This is to certify that the

dissertation entitled

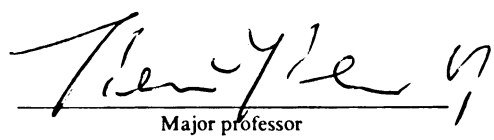
THE HOMOTOPY METHOD FOR THE SYMMETRIC EIGENVALUE PROBLEMS

presented by

Noah H. Rhee

has been accepted towards fulfillment
of the requirements for

PhD degree in Mathematics



Major professor

Date June 30th, 1987

O-12771

THE HOMOTOPY METHOD FOR THE SYMMETRIC EIGEN-VALUE PROBLEMS

By

Noah H. Rhee

A DISSERTATION

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

DOCTOR OF PHILOSOPHY

Department of Mathematics

1987

ABSTRACT

THE HOMOTOPY METHOD FOR THE
SYMMETRIC EIGEN-VALUE PROBLEMS

By

Noah H. Rhee

The homotopy method is applied to solve the linear algebraic eigen-value problems for symmetric matrices. Special homotopy equations for symmetric eigen-value problems $Ax = \lambda x$ are constructed.

It is known that there are n distinct curves connecting trivial solutions to desired eigen-pairs. Each curve is composed of an eigenvector curve in $R^n \times [0,1]$ and an eigenvalue curve in $R \times [0,1]$. In this thesis we show that it is enough to consider only an eigenvalue curve in $R \times [0,1]$.

The homotopy method for calculating eigen-values and eigen-vectors of a matrix is a serious alternative to the currently most popular approach EISPACK for SIMD machine.

The computational results obtained through this thesis are extremely promising. This thesis is the first serious attempt to make the homotopy method for symmetric eigen-value problems efficient by making use of special features of the homotopy method.

ACKNOWLEDGMENTS

I thank God who gave me the gift of life and the chance to study Mathematics at Michigan State University.

I am grateful to Professor Tien-Yien Li, my thesis advisor. His guidance and encouragement made this work possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGROITHMS

INTRODUCTION

Solving an eigenvalue problem $Ax = \lambda x$ for a symmetric $n \times n$ matrix A can be thought of as solving a nonlinear system of polynomial equations.

$$F_\varphi(x,\lambda) = 0,$$

where $F_\varphi : R^n \times R \to R^n \times R$ is defined by

$$F_\varphi(x,\lambda) = \begin{pmatrix} \lambda x - Ax \\ \varphi(x) \end{pmatrix}$$

and φ is a polynomial from R^n to R. For example, if we want eigenvectors to have Euclidean norm 1, we might let $\varphi(x) = x_1^2 + x_2^2 + \ldots + x_n^2 - 1$, where $x = (x_1, x_2, \ldots, x_n)^T$.

From this point of view, many well developed methods can then be employed to find zeros of this F_φ.

We shall assume that all eigenvalues of A are distinct. Then the classical Newton's method and its many improved modifications are applicable for solving $F_\varphi(x,\lambda) = 0$. Unfortunately Newton's method converges to only one zero at a time. In order to

obtain all n-eigenpairs of A, we have to restart the iteration by making n suitable initial guesses, which is difficult in practice.

The homotopy method of finding all the isolated solutions of a system of polynomials has attracted considerable attention recently.

The basic idea of homotopy continuation is to construct a homotopy from a trivial map to the one of interest. Under suitable conditions, a smooth curve starting from the solution of the trivial map will lead us to the desired solution.

The homotopy method for the symmetric eigenvalue problems was studied by Chu [3]. For the general eigenvalue problems, a homotopy was given by Li, Sauer and Yorke [4]. They constructed homotopies which give n-disjoint smooth curves in $R^{n+1} \times [0,1]$ (or $C^{n+1} \times [0,1]$). And each curve leads from an obvious starting point to an eigenpair (x,λ) of the given matrix.

In both [3] and [4] a curve in $R^{n+1} \times [0,1]$ (or $C^{n+1} \times [0,1]$) is composed of an eigenvector curve in $R^n \times [0,1]$ (or $C^n \times [0,1]$) and an eigenvalue curve in $R \times [0,1]$ (or $C \times [0,1]$). In this thesis we consider only an eigenvalue curve in $R \times [0,1]$ by constructing the following homotopy equation:

$$H : R^n \times R \times [0,1] \to R^n$$

such that

(0.1) $H(x,\lambda,t) = (1-t)[\lambda x - Dx] + t[\lambda x - Ax] = 0,$

where D is an $n \times n$ matrix whose eigenpairs can be computed easily.

Under suitable conditions n-distinct smooth eigenvalue curves (in $R \times [0,1]$) of (0.1) exist, and each eigenvalue curve leads from an eigenvalue of D to that of A. Once we find an eigenvalue, the corresponding eigenvector is immediate by Inverse Power Iteration.

Each eigenvalue curve can be characterized by the solution curve of a scalar ordinary differential equation with an initial value an eigenvalue of D. Hence each eigenvalue curve can be followed numerically. Furthermore different eigenvalue curves correspond only to different initial values of the same ordinary differential equation. Following one eigenvalue curve is completely independent of following the other eigenvalue curves. Therefore the homotopy algorithm is an excellent candidate for exploiting the advantages of parallel processing.

Also the homotopy method maintains the structure of the underlying matrix, if there is any.

Chapter I discusses the existence of n-distinct smooth eigenvalue curves and the local conditioning of the eigenvalue curve. Chapter II discusses the curve following algorithm in detail. Chapter III gives several numerical results which show that the homotopy algorithm for eigenvalue problems can be a serious alternative to the QR-algorithm which is currently the most powerful algorithm for eigenvalue problems.

CHAPTER I: THE EIGENVALUE CURVE

The discussion of the eigenvalue curve is divided into two sections: The Existence of n-distinct Eigenvalue Curves, and The Local Conditioning Factors of an Eigenvalue Curve.

§1.1: The Existence of n-Distinct Eigenvalue Curves

From (0.1) we have

$$\begin{aligned} H(x,\lambda,t) &= (1-t)(\lambda x - Dx) + t(\lambda x - Ax) \\ &= \lambda x - (D + t(A-D))x \\ &= \lambda x - A(t)x \end{aligned}$$

where $A(t) = D + t(A-D)$.

We call D the initial matrix and A the final matrix.

Because A is symmetric, we can tridiagonalize A by a standard tridiagonalization process. Hence we shall assume that A is tridiagonal. We may assume that none of the off-diagonal elements is zero, for otherwise we would subdivide A into the direct sum of tridiagonal matrices of lower order and work instead with them.

We choose the initial matrix D such that

(a) D has distinct eigenvalues and all its eigenpairs are available.

(b) $A(t) = D + t(A - D)$ is tridiagonal with non-zero off-diagonal elements unless $t = 0$.

Remark 1.1:

(1) (a) and (b) imply that for each $t \in [0,1]$, $A(t)$ has n-distinct eigenvalues. For a proof see [8, page 124].

(2) The conditions (a) and (b) are easily satisfied, for example, by choosing a diagonal matrix D with distinct diagonal elemtns. ***

We denote the n-distinct eigenvalues of $A(t)$ by $(\lambda^1(t), \lambda^2(t), \ldots, \lambda^n(t))$ and assume that $\lambda^1(t) < \lambda^2(t) < \ldots < \lambda^n(t)$. And we denote the corresponding normalized (with respect to the Euclidean norm) eigenvectors by $(\hat{x}^1(t), \hat{x}^2(t), \ldots, \hat{x}^n(t))$. Henceforth $\hat{}$-notation will be used for a unit vector. ***

The eigenvalues of a matrix are continuous functions of the elements of the matrix [7]. Hence we have the following proposition.

Proposition 1.1: $\lambda^i(t)$ is a continuous function of t for $i = 1, \ldots, n$. ***

We shall denote $\{(\lambda^i(t),t) : 0 \leq t \leq 1\}$ by C_i for $i = 1,2,\ldots,n$. Note that $C_i \cap C_j = \emptyset$ if $i \neq j$ from Remark 1.1: (1).

Remark 1.2:

(1) Proposition 1.1 establishes that there are n-distinct continuous eigenvalue curves $C_1, C_2, \ldots, C_n$ such that C_i joints the ith eigenvalue of D and the ith eigenvalue of A. We call this property the Order Preserving Property of eigenvalues.

(2) The Order Preserving Property of eigenvalues is important, since in application we often need to find few eigenvalues which are either algebraically the largest or smallest. The Order Preserving Property of eigenvalue also provides a valuable checking algorithm as we follow an eigenvalue curve (see Chapter II, section 3.) ***

Now we want to show that C_i is, in fact, a C^∞-curve for $i = 1,2,\ldots,n$.

