# ITERATIVE SOLUTION OF INDEFINITE SYMMETRIC SYSTEMS BY METHODS USING ORTHOGONAL POLYNOMIALS OVER TWO DISJOINT INTERVALS

Y. SAAD

Technical Report # 212
October 12, 1981

This work was supported in part by the U.S. Office of Naval Research under grant N000014-76-C-0277 and in part by NSF Grant.

## Abstract

It is shown in this paper that certain orthogonal polynomials over two disjoint intervals can be particularly useful for solving large symmetric indefinite linear systems or for finding a few interior eigenvalues of a large symmetric matrix. There are several advantages of the proposed approach over the techniques which are based upon the polynomials having the least uniform norm in two intervals. While a theoretical comparison will show that the norms of the minimal polynomial of degree n in the least squares sense differs from the minimax polynomial of the same degree by a factor not exceeding $2(n+1)^{1/2}$, the least squares polynomials are by far easier to compute and to use thanks to their three term recurrence relation. A number of suggestions will be made for the problem of estimating the optimal parameters and several numerical experiments will be reported.

## 1. Introduction

This paper is concerned with the solution of large indefinite linear systems $A x = f$. The number of effective iterative methods for treating such problems is limited. An obvious possibility is to solve the normal equations $A^2 x = A f$ by any available method for solving positive definite systems but this is not recommended in general as it increases the amount of work and above all squares the original condition number. Perhaps the best known contribution in the field was made by Paige and Saunders who adapted the Conjugate Gradient method to indefinite systems [8]. There have also been a few attempts towards the generalization of the Chebyshev iteration algorithm (also called Stiefel Iteration) to the indefinite case by Lebedev [6], Roloff [10] and de Boor and Rice [4]. These works focus on the difficult problem of finding a polynomial of degree n such that $p_n(0)=1$ and having least uniform norm in two intervals containing the eigenvalues of A but not the origin. Like in the positive definite case, such polynomials are the best for making the residual norm small in a certain sense in the Richardson iteration:

$$x_{n+1}=x_n+\tau_n^{-1} r_n$$

where $\tau_n$ is the $n^{th}$ root of the polynomial and $r_n$ is the residual of $x_n$. There are, however, some serious drawbacks of this approach one of the most important being that the mini-max polynomial is rather difficult to obtain numerically. Moreover the above Richardson iteration is unstable in general.

At this point one might ask whether it is essential to use the minimax polynomial. Perhaps the most attractive property of the Chebyshev polynomials is its three term recurrence formula which enables one to generate, in the positive definite case, a sequence of approximate solutions to the linear system which satisfy a three term recurrence. As pointed out by de Boor and Rice [4], it seems

unfortunately impossible to exhibit a three term recurrence for the minimax polynomials over two intervals.

Nevertheless a natural alternative considered in this paper is to construct a sequence of polynomials satisfying $p_n(0)=1$ and which minimize an $L_2$ norm associated with a certain weight function over the two intervals. Such polynomials are easy to compute and can be obtained via a sequence of polynomials satisfying a three term recurrence. Moreover it will be shown that their uniform norm in the two intervals is comparable within a factor of $2(n+1)^{1/2}$ with the norm of the minimax polynomial if n is the degree of both polynomials. We will call Generalized Chebyshev Iteration (GCI) the iterative method which at each step yields a residual vector of the form $r_n=p_n(A)r_0$ where $p_n$ is the minimal polynomial. There are several other applications of the minimal polynomial in addition to the Generalized Chebyshev Iteration. First, a block version of the algorithm can be formulated. It is essentially a generalization of the block Stiefel algorithm described in [11] and is particularly suitable for parallel computation. Secondly there are various ways of using the minimal polynomials for computing eigenvectors associated with interior eigenvalues. For example one can easily adapt the subspace iteration method (see [9]) to interior eigenvalues. A few of the possible generalizations and extensions of the basic Generalized Chebyshev Iteration will be briefly outlined but we will not attempt to describe them in detail.

Section 2 describes polynomial iteration for solving linear systems. In section 3 we will deal with the orthogonal polynomials over two intervals and will introduce the minimal polynomial. Section 4 covers the application of the minimal polynomials to the solution of indefinite systems while section 5 outlines their application to the interior eigenvalue problems. In section 6 a number of practical details, concerning in particular the estimation of the optimal

parameters, are given. Finally some numerical experiments are reported in section 7 and a tentative conclusion is drawn in the last section.

## 2. Polynomial iteration for linear systems.

Consider the N x N linear system

$$A\,x = f \tag{2.1}$$

where A is a nonsingular symmetric matrix. Let $x_0$ be any initial guess of the solution x and $r_0$ the corresponding initial residual $r_0 = f - A\,x_0$. Polynomial iteration methods are methods that provide approximate solutions $x_n$ whose residuals $r_n$ satisfy:

$$r_n = p_n(A)\,r_0 \tag{2.2}$$

where $p_n$ is a polynomial of degree n, often called the residual polynomial. We only consider methods in which $x_n$ is given by

$$x_{n+1} = x_n + \alpha_n u_n \tag{2.3}$$

where $u_n$, the direction of search, is a linear combination of the previous residuals. In this case it can easily be shown by induction that the polynomial $p_n$ satisfies the condition $p_n(0) = 1$.

If $r_0$ is written in the eigenbasis $\{u_i\}_{i=1,N}$ of A as

$$r_0 = \sum_{i=1}^{N} \xi_i u_i$$

then obviously the Euclidean norm of $r_n$ is

$$||r_n|| = \left[ \sum_{i=1}^{N} p_n(\lambda_i)^2\, \xi_i^2 \right]^{1/2} \tag{2.4}$$

where the $\lambda$ ' s are the eigenvalues of A.

Equation 2.4 suggests the general principle that if we want the residual norm $||r_n||$ to be small, we must find a polynomial satisfying $p_n(0) = 1$ which is small in some set containing the eigenvalues of A. One way of choosing a residual polynomial which is small in a set containing the spectrum is to take among the polynomials satisfying $p_n(0) = 1$ the one which has the least uniform norm in that set. When the matrix A is positive definite its eigenvalues can be included in one single interval [a,b] not containing the origin and the polynomial having least uniform norm in [a,b] can easily be expressed in terms of the Chebyshev polynomials. This is the essence of the Chebyshev semi iterative method [5].

When A is not positive definite, i.e. when it has both positive and negative eigenvalues, then its spectrum can be included in two disjoint intervals [a,b], [c,d] with $b < 0 < c$ , and the polynomial having having least uniform norm in [a,b] U [c,d] can no longer be easily expressed. Lebedev [6] considered this problem and gave an explicit solution to it for the very particular case when the two intervals [a,b] and [c,d] have the same length. Lebedev does not however give any suggestion for the general case and de Boor and Rice [4] showed how to practically obtain the polynomial of minimal uniform norm in two disjoint intervals. There remains however several serious difficulties with the use of these minimax polynomials which render the method unattractive and unreliable. Firstly the computation of the minimax polynomial can be particularly difficult and time consuming. Secondly once the minimax polynomial is available one must perform the Richardson iteration:

$$x_{i+1} = x_i + \tau_i r_i \tag{2.5}$$

where the $\tau$ 's are the inverses of the roots of the minimax polynomial. These roots should therefore be computed before 2.5 is started but this can be done at a negligible cost. The main disadvantage of 2.5, however, lies in its unstability. This can easily be understood by deriving from 2.5 the equivalent equation for the

residual vectors

$$r_{i+1}=r_i - \tau_i A r_i = (I - \tau_i A) r_i$$

which shows that

$$r_{i+1} = (I-\tau_{i+1} A)(I - \tau_i A)\ldots(I-\tau_0 A)\, r_0 \qquad (2.6)$$

Although the final residual $r_n$ should be small in exact arithmetic, the intermediate residuals $r_i$ can be very large as is easily seen from 2.6 and this may cause unstability in iteration 2.5. Reordering of the parameters $\tau$ might be used to achieve a better stability [1] although this does not seem to constitute a definitive remedy. Note also that the computation of the roots of the minimal polynomial can be itself an unstable process. Finally another drawback with the use of the minimax polynomial is that if a polynomial of degree m is used then the intermediate vectors $x_i$ with $i<m$ do not approximate the exact solution x. Only the last vector $x_m$ does in theory. The implication of this is that one cannot stop the process and test for convergence before the m steps are all performed. If the accuracy is unsatisfactory after the m steps, one has the alternatives of either recomputinng another polynomial with different degree and restart 2.5 or perform again m steps of iteration 2.5 with the same polynomial.

It will be seen that all of the difficulties mentioned above can be avoided by using polynomials that are small in the two intervals in the least squares sense i.e. polynomials which minimize the $L_2$ norm with respect to some weight function w(x). This brings up orthogonal polynomials in two disjoint intervals.

## 3. Orthogonal polynomials in two Disjoint Intervals

### 3.1. Basic definitions and notation

In this subsection we will describe general orthogonal polynomials in two disjoint intervals. Let (a,b), (c,d) be two open intervals such that $b < 0 < c$. The assumption that the origin lies between the two intervals is not essential for this subsection. We consider a weight function w(x) on the interval (a,d) defined by

$$w(x) = \begin{cases} w_1(x) & \text{for } x \in (a,b) \\ w_2(x) & \text{for } x \in (c,d) \\ 0 & \text{for } x \in (b,c) \end{cases}$$

where $w_1(x)$ and $w_2(x)$ are two nonnegative weight functions on the intervals (a,b) and (c,d) respectively. The inner product associated with w(x) will be denoted by $\langle .,. \rangle$, i.e.:

$$\langle f,g \rangle = \int_a^d f(x)g(x)w(x)dx$$

or:

$$\langle f,g \rangle = \int_a^b f(x)g(x)w_1(x)dx + \int_c^d f(x)g(x)w_2(x)dx \qquad (3.1)$$

The norm associated with this inner product will be denoted by $||.||_C$ i.e. $||f||_C = \langle f,f \rangle^{1/2}$. We will often use the notation $\langle f,g \rangle_1$ and $\langle f,g \rangle_2$ for the first and the second term of the right member of 3.1 respectively.

The theory of orthogonal polynomials shows that there is a sequence $\{p_n\}$ of polynomials which is orthogonal with respect to $\langle .,. \rangle$. All of the properties of the orthogonal polynomials in one interval hold for the case of 2 intervals, provided we consider these polynomials as being orthogonal on (a,d) rather than on

(a,b)U(c,d). For example although we cannot assert that the roots are all in [a,b]U[c,d], we know that they belong to the interval [a,d]. Cases where one root is inside the interval (b,c) can easily be found. But the fondamental three term recurrence relation which holds for all sequences of orthogonal polynomials is obviously valid. Thus the polynomials $p_n$ satisfy a recurrence of the form:

$$\beta_{n+1}p_{n+1}(x) = (x-\alpha_n)p_n(x) - \beta_n p_{n-1}(x) \tag{3.2}$$

with

$$\alpha_n = \langle xp_n, p_n \rangle \tag{3.3}$$

$$\beta_{n+1} = ||\tilde{p}_{n+1}||_C = \langle \tilde{p}_{n+1}, \tilde{p}_{n+1} \rangle^{1/2} \tag{3.4}$$

where

$$\tilde{p}_{n+1}(x) = (x-\alpha_n)p_n(x) - \beta_n p_{n-1}(x) \tag{3.5}$$

One important problem considered in this paper is to find weight functions for which the coefficients $\alpha_i$, $\beta_i$ are easy to generate and to propose algorithms which generate them. Before entering into the details of this question in the next subsection let us briefly outline how these polynomials may be obtained. First select two weight functions $w_1$ and $w_2$ such that for each of the intervals (a,b) and (c,d) the orthogonal polynomials considered separately are known. Then express the orthogonal polynomials in (a,b)U(c,d) in terms of the orthogonal polynomials in (a,b) and in terms of the orthogonals polynomilas in (c,d). We thus obtain two sets of n+1 expansion coefficients expressing the polynomial $p_n$. Using these coefficients we are able to obtain the recurrence parameters $\alpha_n$ and $\beta_n$ of equation 3.2 by formulas 3.3 and 3.4. Hence the two sets of expansion coefficients can now be updated by use of 3.2 to yield the expansion coefficients of the next polynomial $p_{n+1}$ etc... Next the details of this procedure will be developed for a judicious choice of $w_1$ and $w_2$.

