SIAM J. NUMER. ANAL.
Vol. 20, No. 4, August 1983

0036-1429/83/2004-0009 $01.25/0

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

YOUCEF SAAD†

Abstract. In the classical Chebyshev algorithm for solving symmetric positive definite linear systems $Ax = b$, the approximate solution is of the form $x_k = x_0 + s_k(A)r_0$, where $r_0 = b - Ax_0$ is the initial residual and s_k is a polynomial of degree $k-1$ which minimizes $\|1 - \lambda s_k(\lambda)\|_\infty$, where $\|\cdot\|_\infty$ is the uniform norm on some interval S containing the spectrum of A but not the origin. If A is not positive definite, the set S can be taken to be the union of two disjoint intervals $[a, b]$ and $[c, d]$ with $b < 0 < c$ and a generalization of the Chebyshev algorithm based on the minimax polynomial has been considered by several authors. In this paper we propose a new technique based upon the least squares polynomial in the set S, i.e. the polynomial t_k which minimizes $\|1 - \lambda t_k(\lambda)\|_w$, where $\|\cdot\|_w$ is an L_2 norm with respect to a weight function $w(\lambda)$ defined on S. With a suitable weight function $w(\lambda)$, the polynomials s_k can be computed without numerical integration. The convergence of the algorithm is established.

1. Introduction. This paper is concerned with the solution of large indefinite linear systems $Ax = f$. An obvious way of handling such problems is by solving the normal equations $A^2x = Af$ by any method for 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 [13]. There have also been a few contributions towards the generalization of the Chebyshev iteration algorithm (also called Chebyshev semi-iterative method or Stiefel iteration) to the indefinite case by Lebedev [11], Roloff [17] and de Boor and Rice [5]. The approach suggested by these authors is as follows. Consider the Richardson iteration

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

where r_{n-1} denotes the residual vector of the iterate x_{n-1} and τ_n is some coefficient. Then the residual vector r_n satisfies the equation

$$r_n = p_n(A)r_0$$

where p_n is the polynomial of degree n defined by

$$p_n(\lambda) = (1 - \tau_1\lambda)(1 - \tau_2\lambda) \cdots (1 - \tau_n\lambda).$$

The problem is then to find a set of coefficients $\{\tau_i\}_{i=1,n}$, which makes the residual vector as small as possible. Clearly, the coefficients τ_i can be found as the inverses of the roots of a polynomial $p_n(\lambda)$ satisfying the condition $p_n(0) = 1$. Writing the expansion of the residual vector r_n in the eigenbasis of A, it can be seen that, in order to obtain a small residual vector, one has to choose the polynomial p_n so that it is small on the spectrum of A. For positive definite problems the spectrum of A is included in an interval $[a, b]$ with $a > 0$ and a suitable residual polynomial is the minimax polynomial on $[a, b]$, i.e. the one minimizing the uniform norm on $[a, b]$ among all polynomials p of degree n such that $p(0) = 1$. This choice yields the well known Chebyshev iteration.

* Received by the editors November 20, 1981, and in revised form September 8, 1982. This work was supported in part by the U.S. Office of Naval Research under grant N00014-76-C-0277 and in part by the National Science Foundation under grant MCS-8104874.

† Computer Science Department, Yale University, Box 2158, Yale Station, New Haven, Connecticut 06520.

When A is not positive definite the spectrum of A can be included in two disjoint intervals $[a, b]$ and $[c, d]$ with $b < 0 < c$ and the problem is transformed into that of finding the polynomial p_n of degree n satisfying $p_n(0) = 1$ and having the least uniform norm in $[a, b] \cup [c, d]$. The problem of computing such polynomials has been considered by Lebedev [11], Roloff [17], de Boor and Rice [5] and Grcar [9].

However, the above approach has a few drawbacks. In particular, the minimax polynomial, which is no longer related to the Chebyshev polynomials, may be difficult to obtain numerically especially if the degree is high. The computation of its roots, which are needed in the Richardson iteration, can also be troublesome. Moreover, the above Richardson iteration is unstable in general, and requires a special renumbering of the roots [1].

Although the above-mentioned difficulties are not unsurmountable, 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 [5], 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 minimizing an L_2 norm associated with a certain weight function over the two intervals. Such least squares polynomials are easily obtained via a sequence of orthogonal polynomials satisfying a three term recurrence. Moreover, it will be shown that the least squares polynomial of degree n has a uniform norm no larger than $2(n+1)^{1/2}$ times the uniform norm of the minimax polynomial of the same degree. In fact, Johnson, Micchelli and Paul observed, in the positive definite context, that least squares polynomials can yield faster convergence than the minimax polynomials [10].

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 least squares polynomial. Unlike the SYMMLQ algorithm, our method requires the knowledge of some eigenvalues of A. On the other hand, each step requires only $3N$ multiplications versus $9N$ for SYMMLQ. We will see that the performance of our algorithm can be significantly enhanced by combining it with a projection process which provides estimates of the desired parameters and purifies the solution.

A few of the possible generalizations and extensions of the basic generalized Chebyshev iteration will be briefly outlined. Thus, a block version of the algorithm can be formulated. It is essentially a generalization of the block Stiefel algorithm described in [21] and is particularly suitable for parallel computation. Also, there are various ways of using the least squares polynomials for computing eigenvectors associated with interior eigenvalues.

Section 2 describes polynomial iteration for solving linear systems. In § 3 we will deal with the computation of orthogonal polynomials over two intervals and present a first algorithm for solving indefinite linear systems. Section 4 introduces the least squares polynomials and § 5 covers their application to the solution of symmetric indefinite linear systems. Section 6 outlines some applications to the interior eigenvalue problems. Some practical details, concerning in particular the estimation of the optimal parameters, are given in § 7. Finally, a few numerical experiments are reported in § 8.