For fixed v in $[0,1]$, let $(\hat{x}(v),\lambda(v))$ be the ith eigenpair of $A(v)$. Consider the linear functional $\varphi : R^n \to R$ defined by $\varphi x = \hat{x}(v)^T x$ for $x \in R^n$.

Let $\tilde{H} : R^n \times R \times [0,1] \to R^n \times R$ be defined by

$$\tilde{H}(x,\lambda,t) = \begin{pmatrix} \lambda x - A(t)x \\ \hat{x}(v)^T x - 1 \end{pmatrix}$$

Lemma 1.2: The Frechet-Derivative $D_{(x,\lambda)}\tilde{H}(\hat{x}(v),\lambda(v),v)$ of $\tilde{H}$ with respect to x and λ at $(\hat{x}(v),\lambda(v),v)$ is nonsingular.

Proof: See [3]. ***

Notice that $\tilde{H}(\hat{x}(v),\lambda(v),v) = 0$. Hence by the Implicit Function Theorem, there exist $\delta > 0$ and a unique C^∞-curve

$$\Gamma_v = \{\Gamma_v(t) : v - \delta < t < v + \delta\}$$
$$= \{(x(t),\lambda(t),t) : (x(v),\lambda(v),v) = (\hat{x}(v),\lambda(v),v),\ v - \delta < t < v + \delta\}$$

such that $\tilde{H}(x,\lambda,t) = 0$ on Γ_v.

We denote the second component of Γ_v by C_v, that is, $C_v = \{(\lambda(t),t) : v - \delta < t < v + \delta\}$.

Proposition 1.3: C_v is the restriction of C_i to $(v - \delta, v + \delta)$.

Proof: Since (1) $(\lambda(v),v) \in C_i \cap C_v$, (2) C_i is continuous and disjoint from C_j for $j \neq i$, and (3) C_v is smooth, it follows that C_v must be the restriction of C_i to $(v-\delta, v+\delta)$. ***

Remark 1.3: This proposition shows that C_i is a C^∞-curve for $i = 1,2,\ldots,n$. ***

§1.2: The Local Conditioning Factors of an Eigenvalue Curve

Since C_i is C^∞-curve and $C_i = C_v = \{(\lambda(t),t) : v-\delta < t < v+\delta\}$ on $(v-\delta, v+\delta)$ by Proposition 1.3, the bound of the nth derivative of $\lambda(t)$ at $t = v$ is a good measure of determining the local conditioning of C_i. Hence we shall proceed to find a bound of the nth derivative of $\lambda(t)$ at $t = v$.

Proposition 1.4:

$$(D_{(x,\lambda)}\tilde{H}(\hat{x}(v),\lambda(v),v))^{-1} = \left[\begin{array}{c|c} \lambda(v)I - A(v) & \hat{x}(v) \\ \hline \hat{x}(v)^T & 0 \end{array}\right]^{-1} = \left[\begin{array}{c|c} [\lambda(v)I - A(v)]^+ & \hat{x}(v) \\ \hline \hat{x}(v)^T & 0 \end{array}\right]$$

where $[\lambda(v)I - A(v)]^+$ is the Pseudo-Inverse of $\lambda(v)I - A(v)$, and I is the $n \times n$ identity matrix.

Proof: See [2]. ***

Let $d(i,v) = \min_{\nu \neq \lambda(v)} |\lambda(v) - \nu|$, where the minimum is taken over $\sigma(A(v))$ and $\sigma(A(v))$ is the spectrum of $A(v)$. In $d(i,v)$, i means that $\lambda(v)$ is the ith eigenvalue of $A(v)$. We call $d(i,v)$ the eigenvalue separation at $\lambda(v)$. Let $\|\cdot\|$ denote either the Euclidean norm for a vector or the spectral norm for a matrix.

Lemma 1.5: $\| [\lambda(v)I - A(v)]^{+} \| = 1/d(i,v)$.

Proof: From

$$\sigma(\lambda(v) - A(v)) = \{0\} \cup \{\lambda(v) - \nu : \nu \in \sigma(A(v)), \nu \neq \lambda(v)\},$$

we have

$$\sigma([\lambda(v)I - A(v)]^{+}) = \{0\} \cup \{\frac{1}{\lambda(v) - \nu} : \nu \in \sigma(A(v)), \nu \neq \lambda(v)\}.$$

Hence $\| [\lambda(v)I - A(v)]^{+} \| = 1/d(i,v)$. ***

We denote $(d^n/dt^n)\lambda(v)$ by $\lambda^{(n)}(v)$ and $(d^n/dt^n)x(v)$ by $x^{(n)}(v)$. And we denote $\lambda^{(1)}(v)$ and $x^{(1)}(v)$ by $\dot{\lambda}(v)$ and $\dot{x}(v)$ respectively.

Theorem 1.6:

$$|\lambda^{(k)}(v)| < C(k)\|A - D\|^{k}/d(i,v)^{k-1}$$

$\|x^{(k)}(v)\| \leq C(k)\|A - D\|^k/d(i,v)^k$ for $k = 1,2,\cdots$,

where $C(k)$ is a constant, which depends only on k.

Proof: We will use induction to prove this theorem. For $m = 1$, $(d/dt)\tilde{H}(\hat{x}(v),\lambda(v),v) = 0$ implies that

$$\left[\begin{array}{c|c} \lambda(v)I - A(v) & \hat{x}(v) \\ \hline \hat{x}(v)^T & 0 \end{array}\right] \left[\begin{array}{c} \dot{x}(v) \\ \hline \dot{\lambda}(v) \end{array}\right] = \left[\begin{array}{c} (A - D)\hat{x}(v) \\ \hline 0 \end{array}\right]$$

By Proposition 1.5

$$\left[\begin{array}{c} \dot{x}(v) \\ \hline \dot{\lambda}(v) \end{array}\right] = \left[\begin{array}{c|c} [\lambda(v)I - A(v)]^+ & \hat{x}(v) \\ \hline \hat{x}(v)^T & 0 \end{array}\right] \left[\begin{array}{c} (A - D)\hat{x}(v) \\ \hline 0 \end{array}\right]$$

$$= \left[\begin{array}{c} [\lambda(v)I - A(v)]^+(A - D)\hat{x}(v) \\ \hline \hat{x}(v)^T(A - D)\hat{x}(v) \end{array}\right]$$

Thus $\dot{x}(v) = [\lambda(v) - A(v)]^+(A - D)\hat{x}(v)$

$$\begin{aligned} \|\dot{x}(v)\| &= \|[\lambda(v) - A(v)]^+(A - D)\hat{x}(v)\| \\ &\leq \|[\lambda(v) - A(v)]^+\|\|A - D\| \\ &= \|A - D\|/d(i,v) . \end{aligned}$$

Hence $\|\dot{x}(v)\| \leq \|A - D\|/d(i,v)$.

Also

$$\dot{\lambda}(v) = \hat{x}(v)^T (A - D)\hat{x}(v) \tag{1.1}$$

$$|\dot{\lambda}(v)| = |\hat{x}(v)^T (A - D)\hat{x}(v)|$$

$$\leq \|\hat{x}(v)\| \|A - D\| \|\hat{x}(v)\|$$

$$= \|A - D\| .$$

Hence $|\dot{\lambda}(v)| \leq \|A - D\|$. Therefore our Theorem is true for $m = 1$.

Suppose that $\|\lambda^{(m)}(v)\| \leq C(m)\|A - D\|^m / d(i,v)^{m-1}$ and $\|x^{(m)}(v)\| \leq C(m)\|A - D\|^m / d(i,v)^m$ are true for $m \leq k - 1$.