### 3.2. Generalized Chebyshev polynomials.

Let $c_1=(a+b)/2$, $c_2=(c+d)/2$ be the middles of the two intervals and $d_1=(b-a)/2$, $d_2=(d-c)/2$ their half widths. Consider the following weight functions:

$$w_1(x) = (2/\pi)\ [d_1^2 - (x-c_1)^2\ ]^{-1/2} \quad \text{for } x \in (a,b) \tag{3.6}$$

$$w_2(x) = (2/\pi)\ [d_2^2 - (x-c_2)^2\ ]^{-1/2} \quad \text{for } x \in (c,d) \tag{3.7}$$

The orthogonal polynomials in (a,b) with respect to the weight function $w_1$ can be found by a simple change of variables to be:

$$T_n\ [(x-c_1)/d_1\ ] \tag{3.8}$$

where $T_n$ is the Chebyshev polynomial of the first kind of degree n. Likewise in the second interval (c,d) the orthogonal polynomials with respect to $w_2$ are the polynomials:

$$T_n\ [(x-c_2)/d_2\ ] \tag{3.9}$$

We are interested in the polynomials $p_n(x)$ which are orthogonal in $(a,b)\cup(c,d)$ with respect to the weight function w(x) defined by $w_1$ in (a,b) and $w_2$ in (c,d). In order to generate these polynomials we simply express them in terms of the polynomials 3.8 in (a,b) and in terms of the polynomials 3.9 in (c,d) as follows:

$$p_n(x)= \sum_{i=0}^{n} \gamma_i^{(n)}\ T_i[(x-c_1)/d_1\ ] \tag{3.10}$$

$$p_n(x)= \sum_{i=0}^{n} \delta_i^{(n)}\ T_i[(x-c_2)/d_2\ ] \tag{3.11}$$

The following proposition will enable us to compute the coefficients $\alpha_n$ and $\beta_{n+1}$, given the $\gamma$'s and the $\delta$'s .

Proposition 3.1: Let p be any polynomial of degree n having the following expansions in (a,b) and (c,d) :

$$p(x)=\sum_{i=0}^{n} \gamma_i T_i[(x-c_1)/d_1] \quad \text{for } x \in (a,b) \tag{3.12}$$

$$p(x)=\sum_{i=0}^{n} \delta_i T_i[(x-c_2)/d_2] \quad \text{for } x \in (c,d) \tag{3.13}$$

Set:

$$\sigma_1 = 2\gamma_0^2 + \sum_{i=1}^{n} \gamma_i^2 \quad , \qquad \sigma_2 = 2\delta_0^2 + \sum_{i=1}^{n} \delta_i^2$$

$$\tau_1 = 2\gamma_1\gamma_2 + \sum_{i=1}^{n-1} \gamma_i\gamma_{i+1}, \qquad \tau_1 = 2\gamma_1\gamma_2 + \sum_{i=1}^{n-1} \gamma_i\gamma_{i+1}$$

then:

$$\langle p,p\rangle = \sigma_1 + \sigma_2 \tag{3.14}$$

$$\langle xp,p\rangle = c_1\,\sigma_1 + c_2\,\sigma_2 + d_1\tau_1 + d_2\tau_2 \tag{3.15}$$

Proof: Consider 3.14 first. From 3.1 and 3.12, 3.13 we have:

$$\langle p,p\rangle=\langle \sum_{i=1}^{n} \gamma_i T_i[(x-c_1)/d_1], \sum_{i=1}^{n} \gamma_i T_i[(x-c_1)/d_1]\rangle$$

Using the change of variable $y = (x-c_1)/d_1$ and the orthogonality of the Chebyshev polynomials we obtain

$$\langle p\ ,\ p\rangle_1 = 2\gamma_0^2 + \sum_{i=1}^{n} \gamma_i^2 = \sigma_1$$

By the same transformations we would obtain $\langle p,p\rangle_2 = \sigma_2$. Hence $\langle p,p\rangle = \langle p,p\rangle_1 + \langle p,p\rangle_2 = \sigma_1 + \sigma_2$ which is precisely 3.14.

In order to prove 3.15 consider $\langle xp,p\rangle_1$ and $\langle xp,p\rangle_2$ separately. We have:

$$\langle\ xp\ ,\ p\ \rangle_1 = d_1\langle\ \frac{[x-c_1]}{d_1}\cdot p,\ \ p\ \rangle_1 + c_1\langle\ p,\ \ p\ \rangle_1 \tag{3.16}$$

The first inner product can be transformed as follows:

$$\langle \frac{x-c_1}{d_1} p,\ p \rangle_1 = \langle \bar{\sum} \gamma_i \frac{x-c_1}{d_1} T_i[\frac{x-c_1}{d_1}1],\ \bar{\sum} \gamma_i\ T_i[\frac{x-c_1}{d_1}1] \rangle_1$$

$$= \frac{2}{\pi} \int_a^b \bar{\sum} \gamma_i \frac{x-c_1}{d_1} T_i[\frac{x-c_1}{d_1}] . \bar{\sum} \gamma_i\ T_i[\frac{x-c_1}{d_1}] \ dw_1(x$$

where $\bar{\sum}$ stands for $\sum_{i=0}^{n}$ . By the change of variables $y=\frac{x-c_1}{d_1}$ this simplifies into

$$\langle \frac{x-c_1}{d_1} 1\ p\ ,\ p \rangle_1 = \frac{2}{\pi} \int_{-1}^{+1} [\ \bar{\sum} \gamma_i\ y\ T_i(y)\ ]^2 (1-y^2)^{1/2}\ dy \qquad (3.17)$$

Using the relations:

$$y\ T_i(y) = \frac{1}{2}\ [T_{i+1}(y) + T_{i-1}(y)\ ]$$

$$y\ T_0(y) = T_1(y)$$

we find

$$\bar{\sum} \gamma_i y\ T_i(y) = \frac{1}{2}\gamma_1\ T_0(y) + (\gamma_0 + \gamma_2/2)\ T_1(y)$$

$$+ \frac{1}{2}(\gamma_{i-1} + \gamma_{i+1})\ T_i(y) + \ldots + \frac{1}{2}(\gamma_{n-1} + \gamma_{n+1})\ \ T_n(y)$$

For convenience $\gamma_{n+1}$ is set to zero. Replacing this in 3.17 and using the orthogonality of the Chebyshev polynomials with respect to the weight function $(1-y^2)^{1/2}$ we get

$$\langle \frac{x-c_1}{d_1} p\ ,\ p\ \rangle_1 = \frac{1}{2}\gamma_1\gamma_0\ s_0 + (\gamma_1+\gamma_2/2)\gamma_1 s_1 +$$

$$\frac{1}{2}(\gamma_{i+1}+\gamma_{i-1})\gamma_i s_i + \ldots + \frac{1}{2}(\gamma_{n+1} + \gamma_{n-1})\gamma_n s_n$$

with $s_0 = 2$ and $s_i = 1$ for $i \neq 0$ . This finally yields

$$\langle \frac{x-c_1}{d_1} 1p\ ,\ \ p\ \rangle = \tau_1$$

and by 3.14

$$\langle\ xp\ ,\ p\ \rangle_1 = c_1\sigma_1 + d_1\tau_1 \qquad (3.18)$$

A similar calculation would give

$$\langle xp,p\rangle_2 = c_2\sigma_2 + d_2\tau_2 \qquad (3.19)$$

Adding 3.18 and 3.19 finally yields the desired result 3.15.

Q.E.D.

Once the parameters $\alpha_n$ and $\beta_{n+1}$ are computed by an appropriate use of 3.14 and 3.15, it is possible to generate the next orthogonal polynomial $p_{n+1}(x)$. The next proposition provides a simple way of obtaining $p_{n+1}$ when the expansions 3.10 and 3.11 are used.

Proposition 3.2: Let $p_n(x)$, $n=0,1,\ldots$ be the sequence of orthogonal polynomials derived from 3.2. Then the expansion coefficients $\gamma_i^{(n)}$ of the expansion 3.10 satisfy the recurrence relation:

$$\beta_{n+1}\gamma_i^{(n+1)} = \frac{d_1}{2}(\gamma_{i-1}^{(n)}+\gamma_{i+1}^{(n)}) + (c_1-\alpha_n)\gamma_i^{(n)} - \beta_n\gamma_i^{(n-1)}, \quad i=2,3,\ldots n+1 \qquad (3.20)$$

with the initial conditions :

$$\beta_{n+1}\gamma_0^{(n+1)} = d_1\gamma_1^{(n)} + (c_1-\alpha_n)\gamma_0^{(n)} - \beta_n\gamma_0^{(n-1)} \qquad (3.21)$$

$$\beta_{n+1}\gamma_1^{(n+1)} = d_1(\gamma_0^{(n)}+\frac{1}{2}\gamma_2^{(n)}) + (c_1-\alpha_n)\gamma_1^{(n)} - \beta_n\gamma_1^{(n-1)} \qquad (3.22)$$

A relation analogous to 3.20 and in which $c_1$ is replaced by $c_2$ and $d_1$ by $d_2$ holds for the $\delta$'s.

Proof: From 3.2 we have

$$\beta_{n+1}p_{n+1}(x) = (x-\alpha_n)\sum\nolimits' \gamma_i^{(n)}\, T_i[(x-c_1)/d_1] - \beta_n \sum\nolimits' \gamma_i^{(n-1)}\, T_i[(x-c_1)/d_1]$$

Using again the change of variable $y=(x-c_1)/d_1$ for convenience we get

$$\beta_{n+1}p_{n+1} = (d_1y+c_1-\alpha_n)\sum\nolimits' \gamma_i^{(n)}\, T_i(y) - \beta_n \sum\nolimits' \gamma_i^{(n-1)}\, T_i(y)$$

Noticing that $y\,T_i(y) = \frac{1}{2}(T_{i-1}(y) + T_{i+1}(y))$ when $i \geq 1$ and $y\,T_0(y) = T_1(y)$ we obtain:

$$\beta_{n+1}p_{n+1} = \frac{d}{2}1 \bar{\sum} \gamma_i^{(n)} \; T_{i+1}(y) + T_{i-1}(y)] + (c_1-\alpha_n) \bar{\sum} \gamma_i^{(n)} \; T_i(y)$$

$$-\beta_n \bar{\sum} \gamma_i^{(n-1)} \; T_i \; (y)$$

with the notational convention $T_{-1} = T_1$. Therefore

$$\beta_{n+1} \; p_{n+1} = d_1 \; [\gamma_1^{(n)} + (\gamma_0^{(n)} + \gamma_2^{(n)}/2) \; T_1(y)$$

$$+ \bar{\sum} \frac{1}{2}(\gamma_{i-1}^{(n)} + \gamma_{i+1}^{(n)}) \; T_i(y) \; ] + (c_1 - \alpha_n) \bar{\sum} \gamma_i^{(n)} \; T_i(y)$$

$$-\beta_n \bar{\sum} \gamma_i^{(n-1)} \; T_i(y)$$

with the conventions stated in the proposition. Returning to the x variable gives the desired expansion of $p_{n+1}$ in the form 3.10 and 3.11 and establishes the result.

Q.E.D

The results of the previous two propositions can now be combined to provide an algorithm for computing the sequences of orthogonal polynomials. The idea of the algorithm is quite simple. At a typical step the polynomial $p_n$ is available through its expansions 3.10 and 3.11 so one can compute $\alpha_n$ by formula 3.14. Then we can compute the expansion coefficients of $\tilde{p}_{n+1}$ using proposition 3.2. Finally $\beta_{n+1}$ is obtained from 3.4 and 3.14 and $\tilde{p}_{n+1}$ is normalized by $\beta_{n+1}$ to yield the next polynomial $p_{n+1}$. This describes schematically one step of the algorithm given below. A notational simplification is achieved by introducing the symbol $\bar{\sum}'$ defined by

$$\bar{\sum_{i=0}^{i=n}}' \; a_i = 2 \; a_0 + \sum_{i=0}^{i=n} a_i \qquad (3.23)$$

## ALGORITHM 1

### Computation of the Generalized Chebyshev Polynomials

1. **Initialize.** Choose the initial polynomial $p_0(x)$ in such a way that $||p_0||=1$ :

$$\gamma_0^{(0)} := \delta_0^{(0)} := \sigma_1^{(0)} := \sigma_2^{(0)} = \frac{1}{2} \quad , \qquad \tau_1^{(0)} := \gamma_2^{(0)} := \beta_0 := 0$$

2. **Iterate.** For $n=0,1,2,\ldots,n_{max}$ do

- Compute $\alpha_n$ by formula 3.15:

$$\alpha_n := c_1\sigma_1^{(n)} + c_2\sigma_2^{(n)} + d_1\tau_1^{(n)} + d_2\tau_2^{(n)}$$

- Compute expansion coefficients of $\tilde{p}_{n+1}$ by formulas 3.20 - 3.22 and their analogue for the $\delta$ 's.

