta^2 + \beta \ge 0$ since

$$\begin{aligned} \Delta^2 + \beta &= \frac{\gamma^2}{(\delta - \lambda_1)^2} + (\Delta^2 - x_1^T x_1) \\ &= \frac{\gamma^2}{(\delta - \lambda_2)^2} + (\Delta^2 - x_2^T x_2) \end{aligned}$$

is implied by (4.1). Otherwise, it may be assumed without loss of generality that $x_*^T x_* \equiv \Delta^2 < x_1^T x_1 < x_2^T x_2$ and hence that $\lambda_* < \lambda_1 < \lambda_2$. In this case the pole δ satisfies $\delta = max(\frac{x_1^T A x_1}{x_1^T x_1}, \frac{x_2^T A x_2}{x_2^T x_2})$. Observe that the function

$$\rho(\lambda) \equiv \frac{x_\lambda^T A x_\lambda}{x_\lambda^T x_\lambda}$$

is decreasing on the interval (λ_1, δ_1) since the Cauchy-Schwarz inequality implies

$$(6.3) \qquad (x_\lambda^T A_\lambda x_\lambda)(x_\lambda^T A_\lambda^{-1} x_\lambda) \geq (x_\lambda^T A_\lambda^{1/2} A_\lambda^{-1/2} x_\lambda)^2 = (x_\lambda^T x_\lambda)^2$$

and hence

$$\rho'(\lambda) = 2\left[1 - \frac{(x_\lambda^T A_\lambda x_\lambda)(x_\lambda^T A_\lambda^{-1} x_\lambda)}{(x_\lambda^T x_\lambda)^2}\right] \leq 0,$$

for all $\lambda \in (\lambda_1, \delta_1)$. It follows that

$$\frac{(x_\lambda^T A_\lambda x_\lambda)}{x_\lambda^T x_\lambda} = \rho(\lambda) - \lambda \geq \delta - \lambda > 0$$

for all $\lambda \in (\lambda_1, \delta_1)$. From (4.3) it may be found that

$$(6.4) \Delta^2 + \beta = \frac{(x_2^T x_2 - x_*^T x_*)(\delta - \lambda_2)^2 - (x_1^T x_1 - x_*^T x_*)(\delta - \lambda_1)^2}{(\delta - \lambda_1)^2 - (\delta - \lambda_2)^2}.$$

Now, $\lambda_* < \lambda_1 < \lambda_2 < \delta$ implies $\delta - \lambda_* > \delta - \lambda_1 > \delta - \lambda_2 > 0$ so the denominator in (6.4) is positive and the result will be established if it is shown that the function

$$\sigma(\lambda) \equiv (x_\lambda^T x_\lambda - x_*^T x_*)((\delta - \lambda)^2$$

is strictly increasing on the interval (λ_*, δ). Differentiating σ with respect to λ gives

$$\begin{aligned}
(6.5) \quad \sigma'(\lambda) &= 2(\delta - \lambda)[x_\lambda^T A_\lambda^{-1} x_\lambda(\delta - \lambda) - (x_\lambda^T x_\lambda - x_*^T x_*)] \\
&\geq 2(\delta - \lambda)(x_\lambda^T x_\lambda)\left[\frac{(x_\lambda^T A_\lambda x_\lambda)(x_\lambda^T A_\lambda^{-1} x_\lambda)}{(x_\lambda^T x_\lambda)^2} - 1 + \frac{x_*^T x_*}{x_\lambda^T x_\lambda}\right] \\
&> 0
\end{aligned}$$

which again follows from (6.3). This implies $\sigma(\lambda)$ is increasing on the interval $\lambda \in (\lambda_1, \delta_1)$ and since

$$\Delta^2 + \beta = \frac{\sigma(\lambda_2) - \sigma(\lambda_1)}{(\delta - \lambda_1)^2 - (\delta - \lambda_2)^2},$$

it follows that $\Delta^2 + \beta > 0$ when $\lambda_* < \lambda_1 < \lambda_2 < \delta$ and the result is established. □

