112
128
THS

THESIS

MICHIGAN STATE UNIVERSITY LIBRARIES
3 1293 00877 2661

This is to certify that the

dissertation entitled

Parallel Homotopy Algorithm for Symmetric Large Sparse Eigenproblems

presented by

Liang Jiao Huang

has been accepted towards fulfillment
of the requirements for

Ph. D. degree in Mathematics

Major professor

Date 5/27/92

MSU is an Affirmative Action/Equal Opportunity Institution

O-12771

LIBRARY
Michigan State
University

PLACE IN RETURN BOX to remove this checkout from your record.
TO AVOID FINES return on or before date due.

DATE DUE	DATE DUE	DATE DUE

MSU Is An Affirmative Action/Equal Opportunity Institution
c:\circ\datedue.pm3-p.1

PARALLEL HOMOTOPY ALGORITHM FOR LARGE SPARSE SYMMETRIC EIGENPROBLEMS

By

Liang Jiao Huang

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Mathematics

1992

642-0508

ABSTRACT

PARALLEL HOMOTOPY ALGORITHM FOR LARGE SPARSE SYMMETRIC EIGENPROBLEMS

By

Liang Jiao Huang

In this work, we apply homotopy method to solve eigenproblem

$$Ax = \lambda x, \qquad \lambda \in \mathbf{R},\ x \in \mathbf{R}^n \setminus \{0\}$$

for large sparse symmetric matrix A. A one-parameter family of matrices $A(t) = tA + (1-t)D$ is introduced and eigenproblem

$$A(t)x(t) = \lambda(t)x(t)$$

is considered for $t \in [0,1]$. The problem of choosing optimal starting matrix $A(0) = D$ is discussed and partial solutions are obtained. The regularity and bifurcation problems of $\lambda(t)$ and $x(t)$ are also considered. It is found that these functions can be chosen as analytic functions of t and the bifurcations are of simple type. Then a homotopy continuation algorithm is constructed and several new techniques are developed to handle the curve following process more efficiently. Finally the algorithm is implemented in both parallel and vector machines and numerical results are obtained for many typical testing matrices. Our preliminary experiments show that homotopy continuation method is a very promising method.

To my wife Lu Ping

ACKNOWLEDGMENTS

I would like to thank Professor Tien-Yien Li, my dissertation advisor, for suggesting the problem and the directions he provided which made this work possible. I would also like to thank him for for his encouragement and support during my graduate study at Michigan State University.

I would like to thank my dissertation committee members Professor Qiang Du, Professor Dennis Dunninger, Professor Richard Hill, and Professor David Yen for their valuable suggestions and time.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Large scale scientific computing is currently a very active research field. Traditional methods which work well for small problems are often not suitable for large problems, and not suitable for modern computer architectures. For example, the very efficient and widely used method for solving eigenproblems of small matrices — the QR iteration method [25], becomes inapplicable for large sparse eigenvalue problems because, among other things, the process of Householder reduction can quickly destroy the sparse pattern. Moreover, the method is highly serial in nature, it is difficult to fully exploit the power of modern computers. The best known method which can be used to solve large scale eigenvalue problems is the Lanczos method [5]. It can take advantage of the sparseness structure of a given matrix and is good for finding a few extreme eigenvalues. However, it is not a parallel method either, and it is not efficient for finding interior eigenvalues.

In this work, we propose a method, called the homotopy continuation method, which is suitable for parallel solution of large sparse symmetric eigenvalue problems. The basic idea of the method is described in the following (see [17] and [18] for more details).

Given a real symmetric matrix A of order n, instead of solving for all the eigenvalues of A directly, we choose another real symmetric matrix D, called the starting

matrix, and consider the one-parameter family of matrices

$$A(t) = D + t(A - D) \quad \text{for } t \in [0,1]. \tag{1.1}$$

This family has the property,

$$A(0) = D, \quad A(1) = A.$$

For each $t \in [0,1]$, let the eigenvalues of $A(t)$ be $\lambda_1(t) \leq \lambda_2(t) \leq \cdots \leq \lambda_n(t)$. It is well known that eigenvalues of a matrix are continuous functions of its elements, so in this case each $\lambda_i(t)$ is a continuous function of t for $i = 1, 2, \cdots, n$. Therefore, there are n continuous curves, which we shall call *eigencurves.*

Suppose we have the eigenvalues of D — $\lambda_1(0), \lambda_2(0), \cdots, \lambda_n(0)$. From these values, the starting points of n eigencurves are known. By following these continuous eigencurves, we can reach the ending points $\lambda_1(1), \lambda_2(1), \cdots, \lambda_n(1)$ of the curves, which are the eigenvalues of the given matrix A.

The main advantages of our method are:

- It is parallel in nature: tracing of each eigencurve is independent of the others.
- The main calculation is concentrated on solving large sparse linear equations, and unlike solving for eigenvalues directly, several good packages are available for solving such linear equations efficiently [8], [9], [11].
- It can be used to find only a few of the specified eigenvalues. (In contrast, the Lanczos method tends to give extreme eigenvalues on both ends before the emerging of interior eigenvalues , and it can not tell which eigenvalue is found and what multiplicity it has.)

The dissertation proceeds as follows: Chapter 2 discusses the problem of choosing a starting matrix D. Some optimal solutions to this question are found under certain conditions. Chapter 3 addresses the regularity and bifurcation problems. Under the

assumption that A and D is symmetric, the eigensystems are analytic. Also, with A and D symmetric, the bifurcations are easy to handle — bifurcation directions can be readily computed using a simple formula. Chapter 4 describes our homotopy algorithm for following the eigencurves. Chapter 5 presents the test matrices, softwares used, and the numerical results on two typical machines — one a vector machine, and the other a parallel machine. The results show that the homotopy continuation method is very promising for large sparse symmetric eigenproblems.

Chapter 2

The Choice of Starting Matrix

2.1 Introduction

Choosing a good starting matrix D plays a very important role in our homotopy algorithm. This can be seen from the following example.

Example 2.1: Let A be a 10×10 symmetric random matrix, D_1 be a random diagonal matrix, and D_2 be a block diagonal matrix with two diagonal blocks directly from A. That is, if

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

then

$$D_2 = \begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix}$$

where A_{ij} are 5×5 matrices for $i, j = 1, 2$. We construct homotopies (1.1) for each starting matrix D_1 and D_2, and plot eigencurves in Figure 2.1 and Figure 2.2 respectively. It can be seen that, eigencurves corresponding to D_2 are much more straight and thus much easier to follow than those corresponding to D_1. Obviously, D_2 is a better starting matrix than D_1.

The following theorem helps in deciding if a matrix D is a good starting matrix.

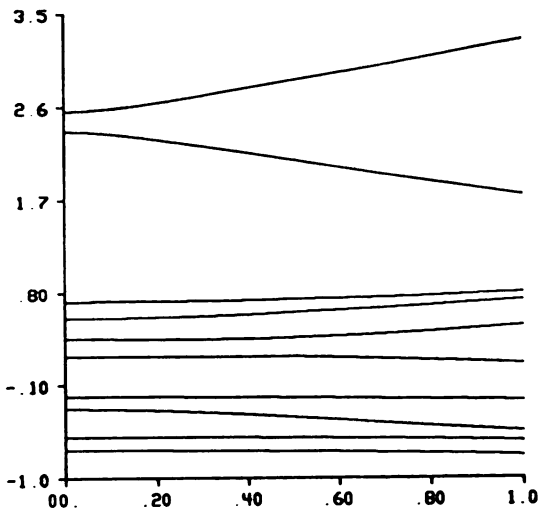


Figure 2.1: Curves corresponding to D_1

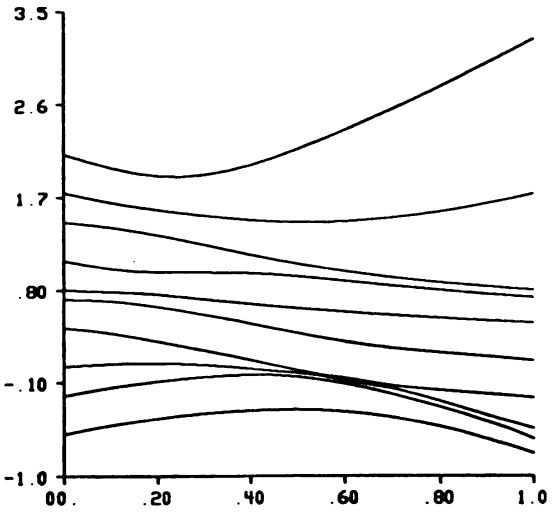


Figure 2.2: Curves corresponding to D_2

Theorem 2.1 *Let A and D be given symmetric matrices, $\lambda_1(t), \lambda_2(t), \cdots, \lambda_n(t)$ be the eigenvalues of $A(t)$ in (1.1), then $|\lambda_i'(t)| \leq \|A - D\|_2$ for each $i = 1, 2, \cdots, n$.*

Proof: By Weyl's Theorem [2],

$$|\lambda_i(t + \Delta t) - \lambda_i(t)| \leq \|A(t + \Delta t) - A(t)\|_2 = \|\Delta t(A - D)\|_2.$$

Hence

$$|\frac{\lambda_i(t + \Delta t) - \lambda_i(t)}{\Delta t}| \leq \|(A - D)\|_2.$$

Because $\lambda_i(t)$ is differentiable function of t [24], letting $\Delta t \to 0$ yields,

$$|\lambda_i'(t)| \leq \|(A - D)\|_2.$$

This completes the proof. ∎

One of our goals in constructing a good homotopy is to find a starting matrix D so that the eigencurves are more straight. The theorem above indicates that one should look for a D which, among other conditions, minimizes $\|A - D\|_2$. For the matrix A in example 2.1, it will be proved in Corollary 2.4 that

$$\|A - D_2\|_2 \leq \|A - D_1\|_2. \tag{2.1}$$

(In fact, we will show

$$\|A - D_2\|_2 = \min_{\mathcal{D}} \|A - D\|_2, \tag{2.2}$$

where $\mathcal{D}$ is the set of matrices having the same structure as D_2). This explains why eigencurves are better behaved when D_2 is used as a starting matrix.

So the problem of finding a good starting matrix becomes a matrix approximation problem. In this chapter, an important type of norms in matrix theory is introduced first, then the approximation problem arises in our homotopy method is considered.

2.2 Unitarily Invariant Norms

Some notations are needed for future reference:

$\mathcal{M}$ = { Complex matrices of order n }

$\mathcal{U}$ = { Unitary matrices of order n }

$\mathcal{D}$ = { Block diagonal matrices of order n with a given block structure }

A *unitarily invariant norm* is a matrix norm which satisfies

$$\|A\| = \|UA\| = \|AV\|$$

for every $A \in \mathcal{M}$ and U, $V \in \mathcal{U}$. A result of J. von Neumann characterizes all unitarily invariant norms by symmetric gauge functions of singular values [28]. Two important classes of unitarily invariant norms are: *Schatten p norms*

$$\|A\|_p = (\sum_{j=1}^{n} s_j(A)^p)^{1/p}, \quad p \geq 1$$

and *Ky Fan k norms*

$$\|A\|_k = \sum_{j=1}^{k} s_j(A), \quad k = 1, 2, \cdots, n$$

where $\{s_j(A), j = 1, 2, \cdots, n\}$ are the singular values of A in descending order. It is easy to see that Schatten *2* norm is the Frobenius norm $\|\cdot\|_F$ and Ky Fan *1* norm is the spectral norm $\|\cdot\|_2$.

