Very Large Symmetric Eigenvalue Problems Solved by Lanczos and Multiple Relatively Robust Representations for Tridiagonals

JOHANNES RYDH

KTH Computer Science
and Communication

Master of Science Thesis
Stockholm, Sweden 2007

Very Large Symmetric Eigenvalue Problems Solved by Lanczos and Multiple Relatively Robust Representations for Tridiagonals

J O H A N N E S R Y D H

Master's Thesis in Numerical Analysis (30 ECTS credits)
at the School of Engineering Physics
Royal Institute of Technology year 2007
Supervisor at CSC was Axel Ruhe
Examiner was Axel Ruhe

TRITA-CSC-E 2007:123
ISRN-KTH/CSC/E--07/123--SE
ISSN-1653-5715

Royal Institute of Technology
School of Computer Science and Communication

KTH CSC
SE-100 44 Stockholm, Sweden

URL: www.csc.kth.se

Abstract

The Lanczos method is the primary algorithm used to solve very large symmetric eigenvalue problems. One of its properties is that explicit reorthogonalization is required, which can be expensive. An alternative solution, first proposed by Cullum and Willoughby, is to perform the Lanczos iteration without reorthogonalization, and apply a criterion to distinguish between the real Ritz values and the so-called "spurious" Ritz values that occur as a result of the non-orthogonal Lanczos basis.

This thesis is a study of one such method in combination with the Multiple Relatively Robust Representations algorithm to solve the tridiagonal eigenproblem, as well as a comparison to a procedure using Lanczos with selective reorthogonalization.

Referat

Lösning av mycket stora symmetriska egenvärdesproblem med Lanczos metod och Multiple Relatively Robust Representations för tridiagonala matriser

Lanczos-metoden är den huvudsakliga algoritmen för att lösa stora symmetriska egenvärdesproblem. En av dess egenskaper är att explicit reortogonalisering krävs, vilket kan vara kostsamt. En alternativ lösning som först föreslogs av Cullum och Willoughby är att utföra Lanczos-iterationerna utan reortogonalisering, och använda ett kriterium för att skilja mellan riktiga Ritzvärden från de falska som förekommer som en följd av den icke-ortogonala basen.

Detta projekt är en studie av en sådan metod, i kombination med algoritmen Multiple Relatively Robust Representations för att lösa det tridiagonala egenvärdesproblemet, samt en jämförelse med en Lanczos-metod med selektiv reortogonalisering

Contents

Chapter 1

Introduction

1.1 Background

One very common problem arising in many engineering and scientific applications is the real symmetric (or Hermitian, in the complex case) eigenvalue problem,

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x}, \qquad \mathbf{A}^T = \mathbf{A}. \tag{1.1}$$

The classic approach to solving such problems is to reduce the matrix to tridiagonal form by orthogonal similarity transformations, followed by an iteration to further reduce the tridiagonal matrix to diagonal form. This method works well and is efficient for small matrices. However, in many applications, for example in quantum chemistry, the size of the matrix is of order 10^4 or 10^5. Using direct methods such as Householder to explicitly manipulate matrix entries in order to introduce zeros then becomes an unacceptably slow process, and in addition generally destroys any sparsity present in $\mathbf{A}$.

Because of this, the use of iterative algorithms is required for large matrices, and today the Lanczos iteration for tridiagonalization is widely used for this purpose. Unfortunately, in the context of a finite precision computation, a straightforward implementation does not yield orthogonal Lanczos vectors, which seemingly destroys the usefulness of the results.

Various strategies of explicit reorthogonalization of the Lanczos vectors have been proposed. Full reorthogonalization of all vectors is very simple to implement and gives very good results, but incurs a substantial computation penalty. Selective reorthogonalization is a more economic variant, requiring a slightly more complicated implementation. A new vector is orthogonalized against the previous vectors only if the loss of orthogonality has passed a certain threshold. This method yields nearly the accuracy of the scheme with full reorthogonalization, while requiring much less work.

1.2 Problem

A natural question arises: is it possible to obtain accurate and orthogonal eigenvectors in reasonable time from Lanczos without any reorthogonalization at all? This would make each iteration very cheap.

1.3 Purpose

In this thesis a method will be presented to solve the symmetric eigenvalue problem using Lanczos tridiagonalization procedure without reorthogonalization. Using a method based upon work by Cullum and Willoughby, spurious Ritz values are removed. To drastically improve the running time of the intermediate tridiagonal eigenvalue problem, allowing larger problems, the recent algorithm Multiple Relatively Robust Representation (MRRR) [7] is used instead of more conventional methods. MRRR has the desirable property of being able to compute n orthogonal eigenvectors in only $O(n^2)$ complexity and is very well suited for this problem.

1.4 Structure of this thesis

Chapter 1 gives a brief overview of the problem and the motivation behind this method. In chapter 2, the details of the Lanczos iteration and the modifications we do are given, together with a short explanation of the MRRR algorithm that is used. The obtained results are presented in chapter 3 and a summary of the conclusions can be found in chapter 4.

Chapter 2

Theory

2.1 The Lanczos method

The Lanczos iteration works by computing the elements of the tridiagonal matrix $\mathbf{T} = \mathbf{V}^T\mathbf{A}\mathbf{V}$ directly. Let

$$\mathbf{T} = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{m-1} \\ & & \beta_{m-1} & \alpha_m \end{bmatrix} \tag{2.1}$$

and let $\mathbf{v}_1 \dots \mathbf{v}_m$ be the columns of the orthogonal matrix $\mathbf{V}$. Rewriting $\mathbf{AV} = \mathbf{VT}$ and equating the columns gives the three-term recursion

$$\mathbf{A}\mathbf{v}_k = \beta_{k-1}\mathbf{v}_{k-1} + \alpha_k\mathbf{v}_k + \beta_k\mathbf{v}_{k+1}. \tag{2.2}$$

Using the orthonormality property of the $\mathbf{v}$-vectors we obtain the following formulas for the tridiagonal elements:

$$\alpha_k = \mathbf{v}_k^T(\mathbf{A}\mathbf{v}_k - \beta_{k-1}\mathbf{v}_{k-1}) \tag{2.3}$$

$$\beta_k = ||\mathbf{A}\mathbf{v}_k - \alpha_k\mathbf{v}_k - \beta_{k-1}\mathbf{v}_{k-1}||_2. \tag{2.4}$$

There are several mathematically equivalent formulas, but it has been shown by Paige [11, 12] that (2.3)–(2.4) is the numerically most stable form.

2.1.1 The Lanczos iteration

Putting this together gives the *Lanczos iteration*:

ALGORITHM 1: THE LANCZOS ITERATION

1. for $k = 1, 2, \ldots, m$
2. if $k = 1$ then
3. $\mathbf{r} = \mathbf{A}\mathbf{v}_k$
4. else
5. $\mathbf{r} = \mathbf{A}\mathbf{v}_k - \beta_{k-1}\mathbf{v}_{k-1}$
6. end
7. $\alpha_k = \mathbf{v}_k^T\mathbf{r}$
8. $\mathbf{r} = \mathbf{r} - \alpha_k\mathbf{v}_k$
9. $\beta_k = ||\mathbf{r}||_2$
10. $\mathbf{v}_{k+1} = \mathbf{r}/\beta_k$
11. end

With $\mathbf{r}$ defined by algorithm 1, the recursion can also be written

$$\mathbf{A}\mathbf{V}_k = \mathbf{V}_k\mathbf{T}_k + \mathbf{r}\mathbf{e}_k^T, \tag{2.5}$$

where $\mathbf{V}_k = [\mathbf{v}_1, \ldots, \mathbf{v}_k]$.