It has just been demonstrated that the iterates are well defined and it is now necessary to establish the local rate of convergence. To this end it is useful to establish a technical lemma that will facilitate the proof of the final desired result

Lemma 6.2 *The intermediate point $\hat{\lambda}$ given by (4.4) satisfies*

$$\hat{\lambda} - \lambda_1 = \left(\frac{\delta - \lambda_1}{2}\right)\left(\frac{\Delta^2 - x_1^T x_1}{\beta + x_1^T x_1}\right) + \mathcal{O}((\lambda_1 - \lambda_*)^2). \tag{6.6}$$

Proof: The result is established using a Taylor expansion of the square root function near 1. The formulas of Algorithm 1 give

$$\begin{aligned}
\hat{\lambda} &= \delta - \sqrt{\frac{\gamma^2}{\Delta^2 + \beta}} \\
&= \delta - (\delta - \lambda_1)\sqrt{\frac{\gamma^2}{\frac{\gamma^2}{(\delta-\lambda_1)^2} + (\Delta^2 - x_1^T x_1)}} \\
&= \delta - (\delta - \lambda_1)\sqrt{\frac{1}{1 + \frac{\Delta^2 - x_1^T x_1}{\beta + x_1^T x_1}}} \\
&= \delta - (\delta - \lambda_1)\left[1 - \frac{1}{2}\left(\frac{\Delta^2 - x_1^T x_1}{\beta + x_1^T x_1}\right)\right] + \mathcal{O}((\lambda_1 - \lambda_*)^2).
\end{aligned}$$

Simplifying this last term yields the desired formula (6.6). □

The updating formula for α will now be used to establish a result to relate $\lambda_+ - \lambda_*$ to $\lambda_1 - \lambda_*$.

Lemma 6.3 *There is a neighborhood $\mathcal{N}$ of λ_* such that the iterate λ_+ produced at Steps 3 and 4 of Algorithm 1 using formula (4.5) to compute α_+ based upon points $\lambda_2, \lambda_1 \in (\mathcal{N})$ will satisfy*

$$(\lambda_+ - \lambda_*) = (\lambda_1 - \lambda_*)\mu(\lambda_1, \lambda_2)\mathcal{O}(1) + \mathcal{O}((\lambda_1 - \lambda_*)^2). \tag{6.7}$$

where

$$\mu(\lambda_1, \lambda_2) \to 0 \quad as \quad \lambda_1, \lambda_2 \to \lambda_*$$

Proof: The proof begins with the formula

$$\alpha_+ \quad = \quad \hat{\lambda} + \eta + \beta(\delta - \hat{\lambda}) + \frac{\gamma^2}{\delta - \hat{\lambda}}.$$

Using the definition

$$\eta \quad = \quad -g^T x_1 - \beta(\delta - \lambda_1) - \frac{\gamma^2}{(\delta - \lambda_1)},$$

and the fact that

$$\beta(\delta - \hat{\lambda}) + \frac{\gamma^2}{\delta - \hat{\lambda}} = 2\beta(\delta - \hat{\lambda}) + (\delta - \hat{\lambda})\Delta^2,$$

and substituting into the above formula gives

$$\alpha_+ = \hat{\lambda} - g^T x_1 + 2\beta(\lambda_1 - \hat{\lambda}) + (\delta - \hat{\lambda})\Delta^2 - (\delta - \lambda_1)x_1^T x_1.$$

Since $-g^T x_1 = \alpha - \lambda_1$ it follows after substitution and simplification that

$$(6.8) \qquad \alpha_+ = \alpha + (\lambda_1 - \hat{\lambda})[2\beta + \Delta^2 - 1] + (\delta - \lambda_1)(\Delta^2 - x_1^T x_1).$$