Let $m = k$ $(k \geq 2)$. By repeatedly differentiating $\tilde{H}(x(t), \lambda(t), t) = 0$ with respect to t and using the fact that $(d/dt)[A(t)] = (d/dt)[D + t(A - D)] = A - D$ and evaluating at $(\hat{x}(v), \lambda(v), v)$, we obtain the following expression:

$$\left[\begin{array}{c|c} \lambda(v) I - A(v) & \hat{x}(v) \\ \hline \hat{x}(v)^T & 0 \end{array}\right] \left[\begin{array}{c} x^{(k)}(v) \\ \hline \lambda^{(k)}(v) \end{array}\right]$$

$$= \left[\begin{array}{c} k(A - D)x^{(k-1)}(v) + g_1 \lambda^{(1)}(v) x^{(k-1)}(v) + g_2 \lambda^{(2)}(v) x^{(k-2)}(v) \\ + \cdots + g_{k-1} \lambda^{(k-1)}(v) x^{(1)}(v) \\ \hline 0 \end{array}\right]$$

where $g_1, g_2, \ldots, g_{k-1}$ are constants, which depend only on k. Let $1 \leq p \leq k - 1$. Then by the induction hypothesis we have

$$|\lambda^{(p)}(v)| = C(p)\|A-D\|^p/d(i,v)^{p-1}$$
$$\|x^{(p)}(v)\| = C(p)\|A-D\|^p/d(i,v)^p.$$

Hence the upper part of the right hand side is bounded by $C(k)\|A-D\|^k/d(i,v)^{k-1}$. Let us denote the right hand side by $[h,0]^T$. Then $\|h\| \le C(k)\|A-D\|^k/d(i,v)^{k-1}$. Hence

$$\begin{bmatrix} x^{(k)}(v) \\ \hline \lambda^{(k)}(v) \end{bmatrix} = \left[\begin{array}{c:c} [\lambda(v)I - A(v)]^+ & \hat{x}(v) \\ \hline \hat{x}(v)^T & 0 \end{array}\right] \begin{bmatrix} h \\ \hline 0 \end{bmatrix}$$

$$= \begin{bmatrix} [\lambda(v)I - A(v)]^+ h \\ \hat{x}(v)^T h \end{bmatrix}$$

Thus

$$\|x^{(k)}(v)\| = \|[\lambda(v)I - A(v)]^+ h\|$$
$$\le \|[\lambda(v) - A(v)]^+\|\|h\|$$
$$\le C(k)\|A-D\|^k/d(i,v)^k$$
$$|\lambda^{(k)}(v)| = |\hat{x}(v)h|$$
$$\le \|x(v)\|\|h\|$$
$$\le C(k)\|A-D\|^k/d(i,v)^{k-1}. \qquad ***$$

Remark 1.4:

(1) The bound of $|\lambda^{(k)}(v)|$ shows that there are two important factors which can influence the local conditioning of eigenvalue curve. First, the poorer the eigenvalue separation is, the

poorer the local conditioning of the eigenvalue curve can be. Second, the closer D is to A, the better the local conditioning of the eigenvalue curve can be.

(2) Since the choice of D is basically free, it may be possible to improve the conditioning of the eigenvalue curve globally by a suitable choice of D, since $\|A - D\|$ does not depend on i or v.

(3) The bound of $|\lambda^{(k)}(v)|$ is independent of the size of the matrix. So the growth of the matrix size does not imply that the eigenvalue curve becomes ill-conditioned.

(4) The equation (1.1) is a known result. (See, for example, [7] or [10, page 389]). It gives the scalar ordinary differential equation which characterizes the eigenvalue curve as its solution curve. ***

CHAPTER II: THE ALGORITHM

Since $\dot{\lambda}(t) = \hat{x}(t)^T (A - D)\hat{x}(t)$ from (1.1), it is feasible to use any available ODE software solvers to follow an eigenvalue curve. However to calculate $\dot{\lambda}(t)$, we need $\hat{x}(t)$. Hence to use any available ODE software solvers as they are, a suitable form of ordinary differential equation is

$$\begin{bmatrix} \dot{x}(t) \\ \hline \dot{\lambda}(t) \end{bmatrix} = \begin{bmatrix} \lambda(t) I - A(t) & \hat{x}(t) \\ \hline \hat{x}(t)^T & 0 \end{bmatrix}^{-1} \begin{bmatrix} (A - D)\hat{x}(t) \\ \hline 0 \end{bmatrix}$$

[see [3]]. Hence to calculate

$$\begin{bmatrix} \dot{x}(t) \\ \hline \dot{\lambda}(t) \end{bmatrix},$$

we need to solve a linear system

$$\begin{bmatrix} \lambda(t) I - A(t) & \hat{x}(t) \\ \hline \hat{x}(t)^T & 0 \end{bmatrix} \begin{bmatrix} \dot{x}(t) \\ \hline \dot{\lambda}(t) \end{bmatrix} = \begin{bmatrix} (A - D)\hat{x}(t) \\ \hline 0 \end{bmatrix}. \tag{2.1}$$

But the matrix which is involved in (2.1) is no longer tridiagonal, which is expensive to solve.

Besides, the accuracy requirement in ODE software solvers keeps the step-sizes in a range where instability does not occur. Hence, stability consideration limits the step-size.

But in our problem $(\hat{x}(t),\lambda(t),t)$ is a solution to $H(x,\lambda,t) = 0$ (0.1), where $(\hat{x}(t),\lambda(t))$ is an eigenpair of $A(t)$. Hence we can locate $(\hat{x}(t),\lambda(t))$ as accurate as we want, for example, by correcting a reasonable approximation by the Newton method. Hence in homotopy method, there is no problem of instability.

Hence it is desirable to develop a numerical algorithm which makes use of the special structure of homotopy method for the symmetric eigenvalue problems instead of using any available ODE software solvers.

The major difficulty in following an eigenvalue curve is that there are $n-1$ other eigenvalue curves. Hence in following an eigenvalue curve, the important question is how we can prevent ourselves from jumping into another eigenvalue curve.

As usual, there are two main steps in following the solution curve of an ordinary differential equation, namely the prediction step and the correction step. In this chapter, in addition to these two steps, we also discuss a checking step.

Therefore the description of the algorithm will be divided into three parts: Prediction, Correction and Checking. Then we summerize the algorithm through the Flow Chart.

Suppose that we have found $(\hat{x}(v),\lambda(v))$, the ith eigenpair of $A(v)$ and $0 \le v \le 1$. Now we want $(\hat{x}(v+h),\lambda(v+h))$, the ith eigenpair of $A(v+h)$.

Throughout this chapter we assume that we are following the ith eigenvalue curve C_i. We shall denote C_i by $\{(\lambda(t),t) : 0 \le t \le 1\}$.

§2.1: Prediction

In this section, we want to determine the next step-size h and prediction for $\lambda(v+h)$ and $\hat{x}(v+h)$.

The description of the prediction is divided into three parts: Eigenvalue Prediction, Eigenvector Prediction and Step-Size Updating.

(a) Eigenvalue Prediction

For now, let us assume that the next step size h was determined.

From (1.1) $\dot{\lambda}(v) = \hat{x}(v)^T (A - D) \hat{x}(v)$. So $\dot{\lambda}(v)$ is immediately available. Thus we use therHermite Interpolation which interpolates $\{\lambda(u), \dot{\lambda}(u), \lambda(v), \dot{\lambda}(v)\}$ to predict $\lambda(v+h)$, where $\lambda(u)$ and $\dot{\lambda}(u)$ are the eigenvalue and eigenvalue derivative at the previous step respectively.

At $v = 0$ we choose the initial matrix D which has a simple structure so that it is easy to get higher derivatives of $\lambda(t)$ at $t = 0$. Thus we used third order Taylor method to predict for the next step eigenvalue.

Let $P(t)$ be the polynomial which interpolates $\lambda(t)$ at $\{\lambda(u), \dot{\lambda}(u), \lambda(v), \dot{\lambda}(v)\}$. Then

$$P(t) = \lambda(u) + \dot{\lambda}(u)(t-u) + \frac{\lambda(v) - \lambda(u) - \dot{\lambda}(u)(v-u)}{(v-u)^2}(t-u)^2 + \frac{(v-u)[\dot{\lambda}(v) + \dot{\lambda}(u)] - 2[\lambda(v) - \lambda(u)]}{(v-u)^3}(t-u)^2(t-v)$$

Let $w = v + h$ and $\lambda_0(w) = P(w)$, that is, $\lambda_0(w)$ is the prediction for $\lambda(w)$. We summarize the eigenvalue prediction as follows:

Algorithm 1: Eigenvalue Prediction Algorithm

(1) Calculate $\dot{\lambda}(v) = \hat{x}(v)^T (A - D) \hat{x}(v)$

(2) Calculate $P(w)$

(3) Set $\lambda_0(w) = P(w)$. ***

(b) Eigenvector Prediction

After the eigenvalue prediction we have $\lambda_0(w)$ and $\hat{x}(v)$. Then we construct the eigenvector prediction, say $\hat{x}_0(w)$, by Inverse Power Iteration as follows:

Algorithm 2: Eigenvector Prediction Algorithm

(1) Set $[\lambda_0(w) - A(w)]y = \hat{x}(v)$.

Solve for y.

(2) Set $\hat{x}_0(w) = y/\|y\|$. ***

(c) Step-Size Updating

Now we discuss how to determine the step-size h to achieve

$$ERR = |\lambda(v+h) - \lambda_0(v+h)| \leq \epsilon$$

where $\epsilon > 0$ is a prescribed accuracy.