2. Polynomial iteration for linear systems. Consider the $N \times N$ linear system

$$Ax = f, \tag{1}$$

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 $r_0 = f - Ax_0$. Consider a method in which the nth approximation x_n is given by

$$x_n = x_{n-1} + d_n \tag{2}$$

where d_n is a linear combination of the previous residual vectors. It can easily be shown by induction that the residual vectors $r_n = f - Ax_n$ satisfy:

$$r_n = p_n(A) r_0 \tag{3}$$

where p_n is a polynomial of degree n such that

$$p_n(0) = 1. \tag{4}$$

Methods that satisfy the conditions (2), (3), and (4) will be referred to as polynomial iteration methods. They were discussed at length by Rutishauser [19].

The polynomial p_n is known as *the residual polynomial.* Noticing that $p_n(\lambda)$ can be written as

$$p_n(\lambda) = 1 - \lambda s_{n-1}(\lambda) \tag{5}$$

where s_{n-1} is some polynomial of degree $n-1$, it is easy to show that the approximation x_n is given by

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

The polynomial s_{n-1} will often be referred to as *the solution polynomial.*

Let μ_i, $i = 1, N$ be the eigenvalues of A with associated orthonormal eigenvectors z_i, $i = 1, \cdots, N$. If r_0 is written in the eigenbasis $\{z_i\}_{i=1,N}$ as $r_0 = \sum_{i=1}^{N} \xi_i z_i$ then the Euclidean norm of r_n is

$$\|r_n\| = \left[\sum_{i=1}^{N} p_n(\mu_i)^2 \xi_i^2 \right]^{1/2}. \tag{6}$$

Equation (6) 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. Let S be a compact set containing the spectrum of A but excluding the origin, and define the uniform norm of p in S by:

$$\|p\|_\infty = \max_{\lambda \in S} |p(\lambda)|. \tag{7}$$

One way of choosing a residual polynomial which is small in S is to take among the polynomials satisfying $p_n(0) = 1$ the one with smallest uniform norm in S. Such a polynomial is referred to as the *minimax polynomial.* In this case the smallness of p_n on the spectrum of A is measured by means of the uniform norm (7). However, as will be seen shortly, the L_2 norm with respect to some weight function $w(\lambda)$ presents some advantages over the uniform norm in the indefinite case.

When the matrix A is positive definite its eigenvalues can be included in a single interval $S = [a, b]$ 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 [8].

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 least uniform norm in $[a, b] \cup [c, d]$ can no longer be explicitly expressed in general. Lebedev [11] considered this problem and

gave an explicit solution to it for the 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. More recently, de Boor and Rice [5] derived a method for computing the minimax polynomial in two general intervals by a Remez-like algorithm.

An alternative, proposed in this paper, is to use 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 $w(\lambda)$. This will necessitate the use of orthogonal polynomials over two intervals which we discuss in detail in the next section.

3. Orthogonal polynomials over two disjoint intervals.

3.1. Basic definitions and notation. In this subsection we will discuss 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 $w(\lambda)$ defined on the interval (a, d) by

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

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

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

or

$$\langle f, g\rangle = \int_a^b f(\lambda)g(\lambda)w_1(\lambda)\, d\lambda + \int_c^d f(\lambda)g(\lambda)w_2(\lambda)\, d\lambda \equiv \langle f, g\rangle_1 + \langle f, g\rangle_2. \tag{8}$$

The norm associated with this inner product will be denoted by $\|\cdot\|_w$, i.e., $\|f\|_w = \langle f, f\rangle^{1/2}$.

The theory of orthogonal polynomials shows that there is a sequence $\{p_n\}$ of polynomials which is orthogonal with respect to $\langle \cdot, \cdot \rangle$. All of the properties of the orthogonal polynomials in one interval hold for the case of two intervals, provided we consider these polynomials as being orthogonal on (a, d) rather than on $(a, b) \cup (c, d)$. For example, although we cannot assert that the roots are all in $[a, b] \cup [c, d]$, 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 fundamental 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:

$$\beta_{n+1}p_{n+1}(\lambda) = (\lambda - \alpha_n)p_n(\lambda) - \beta_n p_{n-1}(\lambda) \tag{9}$$

with

$$\alpha_n = \langle \lambda p_n, p_n\rangle, \qquad \beta_{n+1} = \|(\lambda - \alpha_n)p_n(\lambda) - \beta_n p_{n-1}(\lambda)\|_w.$$

As a consequence, the sequence of orthogonal polynomials can be computed from the algorithm:

ALGORITHM 1

1. Start. $\tilde{p}_0(\lambda) := 1$, $p_0(\lambda) := \tilde{p}_0(\lambda)/\|\tilde{p}_0\|_w$, $\beta_0 := 0$, $p_{-1} := 0$

2. Iterate. For $n = 0, 1, 2, \cdots$ do

(10) • Compute $\alpha_n = \langle \lambda p_n, p_n \rangle$

(11) • Compute the polynomial $\tilde{p}_{n+1}(\lambda) = (\lambda - \alpha_n) p_n(\lambda) - \beta_n p_{n-1}(\lambda)$

(12) • Compute $\beta_{n+1} = \|\tilde{p}_{n+1}\|_w = \langle \tilde{p}_{n+1}, \tilde{p}_{n+1} \rangle^{1/2}$

(13) • Compute $p_{n+1}(\lambda) = \dfrac{1}{\beta_{n+1}} \tilde{p}_{n+1}(\lambda)$

The above algorithm is known as the Stieltjes procedure in the literature [6]. Observe that numerical integration is normally required at each step n, in order to compute the coefficients α_n and β_{n+1}. The discretized Stieltjes procedure, due to Gautschi [6], [7], consists in replacing the integrals by suitable quadrature rules.