Now utilize the relation $\alpha_+ = \lambda_+ - g^T x_+$ and $\alpha_* = \lambda_* - g^T x_*$ to see that

$$\begin{aligned} \alpha_+ - \alpha_* &= \lambda_+ - \lambda_* - g^T(x_+ - x_*) \\ &= \lambda_+ - \lambda_* + g^T(A_+^{-1} - A_*^{-1})g \\ &= \lambda_+ - \lambda_* + g^T A_+^{-1}(A_* - A_+)A_*^{-1}g \\ &= (\lambda_+ - \lambda_*)(1 + x_+^T x_*). \qquad (6.9) \end{aligned}$$

Similarly ,

$$\alpha - \alpha_* = (\lambda_1 - \lambda_*)(1 + x_1^T x_*).$$

Subtracting α_* from both sides of (6.8) above and substituting for $\alpha_+ - \alpha_*$ using (6.9) gives

$$\begin{aligned} (\lambda_+ \;&-\; \lambda_*)(1 + x_+^T x_*) \\ &= (\lambda_1 - \lambda_*)(1 + x_1^T x_*) - (\delta - \lambda_1)(\Delta^2 - x_1^T x_1)\left[\frac{2\beta + \Delta^2 - 1}{2\beta + 2x_1^T x_1} - 1\right] + \mathcal{O}((\lambda_1 - \lambda_*)^2) \\ &= (\lambda_1 - \lambda_*)(1 + x_1^T x_*) - (\Delta^2 - x_1^T x_1)(\delta - \lambda_1)\left[\frac{(\Delta^2 - x_1^T x_1) - (1 + x_1^T x_1)}{2\beta + 2x_1^T x_1}\right] \\ &\qquad + \mathcal{O}((\lambda_1 - \lambda_*)^2). \end{aligned}$$

Since

$$\begin{aligned} \Delta^2 - x_1^T x_1 &= x_*^T x_* - x_1^T x_1 \\ &= (\lambda_* - \lambda_1)[x_*^T A_1^{-1} x_* + x_1^T A_*^{-1} x_1] \end{aligned}$$

and since

$$\begin{aligned} 2(\beta + x_1^T x_1) &= 2\frac{\gamma^2}{(\delta - \lambda_1)^2} \\ &= \frac{[x_2^T A_1^{-1} x_2 + x_1^T A_2^{-1} x_1][(\delta - \lambda_2)]^2}{\delta - \frac{1}{2}(\lambda_1 + \lambda_2)} \\ &= \mathcal{O}(1), \end{aligned}$$

it follows that

(6.10)

$$\begin{aligned} (\lambda_+ &- \lambda_*)(1 + x_+^T x_*) \\ = (&\lambda_1 - \lambda_*)(1 + x_1^T x_*) \\ &- (\lambda_1 - \lambda_*)(1 + x_1^T x_1) \left[\frac{x_*^T A_1^{-1} x_* + x_1^T A_*^{-1} x_1}{x_2^T A_1^{-1} x_2 + x_1^T A_2^{-1} x_1}\right] \left[\frac{(\delta - \frac{1}{2}(\lambda_1 + \lambda_2))(\delta - \lambda_1)}{(\delta - \lambda_2)(\delta - \lambda_2)}\right] \\ &\qquad + \mathcal{O}((\lambda_1 - \lambda_*)^2) \\ = (&\lambda_1 - \lambda_*)(x_+^T x_* - x_1^T x_1) \\ &- (\lambda_1 - \lambda_*)(1 + x_1^T x_1) \left(\left[\frac{x_*^T A_1^{-1} x_* + x_1^T A_*^{-1} x_1}{x_2^T A_1^{-1} x_2 + x_1^T A_2^{-1} x_1}\right] \left[\frac{(\delta - \frac{1}{2}(\lambda_1 + \lambda_2))(\delta - \lambda_1)}{(\delta - \lambda_2)(\delta - \lambda_2)}\right] - 1 \right) \\ &\qquad + \mathcal{O}((\lambda_1 - \lambda_*)^2). \end{aligned}$$