2.3 Block Diagonal Approximants

A good choice for the starting matrix is based on three considerations. First, the eigensystems of D should be easy to find; secondly, D should be as close to A as possible, so that the eigencurves are better behaved [17]; finally, D itself should be easy to obtain. However, these considerations often conflict with each other in the sense that if D is too close to A, then eigensystems of D could be as difficult to find as those of A's; on the other hand, if eigensystems of D are easy to find, then D is usually not very close to A.

One natural candidate for the starting matrix D is the block diagonal matrix

$$D = \begin{bmatrix} D_{11} & & & \\ & D_{22} & & \\ & & \ddots & \\ & & & D_{kk} \end{bmatrix} \tag{2.3}$$

where D_{ii} are smaller square matrices for $i = 1, 2, \cdots, k$. For such a matrix D, its eigensystems can be found from those of $D_{ii}'s$, and each $D_{ii}'s$ eigensystems can be easily found since it is a small matrix. Furthermore, by using multi-processors, eigensystems of D can be computed in parallel. Therefore, matrices of this form satisfy our first consideration.

Let D be of the form in (2.3). The next question is, how to find D of this form which is closest to a given matrix A. In other words, let $\mathcal{D}$ be the set of matrices of the form in (2.3), we want to find $D_0 \in \mathcal{D}$ such that

$$\|A - D_0\| = \min_{D} \|A - D\|. \tag{2.4}$$

To the best of our knowledge, this type of matrix approximation has never been investigated before. To measure the closeness of two matrices, unitarily invariant norms are used as the underlying norms. This is because we are considering eigenvalue problems and these norms are simple functions of eigenvalues (by von Neumann's characterization [28]).

For the Frobenius norm, the solution of (2.4) is obvious. We can partition A into blocks according to the structure in (2.3) and choose D_0 to be the block diagonal matrix whose diagonal blocks are identical to the corresponding diagonal blocks of A. That is, if

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1k} \\ A_{21} & A_{22} & \cdots & A_{2k} \\ \cdots & \cdots & \cdots & \cdots \\ A_{k1} & A_{k2} & \cdots & A_{kk} \end{bmatrix} \tag{2.5}$$

then

$$D_0 = \begin{bmatrix} A_{11} & & & \\ & A_{22} & & \\ & & \ddots & \\ & & & A_{kk} \end{bmatrix}. \tag{2.6}$$

For general unitarily invariant norms, however, the solution is not so obvious. One would hope that D_0 above is the choice since it meets all our three considerations. It turns out that this is true for an important class of matrices.

In this section, we prove some positive results first, and then a counterexample is constructed to show that the result can not be extended to cover all matrices.

Theorem 2.2 *Let A and D_0 be matrices in (2.5) and (2.6). If there exists a unitary matrix $E \in \mathcal{U}$ which commutes with every $D \in \mathcal{D}$ such that*

$$A - D_0 = -E(A - D_0)E^H, \tag{2.7}$$

then D_0 is the best block diagonal approximant of A for every unitarily invariant norm, i.e., $\|A - D_0\| = \min_{D \in \mathcal{D}} \|A - D\|$.

For the proof of Theorem 2.2, the following lemmas are needed. They were first proved by Fan in [10].

Lemma 2.1 *Let $A,B \in \mathcal{M}$,then $\|A\| \leq \|B\|$ holds for all unitarily invariant norms $\|\cdot\|$ if and only if*

$$\|A\|_k \leq \|B\|_k$$

holds for all Ky Fan norms $\|\cdot\|_k$, $k = 1, 2, \cdots, n$.

Lemma 2.2 *Let* $A \in \mathcal{M}$, $X_k = \{x_1, x_2, \cdots, x_k\}$ *and* $Y_k = \{y_1, y_2, \cdots, y_k\}$ *be sets of orthonormal vectors in* $\mathbf{C}^n$, *then*

$$\|A\|_k = \sum_{j=1}^{k} s_j(A) = \max_{X_k, Y_k} \sum_{j=1}^{k} Re\langle x_j, Ay_j\rangle.$$

Proof of Theorem 2.2: Let $A_0 = A - D_0 = U\Sigma V^H$ be a singular value decomposition of A_0, with

$$U = (u_1, u_2, \cdots, u_n),\ \Sigma = diag(s_1(A_0), \cdots, s_n(A_0)),\ V = (v_1, v_2, \cdots, v_n).$$

Because $A_0 = -EA_0E^H$,

$$A_0 = -E(U\Sigma V^H)E^H = (-EU)\Sigma(EV)^H$$

where $-EU$ and EV are unitary matrices. This gives another singular value decomposition of A_0. Thus,

$$\langle u_j, A_0 v_j\rangle = \langle -Eu_j, A_0 Ev_j\rangle = s_j(A_0).$$

By Lemma 2.2, for any $D \in \mathcal{D}$ and $k \le n$,

$$\begin{aligned} \|A - D\|_k &\ge \sum_{j=1}^{k} Re\langle u_j, (A - D)v_j\rangle \\ &= \sum_{j=1}^{k} \langle u_j, A_0 v_j\rangle + \sum_{j=1}^{k} Re\langle u_j, (D_0 - D)v_j\rangle \\ &= \|A - D_0\|_k + \sum_{j=1}^{k} Re\langle u_j, (D_0 - D)v_j\rangle. \end{aligned} \tag{2.8}$$

On the other hand, since $-EU$ and EV are unitary matrices,

$$\begin{aligned} \|A - D\|_k &\ge \sum_{j=1}^{k} Re\langle -Eu_j, (A - D)Ev_j\rangle \\ &= \sum_{j=1}^{k} \langle -Eu_j, A_0 Ev_j\rangle + \sum_{j=1}^{k} Re\langle -Eu_j, (D_0 - D)Ev_j\rangle \\ &= \sum_{j=1}^{k} s_j(A_0) - \sum_{j=1}^{k} Re\langle Eu_j, E(D_0 - D)v_j\rangle \\ &= \|A - D_0\|_k - \sum_{j=1}^{k} Re\langle u_j, (D_0 - D)v_j\rangle. \end{aligned} \tag{2.9}$$

Combining inequalities (2.8) and (2.9),

$$\begin{aligned} \|A - D\|_k &\geq \|A - D_0\|_k + |\sum_{j=1}^{k} Re\langle u_j, (D_0 - D)v_j\rangle| \\ &\geq \|A - D_0\|_k. \end{aligned}$$

This completes the proof because of Lemma 2.1. ∎

Remark 2.1: In fact, it can be seen from the proof that the following more general result has been established: *Let A and D_0 be any two given matrices, if there exists a unitary matrix E that commutes with D_0 such that*

$$A - D_0 = -E(A - D_0)E^H,$$

then D_0 is closest to A among all matrices that commute with E, that is,

$$\|A - D_0\| = \min_{DE=ED} \|A - D\|.$$

An important class of matrices satisfying (2.7) is the class of block tridiagonal matrices.

Corollary 2.3 *If A is a block tridiagonal matrix, then the best block diagonal approximant of A is D_0 for every unitarily invariant norms.*

Proof: Condition (2.7) is satisfied for $E = diag(I_1, -I_2, \cdots, (-1)^{k-1}I_k)$, with each I_j an identity matrix of appropriate order. ∎

Block tridiagonal (in particular, tridiagonal) matrices arise in many applications. In fact, in matrix eigenvalue computations, a given matrix is usually transformed into a compact form — tridiagonal or block tridiagonal form by Householder or Lanczos transformation, then QR iteration, bisection, or homotopy method is applied to obtain the solutions.

Corollary 2.4 *Let A be any square matrix, if we partition A into*

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

then

$$D_0 = \begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix}$$

is the best approximant of A among all block diagonal matrices of the form for every unitarily invariant norm.

Proof: This is a special case of Corollary (2.3) with $k = 2$. ∎

Inequality (2.1) and equation (2.2) of Example 2.1 at the beginning of this chapter follow from this corollary.

It is well known that the time required to find eigensystems of a matrix of order n is proportional to n^3. If A is divided into two blocks of approximately equal sizes as in Corollary 2.3, then D is closest to A and eigensystems of D can be solved by parallel computer using about $1/8$ execution time for solving eigensystems of A. Furthermore, if block sizes of D are still too large to work with, they can be divided into smaller blocks and the "divide and conquer" strategy can be employed.

For general matrices, the conclusion of Theorem 2.2 may not be true. This can be seen in the following example.

Example 2.2: Let

$$A = \begin{bmatrix} 0 & 2 & 2 \\ 2 & 0 & 2 \\ 2 & 2 & 0 \end{bmatrix}.$$

If we choose block size to be *1*, then

$$D_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

The eigenvalues of A are $\lambda_1 = 4$, $\lambda_2 = -2$ and $\lambda_3 = -2$, therefore for spectral norm $\|\cdot\|_2$,

$$\|A - D_0\|_2 = \max_{1 \le i \le n} |\lambda_i| = \lambda_1 = 4.$$

However,

$$\|A - I\|_2 = \max_{1 \le i \le n} |\lambda_i - 1| = 3,$$

i.e.,

$$\|A - I\|_2 < \|A - D_0\|_2.$$

Thus D_0 is not the best approximant. In fact, it can be shown that I is the best approximant in this case: first of all, $A - I$ has orthonormal eigenvectors

$$x_1 = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad x_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}, \quad x_3 = \frac{1}{\sqrt{6}} \begin{bmatrix} -2 \\ 1 \\ 1 \end{bmatrix},$$

with corresponding eigenvalues $\lambda_1 = 3$, $\lambda_2 = -3$ and $\lambda_3 = -3$. If there exists a matrix D such that $\|A - D\|_2 < \|A - I\|_2$, then by Courant-Fischer min-max theorem, we must have

$$\begin{aligned} |\langle x_j, (A - D)x_j \rangle| &= |\langle x_j, (A - I)x_j \rangle + \langle x_j, (I - D)x_j \rangle| \\ &= |\lambda_j + \langle x_j, (I - D)x_j \rangle| \le \|A - D\|_2 \\ &< \|A - I\|_2 = |\lambda_j|, \qquad j = 1, 2, 3. \end{aligned}$$

Let $I - D = \mathrm{diag}(d_1, d_2, d_3)$, then the above inequalities yield

$$\langle x_1, (I - D)x_1 \rangle = (d_1 + d_2 + d_3)/3 < 0, \tag{2.10}$$

$$\langle x_2, (I - D)x_2 \rangle = (d_2 + d_3)/2 > 0, \tag{2.11}$$

$$\langle x_3, (I - D)x_3 \rangle = (4d_1 + d_2 + d_3)/6 > 0. \tag{2.12}$$

It follows from (2.10) and (2.11), $d_1 < 0$. However, (2.10) and (2.12) imply $3d_1 > 0$, Thus such a matrix D does not exist, and the assertion is achieved.