Algorithm 1 has many desirable properties. The matrix $\mathbf{A}$ only appears in a matrix-vector multiplication, which can be greatly optimized (mainly by exploiting the sparsity of $\mathbf{A}$). Also, only the two most recent Lanczos vectors need to be present in memory, and the orthogonality with all previous vectors is mathematically guaranteed (although, as we have already noted, this is not true in practice). This makes the cost of each iteration very low.

2.1.2 Convergence and orthogonality

The Lanczos iteration is coupled with a diagonalization of the resulting $m \times m$ tridiagonal matrix, $\mathbf{T}_m = \mathbf{S}\mathbf{\Lambda}\mathbf{S}^T$. $\mathbf{V}$ being non-singular, the tridiagonalization is a similarity transformation, and the eigenvalues of $\mathbf{T}_m$ (the Ritz values) are also eigenvalues of $\mathbf{A}$, and the corresponding eigenvectors are $\mathbf{X} = \mathbf{V}\mathbf{S}$.

The convergence of a Ritz value θ_k and its corresponding Ritz vector $\mathbf{x}_k = \mathbf{V}\mathbf{s}_k$ to

an eigenpair of the original matrix $\mathbf{A}$ can be observed by estimating the residual:

$$\begin{aligned} ||\mathbf{A}\mathbf{x}_k - \theta_k \mathbf{x}_k||_2 &= ||\mathbf{A}\mathbf{V}_m \mathbf{s}_k - \theta_k \mathbf{V}_m \mathbf{s}_k||_2 \\ &= ||(\mathbf{A}\mathbf{V}_m - \mathbf{V}_m \mathbf{T}_m)\mathbf{s}_k||_2 \\ &= ||\mathbf{r}\mathbf{e}_k^T \mathbf{s}_k||_2 = \beta_m |s_{m,k}|. \end{aligned} \tag{2.6}$$

When this quantity is small, we can expect the Ritz value θ_k to be an approximation to the true eigenvalue λ_k.

Paige showed in [10, 12] that the computed $\mathbf{V}_m$, $\mathbf{T}_m$ and $\mathbf{r}$ satisfies (2.5) to working precision. However, the orthogonality of the computed Lanczos vectors satisfy

$$|\mathbf{v}_{m+1}^T \mathbf{v}_i| \approx \frac{|\mathbf{r}^T \mathbf{v}_i| + \epsilon_{\text{mach}} ||A||_2}{|\beta_m|}, \qquad i = 1, \ldots, m \tag{2.7}$$

where ϵ_{mach} is the machine epsilon. This means that when $|\beta_m|$ is small, there is a loss of orthogonality.

Further analysis in [12] shows that this loss of orthogonality occurs when a Ritz pair converges:

$$|\mathbf{v}_{m+1}^T \mathbf{x}_i| \approx \frac{\epsilon_{\text{mach}} ||\mathbf{A}||_2}{\beta_m |s_{m,i}|}, \qquad i = 1, \ldots, m. \tag{2.8}$$

This tells us that the latest computed Lanczos vector has a large component in the direction of an already converged Ritz vector. It is this observation upon which selective reorthogonalization is based, and it motivates the strategy of orthogonalizing only against the subset of converged Lanczos vectors. The phenomenon is illustrated in figure 2.1, where $||\mathbf{V}_k^T \mathbf{v}_{k+1}||_2$ and $\beta_k |s_{k,i}|$ for the first converging Ritz pair are plotted.

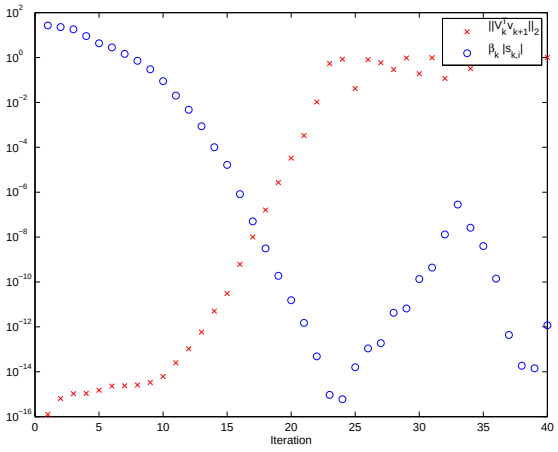

Figure 2.1. *The loss of orthogonality in the Lanczos base is simultaneous with the first convergence of a Ritz pair.*

2.1.3 Lanczos without reorthogonalization

Without explicit reorthogonalization in some form, three concerns appear. Due to the loss of orthogonality of the Lanczos vectors, it is not immediately obvious that we can obtain a good set of orthogonal eigenvectors at all, possibly rendering the method unusable.

Secondly, the Ritz values converge to eigenvalues of $\mathbf{A}$ significantly slower than in the theoretical infinite-precision machine, or with a scheme including reorthogonalization. The number of iterations m required for the desired number of eigenvalues is very large, frequently greater than n itself.

Thirdly, during the iteration there will appear a significant number of so-called "spurious" Ritz values. They converge towards already converged Ritz values but do not necessarily attach themselves to any single eigenvalue. These "ghost" copies do not represent multiplicity of the eigenvalues but are purely an artifact of the non-orthogonality. Measures must be taken to characterize them apart from the desired values. Figure 2.2 shows the appearance of several spurious Ritz values in a Lanczos iteration following algorithm 1 on the Strakoš-30 matrix [9].

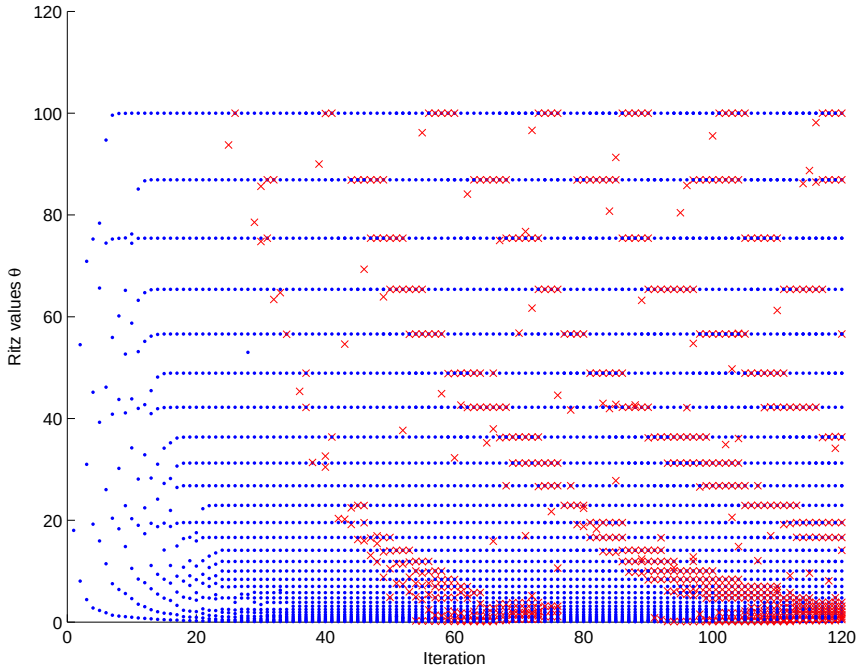

Figure 2.2. *Lanczos iteration on the Strakoš-30 matrix. Crosses (×) represent "spurious" Ritz values.*

For the largest eigenvalue $\lambda = 100$, spurious Ritz values appear and start converging towards it at iterations 25, 39, 55, 72, 85, 100 and 116, and after the 120 iterations

there are thus 8 copies of that Ritz value. Note that according to figure 2.2, the extra copies seem to lose their status as "spurious" Ritz values after a couple of iterations; this will be discussed further in the next section.