$$\tilde{\gamma}_0^{(n+1)} := d_1\gamma_1^{(n)} + (c_1-\alpha_n)\gamma_0^{(n)} - \beta_n\gamma_0^{(n-1)}$$

$$\tilde{\gamma}_1^{(n+1)} := d_1(\gamma_0^{(n)} + \tfrac{1}{2}\gamma_2^{(n)}) + (c_1-\alpha_n)\gamma_1^{(n)} - \beta_n\gamma_1^{(n-1)}$$

$$\tilde{\gamma}_i^{(n+1)} := \tfrac{d_1}{2}(\gamma_{i-1}^{(n)} + \gamma_{i+1}^{(n)}) + (c_1-\alpha_n)\gamma_i^{(n)} - \beta_n\gamma_i^{(n-1)}$$

$$i=2,3,\ldots,n+1$$

$$\tilde{\delta}_0^{(n+1)} := d_2\delta_1^{(n)} + (c_2-\alpha_n)\delta_0^{(n)} - \beta_n\delta_0^{(n-1)}$$

$$\tilde{\delta}_1^{(n+1)} := d_2(\delta_0^{(n)} + \tfrac{1}{2}\gamma_2^{(n)}) + (c_2-\alpha_n)\delta_1^{(n)} - \beta_n\delta_1^{(n-1)}$$

$$\tilde{\delta}_i^{(n+1)} := \tfrac{d_2}{2}(\delta_{i-1}^{(n)} + \delta_{i+1}^{(n)}) + (c_2-\alpha_n)\delta_i^{(n)} - \beta_n\delta_i^{(n-1)}$$

$$i=2,3,\ldots,n+1$$

- Compute $||\tilde{p}_{n+1}||$ by formula 3.14 and get $\beta_{n+1}$ by formula 3.4:

$$\tilde{\sigma}_1^{(n+1)} := \sum_{i=1}^{n+1}, [\tilde{\gamma}_i^{(n+1)}]^2$$

$$\tilde{\sigma}_2^{(n+1)} := \sum_{i=1}^{n+1}, [\tilde{\delta}_i^{(n+1)}]^2$$

$$\beta_{n+1} := [\tilde{\sigma}_1^{(n+1)} + \tilde{\sigma}_2^{(n+1)}]^{1/2}$$

- Normalize $\tilde{p}_{n+1}$ by $\beta_{n+1}$ to get $p_{n+1}$:

$$\gamma_i^{(n+1)} := \tilde{\gamma}_i^{(n+1)}/\beta_{n+1} \quad , \qquad \delta_i^{(n+1)} := \tilde{\delta}_i^{(n+1)}/\beta_{n+1}$$

$$i = 0,1,\ldots,n+1$$

- Compute new $\sigma$'s and $\tau$'s

$$\sigma_1^{(n+1)} := \tilde{\sigma}_1^{(n+1)} / \beta_{n+1}^2 \qquad \sigma_2^{(n+1)} := \tilde{\sigma}_2^{(n+1)} / \beta_{n+1}^2$$

$$\tau_1^{(n+1)} := \sum_{i=0}^{n} \gamma_i^{(n+1)} \gamma_{i+1}^{(n+1)}, \qquad \tau_2^{(n+1)} := \sum_{i=0}^{n} \delta_i^{(n+1)} \delta_{i+1}^{(n+1)}$$

At the $n^{th}$ step of the algorithm one performs approximately $20(n+1)$ arithmetic operations. Concerning the storage requirements, we must save the coefficients $\gamma_i^{(n)}, \delta_0^{(n)}$ as well as the previous coefficients $\gamma_0^{(n-1)}$, and $\delta_0^{(n)}$, which means $4(n_{max} + 1)$ memory locations, if $n_{max}$ is the maximum degree allowed. This suggests that, in practice $n_{max}$ should not be too high. Note that restarting could be used if necessary.

### 3.3. The minimal polynomial

Consider the orthonormal polynomial $p_n$ of degree n produced by Algorithm 1. According to section 2, an interesting approximate solution to the system $A x = b$ would be the one for which the residual vector $r_n$ satisfies:

$$r_n = [p_n(0)]^{-1} \, p_n(A) \, r_0$$

It is easily verified that such an approximation is given by

$$x_n = x_0 + s_{n-1}( A ) \, r_0$$

where

$$s_{n-1}(x) = [ p_n(0)]^{-1} \, [p_n(0) - p_n(x)] / x \tag{3.24}$$

Practically and theoretically however this approach faces several drawbacks. First, it may happen that $p_n(0)$ is zero or very close to zero. In that case the $n^{th}$ approximation $x_n$ either is not defined or becomes a poor approximation to x

respectively. Although this difficulty might be overcome by computing $x_{n+1}$ directly from $x_{n-1}$, this will not be considered.

A second disadvantage of using the orthogonal polynomials $p_n$ is that they do not satify a suitable optimality property. More precisely the norm $||p_n/p_n(0)||_C$ is not minimum over all polynomials of degree not exceeding n satisfying $p_n(0) = 1$. This fact deprives us from any means of comparing the residual norms, and above all does not ensure that the approximate solution $x_n$ will eventually converge towards the exact solution x.

These remarks lead us to resorting to the polynomial $q_n$ of degree n such that $q_n(0) = 0$ which minimizes the norm $||q||_C$ over all polynomials q of degree $\leq n$.

Writing the polynomials q as $q(x) = 1 - x\,s(x)$ where s is a polynomial of degree not exceeding n-1, it is clear that our problem can be reformulated as follows:

Find a polynomial $s^*_{n-1} \in P_{n-1}$ such that:

$$||1-x\,s^*_{n-1}(x)||_C \leq ||1-x\,s(x)||_C, \quad \forall \; s \in P_{n-1} \tag{3.25}$$

Here 1-xs(x) denotes the polynomial q defined by $q(x) = 1 - xp(x)$ rather than the value of this polynomial at x.

From the theory of approximation it is known that the best polynomial $s^*_{n-1}$ exits and is unique [3]. Furthermore the theory of least squares approximation indicates that the solution $s^*_{n-1}$ must satisfy the equations:

$$\langle 1-xs^*_{n-1},\, x\,s(x) \rangle = 0, \quad \forall \; s \in P_{n-1} \tag{3.26}$$

To solve 3.26 let us aassume that we can find a sequence of polynomials $q_j, j=0,1,..n-1,..$ such that the polynomials $xq_j(x)$ form an orthonormal sequence with respect to $\langle\ ,\ \rangle$. Then expressing $s^*_{n-1}$ as

$$s^{*}_{n-1} = \sum_{j=0}^{n-1} \eta_j q_j(x)$$

we obtain from 3.26

$$\eta_j = \langle 1, \ xq_j(x) \rangle \tag{3.27}$$

It turns out that the polynomials $q_j(x)$ can be obtained quite simply by an algorithm which is nothing but algorithm 1 with a different initialization. This is because we actually want to compute an orthogonal sequence of the form $\{xq_n\}$ with respect to the same weight function w(x) as before.

Practically we will have again to express the polynomials $xq_n(x)$ by two different formulas as in subsection 3.2:

$$xq_n(x) = \sum_{i=0}^{n+1} \gamma_i'^{(n)} \ T_i(x), \quad x \in [a,b] \tag{3.28}$$

$$xq_n(x) = \sum_{i=0}^{n+1} \delta_i'^{(n)} \ T_i(x), \quad x \in [c,d] \tag{3.29}$$

Denoting by $\alpha'_n$ and $\beta'_n$ the coefficients of the new three term recurrence which is satisfied by the polynomials $q_n$ we obtain:

$$\beta'_{n+1} \ q_{n+1}(x) = (x- \alpha'_n)q_n(x) - \beta'_n q_{n-1}(x) \tag{3.30}$$

which is equivalent to:

$$\beta'_{n+1} \ xq_{n+1}(x) = (x- \alpha'_n) \ xq_n(x) - \beta'_n \ xq_{n-1}(x) \tag{3.31}$$

The condition that the sequence $\{ xq_j \}_{j=0,1,..n..}$ is orthonormal gives the following expressions for $\alpha'_n$ and $\beta'_n$:

$$\alpha'_n = \langle x \ (xq_n), \ xq_n \rangle \tag{3.32}$$

$$\beta'_{n+1} = ||x\tilde{q}_n||_C = ||(x - \alpha'_n)xq_n(x) - \beta'_n xq_{n-1}(x) \ ||_C \tag{3.33}$$

It is clear from the above formulas that the Algorithm for computing the

sequence $xq_n(x)$ is essentially algorithm 1 in which the initialization is replaced by a new one corresponding to the starting polynomial $xq_0(x) = x$ instead of $p_0(x) = 1$. Thus the orthogonal sequence $xq_n(x)$ replaces the sequence $p_n(x)$ of algorithm 1.

We must now indicate how to obtain the expansion coefficients $\eta_j$ of the polynomial $s_n^*$ in the orthogonal basis $\{ q_j \}$. According to 3.27 we have

$$\eta_j = \langle 1, xq_j \rangle$$

Hence

$$\eta_j = \langle T_0, xq_j \rangle_1 + \langle T_0, xq_j \rangle_2$$

$$\eta_j = 2( \gamma_0'^{(j)} + \delta_0'^{(j)} ) \qquad (3.34)$$

Here we recall that the weight function w has been scaled such that $\langle T_0, T_0 \rangle_i = 2$, $i = 1,2$ and $\langle T_j, T_j \rangle_i = 1$ for $j \neq 0$, $i = 1,2$. As a consequence of 3.34 the expansion coefficients $\eta_n$ can be obtained at a negligible cost. We can now write down the algorithm for computing the orthogonal sequence $q_n(x)$. For simplicity the prime symbols will be dropped.

## ALGORITHM 2

1. **Initialization.** Choose the polynomial $xq_0(x)$ in such a way that $q_0$=constant and $||xq_0||_C = 1$.

$$t2 := 2(c_1^2 + c_2^2) + d_1^2 + d_2^2$$

$$t := t2^{1/2}$$

$$\gamma_0^{(0)} := c_1/t, \quad \delta_0^{(0)} := c_2/t$$

$$\gamma_1^{(0)} := d_1/t, \quad \delta_1^{(0)} := d_2/t$$

$$\sigma_1^{(0)} := [2c_1^2 + d_1^2]/t2, \quad \sigma_2^{(0)} := [2c_2^2 + d_2^2]/t2,$$

$$\tau_1^{(0)} := 2\ \gamma_0^{(0)}\ \gamma_1^{(0)}\ , \qquad \tau_2^{(0)} := 2\ \delta_0^{(0)}\ \delta_1^{(0)}\ ,$$

2. **Iterate**. For $n=0,1,2,\ldots,n_{max}$ do

- Compute $\alpha_n$ by formula 3.15:

$$\alpha_n := c_1\sigma_1^{(n)} + c_2\sigma_2^{(n)} + d_1\tau_1^{(n)} + d_2\tau_2^{(n)}$$

- Compute expansion coefficients of $\tilde{q}_{n+1}$ by formula 3.20 - 3.22 and their analogue for the $\delta$ 's.

$$\tilde{\gamma}_0^{(n+1)} := d_1\gamma_1^{(n)} + (c_1-\alpha_n)\gamma_0^{(n)} - \beta_n\gamma_0^{(n-1)}$$

$$\tilde{\gamma}_1^{(n+1)} := d_1(\gamma_0^{(n)} + \tfrac{1}{2}\gamma_2^{(n)}) + (c_1-\alpha_n)\gamma_1^{(n)} - \beta_n\gamma_1^{(n-1)}$$

$$\tilde{\gamma}_i^{(n+1)} := \tfrac{d_1}{2}(\gamma_{i-1}^{(n)} + \gamma_{i+1}^{(n)}) + (c_1-\alpha_n)\gamma_i^{(n)} - \beta_n\gamma_i^{(n-1)}$$

$$i=2,3,\ldots,n+2$$

$$\tilde{\delta}_0^{(n+1)} := d_2\delta_1^{(n)} + (c_2-\alpha_n)\delta_0^{(n)} - \beta_n\delta_0^{(n-1)}$$

$$\tilde{\delta}_1^{(n+1)} := d_2(\delta_0^{(n)} + \tfrac{1}{2}\gamma_2^{(n)}) + (c_2-\alpha_n)\delta_1^{(n)} - \beta_n\delta_1^{(n-1)}$$

$$\tilde{\delta}_i^{(n+1)} := \tfrac{d_2}{2}(\delta_{i-1}^{(n)} + \delta_{i+1}^{(n)}) + (c_2-\alpha_n)\delta_i^{(n)} - \beta_n\delta_i^{(n-1)}$$

$$i=2,3,\ldots,n+2$$

- Compute $\|\tilde{q}_{n+1}\|$ by formula 3.14 and get $\beta_{n+1}$ by formula 3.4:

$$\tilde{\sigma}_1^{(n+1)} := \sum_{i=1}^{n+1}{}' [\tilde{\gamma}_i^{(n+1)}]^2 \ , \qquad \tilde{\sigma}_2^{(n+1)} := \sum_{i=1}^{n+1}{}' [\tilde{\delta}_i^{(n+1)}]^2$$

$$\beta_{n+1} := [\tilde{\sigma}_1^{(n+1)} + \tilde{\sigma}_2^{(n+1)}]^{1/2}$$

- Normalize $\tilde{q}_{n+1}$ by $\beta_{n+1}$ to get $q_{n+1}$:

$$\gamma_i^{(n+1)} := \tilde{\gamma}_i^{(n+1)}/\beta_{n+1} \ , \qquad \delta_i^{(n+1)} := \tilde{\delta}_i^{(n+1)}/\beta_{n+1}$$

$$i = 0,1,\ldots,n+1$$

- Compute new $\sigma$'s and $\tau$'s

$$\sigma_1^{(n+1)} := \tilde{\sigma}_1^{(n+1)}/\beta_{n+1}^2 \qquad \sigma_2^{(n+1)} := \tilde{\sigma}_2^{(n+1)}/\beta_{n+1}^2$$

$$\tau_1^{(n+1)} := \sum_{i=0}^{n} \gamma_i^{(n+1)}\,\gamma_{i+1}^{(n+1)}, \qquad \tau_2^{(n+1)} := \sum_{i=0}^{n} \delta_i^{(n+1)}\,\delta_{i+1}^{(n+1)}$$

Note that part 2( iterate ) is identical with part 2 of algorithm 1. It is important to realize that the above algorithm does not yield directly the polynomials $q_n$ but rather provides their expansion coefficients in terms of the Chebyshev polynomials in the two intervals. In the subsequent sections we will need to generate the polynomial $s_n^*(x)$. This can be achieved by coupling with part 2 of the above algorithm the following

$$q_{n+1}(x) := \frac{1}{\beta_{n+1}}\,[(x-\alpha_n)q_n(x) - \beta_n q_n(x)]$$

$$\eta_{n+1} := 2(\gamma_0^{(n+1)} + \delta_0^{(n+1)})$$

$$s_{n+1}^* := s_n^* + \eta_{n+1} q_{n+1}$$

starting with

$$q_0(x) = 1/t\,, \qquad s_0(x) = 2(\gamma_0^{(0)} + \delta_0^{(0)})\; q_0(x)$$

where t is defined in the initialization of the algorithm.