Noting that

$$(\lambda_1 - \lambda_*)(x_+^T x_* - x_1^T x_1) = -(\lambda_1 - \lambda_*)^2 x_1^T A_* x_1,$$

and that

$$\frac{(1 + x_1^T x_1)}{(1 + x_+^T x_*)} = \mathcal{O}(1),$$

and substituting into (6.10) establishes

$$(6.11) \qquad (\lambda_+ - \lambda_*) = (\lambda_1 - \lambda_*)\mu(\lambda_1, \lambda_2)\mathcal{O}(1) + \mathcal{O}((\lambda_1 - \lambda_*)^2).$$

where

$$\mu(\lambda_1, \lambda_2) \equiv \left[\frac{x_*^T A_1^{-1} x_* + x_1^T A_*^{-1} x_1}{x_2^T A_1^{-1} x_2 + x_1^T A_2^{-1} x_1}\right] \left[\frac{(\delta - \frac{1}{2}(\lambda_1 + \lambda_2))(\delta - \lambda_1)}{(\delta - \lambda_2)(\delta - \lambda_2)}\right] - 1.$$

Since

$$\mu(\lambda_1, \lambda_2) \to 0 \quad \text{as} \quad \lambda_1, \lambda_2 \to \lambda_*,$$

the proof is complete. □

The previous discussion together with Lemmas (6.1) - (6.3) establishes the following

Theorem 6.4 *Suppose the solution to (2.1) is on the boundary of the trust region. Then there is a neighborhood $\mathcal{N}$ of λ_* such that a sequence of iterates produced by Algorithm 1 using the two point scheme beginning with x_2 x_1 corresponding to $\lambda_2, \lambda_1 \in (\mathcal{N})$ will be well defined, remain in $\mathcal{N}$ and converge superlinearly to x_* and λ_**

7 Computational Results and Conclusions

In this final section, a limited set of computational results shall be presented to illustrate the viability of the approach presented here. These results are not meant to be exhaustive. They should be regarded as preliminary results intended to illustrate selected aspects of the behavior of this approach. A comparison with the corresponding cost of solving the requisite linear systems via conjugate gradients is provided .

The methods described in Sections 3-5 were implemented in in MATLAB, Version 4.1. All experiments were carried out on a SUN SPARC station IPX. The floating point arithmetic is IEEE standard double precision with machine precision of $\epsilon_M \equiv 2^{-52} \approx 2.2204 \cdot 10^{-16}$. In all cases the Implicitly Restarted Lanczos technique described in Section 3 was used to solve the eigenproblems. The number of Lanczos basis vectors was limited to nine. Six shifts (i.e six matrix vector products) were applied on each implicit restart. The iteration was halted as soon as the smallest Ritz value had a Ritz estimate (3.2) below the specified tolerance.

The first experiment presents the performance on the problem (2.1) with the matrix $A = L - 5*I$ where L is set to the standard 2-D discrete Laplacian on the unit square based upon a 5-point stencil with equally-spaced mesh points. The shift of -5 was introduced to make the matrix indefinite. A sequence of 20 related problems were solved. The order of A was $n = 1024$

in all cases. The trust-region radius was fixed at $\Delta = 100$ for all of the problems. For each problem a random vector g was constructed with entries uniformly distributed on (-.5 , .5) and the problem was solved three times with a tolerance of 10^{-4} , 10^{-6} and 10^{-8}. In Table 1 the average number of trust-region iterations and average number of matrix vector products $w \leftarrow Av$ per trust-region iteration are reported. In addition, the average number of matrix-vector products required to solve the system $(A - \lambda I)x = -g$ using the conjugate-gradient method is given. These tests indicate that 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 the conjugate-gradient method. The accuracy requirement of the eigenvalue solution computed by IRAM at each step was relaxed and made proportional to the relative accuracy of the computed solution. More specifically , $\|B_\alpha q - q\lambda\| < \tau_1/1000$ where $\tau_1 = min(10^{-6}, \left|\frac{\|x\|-\Delta}{\Delta}\right|)$. In addition to this, the inner IRAM iteration was initialized with the solution from the previous outer trust-region iteration.