2.2 A criterion to detect spurious Ritz values

In [3], Cullum and Willoughby presented the following method to determine whether a Ritz value should be kept or discarded.

Let $\mathbf{A}$ be the symmetric matrix whose eigenvalues we are interested in and $\mathbf{T}_m$ be the tridiagonal matrix of size m obtained from algorithm 1 applied to $\mathbf{A}$:

$$\mathbf{T}_m = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{m-1} \\ & & \beta_{m-1} & \alpha_m \end{bmatrix}. \tag{2.9}$$

Let $\hat{\mathbf{T}}_{2,m}$ denote the symmetric tridiagonal matrix obtained from $\mathbf{T}_m$ by removing the first column and row, that is

$$\hat{\mathbf{T}}_{2,m} = \begin{bmatrix} \alpha_2 & \beta_2 & & \\ \beta_2 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{m-1} \\ & & \beta_{m-1} & \alpha_m \end{bmatrix}. \tag{2.10}$$

To identify the spurious eigenvalues, we compare the eigenvalues of $\mathbf{T}_m$ and $\hat{\mathbf{T}}_{2,m}$.

Criterion 1 *Any simple eigenvalue of $\mathbf{T}_m$ that is also very close to an eigenvalue of $\hat{\mathbf{T}}_{2,m}$ is labeled "spurious". The remaining eigenvalues are taken as approximations of the eigenvalues of $\mathbf{A}$. In particular, a numerically multiple eigenvalue of $\mathbf{T}_m$ is a good approximation, but only one copy should be kept and the rest discarded.*

Note that in a practical implementation of criterion 1, one would not actually compute any eigenvalues of $\hat{\mathbf{T}}_{2,m}$. Instead, the test can be performed using two Sturm sequences, one on the matrix $\hat{\mathbf{T}}_{2,m} - (\mu+\epsilon)\mathbf{I}$ and one on $\hat{\mathbf{T}}_{2,m} - (\mu-\epsilon)\mathbf{I}$. The number of sign changes in the sequence of determinants of the principle submatrices of the above matrices gives the number of eigenvalues less than $\mu+\epsilon$ and $\mu-\epsilon$, respectively, and this shows if there exists an eigenvalue of $\hat{\mathbf{T}}_{2,m}$ within ϵ from a given eigenvalue μ of $\mathbf{T}_m$.

Once a spurious Ritz value has converged sufficiently towards another Ritz value that they are considered numerically multiple, criterion 1 stops labeling them as "spurious", and they are instead discarded due to the second clause of the criterion.

2.2.1 A plausibility argument for criterion 1

In [2], Cullum and Willoughby relates the Lanczos tridiagonalization to the Conjugate Gradient optimization procedure of solving the linear system $\mathbf{Ax} = \mathbf{v}_1$ starting with the vector $\mathbf{x}_1 = \mathbf{0}$. The arguments assume that $\mathbf{A}$ is positive definite, but the conclusions apply to any symmetric matrix. In theory, for a given symmetric matrix $\mathbf{A}$, the Lanczos recursion would give the same Lanczos vectors for a shifted matrix $\mathbf{A} + \sigma\mathbf{I}$. Thus, if $\mathbf{A}$ is not positive definite, one can consider the following relations between the Lanczos tridiagonalization and the Conjugate Gradient optimization procedure as applying to a shifted matrix, with a shift such that it is positive definite. In practice, of course, shifts are not introduced into the computation.

Let the vectors $\mathbf{v}_i$ and the scalars α_i and β_i be determined by the Lanczos recursions (2.2)–(2.4).

The associated Conjugate Gradient optimization procedure generates the search directions $\mathbf{p}_i$ from the residuals $\mathbf{r}$ by

$$\mathbf{p}_{i+1} = \mathbf{r}_{i+1} + \gamma_i \mathbf{p}_i \tag{2.11}$$

where

$$\gamma_i = -\mathbf{p}_i^T \mathbf{A}\mathbf{r}_{i+1} / \mathbf{p}_i^T \mathbf{A}\mathbf{p}_i. \tag{2.12}$$

The procedure steps forward along $\mathbf{p}_i$ with

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{p}_i / \omega_i \tag{2.13}$$

where

$$\omega_i = \mathbf{p}_i^T \mathbf{A}\mathbf{p}_i / \mathbf{p}_i^T \mathbf{r}_i. \tag{2.14}$$

To relate the two procedures, we define for each $1 \leq i \leq m$ the scalars ω_i and γ_i by the solution of

$$\alpha_{i+1} = \omega_{i+1} + \gamma_i \omega_i, \quad \omega_1 = \alpha_1, \quad \beta_i = -\sigma_i \omega_i, \quad \gamma_i = \sigma_i^2. \tag{2.15}$$

We also define

$$\rho_{i+1} = \sigma_i \rho_i, \quad \rho_1 = 1 \tag{2.16}$$

$$\mathbf{A}\mathbf{u}_i = \beta_i \mathbf{v}_{i+1} + \omega_i \mathbf{v}_i \tag{2.17}$$

$$\mathbf{p}_i = \rho_i \mathbf{u}_i, \mathbf{r}_i = \rho_i \mathbf{v}_i, \tag{2.18}$$

where $\rho_m = ||-\mathbf{A}\mathbf{x}_m + \mathbf{v}_1||$ is the norm of the residual in the associated Conjugate Gradient optimization.

Let $a_m(\mu) = \det(\mathbf{T}_m - \mu\mathbf{I})$ denote the characteristic polynomial of $\mathbf{T}_m$. Similarly, $\hat{a}_{2,m}(\mu) = \det(\hat{\mathbf{T}}_{2,m} - \mu\mathbf{I})$ is the characteristic polynomial of the reduced matrix $\hat{\mathbf{T}}_{2,m}$.

If μ is an eigenvalue of $\mathbf{T}_m$, that is, $a_m(\mu) = 0$, then

$$\hat{a}_{2,m}(\mu)a_{m-1}(\mu) = \prod_{k=1}^{m-1} \beta_k^2.$$

This equality holds for every symmetric tridiagonal matrix and is not related to the Lanczos tridiagonalization.

Using (2.15), we have $\prod_{k=1}^{m-1} \beta_k^2 = \prod_{k=1}^{m-1} \sigma_k^2 \omega_k^2$. From (2.16) we have $\rho_m = \prod_{k=1}^{m} \sigma_k$. It is also easy to show that $\prod_{k=1}^{m} \omega_k = a_m$. If $a_{m-1} = \det(\mathbf{T}_{m-1}) \neq 0$ then

$$\frac{\hat{a}_{2,m}(\mu)}{a_{m-1}} \frac{a_{m-1}(\mu)}{a_{m-1}} = \rho_m^2. \tag{2.19}$$

Even under fairly weak assumptions, this residual can be expected to converge to zero as $m \to \infty$, as shown in [2]. Equation (2.19) then gives that when convergence occurs, μ is close to an eigenvalue of either $\mathbf{T}_{m-1}$ or $\hat{\mathbf{T}}_{2,m}$.