Since the Hermite Interpolation, which interpolates $\{\lambda(u), \dot{\lambda}(u), \lambda(v), \dot{\lambda}(v)\}$, is used to predict $\lambda(v+h)$, the error in $|\lambda(v+h) - \lambda_0(v+h)|$ is related to the 4th derivative of $\lambda(t)$ for $t \in [u, v+h]$. In fact, we have the following error formula:

Theorem 2.1: Let the real function λ be $(n+1)$ times differentiable on the interval $[a,b]$ and consider $(m+1)$ support abscissae $t_i \in [a,b]$ such that $t_0 < t_1 < \ldots < t_m$. If the polynomial $P(t)$ whose degree is at the most n, where $n+1 = \sum_{i=1}^{m} n_i$,

interpolates at each support abscissae t_i not only the value but also the first $n_i - 1$ derivatives of λ, that is, $p^{(k)}(t_i) = \lambda^{(k)}(t_i)$ $(k = 0,1,\ldots,n_i-1)$ for each $i = 0,1,\ldots,m$, then to every $t \in [a,b]$, there exist $\bar{t}$, which belongs to the smallest interval containing t, such that

$$\lambda(t) - P(t) = \frac{(t-t_0)^{n_0}(t-t_1)^{n_1}\ldots(t-t_m)^{n_m}\lambda^{(n+1)}(\bar{t})}{(n+1)!}$$

Proof: See [10]. ***

By Theorem 2.1,

$$\lambda(v+h) - \lambda_0(v+h) = (v+h-u)^2(v+h-v)^2\lambda^{(4)}(\bar{t})/24$$
$$= [h+(v-u)]^2h^2\lambda^{(4)}(\bar{t})/24,$$

where $\bar{t} \in [u,v+h]$.

Suppose that we have an estimate M_v for $|\lambda^{(4)}(\bar{t})|$, $\bar{t} \in [u,v+h]$. To determine the next step size h, which makes the error $|\lambda(v+h) - \lambda_0(v+h)|$ less than ϵ, we set $[h+(v-u)]^2h^2M_v/24 = \epsilon$, that is $[h+(v-u)]^2h^2 = 24\epsilon/M_v$. Let $f(h) = [h+(v-u)]^2h^2$. Then we shall look for the smallest positive solution h of $f(h) = 24\epsilon/M_v$. The function $f(h)$ has two double zeros at $h = 0$ and $h = -v(v-u)$. So we have the following picture:

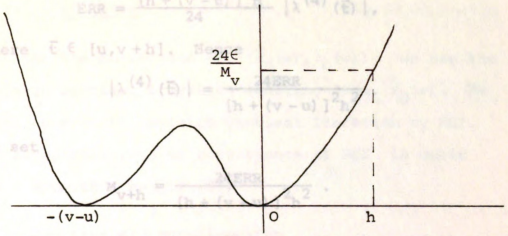


Figure 2.1

Clearly there is a unique positive solution h. Furthermore it is easy to see that Newton Iteration for the equation $f(h) = 24\epsilon/M_v$, starting from the previous step-size $h_0 = v - u$, will produce $\{h_n\}_{n=0}^{\infty}$ such that $h_n \to h$ as $n \to \infty$.

To complete the updating of the step-size, we need to update M_v so that we may have an estimate for M_{v+h}. M_{v+h} will be used for the determination of the next step-size from $t = v + h$.

Note that at this stage h is available. By Algorithm 1, we shall have $\lambda_0(v+h)$. Later by correction we shall have $\lambda(v+h)$. Let $ERR = |\lambda(v+h) - \lambda_0(v+u)|$. Again by Theorem 2.1 we have

$$ERR = \frac{[h + (v - u)]^2 h^2}{24} |\lambda^{(4)}(\bar{t})|,$$

where $\bar{t} \in [u, v+h]$. Hence

$$|\lambda^{(4)}(\bar{t})| = \frac{24ERR}{[h + (v - u)]^2 h^2}.$$

We set

$$M_{v+h} = \frac{24ERR}{[h + (v - u)]^2 h^2}.$$

Therefore, we have the following step-size updating algorithm:

Algorithm 3: Step-Size Updating Algorithm

(1) Set $f(h) = [h + (v - u)]^2 h^2 = 24\epsilon / M_v$. Solve for h by Newton Iteration starting from $h = v - u$, where ϵ is a prescribed accuracy and M_v is an estimate for $|\lambda^{(4)}(t)|$ near $t = v$.

(2) Wait until $\lambda_0(v+h)$ and $\lambda(v+h)$ are available.

(3) Calculate $ERR = |\lambda(v+h) - \lambda_0(v+h)|$.

(4) Set $M_{v+h} = \frac{24ERR}{[h + (v - u)]^2 h^2}$.

Remark 2.1: M_0 is found by calculating $|\lambda^{(4)}(0)|$. Since D has a simple structure, it can be found easily.

§2.2: Correction

For the correction of $(\hat{x}_0(w), \lambda_0(w))$, we use the Rayleigh Quotient Iteration starting from $\hat{x}_0(w)$. We shall abbreviate Rayleigh Quotient Iteration by RQI. The asymptotic rate of convergence of RQI is cubic for a symmetric matrix [8].

Algorithm 4: RQI-Algorithm

Let A be an $n \times n$ symmetric matrix and

$$\rho_A(x) = \frac{x^T a x}{x^T x} .$$

(1) Pick a unit vector $\hat{x}_0$. For $j = 0,1,2,\ldots,$ repeat the following.

(2) Compute $\lambda_{j+1} = \rho_A(\hat{x}_j)$.

(3) Solve $(\lambda_{j+1} I - A) y_{j+1} = \hat{x}_j$ for y_{j+1}.

(4) Set $\hat{x}_{j+1} = y_{j+1} / \|y_{j+1}\|$.

(5) If ($\|y_{j+1}\|$ is big enough) THEN

Accept $(\hat{x}_{j+1}, \lambda_{j+1})$ is an eigenpair of A.

ELSE

$j = j + 1$

Go to (2)

END IF. ***

Remark 2.2: (1) In our algorithm $\hat{x}_0 = x_0(w)$ and $A = A(w)$. We shall write $\hat{x}_j$ and λ_j by $\hat{x}_j(w)$ and $\lambda_j(w)$ respectively, for $j = 1,2,\cdots$. ***

(2) In step 5, that $\|y_{j+1}\|$ is big enough means that the residual is small enough. For

$$(\lambda_{j+1}I - A)\hat{x}_{j+1} = (\lambda_{j+1}I - A)\frac{y_{j+1}}{\|y_{j+1}\|} = \frac{\hat{x}_j}{\|y_{j+1}\|} .$$

Hence

$$\|(\lambda_{j+1}I - A)\hat{x}_{j+1}\| = 1/\|y_{j+1}\| .$$

Now we want to show that for small enough step-size h, RQI-Algorithm with starting point $\hat{x}_0(v+h)$ produces $\hat{x}_j(v+h)$ which converges to $\hat{x}(v+h)$.

It suffices to consider diagonal matrices to understand the geometry and dynamics of symmetric matrices under RQI [1]. Thus we shall assume that $A(v+h)$ is diagonal matrix with $\lambda^1(v+h),\ldots,\lambda^n(v+h)$ as diagonal elements for each h.

Definition 2.2: Let

$$\rho_1(x) = \sum_{j=1}^{i} \lambda^j x_j^2 / \sum_{j=1}^{i} x_j^2$$

$$\rho_2(x) = \sum_{j=i}^{n} \lambda^j x_j^2 / \sum_{j=i}^{n} x_j^2$$

$$E = \{x \in R^n : x_i = 1,\ \rho_1(x) > \frac{\lambda^{i-1} + \lambda^i}{2}$$

$$\text{and}\ \ \rho_2(x) < \frac{\lambda^i + \lambda^{i+1}}{2}\}$$

where x_j is the jth coordinate of x and λ^j is the jth eigenvalue of the associated diagonal matrix for $j = 1,2,\ldots,n$.

Theorem 2.3: If we start RQI-algorithm from any vector in E, the sequence of vectors by RQI-algorithm converges to $e^i = (0,0,\ldots,0,\underset{\text{ith}}{1},0,\ldots,0)$, which is the ith eigenvector of the associated diagonal matrix.

Proof: See [1]. ***

Remark 2.3:

(1) E is not empty since $e^i \in E$.

(2) E is open set in the $x_i = 1$ chart, since $\rho_1(x)$ and $\rho_2(x)$ are continuous functions in the $x_i = 1$ chart.

Theorem 2.4: There exists $\bar{h} > 0$ such that $0 < h < \bar{h}$ implies that RQI-algorithm starting from $\hat{x}_0(v+h)$ produces sequence which converges to e^i.