### 3.4. Comparison with the minimax polynomial.

The purpose of this subsection is to compare the norms of the minimal polynomial described in the previous subsection and the polynomial having the least uniform norm in [a,b] U [c,d] subject to the constraint $p_n(0) = 1$. We will denote by $||p||_\infty$ the uniform norm on [a,b] U [c,d], that is:

$$||p||_\infty = \max_{x \in [a,b] \cup [c,d]} |p(x)|$$

Let us denote by $p_n^*$ the minimal polynomial $1 - xs_{n-1}^*(x)$ of degree n, defined in section 3.3, and by $p_n^\infty$ the polynomial of degree n of least uniform norm in

[a,b]U[c,d]. In other words we have

(3.35)

$$||p_n^*||_C \leq ||p||_C, \quad \forall\, p \in P_n$$

$$||p_n^\infty||_\infty \leq ||p||_\infty, \quad \forall\, p \in P_n \tag{3.36}$$

Since $P_n$ is a finite dimensional space all the norms of $P_n$ are equivalent and the following lemma indicates precisely a comparison of the norms $||.||_C$ and $||.||_\infty$

Lemma 3.3: Let $f_n$ be any polynomial of degree not exceeding n. Then:

$$1) \quad ||f_n||_C \leq 2\, ||f_n||_\infty \tag{3.37}$$

$$2) \quad ||f_n||_\infty \leq (n+1)^{1/2} ||f_n||_C \tag{3.38}$$

Proof. 1)

$$||f_n||_C^2 = \langle f_n , f_n \rangle = \langle f_n , f_n \rangle_1 + \langle f_n , f_n \rangle_2$$

$$= \int_a^b f_n^2(x)\, w_1(x)\, dx + \int_c^d f_n^2(x) w_2(x)\, dx$$

$$\leq ||f_n||_\infty^2 \left[ \int_a^b w_1(x)\, dx + \int_c^d w_2(x)\, dx \right]$$

and finally

$$||f_n||_C^2 \leq 4\, ||f_n||_\infty^2$$

which gives the first result by taking the square root of both members.

2)

$$||f_n||_\infty = \max_{x \in [a,b]U[c,d]} |f_n(x)|$$

$$||f_n||_\infty = \max\left\{ \max_{x\in[a,b]} \left| \sum_{i=0}^{n} \gamma_i T_i[(x-c_1)/d_1] \right| , \max_{x\in[c,d]} \left| \sum_{i=0}^{n} \delta_i T_i[(x-c_2)/d_2] \right| \right\} \tag{3.39}$$

where the $\gamma$'s and the $\delta$'s are the expansion coefficients of $f_n$ in [a,b] and [c,d] with respect to the polynomials 3.8 and 3.9 respectively. Using the Schwartz inequality we get

$$\left| \sum_{i=0}^{n} \gamma_i T_i[(x-c_1)/d_1] \right| \leq \left[ \sum_{i=0}^{n} \gamma_i^2 \right]^{1/2} \left[ \sum_{i=0}^{n} T_i^2[(x-c_1)/d_1] \right]^{1/2}$$

Hence

$$\left| \sum_{i=0}^{n} \gamma_i T_i[(x-c_1)/d_1] \right| \leq (n+1)^{1/2} \left[ \sum_{i=0}^{n} \gamma_i^2 \right]^{1/2} \tag{3.40}$$

Likewise we can show that

$$\left| \sum_{i=0}^{n} \delta_i T_i[(x-c_2)/d_2] \right| \leq (n+1)^{1/2} \left[ \sum_{i=0}^{n} \delta_i^2 \right]^{1/2} \tag{3.41}$$

Replacing 3.40 and 3.41 in 3.39 we get

$$\begin{aligned} \|f_n\|_\infty &\leq (n+1)^{1/2} \max\left\{ \left[ \sum_{i=0}^{n} \gamma_i^2 \right]^{1/2}, \left[ \sum_{i=0}^{n} \gamma_i^2 \right]^{1/2} \right\} \\ &\leq (n+1)^{1/2} \left[ \sum_{i=0}^{n} (\gamma_i^2+\delta_i^2) \right]^{1/2} \\ &\leq (n+1)^{1/2} \left[ \sum_{i=0}^{n} , (\gamma_i^2+\delta_i^2) \right]^{1/2} \\ &= (n+1)^{1/2} \|f_n\|_C \end{aligned}$$

Here we have used the result of proposition 1 and the notation 3.24

Q.E.D

As an immediate consequence of 3.37 and 3.38 we can easily bound each of the norms $\| \|_C$ and $\| \|_\infty$ in terms of the other from the left and the right as in the next inequalities:

$$(n+1)^{-1/2} \|f_n\|_\infty \leq \|f_n\|_C \leq 2 \|f_n\|_\infty \tag{3.42}$$

$$\frac{1}{2} \|f_n\|_C \leq \|f_n\|_\infty \leq (n+1)^{1/2} \|f_n\|_C \tag{3.43}$$

We now state the main result of this subsection which compares with the same

norm the polynomials $p_n^*$ and $p_n^\infty$ .

Theorem 3.4: Let $p_n^\infty$ and $p_n^*$ be the minimax polynomial and the minimal polynomial defined by 3.35 and 3.36. Then the following inequalities hold:

$$||p_n^*||_C \leq ||p_n^\infty||_C \leq 2(n+1)^{1/2} ||p_n^*||_C \tag{3.44}$$

$$||p_n^\infty||_\infty \leq ||p_n^*||_\infty \leq 2(n+1)^{1/2} ||p_n^\infty||_\infty \tag{3.45}$$

Proof: We will only prove 3.44 as the proof for 3.45 is identical. The first part of the double inequality is obvious by equation 3.34. The second part is a simple consequence of lemma 3.3:

$$||p_n^\infty||_C \leq (n+1)^{1/2} ||p_n^\infty||_\infty \leq (n+1)^{1/2} ||p_n^*||_\infty$$

$$\leq 2(n+1)^{1/2} ||p_n^*||_C$$

Q.E.D.

The meaning of the theorem is that the two polynomials $p_n^*$ and $p_n^\infty$ have a comparable smallness in [a,b] U [c,d]. This will have a quite interesting implication for the numerical methods using polynomial iteration since it means that we can replace the minimax polynomial by the least squares polynomial, which is by far easier to generate and to use, and still obtain a residual norm which is not larger than $2(n+1)^{1/2}$ times the norm corresponding to the minimax polynomial. Note that there is no reason a priori why to compare the two polynomials with one particular norm rather than the other. If the norm $||\ ||_C$ is chosen then the polynomial $p_n^*$ is smaller than the polynomial $p_n^\infty$ while with the uniform norm the contrary is true. Whichever norm is used however, the two polynomials have a norm comparable within a factor not exceeding $2(n+1)^{1/2}$, as is asserted in the above theorem.

## 4. Application to the solution of large undefinite systems

### 4.1. The Generalized Chebyshev Iteration

Let us return to the system 2.1 and suppose that we have an initial guess $x_0$ of the solution. We seek for an approximate solution of the form

$$x_n = x_0 + s_{n-1}(A)r_0 \tag{4.1}$$

where $s_{n-1}$ is a polynomial of degree n-1 and $r_0$ the initial residual. The residual vector $r_n$ of $x_n$ is given by

$$r_n = [I-As_{n-1}(A)]\ r_0 \tag{4.2}$$

The development of the previous sections lead us to choose for $s_{n-1}$ the polynomial $s^*_{n-1}$ which minimizes $||1-xs(x)||_C$ over all polynomials s of degree $\leq$ n-1. From subsection 3.3 we know that:

$$s^*_{n-1}(x) = \sum_{j=0}^{n-1} \eta_j q_j(x) \tag{4.3}$$

where the $q_j(x)$ satisfy the recurrence 3.30 and

$$\eta_j = 2(\gamma_0'^{(j)} + \delta_0'^{(j)})$$

Let us assume that the coefficients $\alpha_n$, $\beta_n$ as well as the $\gamma$'s and $\delta$'s are known. Noticing that

$$s^*_{n+1}(x) = s^*_n(x) + \eta_{n+1}\ q_{n+1}(x)$$

and letting $u_j = q_j(A)r_0$, it is immediately seen from subsection 3.3 that the approximate solution $x_n$ can computed from the recurrence:

$$\eta_n = 2[\gamma_0'^{(n)} + \delta_0'^{(n)}] \tag{4.4}$$

$$x_{n+1} := x_n + \eta_n u_n$$

$$u_{n+1} := \frac{1}{\beta'_{n+1}} [(A-\alpha'_n I)u_n - \beta'_n u_{n-1}] \tag{4.5}$$

Clearly the coefficients $\beta'_n$, $\alpha'_n$ and $\eta_n$ can be determined at the same time as 4.5 and 4.4 are performed. Therefore a typical step of the algorithm would be as follows:

**ALGORITHM3**

1. Compute $\alpha'_n$, $\beta'_{n+1}$ and the coefficients $\gamma'^{(n+1)}_i$, $\delta'^{(n+1)}_i$ (see part 2 of Algorithm 2).

2. Compute:

$$\eta_n = 2[\gamma'^{(n)}_0 + \delta'^{(n)}_0] \tag{4.6}$$

$$x_{n+1} = x_n + \eta_n u_n \tag{4.7}$$

$$u_{n+1} := \frac{1}{\beta'_{n+1}} [(A-\alpha'_n I)u_n - \beta'_n u_{n-1}] \tag{4.8}$$

We will refer to this algorithm as the Generalized Chebyshev Iteration algorithm. Although we need not save the coefficients $\alpha'_n$ and $\beta'_n$, if a restarting strategy is used in which the iteration is performed with the same polynomial it will be more efficient to save the coefficients $\alpha'$ and $\beta'$.

Note that the multiplication by a scalar in 4.7 can be avoided by using instead of the vectors $u_n$ the sequence of vectors $u'_n$ defined by:

$$u'_n = \eta_n u_n$$

which can be updated in no more operations than are needed for $u_n$.

Each step of the above algorithm requires one matrix by vector multiplication, 3N additions 3N multiplications and O(n) operations, at the $n^{th}$ step. Four vectors of length N need to be stored in memory plus an extra O(n) locations for the coefficients $\gamma$ amd $\delta$. A feature of the Generalized Chebyshev Iteration, which is particularly attractive in parallel computation is the absence

of any inner product of N-dimensional vectors.

### 4.2. A minimal residual type implementation

A look at the previous algorithm reveals a similarity with one of the various versions of the conjugate gradient method which is described in [2]. One might ask whether it is possible to provide a version of the method which resembles the conjugate gradient or the conjugate residual method. We are about to show here that the answer is yes and we will provide a method which is very similar to the conjugate residual algorithm described in [2].

To be more specific we are looking for an iteration of the form :

$$x_{n+1} = x_n + a_n w_n \tag{4.9}$$

$$r_{n+1} = r_n - a_n A w_n \tag{4.10}$$

$$w_{n+1} = r_{n+1} + b_n w_n \tag{4.11}$$

where the scalars $a_n$, $b_n$ are some coefficients that are computed in such a way that at each step the residual $r_{n+1}$ is precisely the residual obtained by using the minimal polynomial as described earlier.

There are two main reasons why such a new form of algorithm 3 might be sought for. The first is that the above version is slightly more economical than algorithm 3, specifically 4.9 to 4.11 can be performed with N less multiplications. Note also that N less storage locations are required in the case where the matrix by vector multiplication can be performed componentwise, i.e. if the i-th component of Ax can be delivered cheaply for all i. The second reason for looking for a version of the form given above is that the residual vector is available at each step of the iteration.

Some theoritical background is necessary for the understanding of the remainder of this subsection. The development of the conjugate gradient like method to be presented is based on the simple remark that the approximate solution $x_n$ belongs to the affine subspace $x_0+K_n$ where $K_n$ is the so called Krylov subspace spanned by $r_0$, $Ar_0$, $A^2r_0$,... $A^{n-1}r_0$ . Then the key observation is that there is a one to one mapping between the subspace $K_n$ and the space $P_{n-1}$ of polynomials of degree not exceeding n-1. In effect for any element u of $K_n$ of the form $u = \mu_0 r_0 + \mu_1 Ar_0 + \ldots + \mu_{n-1} A^{n-1} r_0$ , we can associate the polynomial $q_u(t) = \mu_0 + \mu_1 t + \ldots + \mu_{n-1} t^{n-1}$. Clearly the transformation which maps u into $q_u$ is an isomorphism between $K_n$ and $P_{n-1}$. We now introduce a new inner product on $K_n$.