	Trust Mvecs	*iters*	*Cg Mvecs*	*Ratio*
$tol = .0001$	59.3	4.2	44.4	1.34
$tol = .000001$	98.1	8.4	58.0	1.69
$tol = .00000001$	132.8	12.3	72.2	1.84

Table 1: Average behavior for different tolerances

The second experiment illustrates how the size of the trust-region parameter Δ may effect the solution process. In these problems the matrices were distributed in the form $A = UDU^T$ with D a diagonal matrix with diagonal elements selected randomly from a uniform distribution on (-.5,.5). The matrix $U = I - 2uu^T$ with the vector u and the vector g constructed with randomly distributed elements and then normalized to have unit length. The matrix A was of order $n = 1000$. The trust-region radius Δ was varied by a factor of 10 through the values $100, 10, ..., .0001$ and each problem was solved to the level $\left|\frac{\Delta - \|x\|}{\Delta}\right| < 10^{-6}$. By way of comparison, the conjugate-gradient method was used to solve the same linear systems $(A - \lambda_j I)x = -g$ using the parameter λ_j provided by the eigen-solution of B_{α_j} at the jth step of the trust-region iteration . Each system was solved by conjugate gradients to the same level of accuracy as the solution provided from the eigenvalue solution. The total number of matrix-vector products required by the eigenvalue method is to be compared to the number required by the conjugate-gradient method. These results are presented in Table 2.

Δ	100	10	1	.1	.01	.001	.0001
Trust Iters	13	8	4	4	4	4	4
Matvecs	579	240	36	36	36	36	36
CG-Matvecs	1307	384	51	39	30	26	24
$\|g + (A - \lambda I)x\|$	(-4)	(-6)	(-12)	(-15)	(-15)	(-15)	(-15)
$\frac{\Delta - \|x\|}{\Delta}$	(-7)	(-7)	(-10)	(-9)	(-11)	(-13)	(-14)

Table 2: Behavior for different trust-region radii

The entries in parentheses in Table 2 represent powers of 10 (i.e. (-4) represents 10^{-4}). The row labeled *Trust Iters* gives the number of iterations required in Algorithm 1. The row labeled *Matvecs* gives the number of matrix-vector products required to solve the resulting eigenvalue problems and the row labeled $CG - Matvecs$ gives the number of matrix-vector products required by the conjugate-gradient iteration to solve the same linear systems. Note that for small trust-region radii there is not a significant difference in the required number of matrix-vector products but the conjugate-gradient method has a much easier time for smaller values of Δ than for larger values. This is because the matrix $A - \lambda I$ will have a very large value of λ and hence will act as though there are essentially two distinct eigenvalues when the value of Δ is small. Just the opposite situation occurs when the value of the trust-region radius gets larger. The eigenvalue problems do get more difficult to solve but the conjugate-gradient method has more trouble with these systems than the eigenvalue method. This phenomena is partially explained in [17]. When the spectrum is not clustered it is often more difficult to solve the linear system by conjugate gradients than it is to find an extreme eigenvalue.