Proof: For given $h > 0$, we associate

$$E(h) = \{x \in R^n : x_i = 1, \rho_1(x) > \frac{\lambda^{i-1}(v+h) + \lambda^i(v+h)}{2}$$

$$\text{and } \rho_2(x) < \frac{\lambda^i(v+h) + \lambda^{i+1}(v+h)}{2}\}.$$

Since $E(h)$ is open in $x_i = 1$ chart, there exists a neighborhood of e^i with radius $\delta(h)$ in $x_i = 1$ chart, say $N_{\delta(h)}(e^i)$, such that $N_{\delta(h)}(e^i)$ is the maximal open ball which is in $E(h)$. Note that $\delta(h)$ is a continuous function of h since everything which is involved is continuous with respect to h.

Now we fix $\tilde{h} > 0$. Let $\delta = \min_{0 \le h \le \tilde{h}} \delta(h)$. Then $\delta > 0$, since $\delta(h)$ is a positive continuous function and $[0,\tilde{h}]$ is compact. Note that as $h \to 0$, $|\lambda^i(v+h) - \lambda_0(v+h)| \to 0$ and thus $\hat{x}_0(v+h) \to e^i$.

Hence there exists $\bar{h}$ such that $0 < \bar{h} < \tilde{h}$ and if $h < \bar{h}$ then $\|x_0(v+h) - e^i\| < \delta$, where $x_0(v+h)$ is the constant multiple of $\hat{x}_0(v+h)$ so that $x_0(v+h)$ may belong to $x_i = 1$ chart. Thus the Theorem immediately follows. ***

Remark 2.4: The convergence in Theorem 2.3 and Theorem 2.4 is the convergence in the real projective space PR^{n-1}. Hence the correction may end up $(-\hat{x}(v+h), \lambda(v+h))$ instead of $(\hat{x}(v+h), \lambda(v+h))$. However, it has no effect in our algorithm at all since

$$\begin{aligned}\dot{\lambda}(v+h) &= x(v+h)^T (A-D)\hat{x}(v+h) \\ &= -\hat{x}(v+h)^T (A-D)(-\hat{x}(v+h)).\end{aligned}$$ ***

§2.3: Checking

Even though we find $(\hat{x}(w), \lambda(w))$ from $(\hat{x}(v), \lambda(v))$ by prediction and correction, where $w = v + h$, $\lambda(w)$ may be on a different eigenvalue curve from what we expect.

There are two reasons for this failure. First, $\lambda^{(4)}(\bar{t})$ $(u \leq \bar{t} \leq w)$ is not known exactly. Second, the eigenvalue separation of $A(w)$ is not known. Fortunately there is a simple and accuract checking algorithm for tridiagonal eigenvalue problem, which can assure that if $\lambda(v)$ was the ith eigenvalue of $A(v)$, then $\lambda(w)$ is also the ith eigenvalue of $A(w)$.

The description of the checking algorithm is divided into two parts: Prediction Checking and Correction Checking.

(a) Prediction Checking

Since $A(t) = D + t(A - D)$ is symmetric for each $t \in [0,1]$, the eigensystem forms a complete

orthonormal system for each t. Hence the eigensystem may be considered as rotating continuously from $t = 0$ to $t = 1$. If the rotation of the eigenvector is too big in two consecutive steps, we might have jumped into another eigenvalue curve.

After eigenvector prediction, we have $\hat{x}_0(w)$. And we hope that the rotation between $\hat{x}_0(w)$ and $\hat{x}(v)$ is not too big. So we have the following prediction checking algorithm:

Algorithm 5: Prediction Checking Algorithm

Let CRI be a positive real number.

(1) Calculate $IP = \hat{x}(v)^T \hat{x}_0(w)$.

(2) If $(|IP| > CRI)$ THEN

Accept $\hat{x}_0(w)$.

ELSE

Cut the current step-size by half.

Go back to eigenvalue prediction.

END IF. ***

Remark 2.5:

(1) The number CRI actually restricts the rotation angle between the eigenvectors.

In our program, we set CRI = .85. It means that no more than $\frac{\pi}{6}$ eigenvector rotation is allowed in two consecutive steps.

(2) If $|IP| \leq$ CRI, then indeed we might have jumped into another eigenvalue curve. However $|IP| >$ CRI does not necessarily imply that $\lambda(w)$ is on the right eigenvalue curve. To be safe, we devise a correction checking algorithm as follows.

(b) Correction Checking

To state correction checking algorithm, we need some background material.

Let $T =$

$$\begin{bmatrix} \alpha_1 & \beta_2 & & & & \\ \beta_2 & \alpha_2 & \ddots & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & \beta_n \\ & & & & \beta_n & \alpha_n \end{bmatrix}$$

where $\beta_i \neq 0$ for $i = 2,3,\ldots,n$.

The characteristic polynominals $P_i(x)$ of the principle minor formed by the first $i \times i$ submatrix of the matrix T satisfy the following recursive relations:

$$P_0(\lambda) = 1$$

$$P_1(\lambda) = \alpha_1 - \lambda$$

$$P_i(\lambda) = (\alpha_i - \lambda)P_{i-1}(\lambda) - \beta_i^2 P_{i-2}(\lambda)$$

for $i = 2,3,\ldots,n$.

Then the roots of P_i strictly separate those of $P_{i-1}(1 \leq i \leq n-1)$ [12]. We call this property of $P_0, P_1, \ldots, P_n$ the Strum Sequence Property.

The Sturm Sequence Property for symmetric tridiagonal matrices with non-zero off diagonal elements forms the basis of the following Theorem.

Theorem 2.5: Let $G(\mu)$ be the number of sign changes of $P_0(\mu), \ldots, P_n(\mu)$ at location μ. Then $G(\mu)$ is the number of roots of $P_n(\lambda)$ with $\lambda < \mu$.

Proof: See [12]. ***

Remark 2.6: If $P_i(\mu) = 0$, we take the sign of $P_i(\mu)$ as that of $P_{i-1}(\mu)$. Note that no two consecutive $P_i(\mu)$ can be zero. ***

Let $Q_i(\lambda) = P_i(\lambda)/P_{i-1}(\lambda)$ $(i = 1,2,\ldots,n)$. Then $Q_i(\lambda)$ satisfies the following relations:

$$Q_1(\lambda) = \alpha_1 - \lambda$$

$$Q_i(\lambda) = (\alpha_i - \lambda) - \beta_i^2/Q_{i-1}(\lambda) \quad (i = 2,\ldots,n)$$

and $G(\mu)$ is given by the number of negative $Q_i(\mu)$. We thus have the Sturm Sequence Algorithm.

Algorithm 6: Strum Sequence Algorithm

(1) Set COUNT = 0

(2) Calculate $Q_1(\mu) = \alpha_1 - \mu$

IF $i = 2,3,\ldots,n$, repeat the following.

(3) IF $(Q_{i-1}(\mu) = 0)$ $Q_{i-1}(\mu) = |\beta_i|$ MACH, where MACH is the smallest number for which $1 + \text{MACH} > 1$ on the computer.

(4) Calculate $Q_i(\mu) = (\alpha_i - \mu) - \beta_i^2/Q_{i-1}(\mu)$.

(5) If $(Q_i(\mu) < 0)$ COUNT = COUNT + 1.

(6) IF $(i = n)$ THEN

Set $G(\mu)$ = COUNT

ELSE

i = i + 1

Go to (3)

END IF. ***

Remark 2.7:

(1) $G(\mu)$ is the number of eigenvalues of T which is less than μ.

(2) When $Q_{i-1}(\mu) = 0$, replacing $Q_{i-1}(\mu)$ by $|\beta_i|$ MACH (note that this is positive: this is essential because if $Q_{i-1}(\mu) = 0$,

it is treated as positive) is equivalent to replacing α_{i-1} by $\alpha_{i-1} + |\beta_i|$ MACH. Hence this algorithm is stable.

(3) For the detailed discussion of this algorithm see [6]. ***

Algorithm 6 can be used to check whether $\lambda(w)$ is on the right eigenvalue curve. By the prediction and correction step, we have the sequence $\{(\hat{x}_j(w), \lambda_j(w))\}_{j=1}^{k}$, where $\|y_k(w)\|$ is big enough [Algorithm 4]. And the pair $(\hat{x}_k(w), \lambda_k(w))$ is considered as the ith eigenpair of $A(w)$. At this stage there are two possibilities:

(1) $\hat{x}_{k-1}(w)^T \hat{x}_k(w) > 0$

or

(2) $\hat{x}_{k-1}(w)^T \hat{x}_k(w) < 0$.

Since $\hat{x}_{k-1}(w)$ and $\hat{x}_k(w)$ have almost the same direction, $\hat{x}_{k-1}(w)^T \hat{x}_k(w)$ cannot be 0.