One important problem considered in this paper is to realize, for some suitable weight functions w_1 and w_2, the above Stieltjes procedure without resorting to numerical integration. This is discussed below.

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

(14) $$w_1(\lambda) = (2/\pi)[d_1^2 - (\lambda - c_1)^2]^{-1/2} \qquad \text{for } \lambda \in (a, b),$$

(15) $$w_2(\lambda) = (2/\pi)[d_2^2 - (\lambda - c_2)^2]^{-1/2} \qquad \text{for } \lambda \in (c, d).$$

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

(16) $$T_n[(\lambda - c_1)/d_1]$$

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

(17) $$T_n[(\lambda - c_2)/d_2].$$

We are interested in the polynomials $p_n(\lambda)$ that are orthogonal in $(a, b) \cup (c, d)$ with respect to the weight function $w(\lambda)$ defined by w_1 in (a, b) and w_2 in (c, d). In order to generate these polynomials, let us express them in two different bases, namely the two bases of polynomials (16) and (17):

(18) $$p_n(\lambda) = \sum_{i=0}^{n} \gamma_i^{(n)} T_i[(\lambda - c_1)/d_1],$$

(19) $$p_n(\lambda) = \sum_{i=0}^{n} \delta_i^{(n)} T_i[(\lambda - c_2)/d_2].$$

Note that the above two expansions are redundant since they express the *same polynomial*. However, even though one of the expansions can theoretically be obtained from the other, the process of changing bases is likely to be unstable and expensive and will not be considered. The following proposition permits to compute the coefficients α_n and β_{n+1} of the three term recurrence (9), given the expansions (18) and (19).

PROPOSITION 1. *Let p be any polynomial of degree n having the following two expansions*:

$$p(\lambda) = \sum_{i=0}^{n} \gamma_i T_i[(\lambda - c_1)/d_1], \tag{20}$$

$$p(\lambda) = \sum_{i=0}^{n} \delta_i T_i[(\lambda - c_2)/d_2]. \tag{21}$$

Set

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

$$\tau_1 = 2\gamma_0\gamma_1 + \sum_{i=1}^{n-1} \gamma_i\gamma_{i+1}, \qquad \tau_2 = 2\delta_0\delta_1 + \sum_{i=1}^{n-1} \delta_i\delta_{i+1}.$$

Then

$$\langle p, p\rangle = \sigma_1 + \sigma_2, \tag{22}$$

$$\langle \lambda p, p\rangle = c_1\sigma_1 + c_2\sigma_2 + d_1\tau_1 + d_2\tau_2. \tag{23}$$

Proof. Consider (22) first. From the definition (8) and from (20), (21) we get:

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

Using the change of variable $\xi = (\lambda - c_1)/d_1$ 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 similar transformations we would obtain $\langle p, p\rangle_2 = \sigma_2$, and (22) follows from the definition (8).

In order to prove (23) consider $\langle \lambda p, p\rangle_1$ and $\langle \lambda p, p\rangle_2$ separately. We have:

$$\langle \lambda p, p\rangle_1 = d_1 \left\langle \frac{\lambda - c_1}{d_1} p, p \right\rangle_1 + c_1 \langle p, p\rangle_1. \tag{24}$$

The first inner product can be transformed as follows:

$$\left\langle \frac{\lambda - c_1}{d_1} p, p \right\rangle_1 = \left\langle \sum \gamma_i \frac{\lambda - c_1}{d_1} T_i\left[\frac{\lambda - c_1}{d_1}\right], \sum \gamma_i T_i\left[\frac{\lambda - c_1}{d_1}\right] \right\rangle_1$$

where $\sum$ stands for $\sum_{i=0}^{n}$. By the change of variables $\xi = (\lambda - c_1)/d_1$ this simplifies into

$$\left\langle \frac{\lambda - c_1}{d_1} p, p \right\rangle_1 = \frac{2}{\pi} \int_{-1}^{+1} \left[\sum \gamma_i \xi T_i(\xi)\right]^2 (1 - \xi^2)^{-1/2}\, d\xi. \tag{25}$$

Using the relations:

$$\xi T_i(\xi) = \tfrac{1}{2}[T_{i+1}(\xi) + T_{i-1}(\xi)], \qquad \xi T_0(\xi) = T_1(\xi),$$

we find that

$$\sum \gamma_i \xi T_i(\xi) = \tfrac{1}{2}\gamma_1 T_0(\xi) + (\gamma_0 + \tfrac{1}{2}\gamma_2) T_1(\xi) + \cdots + \tfrac{1}{2}(\gamma_{i-1} + \gamma_{i+1}) T_i(\xi) + \cdots + \tfrac{1}{2}(\gamma_{n-1} + \gamma_{n+1}) T_n(\xi).$$