Another example of larger order can be found in [22] where the conclusion was justified by numerical results.

Although the question of choosing the starting matrix is not completely answered, the matrix D in (2.6) is the best choice in several important cases (such as block tridiagonal A) and is the best in all cases for Frobenius norm.

Remark 2.2: Our results provide a general guide line for choosing a starting matrix. In a practical problem, usually more special properties about the given matrix A is known, this allows other choices for D as long as it satisfies our three general considerations. For instance, when a physical problem is investigated, in order to determine the final parameters for the problem, it is necessary to do a series of experiments. In each experiment, the parameters are adjusted only by a small amount. If the question is related to an eigenvalue problem, then there is a family of matrices with the same (or similar) structures, each matrix is a small perturbation of the previous one (except the first). In such a case, the natural choice for the starting matrix is one of the matrices in the family whose eigensystems have been found, and we expect eigensystems of the new matrix to be a "small" perturbation of the starting matrix.

Chapter 3

Regularity and Bifurcation

3.1 Introduction

It is well known that eigenvalues are continuous functions of the entries of the matrix. However, they are generally not differentiable. For example, consider

$$A(t) = \begin{bmatrix} 1 & t \\ 1 & 1 \end{bmatrix}, \tag{3.1}$$

its eigenvalues are $\lambda_1(t) = 1 - \sqrt{t}$ and $\lambda_2(t) = 1 + \sqrt{t}$. They are not differentiable at $t = 0$. This happens because, at $t = 0$, $A(0)$ has multiple eigenvalues $\lambda_1(0) = \lambda_2(0) = 1$, that is, eigencurves have a bifurcation point.

The behavior of eigenvectors at a bifurcation is even worse. They may not even be continuous. The following example is attributed to W. Givens in [21], §3.1:

$$A(t) = \begin{bmatrix} 1 + t\cos(2/t) & -t\sin(2/t) \\ -t\sin(2/t) & 1 - t\cos(2/t) \end{bmatrix} \tag{3.2}$$

has eigenvalues $\lambda_1(t) = 1 - t$ and $\lambda_2(t) = 1 + t$. They are analytic functions of t. However, the corresponding eigenvectors are

$$x_1(t) = \begin{bmatrix} \cos(1/t) \\ -\sin(1/t) \end{bmatrix}, \quad x_2(t) = \begin{bmatrix} \sin(1/t) \\ \cos(1/t) \end{bmatrix},$$

which have no limits as $t \to 0$! Again the problem arises because $A(t)$ has a double eigenvalue at $t = 0$.

For our homotopy algorithm, handling the bifurcation efficiently becomes a very important problem. The behavior of eigencurves around such points can be quite complicated. Nevertheless, for symmetric matrices, bifurcation is not as difficult. In fact, a result proved in 1940s ([24], Chapter I) guarantees that $\lambda_i(t)$ and $x_i(t)$ can be chosen in such a way that they are all analytic functions of t. In such a case, the eigencurves through a bifurcation point are well behaved. A more general result of this type is considered one of the major breakthroughs in the past fifty years in eigenvalue perturbation theory [14].

3.2 Regularity

We summarize those results that related to our problem in the following theorem.

Theorem 3.1 *Suppose A and D are both real symmetric matrices. Let*

$$A(t) = tA + (1-t)D = D + t(A - D),$$

then the eigensystems

$$(\lambda_1(t), x_1(t)), (\lambda_2(t), x_2(t)), \cdots, (\lambda_n(t), x_n(t))$$

of $A(t)$ can be chosen in such a way that all functions involved are analytic functions for real t. Furthermore, there are only finitely many $t \in [0,1]$ such that $A(t)$ has multiple eigenvalues.

Proof: see [14]. ∎

Remark 3.1: In our algorithm, preserving the order $\lambda_1(t) \leq \lambda_2(t) \leq \cdots \leq \lambda_n(t)$ is very important. With such an order, we can implement parallel processing and compute only partial eigensystems when necessary. However, when this is imposed,

the conclusion of the above theorem is no longer valid. For example, let

$$A(t) = \begin{bmatrix} 1 & 1-2t \\ 1-2t & 1 \end{bmatrix} \tag{3.3}$$

then $\lambda_1(t) = 2t$, $\lambda_2(t) = 2 - 2t$ and they are analytic. With the ordering, however, $\lambda_1(t) = 1-|1-2t|$, $\lambda_2(t) = 1+|1-2t|$. They are not analytic at $t = 1/2$. Nevertheless, after the ordering, these $\lambda_i(t)$'s are still piecewise analytic and one-sided derivatives $\lambda_i^{(n)}(t+)$ always exist. This is sufficient for our numerical implementation, since we trace eigencurves forward, only the right hand derivatives are needed.

3.3 Bifurcation Directions

Suppose

$$(\lambda_1(t), x_1(t)), (\lambda_2(t), x_2(t)), \cdots, (\lambda_n(t), x_n(t))$$

are the eigensystems of $A(t)$ as in Theorem 3.1, then

$$A(t)x_j(t) = \lambda_j(t)x_j(t). \tag{3.4}$$

Taking the right-hand derivatives with respect to t on both sides of this equation yields

$$A'(t)x_j(t) + A(t)x_j'(t+) = \lambda_j'(t+)x_j(t) + \lambda_j(t)x_j'(t+). \tag{3.5}$$

Multiplying the above equation on the left by $x_j^T(t)$, we have

$$\lambda_j'(t+) = x_j^T(t)A'(t)x_j(t).$$

But $A'(t) = A - D$, so

$$\lambda_j'(t+) = x_j^T(t)(A - D)x_j(t). \tag{3.6}$$

With this last formula, the prediction-correction scheme can be applied to numerically compute the eigensystems. However, at a bifurcation point, eigenvectors are not uniquely defined. For an eigenvalue of multiplicity k, any k orthonormal vectors

from the k dimensional invariant subspace form an eigenbasis for that subspace. By Theorem 3.1, there is at least one way in choosing an appropriate set of eigenvectors for each t such that $x_j(t)$ becomes analytic. This choice is not known beforehand. And (3.6) can only be applied for this set of $x_j(t)$'s. An alternative way of computing these bifurcation directions is given as follows.

Theorem 3.2 *Suppose $\lambda_i(t) = \lambda_{i+1}(t) = \cdots = \lambda_{i+k-1}(t)$ are k multiple eigenvalues of $A(t)$. Let $y_1(t), y_2(t), \cdots, y_k(t)$ be any k orthonormal eigenvectors corresponding to these eigenvalues. Then $\lambda_i'(t+), \lambda_{i+1}'(t+), \cdots, \lambda_{i+k-1}'(t+)$ equal the k eigenvalues of $Y^T(A-D)Y$, where Y is the matrix consisting of $y_i(t), y_{i+1}(t), \cdots, y_{i+k-1}(t)$ as its columns.*

Proof: The proof of a more general result can be found in [14]. By using equation (3.5) a very simple proof can be obtained here.

Let $x_i(t), x_{i+1}(t), \cdots, x_{i+k-1}(t)$ be the analytic eigenvectors corresponding to $\lambda_i(t) = \lambda_{i+1}(t) = \cdots = \lambda_{i+k-1}(t)$ as in Theorem 3.1. Multiply (3.5) by $x_l^T(t)$ from the left,

$$x_l^T(t)(A-D)x_j(t) + \lambda_l(t)x_l^T(t)x_j'(t+) = \lambda_j'(t+)x_l^T(t)x_j(t) + \lambda_j(t)x_l^T(t)x_j'(t+).$$

Since $\lambda_l(t) = \lambda_j(t)$ and $x_l^T(t)x_j(t) = \delta_{lj}$ for $i \le l,\ j \le i+k-1$, the above equation yields

$$x_l^T(t)(A-D)x_j(t) = \delta_{lj}\lambda_j'(t+), \quad \text{for } i \le l,\ j \le i+k-1. \tag{3.7}$$

Let X be the matrix consisting of $x_i(t), x_{i+1}(t), \cdots, x_{i+k-1}(t)$ as its columns. By (3.7),

$$\Lambda'(t+) \equiv X^T(A-D)X = \begin{bmatrix} \lambda_i'(t+) & & & \\ & \lambda_{i+1}'(t+) & & \\ & & \ddots & \\ & & & \lambda_{i+k-1}'(t+) \end{bmatrix} \tag{3.8}$$

Because both $\{y_i(t), y_{i+1}(t), \cdots, y_{i+k-1}(t)\}$ and $\{x_i(t), x_{i+1}(t), \cdots, x_{i+k-1}(t)\}$ form orthonormal bases for the invariant subspace, there exists an orthogonal matrix Q such that $Y = XQ$. Hence

$$Y^T(A-D)Y = Q^T[X^T(A-D)X]Q.$$

It follows that $Y^T(A-D)Y$ and $X^T(A-D)X$ are similar matrices and have the same eigenvalues. By (3.8), they are $\lambda_i'(t+), \lambda_{i+1}'(t+), \cdots, \lambda_{i+k-1}'(t+)$. ∎

3.4 Continuity of Eigenvectors

The Rayleigh quotient iteration (RQI) is one of the most important methods in our algorithm (see Chapter 4). The convergence of RQI depends not only on a good choice of eigenvalue prediction, but also on a good eigenvector prediction. Usually, the eigenvector at step t is used as the starting vector for RQI at step $t + \Delta t$. In earlier discussion, homotopy algorithm computes eigenvectors at $t = 0$ as follows: find the eigenvectors for each diagonal block of $A(0) = D$ then extend them to full length vectors with appropriate zero entries. The following example shows that this can be an inefficient choice.

Let

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad D = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{3.9}$$

and $A(t) = (1-t)D + tA$, then

$$A(t) = \begin{bmatrix} 1 & t \\ t & 1 \end{bmatrix}. \tag{3.10}$$

According to the earlier algorithms, eigenvectors of $A(0) = D$ should be (by considering D as a block diagonal matrix with two 1×1 diagonal blocks),

$$x_1(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad x_2(0) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

However, for any $t > 0$, $A(t)$ has eigenvectors

$$x_1(t) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad x_2(t) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

These eigenvectors are not even continuous at $t = 0$. In fact there is a 45° turn in both vectors. Using $x_1(0)$ and $x_2(0)$ as starting vectors for RQI in the computation of $x_1(t)$ and $x_2(t)$ is obviously a bad choice. Again, the problem occurs because $A(t)$ has multiple eigenvalues at $t = 0$, and any two orthonormal vectors can serve as base vectors for the corresponding invariant subspace.

An important question is then, how can one find a set of eigenvectors at step t that can be turned continuously into a set of eigenvectors at step $t + \Delta t$? Using the notations in Theorem 3.2, we have

Theorem 3.3 *Suppose $Y^T(A - D)Y$ has eigendecomposition $Q\Lambda' Q^T$. If*

$$\lambda_i'(t+), \lambda_{i+1}'(t+), \cdots, \lambda_{i+k-1}'(t+)$$

are distinct, then the columns of YQ are, up to a sign, the analytic eigenvectors described in Theorem 3.1.