Theorem 2.6: Suppose that $\lambda_k(w)$ and $\hat{x}_{k-1}(w)$ are sufficiently close to $\lambda(w) = \lambda^i(w)$ and $\hat{x}(w) = \hat{x}^i(w)$, respectively. Then,

(1) If $\hat{x}_{k-1}(w)^T \hat{x}_k(w) > 0$, then $\lambda_k(w) > \lambda(w)$.

(2) If $\hat{x}_{k-1}(w)^T \hat{x}_k(w) < 0$, then $\lambda_k(w) < \lambda(w)$.

Proof: It is sufficient to prove (1). Let $\{\hat{x}^1(w), \hat{x}^2(w), \ldots, \hat{x}^n(w)\}$ be the set of n orthonormal eigenvectors of $A(w)$. Step (3) in Algorithm 4 with $j = k-1$ can be written as:

$$[\lambda_k(w) I - A(w)]\, y_k(w) = \hat{x}_{k-1}(w).$$

Let

$$\hat{x}_{k-1}(w) = \sum_{j=1}^{n} g_j \hat{x}^j(w),$$

where

$$g_j = \hat{x}_{k-1}(w)^T \hat{x}^j(w) \quad (j = 1,2,\ldots,n). \tag{2.1}$$

Since $\hat{x}_{k-1}(w)$ is close to $\hat{x}^i(w)$ by assumption, g_i is not small. Then,

$$\begin{aligned} y_k(w) &= [\lambda_k(w) I - A(w)]^{-1} \hat{x}_{k-1}(w) \\ &= [\lambda_k(w) I - A(w)]^{-1} \sum_{j=1}^{n} g_j \hat{x}^j(w) \\ &= \sum_{j=1}^{n} \frac{g_j}{\lambda_k(w) - \lambda^j(w)} \hat{x}_j(w) \end{aligned}$$

From (2.1), we obtain

$$\begin{aligned} \hat{x}_{k-1}(w)^T y_k(w) &= \sum_{j=1}^{n} \frac{g_j^2}{\lambda_k(w) - \lambda^j(w)} \\ &\simeq \frac{g_i^2}{\lambda_k(w) - \lambda^i(w)}, \end{aligned}$$

for $\lambda_k(w)$ is, by assumption, sufficiently close to $\lambda^i(w)$, and g_i is not small.

Because $y_k(w)$ and $\hat{x}_k(w)$ have the same direction, $\hat{x}_{k-1}(w)^T \hat{x}_k(w) > 0$ implies $\hat{x}_{k-1}(w)^T y_k > 0$. It follows that $\lambda_k(w) > \lambda^i(w) = \lambda(w)$. ***

Theorem 2.6 and the Sturm Sequence Property lead to the following correction checking algorithm.

Algorithm 7: Correction Checking Algorithm

Suppose that $(\hat{x}(0), \lambda(0))$ is the ith eigenpair of the initial matrix D.

IF $(\hat{x}_{k-1}(w)^T \hat{x}_k(w) > 0)$ THEN

IF $(G(\lambda_k(w)) = i)$ THEN

Accept $(\hat{x}_k(w), \lambda_k(w))$.

ELSE

Cut the current step-size by half

Go back to eigenvalue prediction

END IF

ELSE

IF $(G(\lambda_k(w)) = i - 1)$ THEN

Accept $(\hat{x}_k(w), \lambda_k(w))$

ELSE

Cut the current step-size by half

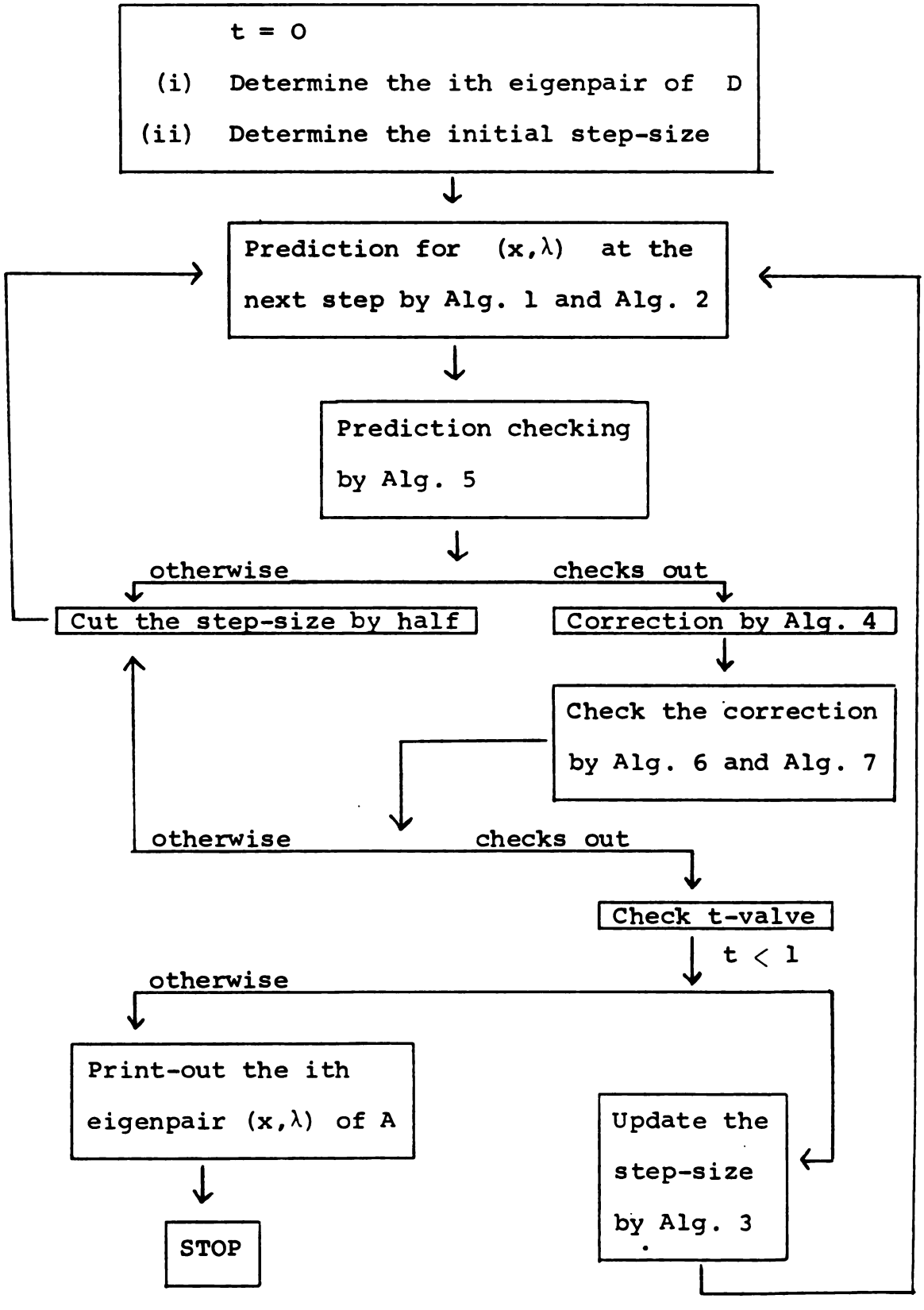
Go back to eigenvalue prediction

END IF

END IF. ***

Remark 2.8: The usage of Sturm Sequence in Algorithm 7 is quite different from that of finding eigenvalues by the method of bisection. The convergence process of the bisection is linear with convergence rate 0.5. But the Strum Sequence Algorithm is used only one time in Algorithm 7 to check whether $\lambda_k(w)$ is on the right eigenvalue curve. ***

§ 2.4: Flow Chart



CHAPTER III: NUMERICAL RESULTS

The computations described in this chapter were performed on the CDC Cyber 750 at Michigan State University.

This chapter is divided into two parts: Examples and Comparison with EISPACK.

§3.1: Examples

Two examples will be presented in this section.

Example 1:

$$\text{Let } A = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \beta_n \\ & & & \beta_n & \alpha_n \end{bmatrix} = \begin{bmatrix} 1 & 1 & & & \\ 1 & 2 & 1 & & \\ & 1 & 3 & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 1 & n \end{bmatrix}$$

That is, $\alpha_i = i \quad (i = 1,2,\ldots,n)$ and $\beta_i = 1 \quad (i = 2,3,\ldots,n)$. This kind of matrices arises in connection with a study of the John-Teller effect [5,11].

The matrix A has the ith eigenvalue nearby i except for few smallest and largest eigenvalues. Hence A has a good eigenvalue separation.