We define γ_{n+1} as being equal to zero for convenience. Replacing this in (25) and using the orthogonality of the Chebyshev polynomials with respect to the weight function $(1-\xi^2)^{-1/2}$ we get

$$\left\langle \frac{\lambda - c_1}{d_1} p, p \right\rangle_1 = \tfrac{1}{2}\gamma_1\gamma_0\kappa_0 + (\gamma_1 + \tfrac{1}{2}\gamma_2)\gamma_1\kappa_1 + \tfrac{1}{2}(\gamma_{i+1} + \gamma_{i-1})\gamma_i\kappa_i + \cdots + \tfrac{1}{2}(\gamma_{n+1} + \gamma_{n-1})\gamma_n\kappa_n$$

with $\kappa_0 = 2$ and $\kappa_i = 1$ for $i \neq 0$. This finally yields

$$\left\langle \frac{\lambda - c_1}{d_1} p, p \right\rangle = \tau_1,$$

and by (22)

$$\langle \lambda p, p \rangle_1 = c_1\sigma_1 + d_1\tau_1. \tag{26}$$

A similar calculation would give

$$\langle \lambda p, p \rangle_2 = c_2\sigma_2 + d_2\tau_2. \tag{27}$$

Finally, adding (26) and (27) yields the desired result (23). □

Once the parameters α_n and β_{n+1} are computed by an appropriate use of (22) and (23), it is possible to generate the next orthogonal polynomial $p_{n+1}(\lambda)$ by using formula (9). The purpose of the next proposition is to show how to do so, when the expansions (18) and (19) are available.

PROPOSITION 2. *Let $p_n(\lambda)$, $n = 0, 1, \cdots$, be the sequence of orthogonal polynomials satisfying the three term recurrence relation* (9). *Then the coefficients $\gamma_i^{(n)}$ of the expansion* (18) *satisfy the relations*

$$\beta_{n+1}\gamma_i^{(n+1)} = \frac{d_1}{2}(\gamma_{i-1}^{(n)} + \gamma_{i+1}^{(n)}) + (c_1 - \alpha_n)\gamma_i^{(n)} - \beta_n\gamma_i^{(n-1)}, \qquad i = 2, 3, \cdots, n+1, \tag{28}$$

starting with

$$\beta_{n+1}\gamma_0^{(n+1)} = d_1\gamma_1^{(n)} + (c_1 - \alpha_n)\gamma_0^{(n)} - \beta_n\gamma_0^{(n-1)}, \tag{29}$$

$$\beta_{n+1}\gamma_1^{(n+1)} = d_1(\gamma_0^{(n)} + \tfrac{1}{2}\gamma_2^{(n)}) + (c_1 - \alpha_n)\gamma_1^{(n)} - \beta_n\gamma_1^{(n-1)}. \tag{30}$$

Relations analogous to (28), (29) *and* (30), *in which c_1 is replaced by c_2 and d_1 by d_2, hold for the coefficients $\delta_i^{(n)}$ of the expansion* (19).

Proof. From (9) we have

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

where $\sum$ stands for $\sum_{i=0}^{n}$. Using again the change of variables $\xi = (\lambda - c_1)/d_1$ for convenience we get

$$\beta_{n+1}p_{n+1}(\lambda) = (d_1\xi + c_1 - \alpha_n)\sum \gamma_i^{(n)}T_i(\xi) - \beta_n \sum \gamma_i^{(n-1)}T_i(\xi).$$

Noticing that $\xi T_i(\xi) = \frac{1}{2}[T_{i-1}(\xi) + T_{i+1}(\xi)]$ when $i \geqq 1$ and $\xi T_0(\xi) = T_1(\xi)$ we obtain:

$$\beta_{n+1}p_{n+1}(\lambda) = \frac{d_1}{2}\sum \gamma_i^{(n)}[T_{i+1}(\xi) + T_{i-1}(\xi)] + (c_1 - \alpha_n)\sum \gamma_i^{(n)}T_i(\xi) - \beta_n \sum \gamma_i^{(n-1)}T_i(\xi)$$

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

$$\begin{aligned}\beta_{n+1}p_{n+1}(\lambda) = {} & d_1[\gamma_1^{(n)} + (\gamma_0^{(n)} + \gamma_2^{(n)}/2)T_1(\xi) + \textstyle\sum \tfrac{1}{2}(\gamma_{i-1}^{(n)} + \gamma_{i+1}^{(n)})T_i(\xi)] \\ & + (c_1 - \alpha_n)\sum \gamma_i^{(n)}T_i(\xi) - \beta_n \sum \gamma_i^{(n-1)}T_i(\xi).\end{aligned}$$

Returning to the λ variable gives the desired expansion of p_{n+1} in the form (18) and (19) and establishes the result. □

The previous two propositions can now be combined with the Stieltjes procedure (Algorithm 1) to derive an algorithm for computing the sequence of orthogonal polynomials. At a given step n the polynomial p_n is available through its expansions (18) and (19) so one can compute α_n by formula (23). Then we can compute the expansion coefficients of $\tilde{p}_{n+1}$ using Proposition 2. Finally β_{n+1} is obtained from (12) and (23) and $\tilde{p}_{n+1}$ is normalized by β_{n+1} to yield the next polynomial p_{n+1} as in (13). A notational simplification is achieved by introducing the symbol $\sum'$ defined by

$$\sum_{i=0}^{i=n}{}' a_i = 2a_0 + \sum_{i=1}^{i=n} a_i. \tag{31}$$

ALGORITHM 2

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

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

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

- Compute α_n from (10) and (23):

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

- Compute expansion coefficients of $\tilde{p}_{n+1}$ by formulas (28)–(30) and their analogue for the δ's:

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

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

$$\tilde{\gamma}_i^{(n+1)} := \frac{d_1}{2}(\gamma_{i-1}^{(n)} + \gamma_{i+1}^{(n)}) + (c_1 - \alpha_n)\gamma_i^{(n)} - \beta_n\gamma_i^{(n-1)}, \qquad i = 2, 3, \cdots, n+1$$