The next results verify the superlinear rate of convergence for the two point iteration. In this case the matrix A is again set to $A = L - .5I$ with L the 2-D discrete Laplacian on the unit square but the order of A was $n = 256$ in this case. The trust-region radius was set at $\Delta = 10$ for all of the problems. Again, a random vector g was constructed with entries uniformly distributed on (-.5 , .5) and the problem was solved with a tolerance of 10^{-11}. In Table 3 the progressive decrease in the magnitude of $\frac{\Delta - \|x\|}{\Delta}$ is charted as the iteration proceeds. The required number of iterations was 6 and it took 144 matrix-vector products to solve the associated eigenvalue problems. Each eigenproblem was solved to the accuracy level $\|B_\alpha v - v\lambda\| \leq 10^{-9}$

$Iter$	$\frac{\Delta - \|x\|}{\Delta}$
1	0.8730
2	-0.1028
3	0.0063
4	7.1389e-05
5	-4.8522e-08
6	1.2491e-12

Table 3: Verification of superlinear convergence

To study the behavior of the algorithm in the hard case, the same matrix $A = L - .5I$ of order 256 was used. In order to generate the hard case the vector g was randomly generated as before and then the operation $g \leftarrow g - q(q^T g)$ was performed to orthogonalize g to the eigenvector q corresponding to the smallest eigenvalue of A. Then a "noise" vector of norm 10^{-8} was added to g. In this test the trust-region radius was $\Delta = 100$. A number of different problems were solved and the following behavior of one of the problems was typical of all of them. In every problem the hard case was detected on the second regular iteration of Algorithm 1 and then the iteration of Algorithm 2 was entered. Table 3 displays the ratio

$$\left| \frac{z^T(A - \lambda I)z}{(g^T p + \lambda \Delta^2)} \right|$$

and the iteration was halted when this ratio was less than $\sigma = .000001$ where σ is the tolerance introduced in Lemma (5.2) (see Step 4 of Algorithm 2).

$Iter$	$\left\| \frac{z^T(A-\lambda I)z}{(g^T p + \lambda\Delta^2)} \right\|$
1	0.2916
2	0.1448
3	0.0631
4	0.0221
5	0.0049
6	3.8942e-04
7	2.9174e-06
8	8.5908e-09

Table 4: The Hard Case

The final solution was on the trust-region boundary to within working precision and it required a total of 11 eigenvalue problems which required 291 matrix-vector products . One of these was done between the transition from Algorithm 1 to Algorithm 2 in order to assure a lower bound on α had been obtained. This step is not reported in the Table 4. The behavior of this iteration seemed to be more sensitive to level of accuracy required by the eigen-solution then in the standard case. A rational approximation was tried instead of the linear interplation and this performed poorly. However, more testing is needed and perhaps a modification of the scheme for the hard case will lead to improvements. No testing was done on large matrices since it was desirable to have complete control over which eigenvectors the vector g would be orthogonalized against. Moreover, no testing was done with higher dimensional eigenspaces corresponding to the smallest eigenvalue. Finally, special consideration may be called for in the case of least squares problems arising from the discretization of ill-posed continuous problems . These problems will be of the form $min\{\|Mx - b\| \, : \, \|x\| \leq \Delta\}$ and for ill posed problems the matrix $A = M^T M$ will be singular or nearly singular and the vector $g = M^T b$ will be orthogonal or nearly orthogonal to the corresponding null space of A. The method described by Golub and von Matt [6] may be better suited to this situation and this comparison should be made.

Although a direct comparison to the Secant method has not been made here, the results that have been compiled with respect to the performance of the conjugate-gradient iteration may be used to draw some conclusions. Two possibilities for a Secant iteration come to mind. The first would be to apply the Secant Method directly to the problem of adjusting λ to obtain

$$\frac{1}{\Delta} - \frac{1}{\|x_\lambda\|} = 0.$$

using the conjugate-gradient method to solve the resulting linear systems of the form $(A - \lambda I)x = -g$. An immediate problem with this approach is to discover the range of λ for which $(A - \lambda I)$ is positive definite. Moreover, the systems that would have to be solved would be as computationally demanding for the conjugate-gradient iteration as the ones arising within the iteration presented here. The computational results indicate this approach would be inferior to the eigenvalue approach for modest to large trust-region radii and roughly comparable for small radii.