Furthermore, one can derive an upper bound for the error in the approximation of a Ritz value μ as an eigenvalue of $\mathbf{A}$[3]:

$$\epsilon_\mu = \frac{\rho_{m-1}}{\sqrt{\frac{a'_m(\mu)}{a_m} \frac{\hat{a}_{2,m}(\mu)}{a_m}}} \tag{2.20}$$

The $a'_m(\mu)$ factor is a measure of how isolated an eigenvalue μ is. When considering simple eigenvalues, $\hat{a}(\mu)$ becomes the dominating factor. When μ is an eigenvalue of $\hat{\mathbf{T}}_{2,m}$ then the error bound is large, which suggests that it is not a very good approximation to an eigenvalue of $\mathbf{A}$. With a high probability, the Ritz value is spurious and should be discarded. If μ is close to another eigenvalue then we have a numerically multiple eigenvalue, and such an eigenvalue is an accurate approximation to an eigenvalue of $\mathbf{A}$ and we need not consider it further.

It should be noted that the arguments behind criterion 1 has not been universally accepted; in [9], Meurant argues that the residual ρ_m of the Conjugate Gradient optimization procedure cannot always be expected to go to zero.

2.2.2 A restatement of the criterion

Criterion 1 can be restated in a simpler form:

Criterion 2 *Let $\mathbf{T}_m = \mathbf{S}\mathbf{\Lambda}\mathbf{S}^T$ be obtained from a Lanczos tridiagonalization procedure without reorthogonalization. A simple Ritz value $\theta_k \in \lambda(\mathbf{T}_m)$ is "spurious" if the first component of the corresponding eigenvector $\mathbf{s}_k$ is small.*

The equivalence of criteria 1 and 2 is trivial to show. Assume that $\mathbf{T}_m\mathbf{s}_k = \theta_k\mathbf{s}_k$ and $|s_{1,k}| < \epsilon$. Then

$$\left\| \hat{\mathbf{T}}_{2,m} \begin{bmatrix} s_{2,k} \\ \vdots \\ s_{m,k} \end{bmatrix} - \theta_k \begin{bmatrix} s_{2,k} \\ \vdots \\ s_{m,k} \end{bmatrix} \right\|_2 = \beta_2|s_{1,k}| < \beta_2\epsilon, \tag{2.21}$$

that is, θ_k is a good approximation to an eigenvalue of the reduced matrix $\hat{\mathbf{T}}_{2,m}$, given that ϵ is sufficiently small. Conversely, assume that

$$\mathbf{T}_m\mathbf{s}_k = \theta_k\mathbf{s}_k, \qquad \left\| \hat{\mathbf{T}}_{2,m} \begin{bmatrix} s_{2,k} \\ \vdots \\ s_{m,k} \end{bmatrix} - \theta_k \begin{bmatrix} s_{2,k} \\ \vdots \\ s_{m,k} \end{bmatrix} \right\|_2 < \epsilon. \tag{2.22}$$

Then $\beta_2|s_{1,k}| < \epsilon$. Since $\mathbf{T}_m$ is the result of a Lanczos iteration, it is unreduced (except in the case of a breakdown) and β_2 is not small. Hence $|s_{1,k}| < \epsilon/\beta_2$ is small.

2.3 The MRRR algorithm

The Multiple Relatively Robust Representations algorithm (MRRR, or MR^3) [7] is a fairly new algorithm for solving symmetric tridiagonal eigenvalue problems. It can easily compete with any existing algorithm in terms of speed, computing k orthogonal eigenvectors in just $O(nk)$ time in the worst case, while one of its primary contester, Divide-and-Conquer, uses $O(n^{2.3})$ on average and may require up to $O(n^3)$ or $O(nk^2)$ time in the worst case. When Lanczos is run without reorthogonalization, the resulting tridiagonal matrix will be huge due to the large amount of iterations. The speedup of this tridiagonal phase of the computation that MRRR provides extends the use of this Lanczos variant to larger matrices and more iterations.

The MRRR algorithm does not work with the elements of the tridiagonal matrix directly. Instead, it performs factorizations of form $\mathbf{T} - \tau\mathbf{I} = \mathbf{L}\mathbf{D}\mathbf{L}^T$, where $\mathbf{L}$ is unit bidiagonal and $\mathbf{D}$ is diagonal. In [13], it is shown that such a representation of the matrix can determine a subset Γ of the eigenvalues to high relative accuracy, that is to say, small relative perturbations of the non-trivial elements of $\mathbf{L}$ and $\mathbf{D}$ only cause small relative changes in the eigenvalues $\lambda \in \Gamma$. It is then called a relatively robust representation (RRR) for Γ.

The central piece of MRRR is the eigenvector computation [6]. Let $\hat{\lambda}$ be an accurate approximation of the eigenvalue λ of $\mathbf{L}\mathbf{D}\mathbf{L}^T$ that has a large relative gap, that is to say,

$$\frac{\text{gap}(\hat{\lambda})}{|\lambda|} > \text{tol}, \quad \text{gap}(\hat{\lambda}) = \min\{|\hat{\lambda} - \mu| : \mu \neq \lambda, \mu \in \text{spectrum}(\mathbf{L}\mathbf{D}\mathbf{L}^T)\}. \tag{2.23}$$

Then $\mathbf{LDL}^T - \hat{\lambda}\mathbf{I}$ is almost singular, and the sought eigenvector is the solution to $(\mathbf{LDL}^T - \hat{\lambda}\mathbf{I})\mathbf{z} \approx \mathbf{0}$. The algorithm constructs the factorizations

$$\mathbf{LDL}^T - \hat{\lambda}\mathbf{I} = \mathbf{L}_+\mathbf{D}_+\mathbf{L}_+^T \tag{2.24}$$

and

$$\mathbf{LDL}^T - \hat{\lambda}\mathbf{I} = \mathbf{U}_-\mathbf{D}_-\mathbf{U}_-^T, \tag{2.25}$$

where $\mathbf{L}_+$ is lower unit bidiagonal and $\mathbf{U}_-$ is upper unit bidiagonal. The matrices $\mathbf{L}_+$ and $\mathbf{U}_-$ can be formed directly the elements of $\mathbf{D}$ and $\mathbf{L}$ without ever having to form the product $\mathbf{LDL}^T$.

Using this, is it possible to form a number of "twisted" factorizations

$$\mathbf{N}_k\mathbf{D}_k\mathbf{N}_k^T = \mathbf{LDL}^T - \hat{\lambda}\mathbf{I}, \quad 1 \leq k \leq n \tag{2.26}$$

where $\mathbf{N}_k$ has the rows 1 to k from $\mathbf{L}_+$ and the rows k to n from $\mathbf{U}_-$. For $n = 6, k = 3$, N_k would have the following structure:

$$N_k = \begin{bmatrix} 1 & & & & & \\ \times & 1 & & & & \\ & \times & 1 & \times & & \\ & & & 1 & \times & \\ & & & & 1 & \times \\ & & & & & 1 \end{bmatrix}. \tag{2.27}$$

$\mathbf{D}_k$ is diagonal with the first $k-1$ diagonal elements from $\mathbf{D}_+$ and the last $n-k-1$ elements from $\mathbf{D}_-$. Let γ_k be the kth diagonal element.

For some choice of k, an approximate eigenvector is obtained by solving $\mathbf{N}_k^T\mathbf{z} = \mathbf{e}_k$, which is equivalent to solving $(\mathbf{LDL}^T - \hat{\lambda}\mathbf{I})\mathbf{z} = \mathbf{N}_k\mathbf{D}_k\mathbf{N}_k^T\mathbf{z} = \gamma_k\mathbf{e}_k$, since $\mathbf{D}_k\mathbf{e}_k = \gamma_k\mathbf{e}_k$ and $\mathbf{N}_k\mathbf{e}_k = \mathbf{e}_k$. The choice of k is made so that the residual is minimized.