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

$$\tilde{\delta}_1^{(n+1)} := d_2(\delta_0^{(n)} + \tfrac{1}{2}\delta_2^{(n)}) + (c_2 - \alpha_n)\delta_1^{(n)} - \beta_n\delta_1^{(n-1)}$$

$$\tilde{\delta}_i^{(n+1)} := \frac{d_2}{2}(\delta_{i-1}^{(n)} + \delta_{i+1}^{(n)}) + (c_2 - \alpha_n)\delta_i^{(n)} - \beta_n\delta_i^{(n-1)}, \qquad i = 2, 3, \cdots, n+1$$

- Compute $\beta_{n+1} = \|\tilde{p}_{n+1}\|$ from (12) and (22)

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

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

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

$$\gamma_i^{(n+1)} := \tilde{\gamma}_i^{(n+1)}/\beta_{n+1}, \quad \delta_i^{(n+1)} := \tilde{\delta}_i^{(n+1)}/\beta_{n+1}, \quad i = 0, 1, \cdots, n+1$$

- Compute new σ's and τ'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 nth step of the algorithm one must perform approximately $20(n+1)$ arithmetic operations. Concerning the storage requirements, we must save the coefficients $\gamma_i^{(n)}$, $\delta_i^{(n)}$ as well as the previous coefficients $\gamma_i^{(n-1)}$, and $\delta_i^{(n)}$, 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 compounding of low degree polynomials could be used if necessary.

3.3. Application to the solution of linear systems. Consider the orthonormal polynomial p_n of degree n produced by Algorithm 2. The normalized polynomial $t_n(\lambda)=p_n(\lambda)/p_n(0)$ is of degree n and satisfies the constraint of being equal to one at the origin. According to § 2, the approximate solution x_n for which the residual is given by $r_n=t_n(A)r_0$, will be a good approximation to the solution x if $t_n(\lambda)$ is small on the set $[a, b]\cup[c, d]$ containing the spectrum of A. From the theory of orthogonal polynomials, it is known that the polynomial obtained by normalizing p_n by its leading coefficient, minimizes the norm $\|\cdot\|_w$ over all *monic* polynomials of degree n. This implies that p_n is small in a certain sense, although no quantitative result seems to be known for the polynomial t_n. The corresponding approximation x_n is given by $x_n = x_0+s_{n-1}(A)r_0$ where $s_{n-1}(\lambda)=[1-t_n(\lambda)]/\lambda$. This relation allows to take advantage of the term recurrence and to derive a suitable algorithm but we omit the details because, as is explained below, this is not reliable and a better alternative will be presented in the next section.

The disadvantages of this approach are both theoretical and practical. First, $p_n(0)$ may vanish or be close to zero. In this case, the nth approximation x_n either is not defined or becomes a poor approximation to the solution respectively. Although there exists a possibility of overcoming this difficulty, namely by computing x_{n+1} directly from x_{n-1}, it will not be considered.

A second disadvantage is that the orthogonal polynomials t_n do not satisfy a suitable optimality property. More precisely, the norm $\|t_n\|_w$ is not minimum over all polynomials t of degree not exceeding n satisfying $t(0)=1$. This fact deprives us of any means of comparing the residual norms, and, above all, of proving the convergence of x_n to the exact solution. The next section describes a better alternative based on a slightly different class of polynomials.

4. The least squares polynomial. The above remarks lead us to seek a residual polynomial which minimizes the norm $\|p\|_w$ over all polynomials p of degree $\leqq n$, satisfying $p(0)=1$. Denoting by $\mathbf{P}_{n-1}$ the space of polynomials of degree not exceeding $n-1$ and writing the polynomials p as $p(\lambda)=1-\lambda s(\lambda)$ where s, the solution polynomial, belongs to $\mathbf{P}_{n-1}$, the problem can be reformulated as follows:

Find a polynomial $s_{n-1}^*\in\mathbf{P}_{n-1}$ *such that*:

$$\|1-\lambda s_{n-1}^*(\lambda)\|_w \leqq \|1-\lambda s(\lambda)\|_w \quad \forall s\in\mathbf{P}_{n-1}. \tag{32}$$

Here $1-\lambda s(\lambda)$ denotes the polynomial p defined by $p(\lambda)\equiv 1-\lambda s(\lambda)$ rather than its value at λ. If we denote by $\mathbf{P}_{n-1}^1$, the space of all polynomials of the form $\lambda s(\lambda)$ with $s\in\mathbf{P}_{n-1}$, then the above problem can be interpreted as a least squares approximation problem, whereby we seek the element $\lambda s_{n-1}^*(\lambda)$ of $\mathbf{P}_{n-1}^1$ which is the best approximation to the function $f(\lambda)\equiv 1$, over the set $[a, b]\cup[c, d]$, in the w-norm sense.