Proof: Let X be the matrix with analytic eigenvectors

$$x_i(t), x_{i+1}(t), \cdots, x_{i+k-1}(t)$$

as its columns. Since both X and Y consist of orthonormal eigenvectors from the same invariant subspace, there is an orthogonal matrix $\tilde{Q}$ such that $X = Y\tilde{Q}$. Substituting into (3.8), yields

$$\tilde{Q}^T(Y^T(A - D)Y)\tilde{Q} = \Lambda',$$

or

$$Y^T(A - D)Y = \tilde{Q}\Lambda'\tilde{Q}^T.$$

On the other hand,

$$Y^T(A - D)Y = Q\Lambda' Q^T,$$

so,

$$Q\Lambda' Q^T = \tilde{Q}\Lambda'\tilde{Q}^T.$$

Because diagonal matrix Λ' has distinct values $\lambda_i'(t+), \lambda_{i+1}'(t+), \cdots, \lambda_{i+k-1}'(t+)$ along its diagonal, the columns of Q and $\tilde{Q}$ can differ at most by a sign. Hence the columns of YQ and $X = Y\tilde{Q}$ can differ at most by a sign. ∎

In conclusion, when bifurcation occurs at t, $Y^T(A - D)Y$ is formed by using the computed eigenvector matrix Y, then the eigendecomposition $Q\Lambda'Q^T$ is computed for this $k \times k$ matrix $Y^T(A - D)Y$. From this decomposition, the bifurcation directions can be obtained. If these directions are mutually distinct, then YQ is formed and its columns are taken as a set of improved eigenvectors at this step, which can be used to speed up the iterations in later steps.

In the example considered at the beginning of this section, a simple computation generates the improved eigenvectors at $t = 0$

$$x_1(0) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad x_2(0) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

They turn out to be the exact eigenvectors at $t = 1$.

3.5 Close Eigenvalues

Results in previous sections are derived under the assumption that $A(t)$ has multiple eigenvalues at some t. In real computations, however, it is hardly distinguishable whether a matrix has true coincidental eigenvalues or pathologically close ones. For example, the famous Wilkinson matrix

$$W_{21}^+ = \begin{bmatrix} 10 & 1 & & & & & & \\ 1 & 9 & \ddots & & & & & \\ & \ddots & \ddots & 1 & & & & \\ & & 1 & 0 & \ddots & & & \\ & & & \ddots & \ddots & 1 & & \\ & & & & 1 & 9 & 1 & \\ & & & & & 1 & 10 & \end{bmatrix}$$

has distinct eigenvalues since it is an irreducible symmetric tridiagonal matrix, but its first two eigenvalues are so close that they agree to 15 significant digits ([29], p.308-309). That is, these eigenvalues are practically indistinguishable. Moreover, even if the eigenvalues are distinct, formula (3.6) may not be appropriate to apply when eigenvalues are close, since in this situation the corresponding eigenvectors can be very sensitive. The computed eigenvector $y_j(t)$ may be significantly different from the true one $x_j(t)$, making $y_j^T(t)(A-D)y_j(t)$ inaccurate. In such a case, we will treat them together as a cluster.

In the rest of this section, we demonstrate that the result of Theorem 3.2 can also be used in the case of close eigenvalues.

Theorem 3.4 *Let $\epsilon > 0$, suppose $A(t)$ has eigenvalues $\lambda_i(t), \lambda_{i+1}(t), \cdots, \lambda_{i+k-1}(t)$ such that $|\lambda_i(t) - \lambda_j(t)| \leq \epsilon$ for $j = i+1, \cdots, i+k-1$. Then there exists a matrix E such that $B(t) = A(t) + E$ has $\lambda_i(t)$ as its k-multiple eigenvalue and $\|E\|_2 \leq \epsilon$.*

Proof: Suppose $A(t)$ has eigendecomposition $Q\Lambda Q^T$, where Q is an orthogonal matrix and $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \cdots, \lambda_n)$. Let $\Delta = \mathrm{diag}(\delta_1, \delta_2, \cdots, \delta_n)$ be the diagonal matrix such that $\delta_1 = \cdots = \delta_i = 0$, $\delta_{i+1} = \lambda_i - \lambda_{i+1}, \cdots, \delta_{i+k-1} = \lambda_i - \lambda_{i+k-1}$, $\delta_{i+k} = \cdots = \delta_n = 0$. Set $E = Q\Delta Q^T$ and $B(t) = A(t) + E$. It is easy to see that $B(t)$ has k-multiple eigenvalue λ_i and $\|E\|_2 = \|Q\Delta Q^T\|_2 = \|\Delta\|_2 \leq \epsilon$. ∎

Suppose $B(t)$ has eigensystems $(\mu_1(t), z_1(t)), (\mu_2(t), z_2(t)), \cdots, (\mu_n(t), z_n(t))$. When $A(t)$ has close eigenvalues, ϵ is small, hence $B(t)$ is a small perturbation of $A(t)$. Although eigenvectors corresponding to close eigenvalues are very sensitive, a collection of sensitive eigenvectors can define an insensitive invariant subspace provided the corresponding cluster of eigenvalues is isolated ([12], p.199-208). So

$$\mathcal{X} = span\{x_i(t), x_{i+1}(t), \cdots, x_{i+k-1}(t)\}$$

and

$$\mathcal{Z} = span\{z_i(t), z_{i+1}(t), \cdots, z_{i+k-1}(t)\}$$

are close. Let $X = [x_i(t)\ x_{i+1}(t) \cdots x_{i+k-1}(t)]$, $Z = [z_i(t)\ z_{i+1}(t) \cdots z_{i+k-1}(t)]$, then the eigenvalues of $X^T(A-D)X$ and $Z^T(A-D)Z$ are close since $X^T(A-D)X$ and $Z^T(A-D)Z$ are the projections of $A-D$ onto $\mathcal{X}$ and $\mathcal{Z}$, respectively.

Now apply Theorem 3.2 to matrix $B(t)$,

$$\sigma(Z^T(A-D)Z) = \{\mu_j'(t) | j = i, i+1, \cdots, i+k-1\}.$$

And the first order prediction gives

$$\mu_j(t+\Delta t) \approx \mu_j(t) + \mu_j'(t)\Delta t = \lambda_i(t) + \mu_j'(t)\Delta t.$$

By Weyl's Theorem [2],

$$|\lambda_j(t+\Delta t) - \mu_j(t+\Delta t)| \leq \|A(t+\Delta t) - B(t+\Delta t)\| = \|E\| \leq \epsilon,$$

hence

$$\lambda_j(t+\Delta t) \approx \mu_j(t+\Delta t) \approx \lambda_i(t) + \mu_j'(t)\Delta t.$$

It follows that we may replace $\mu_j'(t)$ by an eigenvalue of $X^T(A-D)X$. Hence close eigenvalues and multiple eigenvalues can be handled in a uniform way.

Chapter 4

The Algorithm

The tracing of eigencurves consists of the following main steps:

4.1 Choice of the Starting Matrix

To start the algorithm, we need to choose a starting matrix D first. From the results developed in Chapter 2, block diagonal matrix with blocks directly from A will be our choice unless more information about the problem is revealed. At this stage, the sizes of the blocks remain undecided. It is clear that D will be closer to A if its diagonal blocks are larger, and the eigencurves will be better behaved. However, because of the requirement that the eigensystems of D should be much easier to find than those of A, the sizes of the blocks of D should be kept under certain limit. In our algorithm, QR iteration is used to solve for eigensystems of D. This requires that each block size be no more than a few hundred. Another important factor in choosing D is the computer architecture. When more processors are available, one can choose more (hence smaller) blocks; otherwise, fewer (hence larger) blocks should be used. In the test examples we run in BBN Butterfly Machine, we choose the block size to be around one hundred. This seems to be optimal for most of the cases comparing with other choices. In general, no formula is set to automatically generate the block

sizes of D.

4.2 Location of the Starting Points

When the starting matrix D is chosen, we need to find values $\lambda_i(0)$, $x_i(0)$ for $i = 1, 2, \cdots, n$, that is, the eigenvalues and eigenvectors of D. Since D is a block diagonal matrix, its eigenvalues consist of eigenvalues from all diagonal blocks, its eigenvectors can be obtained from eigenvectors of diagonal blocks by extending them to full dimension vectors with appropriate zero entries. In a parallel architecture, each block is assigned a processor to perform QR iterations on the block, and the results are collected to form the eigensystems of D. The QR iteration method is used since it is one of the best methods for matrices of relatively small size. Finally the eigenvalues of D are arranged in descending order to prepare for parallel processing in later steps.

It is clear that this step has high parallelization level and requires relatively small amount of execution time.

4.3 Prediction

From the previous step, eigenvalues $\lambda_j(0)$ and eigenvectors $x_j(0)$ are available for $j = 1, 2, \cdots, n$. Assuming in general that $\lambda_j(t), x_j(t)$ at t have been obtained, we shall apply the prediction – correction scheme to compute the eigenvalues λ_j and eigenvectors x_j at $t + h$. The choice of the step size h will be discussed in a later section. Here, we first consider the prediction procedure. In this step we not only predict the eigenvalue $\lambda_j(t + h)$, but also predict whether or not it is an isolated eigenvalue of $A(t + h)$. For isolated eigenvalues and for clusters, different correction methods are used.

Case 1: If the j-th eigenvalue $\lambda_j(t)$ of $A(t)$ is simple and well separated from the other eigenvalues, then

$$\lambda_j'(t) = x_j^T(t)(A - D)x_j(t).$$

Let the j-th predicted eigenvalue of $A(t+h)$ be

$$\begin{aligned} \lambda_j(t+h) &\approx \lambda_j(t) + \lambda_j'(t)h \\ &= \lambda_j(t) + x_j^T(t)(A-D)x_j(t)h. \end{aligned}$$

All quantities on the right hand side of the above equation are available from step t.

For the eigenvector, we simply use $x_j(t)$ as the prediction for $x_j(t+h)$. There are several reasons for this choice:

- Although in theory one can use

$$x_j(t+h) \approx x_j(t) + x_j'(t)h, \tag{4.1}$$

  but the high cost in computating $x_j'(t)$ makes it an impractical approach.

- Eigenvectors are more sensitive to perturbation than eigenvalues ([29], p.331-335). It is difficult to compute the derivative $x_j'(t)$ to a high accuracy. Hence (4.1) usually does not yield significant improvement of $x_j(t+h)$.

- In our correction step, the eigenvector prediction is not as important as the eigenvalue prediction.

Case 2: $\lambda_j(t)$ is in a cluster of eigenvalues of $A(t)$, then we proceed as follows.

1) Find k — the number of eigenvalues in the cluster;

2) Form $X = [x_i,\ x_{i+1}, \cdots, x_{i+k-1}]$ where the corresponding $\lambda_i, \cdots, \lambda_{i+k-1}$ form a cluster;

3) Form $X^T(A-D)X$;

4) Calculate eigendecomposition $X^T(A-D)X = Q\Omega Q^T$;

5) Set $X = XQ$.