The homotopy algorithm was used to find the first eigenpair of A with n = 20,40, and 80. For each n, we used three different initial matrices:

$$D_1 = \begin{bmatrix} \alpha_1 & & & \\ & \alpha_2 & & \\ & & \ddots & \\ & & & \alpha_n \end{bmatrix}$$

$$D_2 = \begin{bmatrix} \alpha_1 & \beta_2 & & & & \\ \beta_2 & \alpha_2 & & & & \\ & & \alpha_3 & \beta_4 & & \\ & & \beta_4 & \alpha_4 & & \\ & & & & \ddots & \\ & & & & \alpha_{n-1} & \beta_n \\ & & & & \beta_n & \alpha_n \end{bmatrix}$$

$$D_3 = \begin{bmatrix} \alpha_1 & \beta_2 & & & & \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \beta_3 & \alpha_3 & \beta_4 & & \\ & & \beta_4 & \alpha_4 & & \\ & & & & \alpha_5 & \\ & & & & & \alpha_6 \\ & & & & & & \ddots \\ & & & & & & & \alpha_n \end{bmatrix}$$

The first eigenvalue of D_1 is clearly 1 and the corresponding eigenvector is $(1,0,0,\ldots,0)^T$. The first eigenpair of D_2 is easily available since the first eigenpair is essentially the first eigenpair of the first 2×2 submatrix. The first eigenvalue of D_2 is .381966 and the corresponding eigenvector is $(.850651, -.525731,0,\ldots,0)^T$. The first eigenpair of D_3 is essentially the first eigenpair of upper 4×4 matrix and it was found by the EISPACK subroutine IMTQL2. The first eigenvalue of D_3 is .254719 and the corresponding eigenvector is $(.777951, -.579792, .233949, -.0624651, 0,0,\ldots,0)^T$. Here we are taking advantage of the nice performance of EISPACK with small matrices.

We chose the error tolerance $\epsilon = 1E-2$, and corrected eigenpair until $\|y_j\| \geq \epsilon^{-2} = 1E+4$ if $t < 1$, and $\|y_j\| \geq 1E+10$ if $t = 1$. (For notation, see Algorithm 3 and Algorithm 4). We obtain the following results.

	STEPS	L-SYSTEMS	MAXERR	FAILURES
D_1	4	8	.8643E-2	0
D_2	2	4	.9422E-2	0
D_3	1	2	.9067E-5	0

Table 3.1: n = 20

Notations 3.1

STEPS: The total number of steps to find the first eigenpair of A, where a step means a loop of prediction, correction and checking.

L-SYSTEMS: The total number of linear systems solved, (see Algorithm 2 and Algorithm 4).

MAXERR: The maximum error in eigenvalue predictions.

FAILURES: The total number of failures in either prediction checking or correction checking, where a failure means the case in which we need to cut the step-size by half and predict eigenvalue again, (see Algorithm 5 and Algorithm 7). ***

Remark 3.1:

(1) The error control was completely satisfactory.

(2) We keep track of the total number of linear systems solved, since it is the most time consuming process in the homotopy algorithm.

(3) For n = 40 and 80, we obtained almost the same results. Hence the growth of n did not make the first eigenvalue curve ill conditioned.

(4) D_2 gives much better conditioning to the first eigenvalue curve than D_1. D_3 gives still better conditioning than D_2. ***

A similar idea can be used to find the other eigenpair. For example, suppose that we want to find the 50th eigenpair of A. Now we choose D_4 as follows:

$$D_4 = \begin{bmatrix} \alpha_1 & & & & & & & & & \\ & \ddots & & & & & & & & \\ & & \alpha_{47} & & & & & & & \\ & & & \alpha_{48} & \beta_{49} & & & & & \\ & & & \beta_{49} & \alpha_{49} & \beta_{50} & & & & \\ & & & & \beta_{50} & \alpha_{50} & \beta_{51} & & & \\ & & & & & \beta_{51} & \alpha_{51} & \beta_{52} & & \\ & & & & & & \beta_{52} & \alpha_{52} & & \\ & & & & & & & & \alpha_{53} & \\ & & & & & & & & & \ddots \\ & & & & & & & & & \alpha_{80} \end{bmatrix}$$

The 50th eigenpair of D_4 is essentially one of the eigenpairs of the middle 5×5 submatrix, and it was found by IMTQL2. The 50th eigenvalue of D_4 is 50 and the corresponding eigenvector is

$$(0,0,\ldots,0,\underset{48\text{th}}{.301511},\underset{49\text{th}}{.603023},\underset{50\text{th}}{.301511},\underset{51\text{th}}{-.603023},\underset{52\text{th}}{.301511},0,\ldots,0)^T.$$

To find the 50th eigenpair of A, we needed only one step, and the solution of one linear system. Other eigenpairs can be handled in a similar fashion.

Remark 3.2: The above discussion shows that we can make each eigenvalue curve in Example 1 have almost the same conditioning by a suitable choice of the initial matrix D.

Example 2: The following example is from [11].

	α_i	β_i	λ^i
1	.25000 0000	.00000 0000	.06437 9910
2	.76849 1173	.15366 0746	.07359 7119
3	.91955 6756	.46726 0328	.08422 5269
4	.23093 8895	.11925 6498	.09720 9219
5	.13305 3788	.08076 3539	.10321 5761
6	.22254 9575	.03394 7196	.12278 7524
7	.11612 7856	.03609 0904	.14342 2880
8	.12033 9373	.03502 2375	.16632 4602
9	.12371 9912	.02915 7561	.17130 7560
10	.12856 1407	.03745 3705	.17735 6337
11	.10776 8089	.01609 0599	.23163 9484
12	.13703 9203	.02382 6467	.26773 3297
13	.13805 7030	.02946 8449	.46276 6202
14	.10379 6943	.00764 6394	1.33403 4844

Table 3.2

where α_i is a diagonal element $(i = 1,2,\ldots,14)$

β_i is an off-diagonal element $(i = 2,3,\ldots,14)$

λ^i is the ith eigenvalue $(i = 1,2,\ldots,14)$.

Remark 3.3: The eigenvalue separation is poor. For example $\lambda^5 - \lambda^4 = .0060065$. ***

We choose the initial matrix D as follows:

$$D = \begin{bmatrix} \alpha_1 & \beta_2 & & & & & & & & & & & & \\ \beta_2 & \alpha_2 & \ddots & & & & & & & & & & & \\ & \ddots & \ddots & \beta_5 & & & & & & & & & & \\ & & \beta_5 & \alpha_5 & & & & & & & & & & \\ & & & & \alpha_6 & \beta_7 & & & & & & & & \\ & & & & \beta_7 & \alpha_7 & \ddots & & & & & & & \\ & & & & & \ddots & \ddots & \beta_{10} & & & & & & \\ & & & & & & \beta_{10} & \alpha_{10} & & & & & & \\ & & & & & & & & \alpha_{11} & \beta_{12} & & & & \\ & & & & & & & & \beta_{12} & \alpha_{12} & \beta_{13} & & & \\ & & & & & & & & & \beta_{13} & \alpha_{13} & \beta_{14} & & \\ & & & & & & & & & & \beta_{14} & \alpha_{14} & & \end{bmatrix}$$

The choice of D was guided by Theorem 1.6. Since β_6 is less than β_5 and β_7, and β_{11} is less than β_{10} and β_{12}, by the above choice of D, $\|A - D\|$ is minimized and D has reasonably sized sub-blocks. We write

$$D = \begin{bmatrix} U & & \\ & M & \\ & & L \end{bmatrix}$$

and found all eigenpairs of U,M and L by using EISPACK subroutine IMTQL2. We chose the error tolerance ϵ = 1E - 4 and corrected the eigenpair until $\|y_j\| \geq \epsilon^{-2}$ = 1E + 8 if t < 1, and $\|y_j\| \geq$ 1E + 10 if t = 1. We obtained the following result.

i	$\lambda^i(0)$	STEPS	L-SYSTEMS	MAXERR	FAILURES	$\lambda^i(1)$
1	.06634 0723	2	5	1.0880E-4	0	.06437 9909
2	.07878 7312	2	6	1.4849E-4	0	.07359 7119
3	.08852 7020	4	9	.8085E-4	0	.08422 5268
4	.09408 9982	4	10	.8073E-4	0	.09720 9219
5	.10255 7223	2	5	.7869E-4	0	.10321 5760
6	.12351 1014	2	5	.8523E-4	0	.12278 7523
7	.14116 5599	2	4	.9097E-4	0	.14342 2879
8	.16767 9729	2	4	.9339E-4	0	.16632 4601
9	.17206 4026	3	7	.7269E-4	0	.17130 7559
10	.17497 7810	3	7	.7432E-4	0	.17735 6336
11	.23472 2400	3	7	.8802E-4	0	.23163 9484
12	.25880 1504	3	6	.8933E-4	0	.26773 3296
13	.46273 7255	1	2	.4505E-7	0	.46276 6202
14	1.33403 4811	1	1	0	0	1.33403 4842

Table 3.3

Remark 3.4:

(1) Example 2 was chosen to demonstrate the accurate execution of the homotopy algorithm. As we started from the ith eigenpair of D, we arrived at the ith eigenpair of A for each i = 1,2,...,14. There was no FAILURE at all in the checking process.