From approximation theory, it is known that the best polynomial $\lambda s_{n-1}^*(\lambda)$ exists and is unique [3]. Furthermore, by least squares theory, it is known that the residual polynomial $1-\lambda s_{n-1}^*(\lambda)$ must be orthogonal to the space $\mathbf{P}_{n-1}^1$, i.e. the desired *least squares solution polynomial* $s_{n-1}^*(\lambda)$ must satisfy the normal equations:

$$\langle 1-\lambda s_{n-1}^*, t(\lambda)\rangle = 0 \quad \forall t\in\mathbf{P}_{n-1}^1$$

or equivalently,

$$\langle 1-\lambda s_{n-1}^*(\lambda), \lambda s(\lambda)\rangle = 0 \quad \forall s \in \mathbf{P}_{n-1}. \tag{33}$$

4.1. An algorithm for computing the least squares polynomial. In order to compute the least squares solution polynomial $s_{n-1}^*(\lambda)$ from the normal equations (33), let us consider the sequence of polynomials q_j, $j = 0, 1, \cdots, n-1, \cdots$, such that the polynomials $\lambda q_j(\lambda)$ form an orthonormal sequence with respect to $\langle \cdot, \cdot \rangle$. Expanding s_{n-1}^* in the basis $\{q_j\}_{j=0,\cdots,n-1}$ as:

$$s_{n-1}^*(\lambda) = \sum_{j=0}^{n-1} \eta_j q_j(\lambda),$$

we obtain from (33) that

$$\eta_j = \langle 1, \lambda q_j(\lambda)\rangle. \tag{34}$$

We will have more to say on the practical computation of these coefficients in a moment.

We now show how to obtain the sequence of polynomials $\{q_j\}$, or equivalently the sequence of orthogonal polynomials $\{\lambda q_j(\lambda)\}$. Clearly, the sequence of polynomials of the form $\{\lambda q_j(\lambda)\}$, which is orthogonal with respect to *the same weight function* $w(\lambda)$ *as before*, can be obtained by Algorithm 2 starting with the polynomial λ instead of 1. Note that the sequence $\{q_j\}_{j=0,1,\cdots,n}$ is also orthogonal with respect to the weight $\lambda^2 w(\lambda)$. As in § 3.2 let us express the polynomials $\lambda q_n(\lambda)$ by two different formulas:

$$\lambda q_n(\lambda) = \sum_{i=0}^{n+1} \gamma_i^{(n)} T_i[(\lambda - c_1)/d_1], \tag{35}$$

$$\lambda q_n(\lambda) = \sum_{i=0}^{n+1} \delta_i^{(n)} T_i[(\lambda - c_2)/d_2]. \tag{36}$$

Denoting again by α_n and β_n the coefficients of the new three term recurrence which is satisfied by the orthogonal polynomials $\lambda q_n(\lambda)$ we obtain:

$$\beta_{n+1}\lambda q_{n+1}(\lambda) = (\lambda - \alpha_n)\lambda q_n(\lambda) - \beta_n \lambda q_{n-1}(\lambda) \tag{37}$$

or, equivalently:

$$\beta_{n+1} q_{n+1}(\lambda) = (\lambda - \alpha_n) q_n(\lambda) - \beta_n q_{n-1}(\lambda) \tag{38}$$

with

$$\alpha_n = \langle \lambda(\lambda q_n), \lambda q_n\rangle, \tag{39}$$

$$\beta_{n+1} = \|\tilde{\lambda} q_n\|_w = \|(\lambda - \alpha_n)\lambda q_n(\lambda) - \beta_n \lambda q - (n-1)(\lambda)\|_w. \tag{40}$$

Since the quantities α_n, β_n, $\gamma_i^{(n)}$ and $\delta_i^{(n)}$ of the previous section will no longer be referenced in the remainder of the paper, we will use these symbols for the new polynomial $\lambda q_n(\lambda)$. This leads to the following adaptation of Algorithm 2 to the computation of the sequence of orthogonal polynomials $\{\lambda q_n(\lambda)\}$.

ALGORITHM 3

1. Initialize. Choose the polynomial $\lambda q_0(\lambda)$ in such a way that q_0 is a constant and $\|\lambda q_0\|_w = 1$:

$$t2 := 2(c_1^2 + c_2^2) + d_1^2 + d_2^2, \qquad t := t2^{1/2}$$

$$\gamma_0^{(0)} := c_1/t, \delta_0^{(0)} := c_2/t, \qquad \gamma_1^{(0)} := d_1/t, \delta_1^{(0)} := d_2/t$$

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

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

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

- Compute α_n by formula (23):

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

- Compute expansion coefficients of $\tilde{p}_{n+1}$ by formulas (28)–(30) and their analogue for the δ's.

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

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

$$\tilde{\gamma}_i^{(n+1)} := \frac{d_1}{2}(\gamma_{i-1}^{(n)} + \gamma_{i+1}^{(n)}) + (c_1 - \alpha_n)\gamma_i^{(n)} - \beta_n\gamma_i^{(n-1)}, \qquad i = 2, 3, \cdots, n+2$$

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

$$\tilde{\delta}_1^{(n+1)} := d_2(\delta_0^{(n)} + \tfrac{1}{2}\delta_2^{(n)}) + (c_2 - \alpha_n)\delta_1^{(n)} - \beta_n\delta_1^{(n-1)}$$

$$\tilde{\delta}_1^{(n+1)} := \frac{d_2}{2}(\delta_{i-1}^{(n)} + \delta_{i+1}^{(n)}) + (c_2 - \alpha_n)\delta_i^{(n)} - \beta_n\delta_i^{(n-1)}, \qquad i = 2, 3, \cdots, n+2$$

- Compute $\beta_{n+1} = \|\tilde{p}_{n+1}\|$ by formula (22)

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

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

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

$$\gamma_i^{(n+1)} := \tilde{\gamma}_i^{(n+1)}/\beta_{n+1}, \qquad \delta_i^{(n+1)} := \tilde{\delta}_i^{(n+1)}/\beta_{n+1}, \quad i = 0, 1, \cdots, n+2$$