As noted above, this procedure can only be used for eigenvalues that are relatively separated. For each remaining cluster of eigenvalues, the algorithm chooses a new shift τ_c close to the end of the cluster [7] and constructs a new RRR, $\mathbf{L}_c\mathbf{D}_c\mathbf{L}_c^T = \mathbf{LDL}^T - \tau_c\mathbf{I}$, such that those eigenvalues get large relative gaps and the corresponding eigenvectors can be computed. In this way, a computation tree is built up with an RRR at each node [5].

Chapter 3

Results

Most of my tests were performed on a well-known test matrix obtained by a five-point finite difference Laplacian on a two-dimensional "L"-shaped grid. Figure 3.1 shows the sparsity pattern of a small such matrix. The matrix used in the computation was produced using the Matlab commands `numgrid` and `delsq`, with the size $n = 10092$.

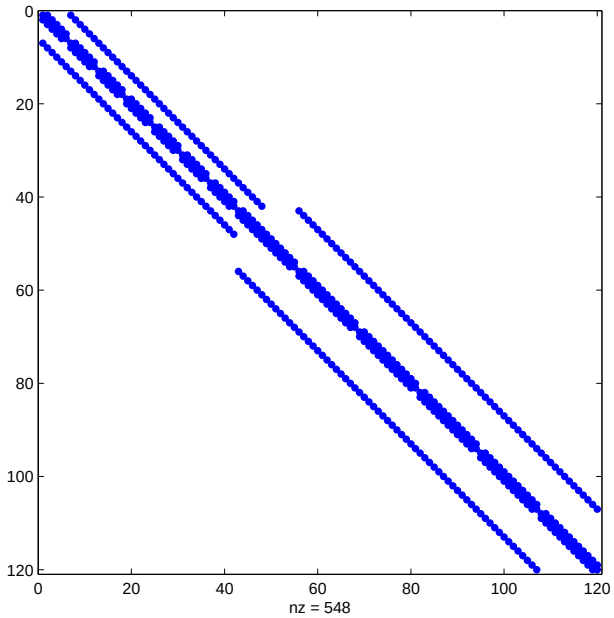

Figure 3.1. *Sparsity pattern of the test matrix.*

After running the algorithm with $m = 6000$ iterations, 77 eigenvalues and eigenvectors have converged at the ends of the spectrum of $\mathbf{A}$. Figure 3.2 show that the

Ritz residuals estimated using (2.6) are all small. In particular, the estimated residual for the smallest eigenvalue 10^{-30}, significantly smaller than for all others. Figure 3.3 show the actual residuals and the orthogonality of the computed eigenvectors. We see that the residuals are all near machine precision, while $||\mathbf{X}^T\mathbf{x}_i - \mathbf{e}_i||_2$ are typically one or two magnitudes larger.

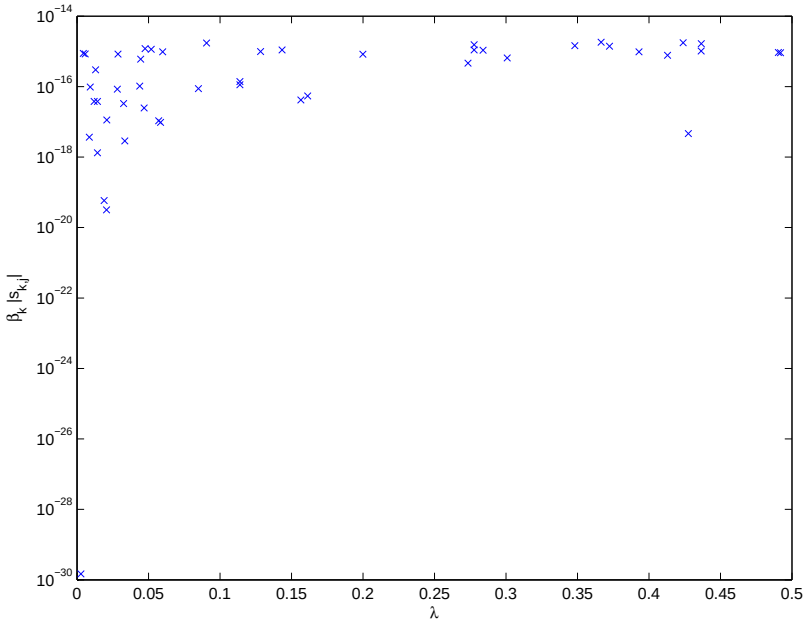

Figure 3.2. *Estimated residuals for Ritz values near the left end of the spectrum. Note the very small estimated residual for the smallest eigenvalue at the lower left corner of the figure.*

The result is thus that even with very non-orthogonal Lanczos vectors we can obtain reasonably orthogonal eigenvectors after weeding out the Ritz vectors corresponding to spurious Ritz values, with good accuracy.

With respect to speed, the scheme with no reorthogonalization is a tradeoff between the cost of each iteration against the number of iterations required. Figure 3.4 and figure 3.5 show the number of eigenvalues converged as a function of iterations, for Lanczos with no and selective reorthogonalization, respectively. It can be seen that the former requires 9000 iterations until 159 eigenvalues have converged, while with reorthogonalization, a similar number is achieved after only 1664 iterations.

Some time measurements were also made, although it should be said that the primary purpose of this thesis was not to give an exhaustive test suite for this. Figure 3.6 shows the cpu time as a function of converged eigenvalues.

The figure shows the computation time with and without the last eigenvector computation at the last step; essentially the matrix-matrix multiplication $\mathbf{X} = \mathbf{VS}$. As

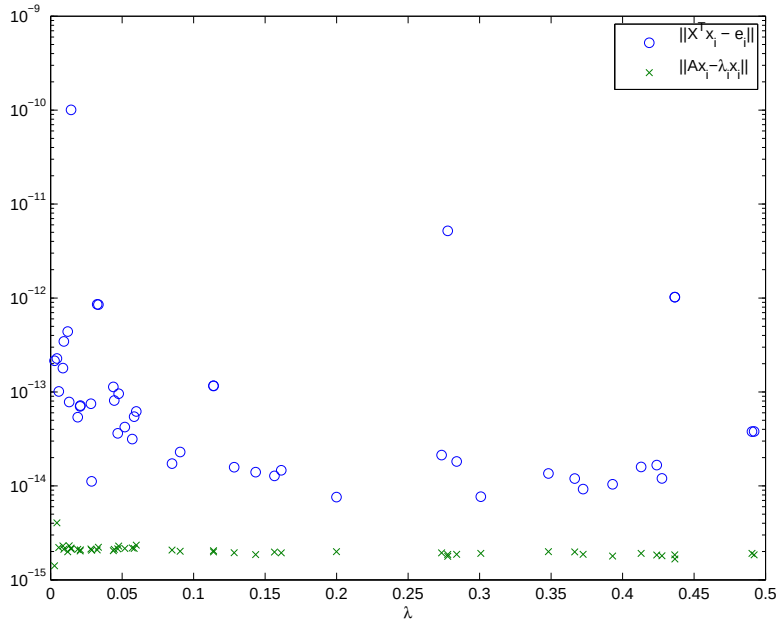

Figure 3.3. *Actual residuals and orthogonality of computed eigenpairs near the left end of the spectrum after 6000 iterations.*