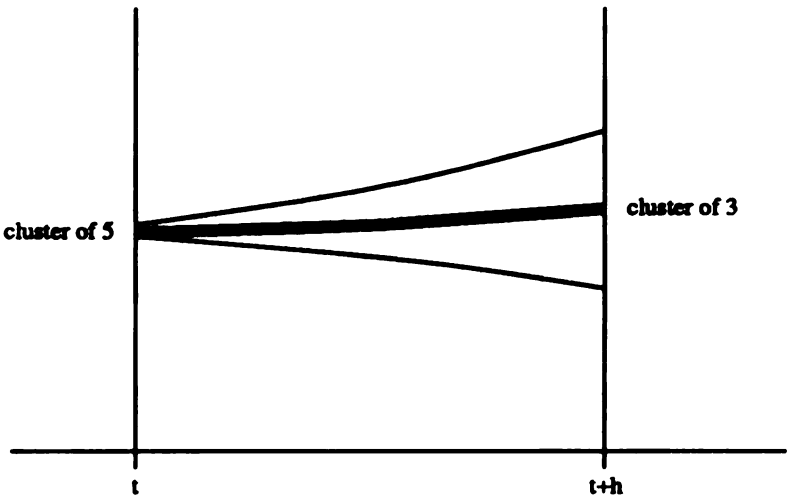


Figure 4.1: The change of clusters from t to $t + h$

Now suppose Ω=diag(ω_i $\omega_{i+1} \cdots \omega_{i+k-1}$). By Theorem 3.2, ω_i, ω_{i+1}, $\cdots$, ω_{i+k-1} are the approximations of the derivatives of $\lambda_j(t)$. Thus the predictions

$$\lambda_j(t+h) \approx \lambda_j(t) + \omega_j h \tag{4.2}$$

$$x_j(t+h) \approx x_j(t). \tag{4.3}$$

can be used. If some ω_j $(i \leq j \leq i+k-1)$ is an isolated value inside ω_i, ω_{i+1}, $\cdots$, ω_{i+k-1}, then $\lambda_j(t+h)$ is likely to be out of the cluster at $t+h$ (Figure 4.1). So it should be labeled as an isolated value and treated as in Case 1. If, on the other hand, some ω_j's are among a cluster of ω_i, ω_{i+1}, $\cdots$, ω_{i+k-1}, the corresponding λ_j's are likely to form a new cluster at $t+h$ (Figure 4.1), hence they are collected in a group and handled together at the correction step. The predictions for them are the same as in (4.2) and (4.3).

4.4 Correction

From the last step, in addition to the predictions for $\lambda_j(t+h)$ and $x_j(t+h)$, the information about the isolation of $\lambda_j(t+h)$ is also available. If it should be treated as

an eigenvalue in a cluster, the number of eigenvalues in the cluster is also known, then Subspace Iteration with Rayleigh-Ritz Procedure (SIRR) is used for the correction:

Suppose $\{\lambda_i, \lambda_{i+1}, \cdots, \lambda_{i+k-1}\}$ is a cluster and the corresponding approximate eigenvectors are $\{x_i, x_{i+1}, \cdots, x_{i+k-1}\}$. Let

$$\lambda = \frac{1}{k}(\lambda_i + \lambda_{i+1} + \cdots + \lambda_{i+k-1}), \quad X = [x_i \; x_{i+1} \cdots x_{i+k-1}].$$

Solve $(A - \lambda I)Y_1 = X$ for Y_1, then repeat the following for $\nu = 1, 2, \cdots$:

1) Decompose $Y_\nu = Q_\nu R_\nu$;

2) Solve $(A - \lambda I)Y_\nu = Q_\nu$;

3) Form $H_\nu = Q_\nu^T(A - D)^{-1}Q_\nu = Q_\nu^T Y_\nu$;

4) Decompose $H_\nu = G_\nu \Theta_\nu G_\nu^T$, where G is orthogonal matrix, Θ_ν=diag$(\theta_i, \theta_{i+1}, \cdots, \theta_{i+k-1})$;

5) Form $Y_{\nu+1} = Q_\nu G_\nu$

6) Test for the convergence of $\theta_i, \theta_{i+1}, \cdots, \theta_{i+k-1}$. Goto 1) until all $\theta_i, \theta_{i+1}, \cdots, \theta_{i+k-1}$ converge.

If $\lambda_j(t + h)$ can be treated as an isolated eigenvalue, Rayleigh Quotient Iteration (RQI) is the choice. However, in our algorithm, RQI is not applied directly at the very first step because of the following two observations:

a) Although $\lambda_j(t)$ is isolated, $\lambda_j(t + h)$ may not be so (Figure 4.2), and our prediction step can not detect such a situation. When this happens, if RQI is used, those undesirable behaviors of the algorithm, such as converging to an eigenvalue far from the original prediction, may occur [20].

b) Although RQI converges cubically, it usually needs a few iterations before achieving this rate (unless the starting values are extremely good). It is quite expensive to use when cubic rate can not be achieved.

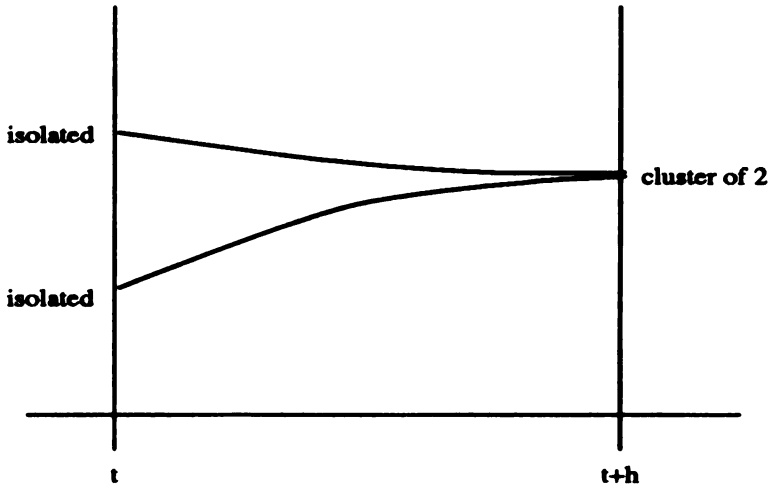


Figure 4.2: An isolated eigenvalue at t becomes nonisolated at $t + h$

To overcome these difficulties, two additional procedures are introduced. The first one is the Inverse Iteration (INVIT) procedure. It can be used to compute a better approximate eigenpair (λ, x) by using inverse iteration recursively. It is cheaper than RQI. The second one is called SUBDIM, which is designed to detect whether an approximate eigenvalue falls inside a cluster of eigenvalues. If it does, SUBDIM will compute the dimension of the invariant subspace corresponding to this cluster of eigenvalues and also generates an approximate basis for the subspace. Otherwise, SUBDIM is equivalent to an additional step of inverse iteration.

With the help of these two procedures, the correction goes as follows: First, a few steps of INVIT are used to obtain a better approximate eigenpair (λ, x), then SUBDIM is called to determine if λ is an isolated eigenvalue. If so, the faster method — RQI is applied for the rest of the correction steps. The starting values (λ,x) for the iteration are now inside or closer to the cubic converging range of RQI. If λ is not an isolated eigenvalue, then from SUBDIM, the dimension of an invariant subspace corresponding to eigenvalues close to λ and an approximate basis for the subspace are available. Then, similar to the first case described at the beginning of this section, subroutine SIRR is applied to further improve the subspace to the desired accuracy.

The detail of INVIT and SUBDIM is described in the rest of this section.

(a). **INVIT:** This is a slightly different version of ordinary inverse iteration procedure:

0) set $x^{(0)} = x_i(t_j)$, $\mu = \tilde{\lambda}_i(t_{j+1})$, $k = 0$

1) solve

$$(A(t_{j+1}) - \mu I)y^{(k)} = x^{(k)} \tag{4.4}$$

for $y^{(k)}$.

2) compute $\gamma_k = \|y^{(k)}\|$ and let

$$x^{(k+1)} = \frac{y^{(k)}}{\gamma_k}.$$

3) if $\gamma_k \leq M$ and $\gamma_k/\gamma_{k-1} \geq \alpha$ then k=k+1, goto 1)
else let $\lambda = \rho(x^{(k+1)})$, x = $x^{(k+1)}$, quit;

where

$$\rho(x^{(k+1)}) = (x^{(k+1)})^T A(t_{j+1})x^{(k+1)}$$

is the Rayleigh quotient.

There are two control parameters in this procedure, namely M and α. In our algorithm, we set $M = 10^4/\mu$ and $\alpha = 1.2$. It is well known that $x^{(k)}$ will converge to an invariant subspace corresponding to the eigenvalues close to μ and

$$\gamma_k \to \frac{1}{\min|\mu - \lambda_i|}.$$

The convergence comes in two ways: if the prediction μ is very close to some true eigenvalues ($\min|\mu - \lambda_i| < 1/M$), $\gamma_k > M$ is quickly satisfied; on the other hand, if μ is not close to any particular eigenvalue ($\min|\mu - \lambda_i| > 1/M$), then we must wait for the second condition $\gamma_k/\gamma_{k-1} < \alpha$ to be satisfied. Inequality $\gamma_k/\gamma_{k-1} < \alpha$ indicates that γ_k will not grow significantly in the following iterations. In this case, the inverse iteration procedure has been stabilized or converged. For our purpose, this procedure

is not affected by close eigenvalues, since one of the two conditions must be satisfied in the end. There are other choices for M and α. Since we will switch to a faster method eventually, it is appropriate to impose a loose converging condition.

(b). **SUBDIM** — for computing the dimension and basis of the invariant subspace corresponding to the eigenvalues close to λ:

Suppose $(\lambda_1, x_1), (\lambda_2, x_2), \cdots, (\lambda_n, x_n)$ are n eigenpairs of $A(t_{j+1})$, let $\{y_1, y_2, \cdots, y_n\}$ be a set of linearly independent vectors chosen beforehand. Take y_1, orthogonalize it against x_1 and call it y_1 again. Then

0) let k=1, $z_1 = x$

i) solve

$$(A(t_{j+1}) - \lambda I)z = y_k \tag{4.5}$$

for z

ii) compute $\|z\|$.

If $\|z\|$ is large, then

let $z_{k+1} = z/\|z\|$, take y_{k+1}, use MGS to orthogonalize it against $\{z_1, z_2, \cdots, z_{k+1}\}$, let iterat=1,k=k+1, goto i).

else

if iterat = 1, orthogonalize z against $\{z_1, z_2, \cdots, z_k\}$, let $y_k = z/\|z\|$, iterat = iterat+1, goto i).

end if.

where MGS stands for modified Gram-Schmidt process [12].

From this procedure, k is the dimension of invariant subspace and $\{z_1, z_2, \cdots, z_k\}$ is an approximate basis.

The theoretical background for this procedure is the following.