- Compute new σ's and τ's

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

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

Notice that part 2 (iterate) is identical with part 2 of Algorithm 2. We will now show how to obtain the expansion coefficients η_j of the polynomial s_{n-1}^* in the basis $\{q_j\}$, and how to compute s_{n-1}^*. According to (34) we have $\eta_j = \langle 1, \lambda q_j\rangle$. Denoting by $T_n^{(1)}$ the polynomials defined by (16) and by $T_n^{(2)}$ the polynomials defined by (17), and recalling that the weight function w has been scaled such that $\langle T_0^{(i)}, T_0^{(i)}\rangle_i = 2$, $i = 1, 2$ and $\langle T_j^{(i)}, T_j^{(i)}\rangle_i = 1$, for $j \neq 0$, $i = 1, 2$, we obtain:

$$\eta_j = \langle 1, \lambda q_j(\lambda)\rangle = \langle T_0^{(1)}, \lambda q_j(\lambda)\rangle_1 + \langle T_0^{(2)}, \lambda q_j(\lambda)\rangle_2$$

hence

$$\eta_j = 2(\gamma_0^{(j)} + \delta_0^{(j)}). \tag{41}$$

As a consequence of (41) the expansion coefficients η_j can be obtained at no expense since the scalars $\gamma_0^{(j)}$ and $\delta_0^{(j)}$ are available when performing Algorithm 3.

The above algorithm does not yield directly the polynomials q_n but rather provides their expansion coefficients in the expansions (35)–(36). In order to generate the least squares solution polynomial $s_n^*(\lambda)$, we can couple with part 2 of the above algorithm

the following computation:

$$\eta_n := 2(\gamma_0^{(n)} + \delta_0^{(n)}), \tag{42}$$

$$s_n^*(\lambda) := s_{n-1}^*(\lambda) + \eta_n q_n(\lambda), \tag{43}$$

$$q_{n+1}(\lambda) := \frac{1}{\beta_{n+1}}[(\lambda - \alpha_n)q_n(\lambda) - \beta_n q_n(\lambda)], \tag{44}$$

starting with

$$q_0(\lambda) = 1/t, \qquad s_0(\lambda) = 2(\gamma_0^{(0)} + \delta_0^{(0)})q_0(\lambda) \tag{45}$$

where t, defined in the initialization part of the algorithm, is equal to $\|\lambda\|_w$. These relations are useful for computing the approximate solution x_n of the linear system, and will be exploited in § 5. Note also that there is an alternative for computing the solution polynomial by use of the theory of kernel polynomials [4], [23].

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

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

Let us denote by p_n^* the least squares polynomial $1 - \lambda s_{n-1}^*(\lambda)$ of degree n, defined in § 4, and by p_n^∞ the polynomial of degree n of smallest uniform norm in $[a, b] \cup [c, d]$. In other words we have

$$p_n^*(0) = p_n^\infty(0) = 1,$$

$$\|p_n^*\|_w \leqq \|p\|_w \quad \forall p \in \mathbf{P}_n, \quad p(0) = 1, \tag{46}$$

$$\|p_n^\infty\|_\infty \leqq \|p\|_\infty \quad \forall p \in \mathbf{P}_n, \quad p(0) = 1. \tag{47}$$

Since $\mathbf{P}_n$ is a finite dimensional space all the norms of $\mathbf{P}_n$ are equivalent and the following lemma compares precisely the norms $\|\cdot\|_w$ and $\|\cdot\|_\infty$.

LEMMA 3. *Let f_n be any polynomial of degree not exceeding n. Then*:

$$1)\ \|f_n\|_w \leqq 2\|f_n\|_\infty, \tag{48}$$

$$2)\ \|f_n\|_\infty \leqq (n+1)^{1/2}\|f_n\|_w. \tag{49}$$

Proof. 1) We have

$$\|f_n\|_w^2 = \langle f_n, f_n \rangle = \langle f_n, f_n \rangle_1 + \langle f_n, f_n \rangle_2 = \int_a^b f_n^2(\lambda) w_1(\lambda)\, d\lambda + \int_c^d f_n^2(\lambda) w_2(\lambda)\, d\lambda$$

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

and hence

$$\|f_n\|_w^2 \leqq 4\|f_n\|_\infty^2,$$

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

2) We have

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

$$\|f_n\|_\infty = \max\left\{ \max_{\lambda\in[a,b]} \left| \sum_{i=0}^{n} \gamma_i T_i[(\lambda - c_1)/d_1] \right|, \max_{\lambda\in[c,d]} \left| \sum_{i=0}^{n} \delta_i T_i[(\lambda - c_2)/d_2] \right| \right\} \tag{50}$$

where the γ's and the δ's are the expansion coefficients of f_n in $[a, b]$ and $[c, d]$ with respect to the polynomials (16) and (17) respectively. Using the Schwarz inequality we get

$$\left| \sum_{i=0}^{n} \gamma_i T_i[(\lambda - c_1)/d_1] \right| \leqq \left[\sum_{i=0}^{n} \gamma_i^2 \right]^{1/2} \left[\sum_{i=0}^{n} T_i^2[(\lambda - c_1)/d_1] \right]^{1/2}, \qquad \lambda \in [a, b].$$

But since $|(\lambda - c_1)/d_1| \leqq 1$ for $\lambda \in [a, b]$, we have that $|T_i[(\lambda - c_1)/d_1]| \leqq 1$. Hence:

$$\left| \sum_{i=0}^{n} \gamma_i T_i[(\lambda - c_1)/d_1] \right| \leqq (n+1)^{1/2} \left[\sum_{i=0}^{n} \gamma_i^2 \right]^{1/2}, \qquad \lambda \in [a, b]. \tag{51}$$