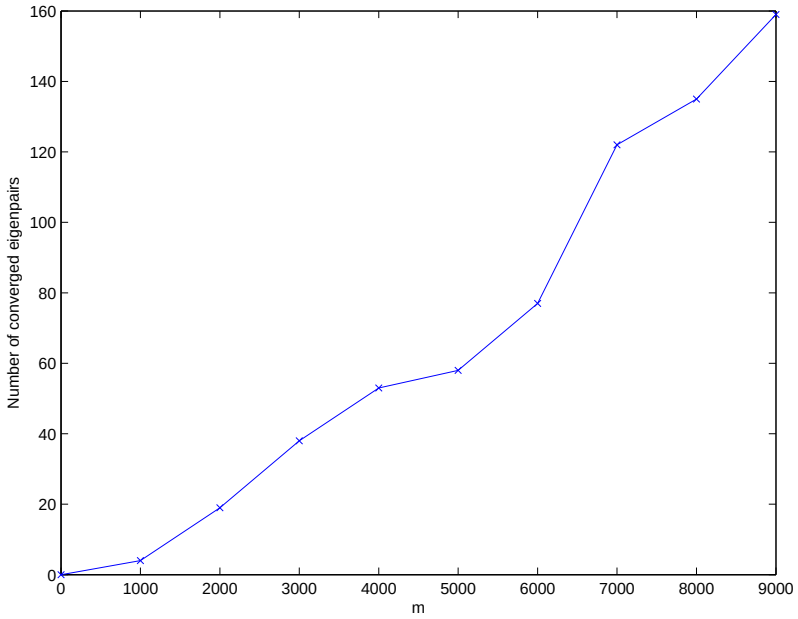

Figure 3.4. *Number of converged eigenpairs as a function of the number of iterations using Lanczos with no reorthogonalization.*

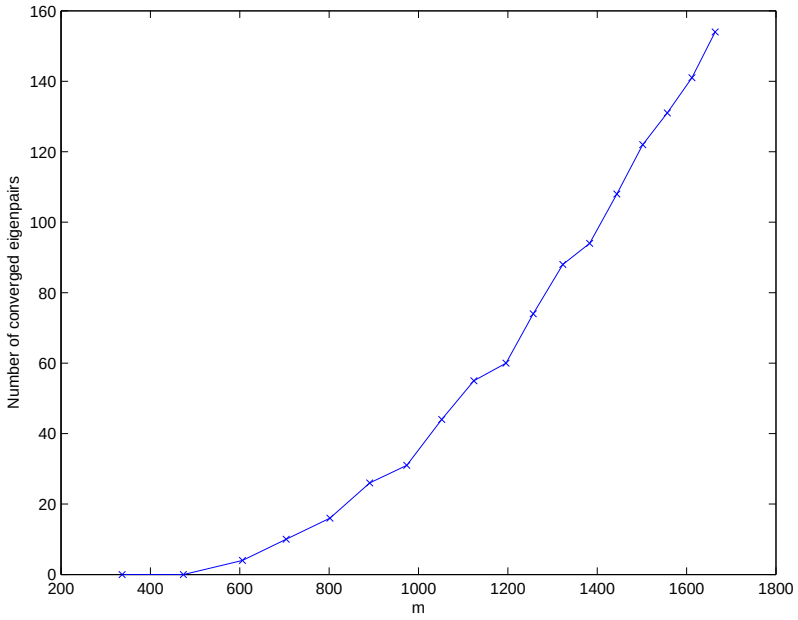

Figure 3.5. *Number of converged eigenpairs as a function of the number of iterations using Lanczos with selective reorthogonalization.*

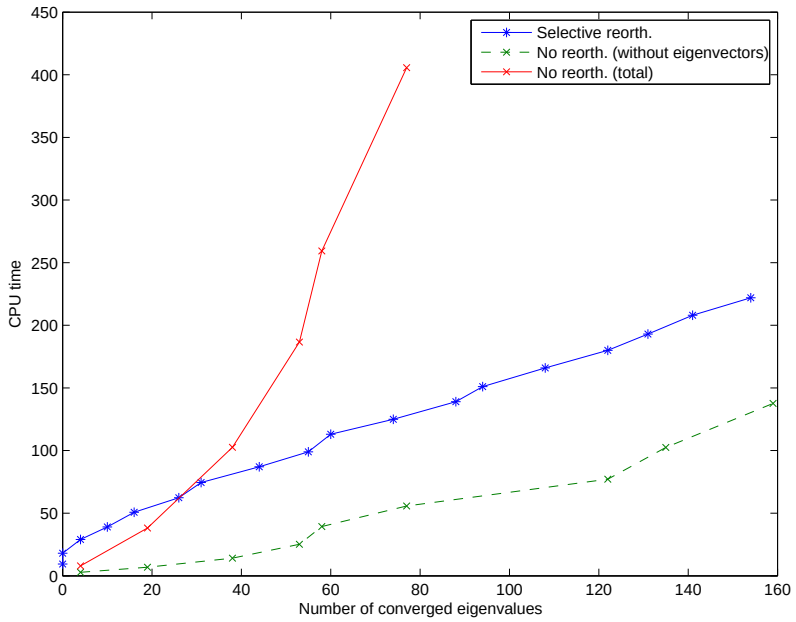

Figure 3.6. *CPU time as a function of the number of converged eigenvalues.*

the figure shows, this is completely dominating. It is unproportionally slow due to the fact that the Lanczos parts of the algorithm was done in Matlab, which in this case produces code which is far from optimal for the computation. Also limiting the test was the amount of available memory on the workstation before swapping became an issue, since the $\mathbf{S}$ matrix becomes very large at higher iteration counts. It would be beneficial if it was possible to extract the first and last components of the eigenvectors from the MRRR algorithm for the convergence check and the application of criterion 2, so that not all eigenvectors need to be computed, reducing the memory footprint. However, that is not possible with the current version.

To give an accurate picture of the performance, one would likely need a more careful implementation of the Lanczos method in Fortran or similarly, and tested in a more suitable environment. However, even with the simple implementations we see that no reorthogonalization can be faster than selective reorthogonalization in some cases, with comparable accuracy.

3.1 Multiple eigenvalues

As a variant of Lanczos, we would not expect to find any multiple eigenvalues of $\mathbf{A}$ with this method. This is due to the fact that $\mathbf{T}_m$ is an unreduced tridiagonal matrix, and such matrices cannot have multiple eigenvalues. We may find numerically multiple eigenvalues of $\mathbf{T}_m$, but only one copy can be used. All but one can be expected to have eigenvectors with a small first component. The code chooses the vector with the largest first component as the eigenvector and applies a Householder reflection to explicitly set the first components of all other vectors to zero.

The above described test matrix does have many multiple eigenvalues, the first appearing being the 8th and 9th eigenvalues ($\lambda_{8,9} = 0.1269$). This is due to the shape of the membrane (see figure 3.7). Seemingly inexplicably, the method *does* give both copies of this eigenvalue. However, their separation is several orders of magnitude larger than the tolerance used, and they are treated as two different eigenvalues, although located very close to each other. This phenomena appears using several other similar iterative algorithms (such as Matlab's `eigs`, which uses implicitly restarted Lanczos) and as such should not be considered to be a property of this particular method.

The MRRR algorithm provides a way to produce several orthogonal eigenvectors to an invariant eigenspace. By selecting different "twisted" factorizations, one can obtain multiple orthogonal eigenvectors.