Definition: We will call Generalized Chebyshev inner product on $K_n$ and will denote by $(.,.)_C$ the bilinear form defined by :

$$(u,v)_C = \langle q_u, q_v \rangle \tag{4.12}$$

where $\langle .,. \rangle$ is the the inner product on $P_{n-1}$ defined by 3.1, 3.6 and 3.7.

That this constitutes an inner product is an immediate consequence of the isomorphism introduced above. There is no ambiguity in denoting again by $||.||_C$ the norm on $K_n$ induced by this inner product . We then have the following immediate result.

Proposition 4.1: At each step of the Generalized Chebyshev iteration, the approximate solution $x_n$ minimizes the generalized Chebyshev norms $||f-Ax||_C$ of the residual vectors f-Ax over all vectors x of $K_n$.

In other words we have:

$$||f - A x_n|| \leq ||f - Ax||_C \quad \forall x \in K_n$$

The proof of the proposition is straightforward.

The most convenient way of deriving a version of algorithm 3 equivalent to 4.9-4.11 is to use a little analogy. Indeed since we want to minimize the residual norm with respect to the generalized Chebyshev inner product 4.12 we can use an adapted version of the minimal residual method which achieves the same goal with a different inner product. The version given below is precisely an adaptation of the conjugate residual method described in [2].

## ALGORITHM 4

1. Initialize. Choose an initial vector $x_0$ and compute $r_0=f-Ax_0$, $w_0=r_0$.
2. Iterate.

$$a_n := \langle xq_{r_n}, q_{r_n} \rangle / \langle xq_{w_n}, xq_{w_n} \rangle \tag{4.13}$$

$$x_{n+1} := x_n + a_n w_n \tag{4.14}$$

$$r_{n+1} := r_n - a_n A w_n \tag{4.15}$$

$$b_n := \langle xq_{r_{n+1}}, q_{r_{n+1}} \rangle / \langle xq_{r_n}, q_{r_n} \rangle \tag{4.16}$$

$$w_{n+1} := r_{n+1} + b_n w_n \tag{4.17}$$

$$q_{w_{n+1}} := q_{r_{n+1}} + b_n q_{w_n} \tag{4.18}$$

The computation of the inner products involved in 4.13 and 4.16 can be carried out in a similar way as before by use of proposition 3.1 together with expressions similar to 3.12, 3.13 for each of the polynomials $q_{r_n}$ and $q_{w_n}$ but we omit the details.

As with the conjugate residual method in the indefinite case, the above algorithm faces a risk of breakdown in equation 4.16 because $\langle xq_{r_n}, q_{r_n} \rangle$ can be equal to zero. We do not know under what circumstances the inner product $\langle xq_{r_n}, q_{r_n} \rangle$ vanishes but our short experience is that this does not often happen.

We will prefer algorithm 3 in general despite the little extra work and possibly storage involved. Note that it is also possible to write a conjugate gradient type method which would correspond to using as residual polynomial the orthogonal polynomial of subsection 3.2, instead of the minimal polynomial of subsection 3.3.

### 4.3. A Block Generalization

A block algorithm can be derived from the Generalized Chebyshev Iteration described above. The principle of the Block algorithm is similar to that of the Block Stieffel iteration proposed in [11].

As shown in fig. 4-1 if an eigenvalue $\lambda_i$ lies inside the interval [b,c] then $p^*(\lambda_i)$ is large in comparison with the other $p^*(\lambda_j)$, with $j \neq i$.

As a consequence the residual $r_n = p_n^*(A)r_0$ will be mainly in the direction of the eigenvector $z_i$ asociated with $\lambda_i$. Therefore by a projection process the component of the residual in this direction can be eliminated leaving only the small components in the directions $z_j$ with $j \neq i$. Let us now assume that instead of one eigenvalue $\lambda_i$, we had m eigenvalues $\lambda_i, \lambda_{i+1}, \ldots, \lambda_{i+m-1}$ inside [b,c], and that we are performing m different Generalized Chebyshev iteration, using m different starting vectors $x_{0,1}, x_{0,2} \ldots, x_{0,m}$. We can again make the same argument: the residuals $r_{n,k}$ at the $n^{th}$ step will have their dominant componants in the eigenvectors $z_i, z_{i+1}, \ldots, z_{i+m-1}$ and we can eliminate these components by projecting onto the subspace spanned by $\{r_{n,j}\}_{j=1,2,..m}$. It was shown in [11] that in the positive definite case, the rate of convergence of the Block Stiefel Iteration is in general substantially superior to that of the single vector Stiefel Iteration.

The block algorithm is more particularly suitable for parallel computation since the m Generalized Chebyshev iterations involved s, can be performed

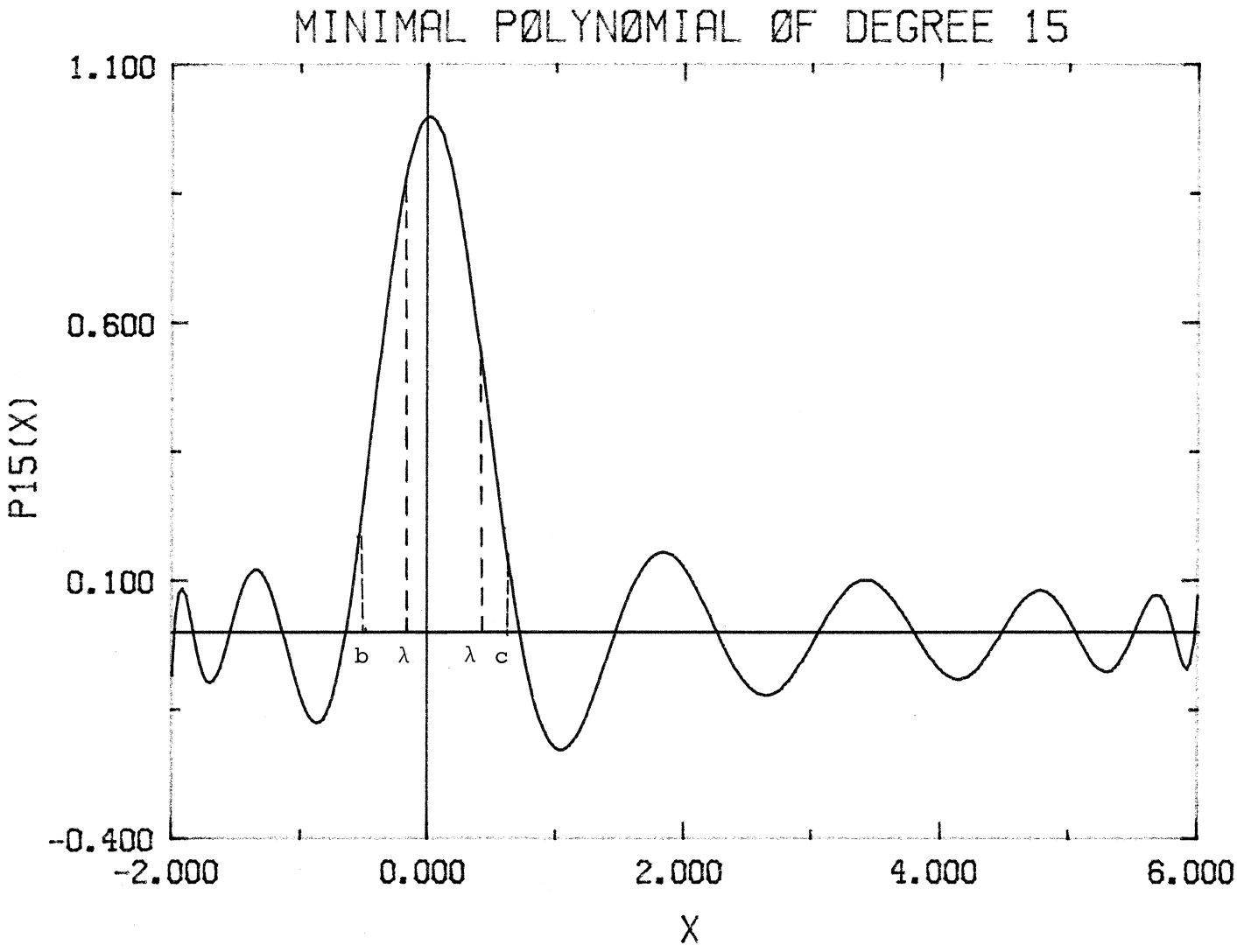


**Figure 4-1: Minimal polynomial with some eigenvalues in [b,c]**

simultaneously. Ignoring the matrix by vector multiplication, intercommunication between the processors is needed only when the projection process takes place, which is infrequent. As mentioned earlier the fact that there are no inner products of N dimensional vectors is a further advantage of this algorithm in parallel computation.

## 4.4. A Priori Error Bounds.

We now establish an a priori error bound which will show that the Generalized Chebyshev Iteration is convergent. We should emphasize however that the bound given here is proposed for the sole purpose of demonstrating the global convergence of the process and does not constitute a sharp error bound in general.

The next theorem is based on the following estimate of the residual norm which is an immediate consequnece of lemma 3.3

Lemma 4.2: At the n-th step of of the Generalized Chebyshev iteration the residual norm $r_n$ of the approximate solution $x_n$ satisfies:

$$||r_n|| \leq (n+1)^{1/2} \, ||p_n^*||_C \, ||r_0|| \tag{4.19}$$

Proof: From equation 2.4 we have

$$||r_n|| \leq ||r_0|| \max_{i=1,..N} |p_n^*(\lambda_i)|$$

Therefore

$$||r_n|| \leq ||r_0|| \, ||p_n^*||_\infty$$

The result follows from inequality 3.37 of lemma 3.3.

Q.E.D.

Theorem 4.3: Suppose that the spectrum of A is contained in [a,b]U[c,d] and let

$$m = \min\{|b|,|c|\},\ M = \max\{|a|,|d|\}$$

Then at the $n^{th}$ step of the Generalized Chebyshev Iteration the residual vector $r_n$ satisfies the inequality:

$$||r_n|| \leq 2(n+1)^{1/2} \frac{||r_0||}{T_{n'}\left[\frac{M^2+m^2}{M^2-m^2}\right]} \tag{4.20}$$

where $n'$ is the integer division of $n$ by 2 and $T_k$ represents the Chebyshev polynomial of degree $k$ of the first kind.

Proof: By equation 4.19 we have

$$||r_n|| \leq (n+1)^{1/2}\ ||p_n^*||_C\ ||r_0|| \tag{4.21}$$

$$\leq (n+1)^{1/2}\ ||p||_C\ ||r_0||$$

for any polynomial p of degree $\leq n$ such that $p(0) = 1$. Consider the polynomial:

$$t_n(x) = T_{n'}[q(x)] \ / \ T_{n'}[q(0)]$$

with

$$q(x) = [M^2 + m^2 - x^2\ ]/[M^2 - m^2]$$

The degree of $t_n$ is $\leq n$ and we have $t_n(0) = 1$. Therefore from 4.21 and lemma 3.3 we get

$$||r_n|| \leq (n+1)^{1/2}\ ||t_n||_C\ ||r_0|| \leq 2(n+1)^{1/2}\ ||t_n||_\infty ||r_0||$$

To complete the proof we observe that $t_n$ has been chosen in such a way that $||t_n||_\infty = [T_m(q(0))\ ]^{-1}$ because for x belonging to $[a,b]\cup[c,d]$, $|t_n(x)| \leq 1$, the value 1 being reached for some x's in $[a,b]\cup[c,d]$ .

Q.E.D.

Note that ideally $M^2$ and $m^2$ are the largest and the smallest eigenvalues of $A^2$. Hence an interpretation of 4.20 is that n steps of our algorithm will yield a residual norm not larger than $2(n+1)^{1/2}$ times the residual norm obtained from $\lceil n/2 \rceil$ steps of the classical chebyshev iteration applied to the normal equations $A^2 x = A b$. This result is however a pessimistic estimate as the numerical experiments show.

## 5. Application to the computation of eigenvectors associated with interior eigenvalues.

Consider the eigenvalue problem

$$A z = \lambda z \tag{5.1}$$

where A is N dimensional and symmetric. When the desired eigenvalue $\lambda$ is one of the few extreme eigenvalues of A, there are a host of methods for solving 5.1, among which the Lanczos algorithm appears as one of the most powerful. When $\lambda$ is in the interior of the spectrum then problem 5.1 becomes more difficult to solve numerically. The Lanczos algorithm often suffers from a slow rate of convergence in that case and may require a number of steps as high as several times the dimension N of A. A further disadvantage of this is that the Lanczos vectors must be kept in secondary storage if the eigenvectors are wanted. This might be acceptable if a large number of eigenvalues are needed but when one only seeks for a small number of them, possibly with a moderate accuracy, then the use of techniques based upon subspace iteration should not be discarded.