Suppose $\lambda_1, \lambda_2, \cdots, \lambda_m$ are in a cluster for some $1 \le m < n$ such that

$$\max_{1 \le l \le m} |\lambda_l - \lambda| \le \epsilon, \qquad \min_{m+1 \le l \le n} |\lambda_l - \lambda| \ge \sigma > \epsilon. \tag{4.6}$$

Expand the vector y_k in (4.5) in terms of $x_1, x_2, \cdots, x_n$:

$$y_k = \sum_{l=1}^{n} a_l^{(k)} x_l = \sum_{l=1}^{m} a_l^{(k)} x_l + \sum_{l=m+1}^{n} a_l^{(k)} x_l \stackrel{\text{def}}{=} y_k^{(0)} + y_k^{(1)},$$

then the solution of (4.5) is

$$\begin{aligned} z &= (A(t_{j+1}) - \lambda I)^{-1} y_k \\ &= (A(t_{j+1}) - \lambda I)^{-1} \sum_{l=1}^{n} a_l^{(k)} x_l \\ &= \sum_{l=1}^{n} \frac{a_l^{(k)}}{\lambda_l - \lambda} x_l \\ &= \sum_{l=1}^{m} \frac{a_l^{(k)}}{\lambda_l - \lambda} x_l + \sum_{l=m+1}^{n} \frac{a_l^{(k)}}{\lambda_l - \lambda} x_l. \\ &\stackrel{\text{def}}{=} z^{(0)} + z^{(1)}. \end{aligned}$$

The norm of $z^{(1)}$ is not big, in fact,

$$\|z^{(1)}\| = \sqrt{\sum_{l=m+1}^{n} \frac{(a_l^{(k)})^2}{|\lambda_l - \lambda|^2}} \le \frac{\sqrt{\sum_{l=1}^{n} (a_l^{(k)})^2}}{\min_{m+1 \le l \le n} |\lambda_l - \lambda|} \le \frac{1}{\sigma}.$$

If $k < m$, y_k is orthogonal to $\tilde{V}_k = \{z_1, z_2, \cdots, z_k\}$, hence y_k has nontrivial component $y_k^{(0)}$ in V_m, and the norm of z grows rapidly during the iteration; in fact,

$$\|z\| \ge \|z^{(0)}\| = \sqrt{\sum_{l=1}^{m} \frac{(a_l^{(k)})^2}{|\lambda_l - \lambda|}} \ge \frac{1}{\epsilon} \|y_k^{(0)}\|.$$

Hence $\|z\|$ becomes large in one or two iterations, and when the procedure continues, the subspace dimension continues to grow. When one reaches $k = m$, however, y_k is orthogonal to the whole subspace $\tilde{V}_m$ which is a good approximation to V_m, hence $y_k^{(0)} \approx 0$, and $\|z\|$ is not large. Then the procedure ends. In such a way, we can detect the size of the cluster and generate a set of vectors which is a good approximate basis for the corresponding invariant subspace.

It is necessary to be specific about "large" and "not large" when this algorithm is used. Notice that keeping on iterating (4.5) yields,

$$\|y_k^{(0)}\| \to 1, \qquad \|y_k^{(1)}\| \to 0,$$

hence in the end

$$\|z\| \geq \frac{1}{\epsilon}.$$

Therefore, we can say, for example, $\|z\|$ is large if it is greater than $1/2\epsilon$, it is not large otherwise, where ϵ is as in (4.6). Our experience is that, one needs not be too restrictive, because underestimating the cluster dimension is more serious than overestimating it, which can cause the slow convergence of the subspace iteration. One remedy for the possible incorrect dimension count is that one can use "dynamic" scheduling (describe in the next section) during the progress of the subspace iteration.

4.5 Dynamic Subspace Iteration

When the convergence of the Rayleigh quotient iteraion or subspace iteration is too slow (more than three iterations, for instance), it indicates that the estimate of the subspace dimension is inappropriate, either too big or too small. The iteration process needs to be monitored more carefully, and the subspace dimension should be ajusted during the progress when it becomes necessary: after a few steps of Rayleigh-Ritz procedure [23], only those Ritz vectors with their Ritz values close together are kept for further iterations. The subspace dimension is then reduced and the condition of the subspace is improved. This in turn will reduce the computation and speed up the convergence. If, on the other hand, all the Ritz values are close and convergence is still slow, then it becomes necessary to enlarge the subspace. This can be achieved by calling procedure SUBDIM to determine a more accurate dimension for the invariant subspace, and the subspace iteration resumes after this. Such a process often accelerates the convergence considerably.

Remark 4.1: For large sparse matrix, factorization is feasible, but it is still one of the major parts of the computation when solving an equation. Therefore we keep the number of factorizations to a minimum in our algorithm by including all the procedures such as INVIT, SUBDIM, SIRR, and RQI in the algorithm. INVIT, SUBDIM, and SIRR are all iterative methods. They require solving the systems with different right hand sides only, that is, for different iteration step, new factorization is not needed. These methods are not the fastest, but by combining with faster methods such as RQI, one can still achieve high convergent rate to reduce the computation time.

4.6 Checking

When RQI or SIRR is used, equations of the type

$$(A(t) - \mu I)x = y$$

are solved. By taking advantage of the symmetry, the so called symmetric Gauss elimination method [3] can be applied. That is, $A(t) - \mu I$ can be decomposed as LDL^T (instead of the traditional LU form), where L is a lower triangular matrix and D is a block diagonal matrix with diagonal blocks of order one or two. Using Sylvester's Theorem, the inertia (ν, ζ, π) of $A(t) - \mu I$ is available as a by-product of the decomposition, where ν, ζ, and π are the number of eigenvalues of $A(t)$ which is greater than, equal to and less than μ, repectively. With the availability of the inertia, the location of μ relative to other eigenvalues of $A(t)$ is known. To see how the checking is done, let's assume the eigenvalues of $A(t)$ are $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ and RQI is applied to compute the eigenvalues. Suppose the k-th step of RQI is:

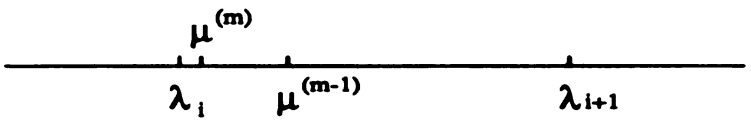
1). Solve $(A(t) - \mu^{(k)} I)y^{(k)} = x^{(k)}$ for $y^{(k)}$.

2). Compute $\gamma_k = \|y^{(k)}\|$ and set $x^{(k+1)} = y^{(k)}/\gamma_k$.

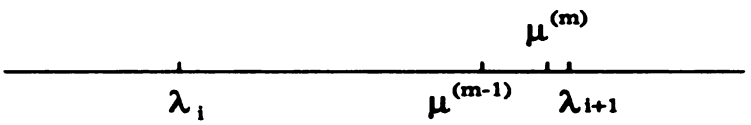
3). Set $\mu^{(k+1)} = (x^{(k+1)})^T A(t) x^{(k+1)}$ and goto 1).

Suppose the convergence is observed at step m. Let the inertias corresponding to $A(t) - \mu^{(m-1)}I$ and $A(t) - \mu^{(m)}I$ be $(\nu_{m-1}, \zeta_{m-1}, \pi_{m-1})$ and (ν_m, ζ_m, π_m) respectively. If we denote $i = \nu_{m-1}$, then $\lambda_i \leq \mu^{(m-1)} \leq \lambda_{i+1}$. Now the checking can be done as follows:

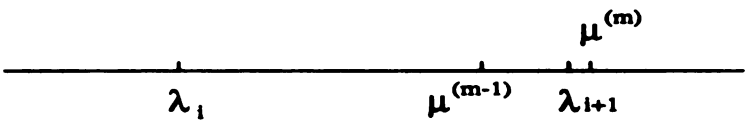
1). If $\nu_m = i$ and $\mu^{(m)} \leq \mu^{(m-1)}$, set $\lambda_i = \mu^{(m)}$.



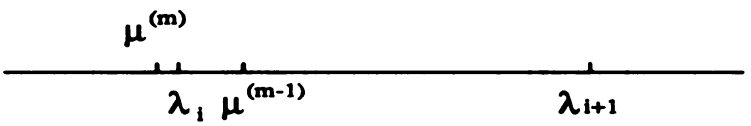
2). If $\nu_m = i$ and $\mu^{(m)} > \mu^{(m-1)}$, set $\lambda_{i+1} = \mu^{(m)}$.



3). If $\nu_m = i + 1$, set $\lambda_{i+1} = \mu^{(m)}$.



4). If $\nu_m = i - 1$, set $\lambda_i = \mu^{(m)}$.



Hence with minimum computations, we know whether the process converges to the eigencurve being followed. In general, jumping to a neighboring curve does not occur when eigenvalue separation is good. When the separation is poor, a cluster of eigenvalues are computed which usually includes the one we wanted. Our iterations in the correction steps are successful most of the time. In case the process does converge to a neighboring eigenvalue, there is a good chance that this value is also needed and the corresponding eigencurve has not been traced. Usually, the number of eigenvalues sought is larger than the number of processors available, so a total waste is unlikely.

For example, suppose we have 100 eigencurves to trace and there are 10 processors available for the purpose. Assume the scheduling is arranged as follows: The first processor is assigned to trace $\lambda_1(t), \lambda_2(t), \cdots, \lambda_{10}(t)$; the second processor is assigned to trace $\lambda_{11}(t), \lambda_{12}(t), \cdots, \lambda_{20}(t)$; and so on. If jumping occurs when the first processor is following $\lambda_1(t)$, then the computed eigenvalue should be one among $\lambda_2(t), \lambda_3(t), \cdots, \lambda_{10}(t)$, say $\lambda_3(t)$. Then tracing of the third eigencurve can be skipped.

4.7 Clearing Up

During the tracing of eigencurves, a significant amount of time is spent on keeping the process on the right curve. There are several existing techniques for this purpose [17],[18]. However, there are cases where these techniques are too costly. Several new approaches have been introduced in our algorithm. They are designed especially for large matrices. Instead of imposing very strong restrictions on those control parameters, our strategy is to abandon the process of tracing the curve if convergence to a desired value is not observed after reasonable efforts have been made. In the end, most of the needed eigenvalues of A are computed. A few of those missed are scattered across the spectrum of A and are isolated by those found, that is, there are several small intervals containing those missed values. By counting the inertias corresponding to the end points of each interval, the number of eigenvalues inside such an interval is known. These values are well separated from the neighboring ones since clusters have been found by SUBDIM in the algorithm. Under these favorite conditions, subspace iteration method can be applied in each interval efficiently to find the eigenvalues inside it, and this can be done in parallel.

4.8 Step Size Control

Our extensive numerical experience shows that the step size should be chosen as large as possible. Allowing small step size can lead to inefficiency. In our algorithm, the

first attempt of the step size h is chosen to be $\max\{0.25, (1-t)/2\}$. If convergence is not achieved at some step and $h > 0.25$, the step size is cut in half and the process is repeated. If the failing occurs when $h \leq 0.25$, we will simply abandon the process of following this specific curve. Then the eigenvalue may be computed by other process (such as SUBDIM if the value is among a cluster) or may be considered missed. Because we have a back up procedure for those eigenvalues missed due to the abandence of tracing the corresponding curves, this "giving up" strategy does not cause problem, instead, it is more efficient to have a lower bound imposed on h. There are cases that, under no control limit, h becomes very small, hence causing the long execution time at a single "bad point". In our many examples, the largest number of missed eigenvalues is usually less than 10% of the total values sought.