A second possibility would be to use the eigenvalue formulation (2.2) to obtain points $x_{\lambda(\alpha)}$ but to apply the Secant Method to the problem

$$\frac{1}{\Delta} - \frac{1}{\|x_{\lambda(\alpha)}\|} = 0$$

in order to adjust the parameter α instead of using the specialized iteration derived in Section 4. This method was coded and computational tests showed it to be inferior to the method presented here. It took many more iterations in general than the specialized iteration based upon rational interpolation.

These results indicate promise for this approach to solving the large scale trust-region subproblem. The examples given here were solved to tolerances which are unlikely in to arise in most applications. This was done to get some indication of the asymptotic behavior and to verify the convergence results presented in Section 6. While these preliminary tests are very encouraging, further experience with testing and with actual application will be necessary.

References

[1] R. H. Byrd , R. B. Schnabel and G. A. Shultz, A trust region algorithm for nonlinearly constrained optimization, *SIAM J. Numer. Anal.* 24, 1152–1170, (1987).

[2] A.K. Cline, C.B. Moler, G.W. Stewart, J.H. Wilkinson, An estimate for the condition number of a matrix, *SIAM J. Numer. Anal.* 16, 368-375,(1979).

[3] M. Celis and J. E. Dennis and R. A. Tapia, A trust region strategy for nonlinear equality constrained optimization,*Numerical Optimization 1984* , SIAM, Philadelphia,71–82, (1985).

[4] M. El–Alem, A global convergence theory for the Celis–Dennis–Tapia trust region algorithm for constrained optimization, *SIAM J. Numer. Anal.* 28, 266–290,(1991).

[5] W. Gander, Least squares with a quadratic constraint. *Numer. Math.* 36, 291-307,(1981).

[6] G.H. Golub and U. von Matt, "Quadratically constrained least squares and quadratic problems" , *Num. Math.* vol. 59, pp. 561-580 (1991).

[7] W. Menke, *Geophysical Data Analysis: Discrete Inverse Theory* , Academic Press, San Diego, (1989).

[8] J.J. More', Recent developments in algorithms and software for trust region methods, *Mathematical Programming: the State of the Art* , A. Bachem, M. Grotschel, B. Korte eds., Springer-Verlag, Berlin, 258-287, (1983).

[9] J. J. More' and D.C. Sorensen, Computing a Trust Region Step, *SIAM J. on Scientific and Statistical Computing*,4 , 553-572 (1983)

[10] B.N. Parlett, The Symmetric Eigenvalue Problem , Prentice-Hall, Englewood Cliffs, NJ. (1980).

[11] M. J. D. Powell and Y. Yuan, A trust region algorithm for equality constrained optimization, *Mathematical Programming* 49 ,189–211, (1991).

[12] D. C. Sorensen, "Implicit application of polynomial filters in a k-step Arnoldi method," *SIAM J. Matrix Anal. Appl.*, vol. 13, pp. 357–385, 1992.

[13] D.C. Sorensen, Newton's method with a model trust region modification, *SIAM J. Numer. Anal.*, 16, 409-426 (1982).

[14] R. Lehoucq, D.C. Sorensen, P.A. Vu, ARPACK: Fortran subroutines for solving large scale eigenvalue problems, Release 2.1, available from sorensen@rice.edu, (1994).

[15] A. Tarantola, *Inverse Problem Theory*, Elsevier, Amsterdam, (1987).

[16] T. Steihaug, The conjugate gradient method and trust regions in large scale optimization, *SIAM J. Numer. Anal.*, 20, 626–637, (1983).

[17] C.C. Paige, B.N. Parlett, H.A. Van der Vorst, Approximate solutions and eigenvalue bounds from Krylov subspaces, U. Utrecht rept. (1993).

[18] Ph. L. Toint , Towards an efficient sparsity exploiting Newton method for minimization, *Sparse Matrices and Their Uses* I. S. Duff,ed., Academic Press, New York, 57–87, (1981).