We would like to indicate in this section how the polynomials described in section 3 can effectively be used to obtain approximations to eigenvectors associated with interior eigenvalues. Consider an approximation to the eigenvector z of 5.1 of the form:

$$v_n = p_n(A)\, v_0 \tag{5.2}$$

where $v_0$ is some initial approximation of z and $p_n$ a polynomial of degree n. Writing $v_0$ in the eigenbasis $\{z_i\}_{i=1,..N}$ as

$$v_0 = \sum_{i=1}^{N} \xi_i z_i$$

we obtain:

$$v_n = \sum_{i=1}^{N} \xi_i\, p_n(\lambda_i)\, z_i \tag{5.3}$$

which shows that if $v_n$ is to be a good approximation of the eigenvector $z_i$ then every $p_n(\lambda_j)$ with $j \neq i$ must be small in comparison with $p_n(\lambda_i)$ . This leads to seek for a polynomial which is small in $[\lambda_1,\lambda_{i-1}]U[\lambda_{i+1},\lambda_N]$ and which satisfies $p_n(\lambda_i) = 1$. In other words we need to find a polynomial of the form $1-(x-\lambda_i)t_{n-1}(x-\lambda_i)$ which is small in these intervals in the least squares sense. The simple change of variables $t = x-\lambda_i$ transforms this problem into the one of subsection 3.3 thus providing an algorithm for computing an approximation to the desired eigenvector associated with $\lambda_i$. Note that in practice $\lambda_i$ is not available and is replaced by some approximation $\mu$ which is improved during the process. We will find a polynomial in the variable t which corresponds to the operator $A - \mu I$ i.e. the approximate eigenvector $v_n$ will have the form

$$v_n = p_n^*(A-\mu I)\, v_0 = [\, I - (A-\mu I)\, s_{n-1}^*(A-\mu I)\, ]\, v_0$$

where $p_n^*$is the least squares polynomial obtained from algorithm 2 with :

$$a = \lambda_1 - \mu, \quad b = \lambda^- - \mu, \quad c = \lambda^+ - \mu, \quad d = \lambda_N - \mu$$

where $\lambda_{i-1}$ and $\lambda_{i+1}$ are replaced by $\lambda^-$ and $\lambda^+$ respectively.

An interesting problem which arises here is to estimate the interior parameters b and c, or more precisely to refine dynamically a given pair of

starting parameters b,c. This is discussed in the next section and a few suggestions are given there.

An obvious extension of this algortihm which we omit to consider here is the subspace iteration otherwise known as simultaneous iteration method ( See e.g. [9])in which one iterates with a block of vectors simultaneously and uses the Rayleigh Ritz projection approximations from the subspace spanned by the blocks.

## 6. Some practical considerations

### 6.1. Projection techniques and the estimation of the optimal parameters.

Little was said in the previous sections about the determination of the parameters a,b,c,d. It is not often the case that these parameters are known beforehand, and one has to provide means for computing them dynamically. This part of the algorithm is a determinant factors of the efficiency. Let us recall that a,d are ideally the smallest and largest eigenvalues $\lambda_1$ and $\lambda_N$ of A, while b and c should be equal to the eigenvalues $\lambda^-$ and $\lambda^+$ closest to the origin with $\lambda^- < 0$ and $\lambda^+ > 0$. The smallest and largest eigenvalues $\lambda_1$ and $\lambda_N$ are easier to estimate and we have several ways of doing so, the simplest being perhaps the use of Gershgorin disks. The use of the Lanczos algorithm is also particularly interesting since it yields at the same time a good estimate of the parameters a,d and a fairly accurate starting vector for the iteration(see e.g. [7]). It is a fact that the extremal parameters a,d which are easier to compute are also more important for the convergence: if a is larger than $\lambda_1$ or d is less than $\lambda_N$ then there is a high risk that the process will diverge. This is because outside the interval [a,d] the residual polynomial can have huge values as is seen in figure 6-1. When this happens however nothing is lost because we can stop and use the (huge) residuals as approximate eigenvectors associated with $\lambda_1$ and $\lambda_N$ thus providing more accurate

estimates of both a and d, with the help of the Rayleigh quotient. (See figure 6-1)

For the interior parameters b and c a projection technique based on a similar principle can be developed. In effect if we start with two approximations b and c such that $b < \lambda^-$ and $c > \lambda^+$, then as is seen from figure 4-1 the residual vectors will have large components in the eigenvectors associated with the eigenvalues $\lambda^-$ and $\lambda^+$ closest to the origin. A Rayleigh Ritz approximation can therefore by applied to extract more accurate estimates of $\lambda^-$ and $\lambda^+$ from two or more residual vectors.

In fact a more complete procedure can be described as follows. Suppose that we start with an interval [b,c] sufficiently wide to contain the eigenvalues $\lambda^+$, and $\lambda^-$. Then after a number of steps the residual vectors will give an accurate representation of the eigenspace corresponding to the eigenvalues contained in the interval $[\lambda^-,\lambda^+]$. Therefore a projection process onto the subspace spanned by a small number of successive residual vectors will provide not only a good approximation of the eigenvalues $\lambda^-$ and $\lambda^+$, but also the Rayleigh Ritz approximate solution wiil be more accurate than the current approximation. In practice we will iterate with a polynomial of fixed degree. We will orthonormalize the m latest residuals where m is usually 5 or 6 (in order to have an accurate representation of at least three eigenvectors ). Calling Q the orthonormal set of vectors obtained from this orthogonalization, we compute the Ritz matrix $Q^T A Q$ and its eigenvalues by e.g. the QR algorithm. We then compare the Ritz eigenvalues thus obtained with those of the previous projection. After a certain number of iterations some of the Ritz eigenvalues will converge to a moderate accuracy (say to 2 or 3 digits of accuracy). As soon as an eigenvalue which corresponds to either $\lambda^-$, or $\lambda^+$ converges, we replace b or c by that eigenvalue. When both of the eigenvalues have converged, we replace the approximate solution

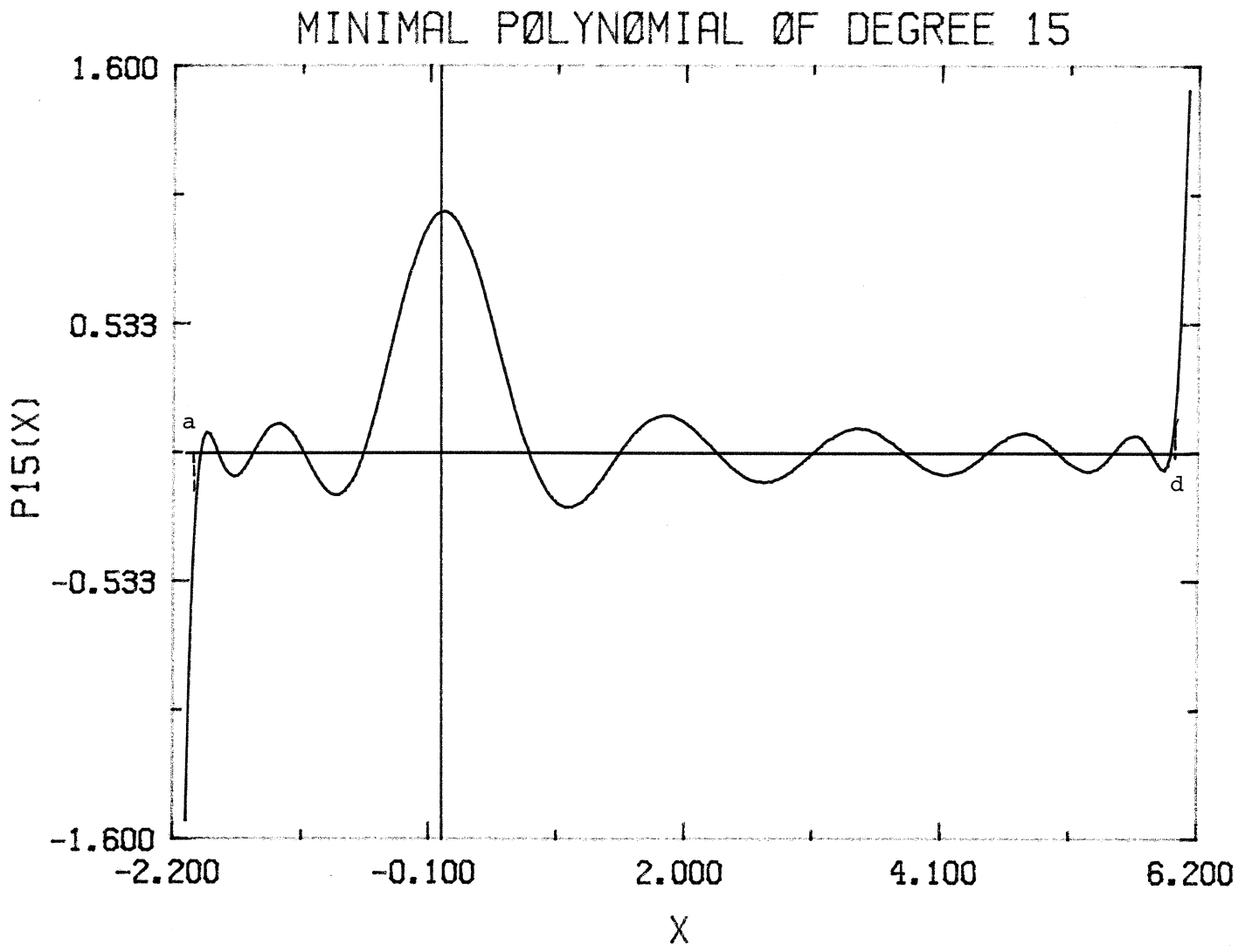


**Figure 6-1: Minimal polynomial with some eigenvalues outside [a,d]**

by the Rayleigh Ritz approximation $\tilde{x} = x_n + \sum [\lambda_j]^{-1} z_j^T r_n$ where the sum is over the converged Ritz eigenvalues $\lambda_j$ and the $z_j$ are the associated approximate Ritz eigenvectors. Recall that $z_j$ is defined by $z_j = Q.s_j$ where $s_j$ is an eigenvector of $Q^T A Q$ associated with the Ritz value $\lambda_j$. The effect of this important projection step is to eliminate the (large) components of the residual in the direction of the converged eigenvectors and is quite effective as will be seen in some experiments described in section 7. A useful variation here is to use the latest directions of search $u_i$ instead of the latest residuals. We thus avoid to compute residual vectors. The reason for this is that $q_n(A)r_0$ is a vector having large components in the directions of the eigenvectors associated with the eigenvalues nearest to zero. Our experience is that the results provided by the use of $u_n$ instead of $r_n$ are often slightly better than those using the uneconomical residuals.

The above process is even more efficiently implemented in a Block version, because then we have a good approximation to several eigenvalues at the same time. Note that in this case we need to have the interval [b,c] enclose at least m-1 eigenvalues if m is the dimension of the blocks.

The same projection technique as the one described above can be implemented for the problem of computing an interior eigenvalue $\lambda_i$ and its corresponding eigenvector. We will drop the subscript i in the following discussion. We are again assuming here that we already know a good estimate of the extreme parameters a, d and that we start with an interval [b,c] sufficiently wide to contain the eigenvalues $\lambda$, $\lambda^+$, and $\lambda^-$. Then after a number of steps of the iteration described in section 5 the approximate eigenvectors will give an accurate representation of the eigenspace corresponding to the eigenvalues contained in the interval [b,c]. Hence a projection process onto the subspace spanned by a small number of successive approximate eigenvectors will provide good approximations to

the eigenvalues $\lambda$, $\lambda^-$ and $\lambda^+$. Also the Ritz eigenvector will be a much better approximation than the current approximation. Practically we will orthonormalize the m latest approximations into the N x m orthonormal system Q. We then compute the Ritz matrix $Q^T A Q$ and its eigenvalues by e.g. the QR algorithm and compare the Ritz eigenvalues obtained with those of the previous projection. As soon as an eigenvalue which corresponds to either $\lambda$, $\lambda^-$, or $\lambda^+$ starts converging, we replace $\mu$, b or c by that eigenvalue. When the three of the eigenvalues have converged, we replace the approximate eigenvector by the Ritz vector $z = Q s$, where s is the eigenvector of $Q^T A Q$ associated with $\lambda$. This process is tested in an experiment in section 7.

### 6.2. Using high degree polynomials

One might ask whether it is possible to use high degree polynomials in practice. Our experience is that despite the fact that we often encounter underflow situations with high degree polynomials, if these underflows are handled simply by replacing the underflowing numbers by zeroes, it is always better to use a high degree polynomial than a low degree polynomial. This fact will be illustrated in an experiment in the next section. Polynomials of degree as high as 200 or 300 are quite useful for badly conditioned problems. We open a parenthesis here to point out the following interesting observation. It has been observed during the numerical experiments that the <u>last</u> coefficients $\delta_i^{(n)}$, $\gamma_i^{(n)}$ become tiny as <u>i and n increase.</u> This is the cause of the underflowing conditions mentioned above. The practical consequences of this observation is that after a certain number of steps we do not need to save those tiny elements. For high degree polynomials this will result in a nonnegligible cut off in memory needs. In fact a more important observation is that the coefficients $\alpha_n$ and $\beta_n$ are cyclically converging more precisely it seems that $\alpha_{3k+j}$ is a converging sequence for fixed j. The same phenomenon is true for the $\beta$'s. A proof of this phenomenon

is not available however. As a consequence there is a hope that after a certain step we can simply use the previous $\alpha$'s and $\beta$'s thus avoiding further calculations of these cofficients.