Chapter 5

Numerical Experiments

5.1 Introduction

Our algorithm has been implemented in several machines using FORTRAN language. The sparse matrix A is held using *coordinate scheme* ([8], p.23-24), that is, the matrix is specified as a set of triples (a_{ij}, i, j), they are stored in one real array and two integer arrays. Because A is symmetric, only the upper triangular part is stored.

In our algorithm, sparse linear systems of the type

$$(A(t) - \lambda I)y = x \tag{5.1}$$

need to be solved frequently. The intensive research in sparse matrix techniques in the past two decades resulted in several efficient codes for solving such systems. We choose MA27 subroutines from *Harwell Subroutine Library* for this purpose. This code can solve indefinite symmetric systems, such as (5.1), stably and with minimum overhead above the code for positive definite systems. It can also obtain the inertia of $A(t) - \lambda I$ as a by-product. Both of these features are vitally important to our algorithm.

The code MA27 has three separate steps: *Symbolic Analyse, Factorize*, and *Solve*. *Symbolic Analyse* exploits the sparse structure of the matrix and estimates the working space needed for later steps. *Factorize* implements a version of Gauss Elimination

to compute the LDL^T decomposition of the matrix. The last step, *Solve*, uses this decomposition to actually solve the matrix system. Since the underlying matrices $A(t)$ have the same sparse structure for all $t \in [0,1]$, *Symbolic Analyse* is required only once in the whole algorithm. In addition, for inverse iterations, one needs to solve systems with different right hand sides only, new LDL^T decomposition is not necessary, i.e., only the last step, *Solve*, is called for each iteration. This leads to significant saving in time because *Factorize* is the most expensive one of the three.

5.2 Test Matrices

Four types of matrices are used to test our algorithm. The first two can be found in [5]. They are common matrices for testing purpose.

1. Diagonally-disordered matrices:

These matrices arise in the study of two-dimensional $NX \times NY$ arrays of atoms in disordered systems [15]. The associate matrix A has order $N = NX \times NY$,

$$A = \begin{bmatrix} B & I & & I \\ I & B & \ddots & \\ & \ddots & \ddots & I \\ I & & I & B \end{bmatrix}, \quad B = \begin{bmatrix} x & 1 & & 1 \\ 1 & x & \ddots & \\ & \ddots & \ddots & 1 \\ 1 & & 1 & x \end{bmatrix}.$$

The matrix A is almost block tridiagonal. I is the identity matrix of order NX. Each B block is $NX \times NX$ and there are NY blocks down the diagonal of A. The diagonal elements of B are randomly generated numbers. A scaling parameter bounds the magnitudes of these disordered terms. In the simplest case, these entries are chosen from a uniform random distribution over an interval [-SCALE, SCALE] determined by the user-specified scaling parameter SCALE. A second class of diagonally-disordered matrices is obtained by choosing the diagonal elements randomly as either 0 or SCALE. If NX and NY are coprime, then all of the eigenvalues of A are distinct.

2. Poisson matrices:

When the Laplace equation

$$u_{xx} + u_{yy} = \lambda u, \quad R = \{(x,y)|0 \le x \le X, 0 \le y \le Y\}$$

is solved numerically, the differential equation is replaced by an algebraic linear system, $Ax = \lambda x$, obtained from discretizing the Laplace operator on the rectangle. The order of the resulting matrix is $N = NX \times NY$ where NX is the number of subdivisions in x-direction and NY is the number of subdivisions in y-direction. The matrix A is block tridiagonal,

$$A = \begin{bmatrix} B & C & & & \\ C & B & \ddots & & \\ & \ddots & \ddots & C & \\ & & C & B & sC \\ & & & sC & B \end{bmatrix}, \quad B = \begin{bmatrix} 1 & -sc & & & & \\ -sc & 1 & -c & & & \\ & -c & \ddots & \ddots & & \\ & & \ddots & \ddots & -c & \\ & & & -c & 1 & -sc \\ & & & & -sc & 1 \end{bmatrix}.$$

The parameter c is user-specified and must be chosen such that $0 < c \le 0.5$. Here, $C = -(0.5 - c)I$. For Dirichlet boundary conditions $s = 1$, and for Neumann conditions $s = \sqrt{2}$. Under Dirichlet conditions, the eigenvalues (and eigenvectors) of A are known,

$$\lambda_{ij} = [1 - 2c\cos(\pi i/(NX+1)) - 2(0.5-c)\cos(\pi j/(NY+1))],$$

$$1 \le i \le NX, \ 1 \le j \le NY.$$

Under Neumann conditions, however, the eigenvalues and eigenvectors are not known a priori. By varying the value of c, many different eigenvalue distributions can be obtained.

3. Random matrices:

The sparse matrices in this group are generated in three steps: Let N be the order of the matrix. We use *rand* to represent random number generator that produces uniform distribution on interval (0,1). Then

0) Clear the matrix:

for I=1,$\cdots$, N; J=I,$\cdots$, N

$A(I,J) = 0.0$.

1) Generate N entries randomly in the upper triangular part of A:

for i=1,$\cdots$, N

*I=Int(N*rand(i)+1), J=Int(N*rand(I)+1)*

if I<J then A(I,J)=rand(i)+A(I,J) else A(J,I)=rand(i)+A(J,I).

2) Generate N entries randomly along upper 5 diagonals:

for i=1,$\cdots$, N

*I=Int(N*rand(i)+1), J=Min{N, I+Int(5*rand(I))}*

A(I,J)=rand(i)+A(I,J).

3) Generate N entries randomly along the main and upper diagonal:

for i=1,$\cdots$, N

*I=Int(N*rand(i)+1), J=Min{N, I+Int(2*rand(I))}*

A(I,J) = rand(i)+A(I,J).

A matrix generated this way has nonzero entries concentrated near the main diagonal, a pattern shared by many sparse matrices in applications.

4. Tridiagonal matrices:

The familiar Toeplitz matrices

$$A = [1\ 2\ 1] = \begin{bmatrix} 2 & 1 & & \\ 1 & 2 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & 2 \end{bmatrix}$$

of different orders are chosen as test examples in this group.

5.3 Results on IBM 3090 Vector Machine

We compare our algorithm with EA15 — an algorithm for eigenproblems using Lanczos method. This algorithm is made up of several subroutines in *Harwell Subroutine Library* . The user supplies an interval which contains all the eigenvalues of interest and the algorithm finds all eigenvalues inside the interval and the corresponding eigenvectors.

Table 5.1 contains a list of execution times for computing the first 50 eigenpairs for different matrices. It can be seen that our algorithm runs behind Lanczos algorithm in IBM 3090, a sequential machine.

Matrix	Order	Homotopy	Lanczos	Ratio
Disordered	500	85.34	43.91	0.515
	1000	181.01	94.49	0.522
	5000	1198.72	631.73	0.527
Poisson	500	65.29	37.44	0.573
	1000	133.54	75.61	0.579
	5000	676.41	394.35	0.583
Random	500	46.83	23.65	0.505
	1000	94.66	49.04	0.518
	5000	481.37	253.68	0.527

Table 5.1: Test data from IBM

However, in the following important situations, our algorithm outperforms Lanczos algorithm.

1). *Interior Eigenvalues*: The above comparision with EA15 are based on the computing of some extremal eigenvalues. It is well known that Lanczos algorithm is

most efficient for such problems. However, in many applications, interior eigenvalues are needed. In such a circumstance, our homotopy method is much faster than Lanczos algorithm. The reason is that, during Lanczos iterations, extremal eigenvalues usually emerge before the interior ones.

Table 5.2 shows the execution times for the computation of 20 middle eigenpairs of the matrices listed.

Matrix	Order	Hom.	EA15	Ratio
Disordered	500	42.68	86.79	2.03
	1000	82.38	332.07	4.03
	5000	354.10	2487.48*	7.02
Poisson	500	33.52	70.73	2.11
	1000	63.91	270.34	4.23
	5000	347.64	2471.72*	7.11
Random	500	24.78	55.75	2.25
	1000	49.46	214.16	4.33
	5000	251.47	1750.23*	6.95

Table 5.2: Interior 20 eigenpairs on IBM 3090

* Eigenvalues only

2). *Better Initial Matrix*: As mentioned in Remark 2.2, for a typical application problem, one needs to solve a series of matrix eigenvalue problems. When this situation occurs, using a matrix in the sequence as the initial matrix, we expect to have a much better result than using the block diagonal initial matrix. An experiment is constructed to illustrate this: randomly choose n/2 nonzero entries in A, perturb them by small random numbers in (-0.05, 0.05). Then use the perturbed matrix

Matrix	Order	Homotopy	EA15	Ratio
Disordered	500	11.96	43.91	3.67
	1000	33.39	94.49	2.83
	5000	199.28	631.73	3.17
Poisson	500	11.52	37.44	3.25
	1000	18.72	75.61	4.04
	5000	119.14	394.35	3.31
Random	500	9.98	23.65	2.37
	1000	23.69	49.04	2.07
	5000	85.70	253.68	2.96

Table 5.3: First 50 eigenpairs on IBM 3090 with perturbed starting matrix

as the initial matrix D. Table 5.3 shows the execution times for both algorithms — homotopy and EA15. The time needed for the computation of initial eigensystems is not included in homotopy algorithm since it is assumed that eigensystems for that matrix are obtained in the previous step of the sequence.

Test results for tridiagonal matrices are not included in the tables above. Our algorithm runs successfully for this group. However, the code EA15 can not find all the eigenvalues in a given interval and no error messasge was signaled. It is known that Lanczos algorithm can be very fast for many problems, but it may not be suitable for general problems, even some well conditioned ones. In contrast, our algorithm seems more robust.

In sparse matrix computations, it is common experience that any method involving $O(n^2)$ or more computations is not recommendable. When the number of needed eigenpairs is fixed, it can be seen from Figure 5.1 that our algorithm's execution

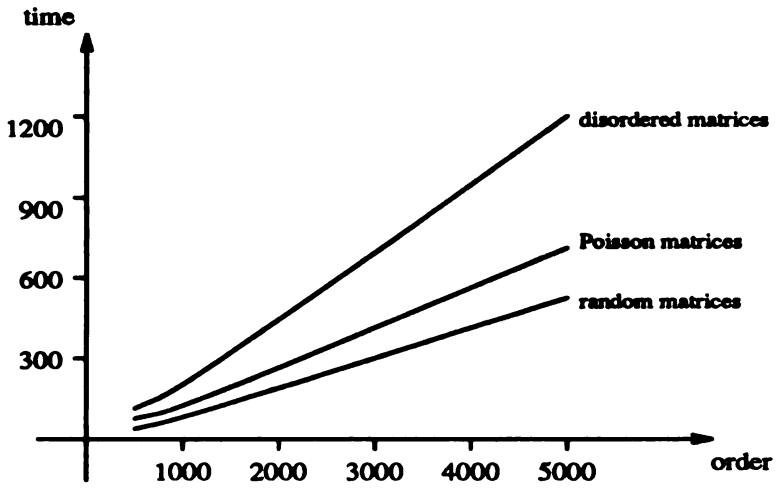


Figure 5.1: Execution time vs. matrix order

time is approximately proportional to the order n of the matrix. It appears that our homotopy method is an excellent candidate for the parallel computation of eigenvalues of sparse matrices.