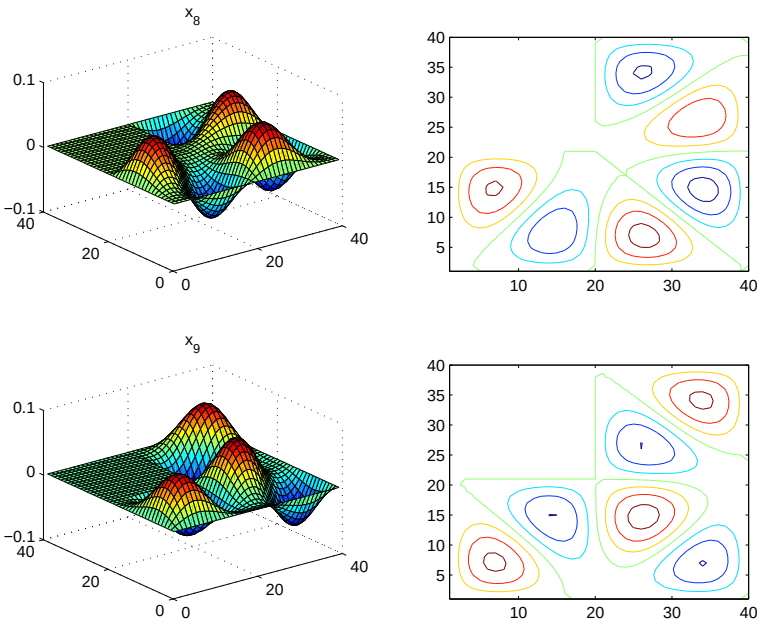

Figure 3.7. *The eigenvectors* $\mathbf{x}_8$ *and* $\mathbf{x}_9$.

Chapter 4

Conclusions

This thesis has described a method to compute eigenvalues and eigenvectors of large symmetric tridiagonal matrices with Lanczos without reorthogonalization. It has been shown that by applying a criterion to select a subset of Ritz values and corresponding Ritz vectors, one obtains a set of eigenvalues and reasonably orthogonal eigenvectors with good precision. Disadvantages include the need to perform a large number of iterations which may result in very large full matrices being formed. It is hard to evaluate the efficiency of the method as compared to Lanczos with selective reorthogonalization. It is clear that there are at least some cases where this method can perform better, but a more careful implementation is required to give an adequate answer to this question.

Bibliography

[1] Paolo Bientinesi, Inderjit S. Dhillon, and Robert A. van de Geijn. A Parallel Eigensolver for Dense Symmetric Matrices Based on Multiple Relatively Robust Representations. *SIAM J. Sci. Comput.*, 27(1):43–66, 2005.

[2] J. Cullum and R. A. Willoughby. The Lanczos Phenomen—An Interpretation Based upon Conjugate Gradient Optimization. *Linear Algebra and Applications*, 29:63–90, 1980.

[3] J. Cullum and R. A. Willoughby. Computing Eigenvalues of Very Large Symmetric Matrices—An Implementation of a Lanczos Algorithm with No Reorthogonalization. *Journal of Computational Physics*, 44:329–358, 1981.

[4] J. W. Demmel. *Applied Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, Philadelphia, USA, 1997.

[5] Inderjit S. Dhillon and Beresford N. Parlett. Multiple Representations to Compute Orthogonal Eigenvectors of Symmetric Tridiagonal Matrices. *Linear Algebra and its Applications*, 387:1–28, 2004.

[6] Inderjit S. Dhillon and Beresford N. Parlett. Orthogonal Eigenvectors and Relative Gaps. *SIAM J. Matrix Anal. Appl.*, 25(3):858–899, 2004.

[7] Inderjit S. Dhillon, Beresford N. Parlett, and Christof Vömel. The Design and Implementation of the MRRR Algorithm. *ACM Trans. Math. Softw.*, 32(4):533–560, 2006.

[8] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The John Hopkins University Press, Baltimore, USA, third edition, 1996.

[9] Gérard Meurant. *The Lanczos and Conjugate Gradient Algorithms: From Theory to Finite Precision Computations*. SIAM, Philadelphia, USA, 2006.

[10] C. C. Paige. *The Computation of Eigenvalues and Eigenvectors of Very Large Sparse Matrices*. PhD thesis, London University, London, UK, 1971.

[11] C. C. Paige. Computational Variants of the Lanczos Method for the Eigenproblem. *IMA J. Appl. Math.*, 10:373–381, 1972.

[12] C. C. Paige. Error Analysis of the Lanczos Algorithm for Tridiagonalizing a Symmetric Matrix. *IMA J. Appl. Math.*, 18:341–349, 1976.

[13] Beresford N. Parlett and Inderjit S. Dhillon. Relatively Robust Representations of Symmetric Tridiagonals. *Linear Algebra and its Applications*, 309:121–151, 2000.

[14] Lloyd N. Trefethen and David Bau, III. *Numerical Linear Algebra*. SIAM, Philadelphia, USA, 1997.

Appendix A

Code

The code used for this thesis can also be sent in electronic format upon request.

A.1 lancno.m

```
% lancno.m
% Lanczos with no reorthogonalization

clear all

% matrix
G = numgrid('L',118);
A = delsq(G);
n = length(A);

% random starting vector
%z = randn(n,1); startvec = z/norm(z);
load sv10k.mat % pre-generated random startvec
v = startvec;

% initial Lanczos iteration, m steps
t = cputime;
m = 6000;
a = zeros(m,1); b = zeros(m-1,1);
for k = 1:m
  if k == 1
    r = A*v;
  else
```

```
    r = A*v - b(k-1)*v2;
  end
  a(k) = v'*r;
  r = r - a(k)*v;
  b(k) = norm(r);
  v2 = v;
  v = 1/b(k)*r;

  % estimate |A|_2 by |T|_1
  if k == 1
    anorm = abs( a(1)+b(1) );
  else
    anorm = max( anorm, b(k-1)+abs(a(k))+b(k) );
  end;
end

t1 = cputime-t;
fprintf( 'Initial Lanczos iteration completed. (t = %2.2f)\n', t1 )

% tridiagonal matrix
[ritz,S] = mrrr(a,b); % calls the LAPACK routine DSTEGR

lres = abs(b(m)*S(m,:))'; % residual estimation
cul = abs(S(1,:))';

% remove non-converged and spurious Ritz values
tol = eps*anorm;
idx = (lres < tol) & (cul > tol);
ritz = ritz(idx);
lres = lres(idx);
cul = cul(idx);
S = S(:,idx);
mm = sum(idx);

% distinguish the clusters of eigenvalues
tol = 1e-8;
i = 1; ci = {};
for k = 2:mm
  if abs(ritz(k)-ritz(i)) > tol
    ci{end+1} = (i:k-1)';
    i = k; k = k + 1;
  end
end
ci{end+1} = (i:mm)';
```

```
% reflect using Householder
e = []; c = []; S2 = zeros(m,length(ci));
for i = 1:length(ci)
  cii = ci{i};
  [y,j] = max(cul(cii));
  ji = cii(1)+j-1;

  x = S(1,cii)'; u = x;
  u(j) = u(j) + sign(x(j))*norm(x);
  s = S(:,ji)-2/(u'*u)*(S(:,cii)*u)*u(j);
  S2(:,i) = s;

  idx(cii) = false; idx(ji) = true;
  e = [e; ritz(ji)];
  c = [c; lres(ji)];
end

t2 = cputime-t-t1;
fprintf('Eigenvalue computation complete. (t = %2.2f)\n',t2)

% compute eigenvectors
v = startvec;
X = zeros(n,size(S2,2));
for k = 1:m-1
  X = X + v*S2(k,:);
  if k == 1
    r = A*v;
  else
    r = A*v - b2*v2;
  end
  a = v'*r;
  r = r - a*v;
  b = norm(r);
  v2 = v; a2 = a; b2 = b;
  v = 1/b*r;
end
X = X + v*S2(m,:);
X = X*diag(1./normc(X));

t3 = cputime-t-t1-t2;
fprintf('Eigenvector computation complete. (t = %2.2f)\n', t3)
t = cputime-t;
```

```
% output
fprintf(['%d/%d eigenvalues converged.\n' ...
     '||AX-XD|| = %e\n||X''X-I|| = %e\n' ...
     'CPU time used: %2.2f\n'], length(e), n, ...
     norm(A*X-X*diag(e))/anorm, norm(X'*X-eye(size(X,2))), t)
```