## 7. Numerical experiments

In this section we will describe a few numerical experiments and give some more details on the pactical use of the Generalized Chebyshev Iteration. All the experiments have been performed on a DEC 20, using double precision (unit roundoff of order $10^{-17}$). Any mention to the Generalized Chebyshev iteration refers to algortihm 3 rather than algorithm 4.

### 7.1. Comparison with SYMMLQ

The SYMMLQ algorithm described in [8] and the various versions of it such as MINRES [8] SYMMBK [2] all based on the Lanczos algorithm, are very probably the best known iterative methods for solving large indefinite systems at the present time so it is important to compare the performances of our algorithm with this class of methods. We will mainly consider SYMMLQ although some of the other versions are slightly faster (but also slightly less stable). Although it is difficult to make any definitive conclusion we will see that there are instances where the Generalized Chebyshev iteration has a better performance than SYMMLQ. Table 1 shows the work per iteration and the storage requirements of both algorithms when the number of nonzero elements in the matrix A is equal to NZ (We have not counted the storage required for the matrix A). An obvious advantage of the Generalized Chebyshev iteration is its low storage requirement. Note that with algorithm 4 GCI would require 2N+NZ operations instead of 3N+NZ.

TABLE1

| | Add/Mult-s | Storage |
|---|---|---|
| SYMMLQ | 9 N + NZ | 6 N |
| GCI | 3 N + NZ | 4 N |

Let us discuss the above table under the assumption that NZ = 5 N, which

would correspond for example to applying the inverse iteration method to compute an eigenvector of a Laplace operator. Then it is easily seen that our algorithm becomes competitive if the number steps for GCI does not exceed 1.75 times the number of steps required by SYMMLQ. It is observed in practice that this ratio is seldom reached. In fact as the next experiment will show, when the problem is well conditioned i.e. when both b and c are not too small relative to a and d, then the number of steps is not too different for the two algorithms. For the first experiment we have chosen a diagonal matrix diag($\lambda_i$) where the eigenvalues $\lambda_i$ are distributed uniformely in each interval [a,b] and [c,d]. We have taken N=200, a=-2.0, b=-0.5, c=0.5 and d=6.0. The eigenvalues $\lambda_i$ are defined by:

$$\text{define } i_1 = \lceil N(b-a)/(b-a+d-c) \rceil$$

$$\text{for } i=1,2..i_1: \qquad \lambda_i = a + (i-1).h_1, \text{ with } h_1=(b-a)/(i_1-1) \tag{7.1}$$

$$\text{for } i=i_1+1,...N: \qquad \lambda_i = c + (i-1).h_2, \text{ with } h_2=(d-c)/(N-i_1) \tag{7.2}$$

The right hand side f has been taken equal to f = A e, where $e^T=(1,1,1...1)$. The initial vector was a random vector, the same for both SYMMLQ and GCI. The Generalized Chebyshev iteration using the exact parameters a,b,c,d is run with a residual polynoomial of degree 25. The residual norms are plotted in figure 7-1 in logarithmic scale.This is done every 5 steps for a total of 75 steps. Observe that the behaviors of both algorithms are almost identical.

This example shows the following interesting property: when the origin is well separated from the spectrum the C-G like methods will converge with about the same speed, but each step is more expensive as shown in table 1.

The next test compares the two algorithms in presence of a mildly badly conditioned problem. We have taken the same example as above but the values of b and c have been changed into -0.05 and 0.05 respectively, and N has been increased

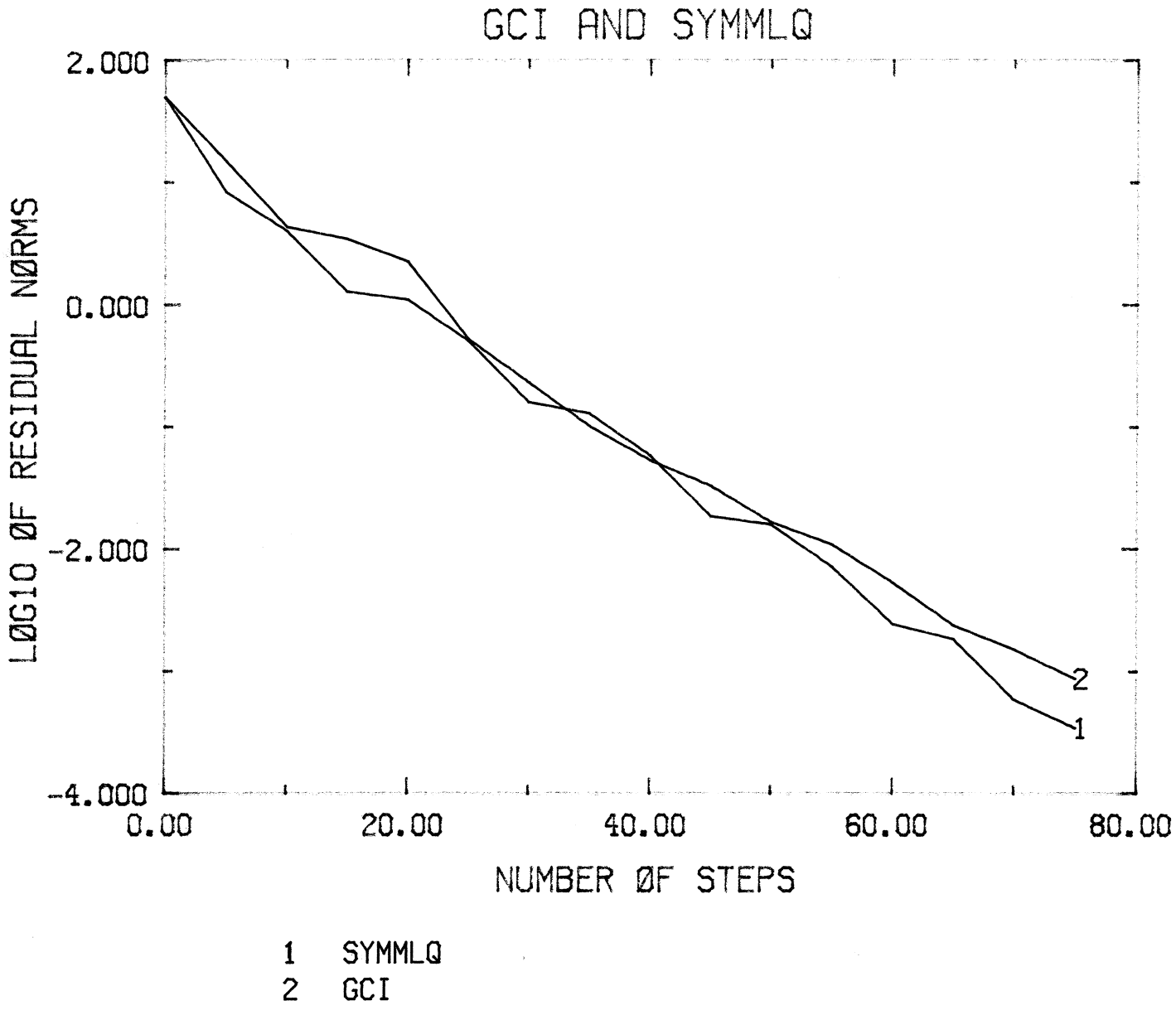


**Figure 7-1:** Comparison of SYMMLQ and GCI. N = 200, a=-2.0,b=-0.5,c=0.5,d=6.0

to 500. The eigenvalues $\lambda_i$ are defined again by 7.1 and 7.2. This is more difficult to solve than our previous example. We have plotted in fig.7-2 the logarithms of the residual norms for a total number of steps of 500. The inner loop of GCI uses a polynomial of degree 250. (therefore 2 outer iterations were taken.) The initial vector has been chosen randomly and has an initial residual norm of.719 E+02. Although SYMMLQ requires less steps here to achieve a reduction of the residual norm by an order of 1.E-06, observe that in the beginning the GCI has a better performance than SYMMLQ. But as the number of steps approaches the dimension N of A, we observe a speeding up of SYMMLQ. Note that in exact arithmectic SYMMLQ would have converged exactly at the 500-th steps. As a comparison we plotted on the same figure the residual norms obtained with GCI using a low degree polynomial. Observe the slowing down of GCI after 130 steps due to the fact that the residuals start being in the direction of some eigenvectors. This shows the superiority of using a higher degree polynomial when possible.

In the above experiments we have assumed that the optimal parameters a,b,c,d are available. This is unrealistic in practice unless another linear system has already been solved with the same matrix A. We would like next to show the effectiveness of the projection procedure described in section 6. In the same example of dimension 500 as above we have taken as extremal parameters the exact values a and d. The interior parameters b and c were initially set to -0.15 and 0.15 respectively, instead of -0.05 and 0.05. We iterated with a polynomial of degree 50. After each (outer ) iteration a Rayleigh Ritz step using the 10 latest vectors $u_i$ as suggested in section 6 was taken. The convergence of the Ritz eigenvalues is very slow in this example because the relative gap between the eigenvalues of A is quite small. At the 9-th iteration no Ritz value has converged to the demanded accuracy $10^{-3}$ and it was decided to take a projection step anyway with those eigenelements which have converged to a relative accuracy of 0.2. The code then gave as eigenvalues $\lambda^-$=-0.0504... and $\lambda^+$=0.0507 and the

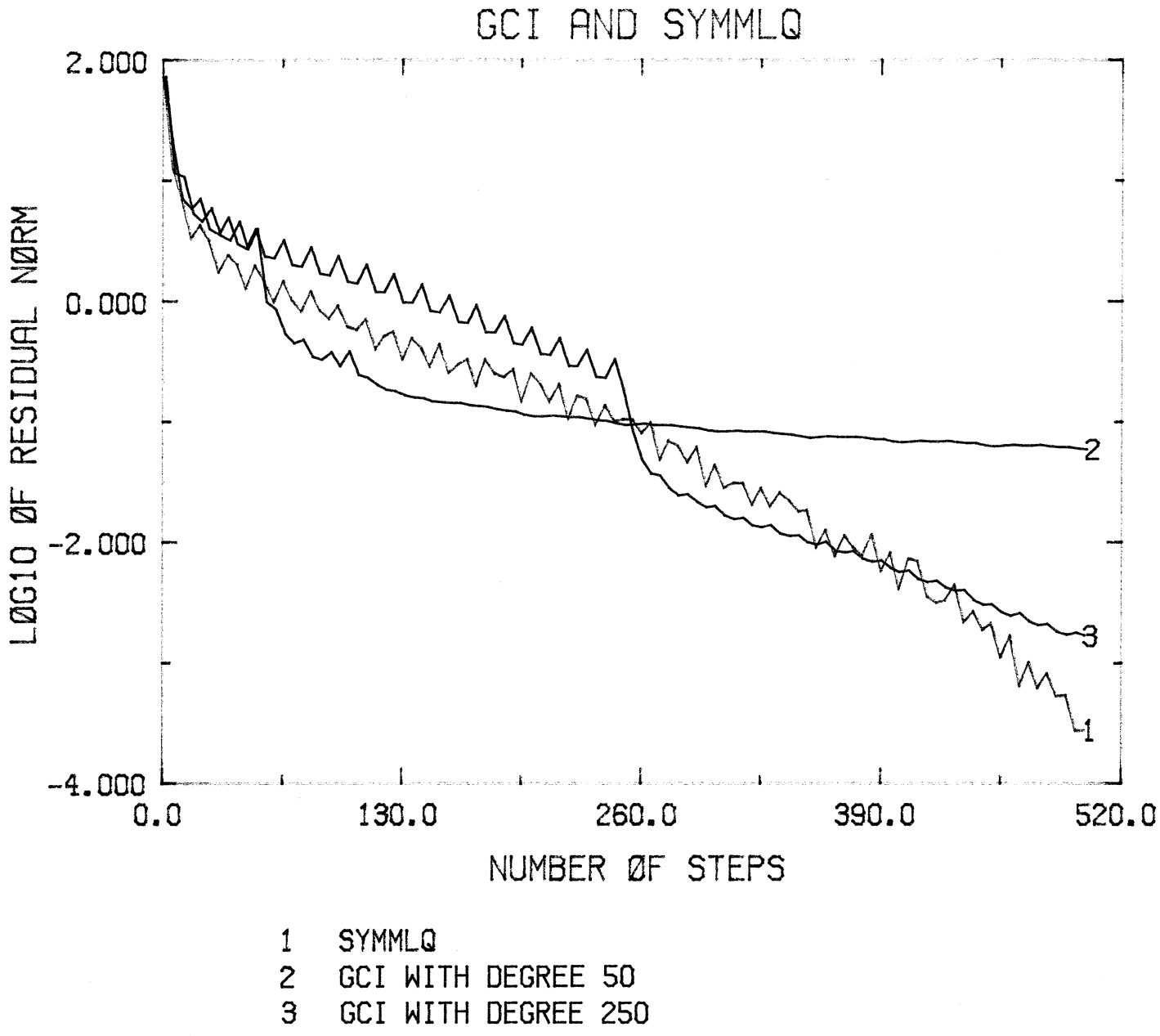


**Figure 7-2:** Comparison of SYMMLQ and GCI. N = 500, a=-2.0,b=-0.05,c=0.05,d=6.0

final (10-th) iteration was pursued with these values for b and c. Figure 7-3 is a plot of the residual norms obtained with this technique together with those previously obtained by using the exact parameters with a residual polynomial of degree 250. It is remarkable that the final results are not too far from those using the optimal parameters.