5.4 Results on BBN Butterfly Parallel Machine

BBN Butterfly machine is a share-distributed memory architecture with 96 MC68020 /MC68882 BBN GP1000 processors, the maximal number of processors available to users is 90. The results in Table 5.4 and Table 5.5 are obtained from averaging the times used in computing the first 100 eigenpairs of 1000 by 1000 diagonally-disordered matrix, Poisson matrix, and random matrix.

Speed-up in Table 5.4 measures the parallelization level of the algorithm itself. It is close to the " perfect" speed-up when the number of processors is less than or equal to 16 (15 times faster using 16 processors). After that, however, the efficiency (speed-up/number of processors) becomes lower. This is caused by the computation of initial eigensystems (those of D's), since the tasks available at this step is far less than the number of processors. If the problem is of sequential type (Remark 2.2), this "bottle neck" will disappear. Another possible way of improving our algorithm

Number of processors	1	2	4	8	16	32	64
Time for homotopy	2186.85	1095.73	565.08	282.90	145.61	88.14	74.08
Speed-up	1	1.99	3.87	7.73	15.02	24.81	29.52

Table 5.4: Speed-up for homotopy algorithm

here is to apply homotopy method recursively — using the idea of "Divide and Conquer". Nevertheless, our present algorithm can achieve high speed-up close to 30.

Number of processors	1	2	4	8	16	32	64
Time for homotopy	2186.85	1095.73	565.08	282.90	145.61	88.14	74.08
Time for EA15	998.17	998.17	998.17	998.17	998.17	998.17	998.17
Speed-up	0.46	0.91	1.77	3.41	6.86	11.33	13.47

Table 5.5: Speed-up over EA15

Table 5.5 lists the speed-up of our algorithm over EA15. It can be seen that our algorithm can be more than 13 times faster than EA15, although it is slower on sequential machines.

The following two figures are the graphs of speed-up curves.

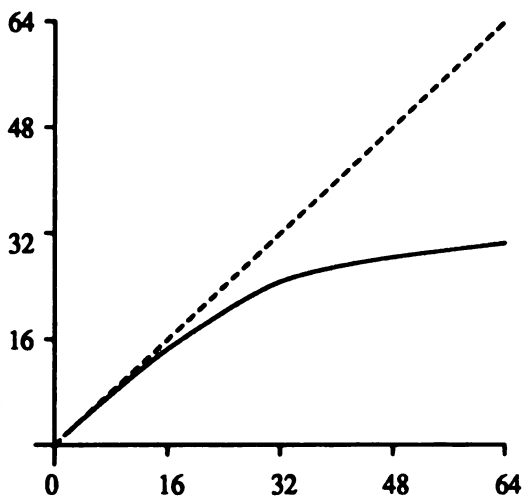


Figure 5.2: Speed-up for homotopy

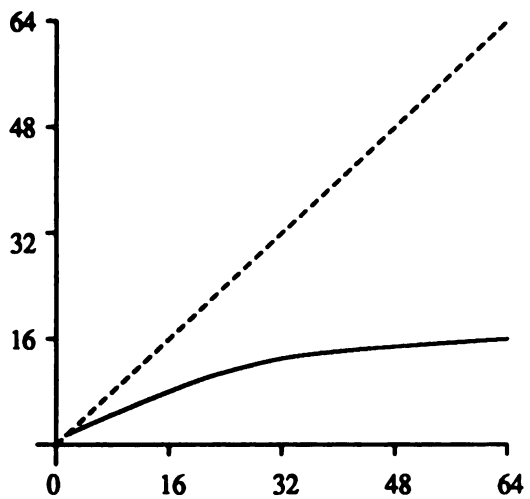


Figure 5.3: Speed-up over EA15

5.5 Accuracy and Orthogonality

The accuracy of our algorithm is also very satisfactory compare to EA15. Table 5.6 lists the maximal residuals

$$\max_{1 \le i \le 50} \|Ax_i - \lambda x_i\|$$

for the computed eigenpairs from these two algorithms. The data used here are collected from IBM 3090. The residual measures the accuracy for both the eigenvalue and eigenvector.

Matrix	Order	Homotopy	Lanczos
Disordered	500	0.97e-15	0.81e-15
	1000	0.13e-14	0.94e-15
	5000	0.27e-14	0.13e-14
Poisson	500	0.32e-14	0.74e-15
	1000	0.51e-14	0.79e-15
	5000	0.76e-14	0.11e-14
Random	500	0.27e-15	0.18e-15
	1000	0.22e-15	0.20e-15
	5000	0.23e-15	0.39e-15

Table 5.6: The residuals of eigenpairs

Table 5.7 compares the orthogonality among the computed eigenvectors. Because A is symmetric, these eigenvectors are orthogonal in theory, i.e., $\|X^T X - I\| = 0$. Listed in the last two columns are the maximal entries of the computed matrices $X^T X - I$ from these two algorithms.

Matrix	Order	Homotopy	Lanczos
Disordered	500	0.36e-15	0.52e-15
	1000	0.66e-15	0.71e-15
	5000	0.67e-15	0.82e-15
Poisson	500	0.27e-14	0.75e-15
	1000	0.23e-14	0.84e-15
	5000	0.59e-14	0.20e-14
Random	500	0.87e-16	0.13e-15
	1000	0.19e-15	0.36e-15
	5000	0.25e-15	0.47e-15

Table 5.7: The orthogonality of eigenvectors

Bibliography

[1] P. Arbenz and G.H. Golub. On the spectral decomposition of hermitian matrices modified by low rank perturbations with applications. *SIAM J. Matrix Anal. Appl.*, 9(1):40–58, Jan. 1988.

[2] R. Bhatia. *Perturbation Bounds for Matrix Eigenvalues.* Longman Scientific & Technical, New York, 1985.

[3] J.R. Bunch and B.N. Parlett. Direct methods for solving symmetric indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 8(4):639–655, 1971.

[4] E. Chu, A. George, J. Liu, and E. Ng. Sparspack: Waterloo sparse matrix package user's guide for sparspack-a. 1984.

[5] J.K. Cullum and K.A. Wiloughby. *Lanczos Algorithms for Large Symmetric Eigenvalue Computations*, volume I:Theory. Birkhauser, Boston, 1985.

[6] J.J.M. Cuppen. A divide and conquer method for the symmetric eigenproblem. *Numer. Math.*, 36:197–95, 1981.

[7] J.J. Dongarra and D.C. Sorensen. A fully parallel algorithm for the symmetric eigenvalue problem. *SIAM J. Sci. Stat. Comput.*, 8(2):139–54, March 1987.

[8] I.S. Duff, A.M. Erisman, and J.K. Reid. *Direct Methods for Sparse Matrices.* Clarendon Press, Oxford, 1986.

[9] S.C. Eisenstat, M.C. Gursky, M.H. Schultz, and A.H. Sherman. Yale sparse matrix package. 1: The symmetric codes. *Int. J. Numer. Meth. Engng.*, 18:1145–51, 1982.

[10] K. Fan. Maximum properties and inequalities for eigenvalues of completely continuous operators. *Proc. Nat. Acad. Sci. U.S.A.*, 75:760–766, 1951.

[11] A. George and J.W.H. Liu. *Computer Solution of Large Sparse Positive- definite Systems.* Prentice-Hall, New York, 1981.

[12] G.H. Golub and C.F. Van Loan. *Matrix Computation.* The Johns Hopkins University Press, Baltimore, 1983.

[13] P.S. Jenson. The solution of large symmetric eigenproblems by sectioning. *SIAM J. Numer. Anal.*, 9:534–545, 1972.

[14] T. Kato. *Perturbation Theory for Linear Operators.* Springer-Verlag, New York, 1966.

[15] S. Kirkpatrick and T.P. Eggarter. Localized states of a binary alloy. *Phys. Rev.*, B 6:3589–3600, 1972.

[16] K. Li and T.Y. Li. An algorithm for symmetric tridiagonal eigenproblems — divide and conquer with homotopy continuation. *To Appear.*

[17] T.Y. Li and N.H. Rhee. Homotopy algorithm for symmetric eigenvalue problems. *Numer. Math.*, 55(3):265–80, 1989.

[18] T.Y. Li, H.Z. Sun, and X.H. Sun. Parallel homotopy algorithm for symmetric tridiagonal eigenvalue problem. *SIAM J. Sci. Stat. Comput.*, 12(3):155–65, May 1988.

[19] S.S. Lo, B. Philippe, and A. Sameh. A multiprocessor algorithm for the symmetric eigenvalue problem. *SIAM J. Sci. Stat. Comput.*, 8(2):155–65, March 1987.

[20] S.C. Ma, M.L. Patrick, and D.B. Szyld. A parallel, hybrid algorithm for the generalized eigenproblem. *Preprint*, 1988.

[21] J. Ortega. *Numerical Analysis; a second course.* Academic Press, New York, 1972.

[22] M.L. Overton. On minimizing the maximum eigenvalue of a symmetric matrix. *SIAM J. Matrix Anal. Appl.*, 9(2):256–68, April 1988.

[23] B.N. Parlett. *The Symmetric Eigenvalue Problem.* Prentice-Hall, Englewood Cliff, New York, 1980.

[24] F. Rellich. *Perturbation Theory of Eigenvalue Problems.* Gordon and Breach, New York, 1969.

[25] B.T. Smith, J.M. Boyle, J.J. Dongarra, B.S. Garbow, Y. Ikebe, V.C. Klema, and C.B. Moler. *Matrix Eigensystem Routines — EISPACK Guide, Lecture Notes in Computer Science*, volume 6. Springer-Verlag, Berlin, 2 edition, 1976.

[26] G.W. Steward. *Introduction to Matrix Computations.* Academic Press, New York, 1973.

[27] D. Szyld. Criteria for combining inverse and rayleigh quotient iterations. *SIAM J. Numer. Anal.*, 25(6):1369–75, 1988.

[28] J. von Neumann. Some matrix inequalities and metrization of matrix space. *Tomk. Univ. Rev.*, 1:286–300, 1937.

[29] J.H. Wilkinson. *The Algebraic Eigenvalue Problem.* Oxford Univ. Press, Oxford, 1965.

[30] Z. Zeng, T.Y. Li, and L. Cong. Solving eigenvalue problems of real nonsymmetric matrices with real homotopies. *SIAM J. Numer. Anal.*, 29(1):229–248, 1992.


MICHIGAN STATE UNIV. LIBRARIES
31293008772661