A.2 lancso.m

```
% lancso.m
% Lanczos with selective reorthogonalization

clear all

% matrix
G = numgrid('L',118);
A = delsq(G);
n = length(A);

% number of eigenvalues sought
nev = 150;

% requested accuracy of eigenvalues
tolconv = eps;

% maximum size of basis, max number of matrix vector multiplies
jmax = n;

% starting vector
load sv10k.mat
vstart = startvec;
%vstart = randn(n,1);


cpustart=cputime;
wjm1=[0 0];
wj=[0 eps eps];
wjp1=[];
wjps=[];
flagreort=0;
nconv=0;
r=vstart;
v=[];
```

```
alpha=[];
beta=[];
beta(1)=norm(r);
for j=1:jmax,
  % Basic recursion
  v(:,j)=r/beta(j);
  r = A*v(:,j);
  if j>1, r=r-v(:,j-1)*beta(j); end;
  alpha(j)=r'*v(:,j);
  r=r-v(:,j)*alpha(j);
  beta(j+1)=norm(r);

  % Estimate |A|_2 by |T|_1
  if j==1
    anorm=abs(alpha(1)+beta(2));
  else
    anorm=max(anorm,beta(j)+abs(alpha(j))+beta(j+1));
  end;
  epert=anorm*eps;

  % Update recurrence to determine selective reorth
  wjp1(1)=0;
  jv=2:j;
  wjp1(jv)=beta(jv).*wj(jv+1)+(alpha(jv-1)-alpha(j))...
       .*wj(jv)+beta(jv-1).*wj(jv-1)-beta(j)*wjm1(jv);
  wjp1(jv)=(wjp1(jv)+sign(wjp1(jv))*epert)/beta(j+1);
  wjp1(j+1)=epert/beta(j+1);
  wjp1(j+2)=eps;
  wjps(j)=max(abs(wjp1));

  % Test if it is time to reorthogonalize
  if max(abs(wjp1)) > sqrt(eps),
    flagreort=1;
    vjj1=[v(:,j) r];
    h1=v(:,1:j-1)'*vjj1;
    vjj1=vjj1-v(:,1:j-1)*h1;
    if norm(h1)>10*sqrt(eps),
      vjj1=vjj1-v(:,1:j-1)*(v(:,1:j-1)'*vjj1);
    end;
    r=vjj1(:,2)-vjj1(:,1)*(vjj1(:,1)'*vjj1(:,2));
    v(:,j)=vjj1(:,1);
    wjp1(2:j+1)=eps;
    wj(2:j)=eps;
  end;
```

```
  wjm1=wj;
  wj=wjp1;
  % Is it time to test for convergence
  if flagreort | j==jmax | beta(j+1)<epert,
    [d s] = mrrr(alpha,beta(2:end));
    %[s d]=eig(diag(alpha)+diag(beta(2:j),1)+diag(beta(2:j),-1));
    bndv=abs(beta(j+1)*s(j,:));
    convd=bndv<tolconv*anorm;
    nconv=sum(convd);
    fprintf('%4.0f %3.0f %8.2e %8.2e %15.8e %15.8e \n',j, ...
            nconv,cputime-cpustart,wjps(j),alpha(j),beta(j+1));
    flagreort=0;
  end;
  if beta(j+1)<epert | nconv>=nev , break; end;
end;
iconv=find(convd);
[lmb ilmb]=sort(d(iconv));
xv=v*s(:,iconv(ilmb));
t = cputime-cpustart;
fprintf(['%d/%d eigenvalues converged.\n' ...
    '||AV-VD|| = %e\n||V''V-I|| = %e\n' ...
    'CPU time used: %2.2f\n'], nconv, n, ...
    norm(xv'*xv-eye(size(xv,2))), ...
    norm(A*xv-xv*diag(lmb))/anorm, t)
```

A.3 mrrr.c

Note: this is the interface to be able to call the MRRR routine from LAPACK in Matlab.

```
#include "mex.h"

extern void dstegr_( char* jobz, char* range, int* n, double* d,
                     double* e, double* vl, double* vu, int* il,
                     int* iu, double* abstol, int* m, double* w,
                     double* z, int* ldz, int* isuppz, double* work,
                    int* lwork, int* iwork, int* liwork, int* info );

void mexFunction( int nlhs, mxArray *plhs[],
                  int nrhs, const mxArray *prhs[] )
{
    if( nrhs != 2 || nlhs != 2 )
```

```
        mexErrMsgTxt( "Expects two input and output arguments" );

    // matrix size
    int n1 = mxGetM( prhs[0] );
    int n2 = mxGetN( prhs[0] );
    int n = (n1 > n2 ? n1 : n2);

    char* jobz = "V";
    char* range = "A";

    // main diagonal
    double* d = mxCalloc( n, sizeof(double) );
    double* temp = mxGetPr( prhs[0] );
    int i;
    for( i = 0; i < n; ++i )
        d[i] = temp[i];

    // sub-/superdiagonal
    double* e = mxCalloc( n, sizeof(double) );
    temp = mxGetPr( prhs[1] );
    for( i = 0; i < n; ++i )
        e[i] = temp[i];

    double vl = 0, vu = 0;  // unused
    int il = 0, iu = 0;     // unused
    double abstol = 1e-13;  // unused?
    int m;  // number of eigenvalues found

    // computed eigenvalues
    plhs[0] = mxCreateDoubleMatrix( n, 1, mxREAL );
    double* w = mxGetPr( plhs[0] );

    // computed eigenvectors
    plhs[1] = mxCreateDoubleMatrix( n, n, mxREAL );
    double* z = mxGetPr( plhs[1] );

    // misc arguments
    int ldz = n;
    int* isuppz = (int*) mxCalloc( 2*n, sizeof(int) );
    double* work1 = (double*) mxCalloc( 1, sizeof(double) );
    int lwork = -1;
    int* iwork1 = (int*) mxCalloc( 1, sizeof(int) );
    int liwork = -1;
    int info;
```

```
    // query for workspace size
    dstegr_( jobz, range, &n, d, e, &vl, &vu, &il, &iu, &abstol,
             &m, w, z, &ldz, isuppz, work1, &lwork, iwork1, &liwork,
             &info );

    if( info < 0 )
        mexErrMsgTxt( "Input to dstegr had an illegal value" );

    lwork = (int) work1[0];
    double* work = (double*) mxCalloc( lwork, sizeof(double) );
    liwork = (int) iwork1[0];
    int* iwork = (int*) mxCalloc( liwork, sizeof(int) );

    // call LAPACK routine
    dstegr_( jobz, range, &n, d, e, &vl, &vu, &il, &iu, &abstol,
             &m, w, z, &ldz, isuppz, work, &lwork, iwork, &liwork,
             &info );

    if( info < 0 )
        mexErrMsgTxt( "Input to dstegr had an illegal value" );

    mxFree( d );
    mxFree( e );
    mxFree( isuppz );
    mxFree( work1 );
    mxFree( work );
    mxFree( iwork1 );
    mxFree( iwork );
}
```

TRITA-CSC-E 2007:123
ISRN-KTH/CSC/E--07/123--SE
ISSN-1653-5715

www.kth.se