(2) The average number of STEPS is 2.4286 and the average number of L-SYSTEMS is 5.5714, to find one eigenpair of A. Thus we need to solve an average of 2.2941 linear systems in each STEP. This is typical of our homotopy algorithm. ***

Example 3: The following example is from [8, page 125].

Let A =

$$\begin{bmatrix} 7 & 1 & & & & & & \\ 1 & 6 & \ddots & & & & & \\ & 1 & \ddots & \ddots & & & & \\ & & \ddots & 1 & \ddots & & & \\ & & & \ddots & 0 & \ddots & & \\ & & & & \ddots & 1 & \ddots & \\ & & & & & \ddots & \ddots & \ddots \\ & & & & & & \ddots & 6 \ 1 \\ & & & & & & & 1 \ 7 \end{bmatrix}$$

That is, $\alpha_i = 8 - i$ $(i = 1,2,\ldots,8)$.

$$\alpha_i = i - 8 \quad (i = 9,10,\ldots,15)$$

and

$$\beta_i = 1 \quad (i = 2,3,\ldots,15).$$

$$\begin{aligned}\sigma(A) = \{&-1.125441522005,\ .253805837119,\\ &.947534612211,\ 1.789326378193,\\ &2.130221682144,\ 2.961274130561,\\ &3.043336908165,\ 4.000000000000,\\ &4.008304183180,\ 5.038725869439,\\ &5.039166155057,\ 6.210673621807,\\ &6.210683778125,\ 7.746194162881,\\ &7.746194203123\}.\end{aligned}$$

We tried to locate the biggest eigenvalue of A by using homotopy methods starting from the initial matrix

$$D = \begin{bmatrix} 7 & 1 & & & & & & & & & & & & & \\ 1 & 6 & 1 & & & & & & & & & & & & \\ & 1 & 5 & 1 & & & & & & & & & & & \\ & & 1 & 4 & 3 & & & & & & & & & & \\ & & & & & 2 & & & & & & & & & \\ & & & & & & 1 & & & & & & & & \\ & & & & & & & 0 & & & & & & & \\ & & & & & & & & 1 & & & & & & \\ & & & & & & & & & \ddots & & & & & \\ & & & & & & & & & & & & & & 7 \end{bmatrix}$$

The biggest eigenvalue of D is 7.745281240174 and the corresponding eigenvector is (.7779570546743, .579791948271, .233949463778, .062465125788, 0,...,0)^T.

We chose the error tolerance ϵ = 1E-4, and corrected eigenpair until $\|y_j\| \geq \epsilon^{-2}$ = 1E+8 if $t < 1$, and $\|y_j\| \geq$ 1E+10 if $t = 1$. However the second largest eigenvalue of A agrees the largest eigenvalue of A up to 7 decimal places. Hence we may expect some failures in correction checking.

Indeed 7 failures in correction checking were observed until the homotopy algorithm located the largest eigenvalue of A. Even if the homotopy algorithm could locate the largest eigenvalue of A, many failures made it inefficient.

Remark 3.5: Many failures in correction checking occurred due to the fact that $\lambda^{14}(1)$ and $\lambda^{15}(1)$ are very close. Hence if A has very close algorithm in this thesis can be inefficient. Hence we need to develop another homotopy algorithm for the case in which A has very close eigenvalues. This will be one of the further research topics.

§3.2: Comparison With EISPACK

In this section we consider Example 1 with the choice of D_3 as the initial matrix. We also found all the eigenpairs of A in Example 1 with n = 20, 40,80 by using IMTQL2 subroutine in EISPACK.

IMTQL2 determines the eigenvalues and eigenvectors of a symmetric tridiagonal matrix. IMTQL2 uses the implicit QL method to compute the eigenvalues and accumulates the QL transformations to compute eigenvectors [9].

(a) Execution Time

We obtained the following results:

	1	2	3	4	5	6	7
n	(sec) H_n	(sec) E_n	$\frac{H_{2n}}{H_n}$	$\frac{E_{2n}}{E_n}$	$\log_2\frac{H_{2n}}{H_n}$	$\log_2\frac{E_{2n}}{E_n}$	$\frac{E_n}{n\cdot H_n}$
10	.000375*	.004	——	——	——	——	1.0667
20	.00075*	.026	——	6.5000	——	2.7004	1.7333
40	.0015*	.196	——	7.5385	——	2.9143	3.2667
80	.003	1.226	——	6.2551	——	2.6450	5.1083
160	.006	7.993	2.0000	6.5196	1	2.7048	8.3260
320	.011	54.988	1.8333	6.8795	.8744	2.7823	15.6216
640	.024	——	2.1818	——	1.1255	——	——
1280	.047	——	1.9583	——	.9696	——	——

Table 3.4

Notation 3.2

H_n: The execution time in seconds to find the first eigenpair of $n \times n$ matrix in Example 1 by the homotopy algorithm with the initial matrix D_3.

E_n: The execution time in seconds to find all eigenpairs of the $n \times n$ matrix in Example 1 by using IMTQL2.

Remark 3.6:

(1) In Column 1 * means estimated time. ($n = 10,20,40$, actual measurement of H_n is not reliable since H_n is too short).

(2) In Column 2, E_{640} and E_{1280} become too long (that is, too expensive).

(3) Column 7 shows that if $n \geq 20$, the homotopy algorithm is better than EISPACK even with one CPU. As n gets bigger the homotopy algorithm becomes increasingly better.

(b) Computational Complexity

Column 5 in Table 3.4 shows that the complexity of following one eigenvalue curve is $O(n)$. Hence the complexity of finding all eigenpairs of a matrix by the homotopy algorithm is $O(n^2)$ by assuming that each eigenvalue curve has more or less the same

conditioning, which is true in this Example [Remark 3.2]. Therefore, if we adopt parallel processing the complexity of the homotopy algorithm is better than $O(n^2)$. If we assume that there are n parallel processors, then the complexity becomes $O(n)$.

However column 6 in Table 3.4 shows that the complexity of IMTQL2 is worse than $O(n^{2.6})$.

Remark 3.7: The discussions in (a) and (b) are not general assertions. They are based on Example 1. Hence the discussions in (a) and (b) are true for matrices which have good eigenvalue separations. ***

(c) Storage Consideration

In IMTQL2, Z is a real two dimensional variable with row dimension at least n and column dimension at least n. If the eigenvectors of the symmetric tridiagonal matrix are desired, then on input, Z contains the identity matrix of order n, and on output the orthonormal eigenvectors of this tridiagonal matrix. Hence to use IMTQL2, at least n^2-storage is needed [9]. For the homotopy algorithm storage needed equals only a few multiples of n. According to our program the storage requirement of the homotopy algorithm is around 20n.

REFERENCES

[1] Batterson, Steve and Smillie, John, "The Dynamics of Rayleigh Quotient Iteration", Preprint.

[2] Blattner, J.W. (1962), "Bordered Matrices", J. Soc. Indust. Appl. Math. 10: p. 528-536.

[3] Chu, M.T. (1984), "A simple application of the homotopy method to symmetric eigenvalue problems", Linear Algebra Appl. 59: p.85-90.

[4] Li, T.Y., Sauer, T. and Yorke, J., "Numerical solution of a class of deficient polynomial systems", to appear.

[5] Longuet-Higgins, H.C., Opik, U., Pryce, M.H.L. and Sack, R.A. (1958), "Studies of the John-Teller effect", Preceedings of the Royal Society Vol. 244: p.1-16.

[6] Martin, R.S. and Wilkinson, J.H. (1967), "Calculation of the eigenvalues of a symmetric tridiagonal matrix by the method of bisection", Numerical Math. 9: p.386-393.

[7] Ortega, J.M. (1972), "Numerical Analysis: a second course", New York. Academic Press.

[8] Parlett (1980), "The Symmetric Eigenvalue Problem", Englewood Cliffs, N.J. Prentice-Hall.

[9] Smith, B.T., et. al (1976), "Matrix Eigensystem Routines, EISPACK Guide", 2nd Edition, Springer-Verlag, New York.

[10] Store, J. and Bulirsch, R. (1980), "Introduction to Numerical Analysis", Springer-Verlag, New York Inc.

[11] Wilkinson, J.H. (1958), "The calculation of the eigenvectors of codiagonal matrices", Computer Journal 1: p.90-96.

[12] Wilkinson, J.H. (1965), "The Algebraic Eigenvalue Problem", Oxford University Press, New York.