## 7.2. Computing interior eigenelements.

Consider the following N x N five-point discretization matrix :

$$A = \begin{pmatrix} B & -I & & & \\ -I & B & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -I \\ & & & -I & B \end{pmatrix} \qquad \text{with } B = \begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 4 \end{pmatrix}$$

and dim(A) = 150, dim(B) = 10

By Gershgorin's theorem all the eigenvalues of A lie in the interval [0,8]. Suppose that we would like to compute the eigenvalue nearest to the value 2.5 and its corresponding eigenvector. Assuming that we do not have at our disposal any estimate of the eigenvalues nearest to 2.5, we start the iteration described in section 5 with $\mu$ equal precisely to 2.5. Also as the first estimates of the eigenvalues $\lambda^+$ and $\lambda^-$, the smallest eigenvalue at the right of $\lambda$ and the largest eigenvalue at the left of $\lambda$ respectively, we take 3.0 and 2.0. The initial vector $v_1$ is generated randomly. After each outer iteration using a polynomial of degree 40 a projection step using the last six vectors is taken. The Ritz eigenvalues are computed and compared with those obtained at the previous projection step as described in section 6. When either of the eigenvalues corresponding to $\lambda^+$, $\lambda$, or $\lambda^-$ has converged to a relative accuracy of $10^{-3}$ then b, $\mu$, or c is replaced accordingly. when the three eigenvalues have converged the approximate

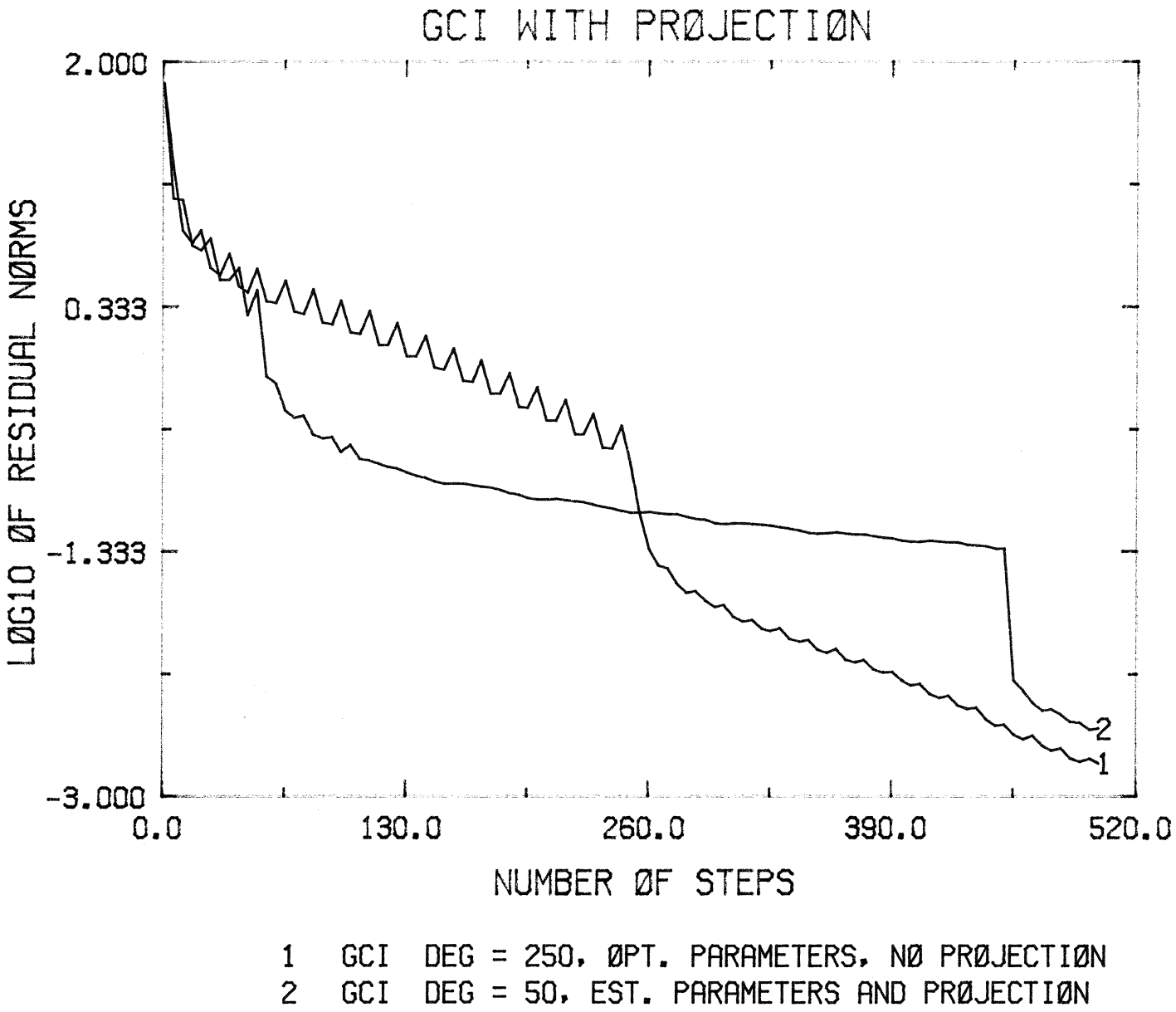


**Figure 7-3:** GCI with dynamical estimation of paramaters and projection. N = 500

eigenvector is replaced by the Ritz vector. Figure 7-4 shows 480 steps of such a process (curve 1). Note the dramatic drop of the residual norm after the projection process. Clearly such an improvement is not only due to the fact that we now use better parameters b and c, but also to the fact that the current approximate eigenvector is now purified from the components in the directions of the undesired eigenvectors.

As a comparison the residual norms are plotted for the case where the exact parameters $\lambda^- = 2.436872$, $\lambda = 2.471196$, and $\lambda^+ = 2.604246$ are used. In this case we iterated with a polynomial of degree 100 (Iteration with a polynomial of degree 40 gave much slower convergence) . Note that the process with orthogonal projection is quite successful here since it does a better job than the algorithm using the optimal parameters and a reasonably high degree polynomial.

## 8. Conclusion.

The numerical experiments suggest that the use of orthogonal polynomials for solving indefinite linear systems as described in this paper can be effective especially if one or more of the following conditions are met:

- The system to solve is not too badly conditioned.
- A moderate accuracy is required
- The operations $y = A x$ are very cheap
- Several systems with the same matrix A are to solved. In that case the parameters are estimated only once .

The Generalized Chebyshev Iteration is a stable process and relatively high degree polynomials can be used without difficulty to achieve a better performance. In order to estimate the optimal parameters we resort to a projection procedure which incidentally can also be used to improve the current approximate solution by removing the undesired components.

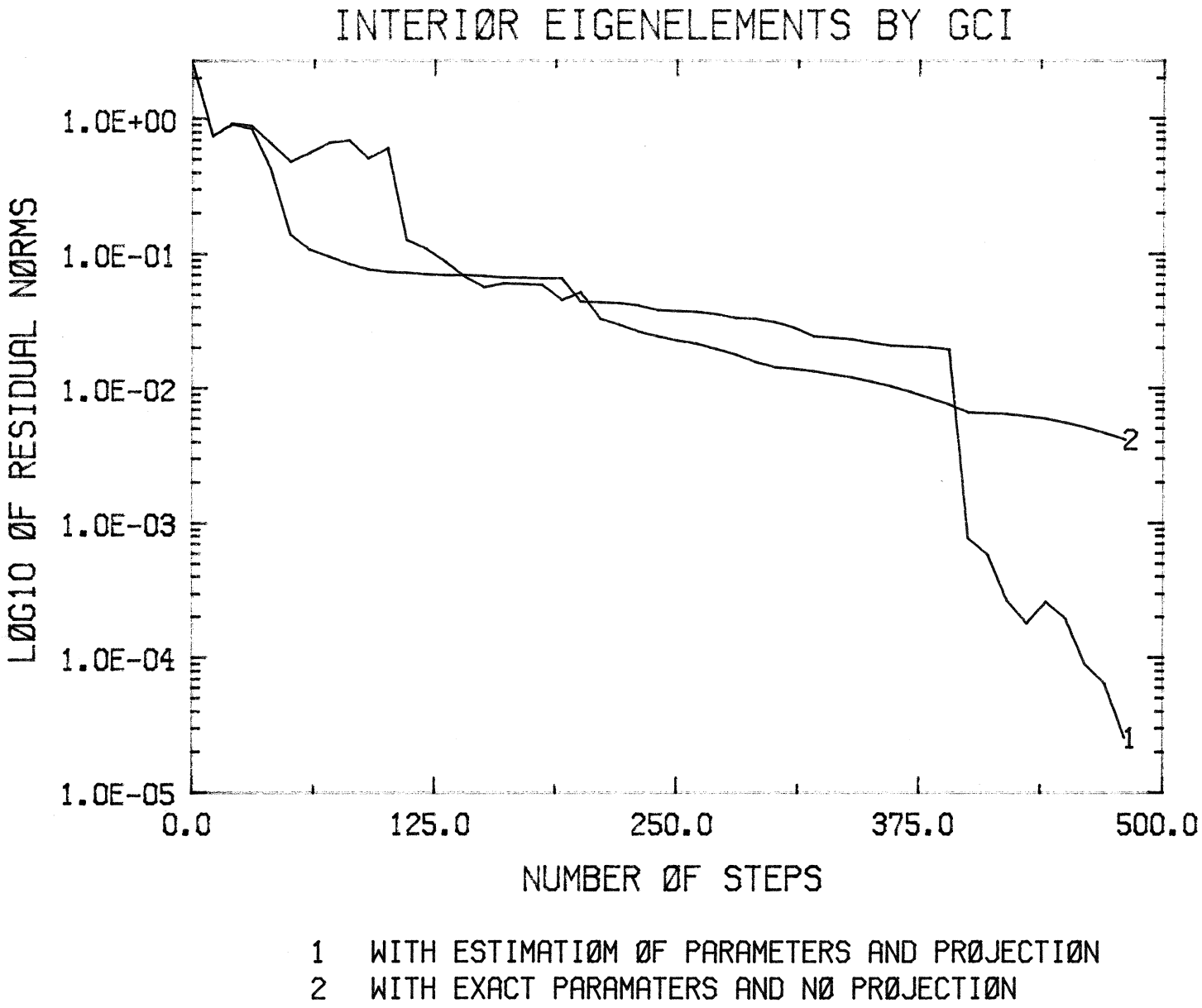


**Figure 7-4: Computing an interior eigenpair with GCI. N=150, $\lambda$ = 2.47119...**

Several problems both theoretical and practical remain to be solved and more numerical experience is needed before this class of techniques will become reliable.

It is also hoped that besides their use in Numerical Linear Algebra, the polynomials introduced in this paper will find applications in other fields of numerical analysis such as in approximation of functions.

Acknowledgements. Most of the ideas of this paper were developed during a visit at the University of California at Berkeley. I would like to express my gratitude to Prof. B.N. Parlett for his hospitality at Berkeley and for valuable comments about this work. I also benefited from helpful discussions with Stan Eisenstat.

## REFERENCES


[1] R.S. Anderson , G.H. Golub. Richardson's non stationary matrix iterative procedure. Technical Report STAN-CS-72-304, Stanford University, 1972.

[2] R. Chandra. Conjugate Gradient Methods for Partial Differential Equations. Technical Report 129, Yale University, 1981. PhD Thesis.

[3] C.C. Cheney. Introduction to Approximation Theory. Mc Graw Hill, N.Y., 1966.

[4] C. de Boor , J.R. Rice. Extremal polynomials with applications to Richardson iteration for indefinite systems. Technical Report 2107, M.R.C., 1980.

[5] G.N. Golub , R.S. Varga. Chebyshev semi iterative methods successive overrelaxation iterative methods and second order Richardson iterative methods. Numer. Mat 3:147-168, 1961.

[6] V.I. Lebedev. Iterative methods for solving operator equations with a spectrum contained in several intervals. USSR Comp Math Math Phys 9(6):17-24, 1969.

[7] D. A. O'Leary. Hybrid Conjugate Gradient algorithms. Computer Science Dpt. Stanford University, Stanford, CA, 1976. PhD dissertation.

[8] C.C. Paige and M.A. Saunders. Solution of sparse indefinite systems of linear equations . SIAM j. of Numer. Anal. 12:617-624, 1975.

[9] B.N. Parlett. The Symmetric Eigenvalue Problem. Prentice Hall, Englewood Cliffs, 1980.

[10] R. R. Roloff. Iterative solution of matrix equations for symmetric matrices possessing positive and negative eigenvalues. Technical Report UIUCDCS R-79-1018, University of Illinois, 1979.

[11] Y. Saad , A. Sameh. A parallel Block Stiefel method for solving positive definite systems. In M.H. Schultz, Editor, Proceedings of the Elliptic Problem Solver Conference, Academic Press, 1980, pp. 405-412.