Likewise, we can show that

$$\left| \sum_{i=0}^{n} \delta_i T_i[(\lambda - c_2)/d_2] \right| \leqq (n+1)^{1/2} \left[\sum_{i=0}^{n} \delta_i^2 \right]^{1/2}, \qquad \lambda \in [c, d]. \tag{52}$$

Replacing (51) and (52) in (50) we get

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

Here we have used the result of Proposition 1 and the notation (31). □

As an immediate consequence of (48) and (49) we can easily bound each of the norms $\|\cdot\|_w$ and $\|\cdot\|_\infty$ in terms of the other from the left and the right as shown below:

$$(n+1)^{-1/2} \|f_n\|_\infty \leqq \|f_n\|_w \leqq 2\|f_n\|_\infty, \tag{53}$$

$$\tfrac{1}{2}\|f_n\|_w \leqq \|f_n\|_\infty \leqq (n+1)^{1/2} \|f_n\|_w. \tag{54}$$

We now state the main result of this subsection, which compares the polynomials p_n^* and p_n^∞.

THEOREM 4. *Let p_n^∞ and p_n^* be the minimax polynomial and the least squares polynomial defined by* (46) *and* (47). *Then the following inequalities hold*:

$$\|p_n^*\|_w \leqq \|p_n^\infty\|_w \leqq 2(n+1)^{1/2} \|p_n^*\|_w, \tag{55}$$

$$\|p_n^\infty\|_\infty \leqq \|p_n^*\|_\infty \leqq 2(n+1)^{1/2} \|p_n^\infty\|_\infty. \tag{56}$$

Proof. We will only prove (55) as the proof for (56) is identical. The first part of the double inequality is an obvious consequence of equation (46). The second part is a simple consequence of Lemma 3:

$$\|p_n^\infty\|_w \leqq (n+1)^{1/2} \|p_n^\infty\|_\infty \leqq (n+1)^{1/2} \|p_n^*\|_\infty \leqq 2(n+1)^{1/2} \|p_n^*\|_w. \qquad \square$$

The meaning of the theorem is that the two polynomials p_n^* and p_n^∞ are comparably small in $[a, b] \cup [c, d]$. This implies that, for the numerical methods for solving linear systems using polynomial iteration, we can replace the minimax polynomial by the least squares polynomial, which is 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 a priori reason to prefer one particular norm over the other. If the norm $\| \ \|_w$ is chosen, then the polynomial p_n^* is smaller than the polynomial p_n^∞ 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 $2(n+1)^{1/2}$, by the above theorem.

5. Application of the least squares polynomials to the solution of indefinite linear systems.

5.1. The generalized Chebyshev iteration. Let us return to the system (1) of § 2. We seek an approximate solution to (1) of the form

$$x_n = x_0 + s(A) r_0 \tag{57}$$

where $s(\lambda)$ is a polynomial of degree $n-1$ and r_0 the initial residual. The residual vector r_n of x_n is given by

$$r_n = p(A) r_0 \tag{58}$$

where $p(\lambda) = 1 - \lambda s(\lambda)$ is the residual polynomial. The developments of the previous sections lead us to choose for s the polynomial s_{n-1}^* which minimizes $\|1 - \lambda s(\lambda)\|_w$ over polynomials s of degree $\leqq n-1$. Let us assume that the coefficients α_n, β_n and the γ's and δ's of the orthonormal sequence $\{\lambda q_n\}$ have been determined. Letting $u_j = q_j(A) r_0$, it is immediately seen from equations (42)–(45) of the previous section that the approximate solution x_{n+1} can be computed from the recurrence:

$$\eta_n = 2[\gamma_0^{(n)} + \delta_0^{(n)}], \tag{59}$$

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

$$u_{n+1} := \frac{1}{\beta_{n+1}} [(A - \alpha_n I) u_n - \beta_n u_{n-1}]. \tag{60}$$

Clearly, the coefficients β_n, α_n and η_n can be determined when (60) and (59) are performed. Therefore, we have the following algorithm:

ALGORITHM 4. *Generalized Chebyshev iteration*

1. Initialize. Choose initial vector x_0 and compute $r_0 = f - A x_0$. Compute $u_0 := r_0/t$, where t is defined in initialization of Algorithm 3.

2. Iterate. For $n = 0, 1, \cdots$, until convergence do:

- Compute α_n, β_{n+1} and the coefficients $\gamma_i^{(n+1)}$, $\delta_i^{(n+1)}$ (see Algorithm 3)
- Compute:

$$\eta_n := 2[\gamma_0^{(n)} + \delta_0^{(n)}] \tag{61}$$

$$x_{n+1} := x_n + \eta_n u_n \tag{62}$$

$$u_{n+1} := \frac{1}{\beta_{n+1}} [(A - \alpha_n I) u_n - \beta_n u_{n-1}] \tag{63}$$

Note that the multiplication by a scalar in (62) can be avoided by using instead of the vectors u_n the sequence of vectors d_n defined by:

$$d_n = \eta_n u_n$$

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

With this simplification each step of the above algorithm requires one matrix by vector multiplication, $3N$ additions, $3N$ multiplications and $O(n)$ operations, at the nth step. Four vectors of length N need to be stored in memory plus an extra $4(n+1)$ locations for the coefficients γ and δ. However, N fewer storage locations are required in the case where the matrix by vector multiplication can be performed componentwise, i.e. if the ith component of Ax can be delivered inexpensively for all i.

5.2. A conjugate 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 derive a version of the method which resembles the conjugate residual method [23], [2]. We wish to show that the answer is positive.

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

$$x_{n+1} = x_n + a_n w_n, \tag{64}$$

$$r_{n+1} = r_n - a_n A w_n, \tag{65}$$

$$w_{n+1} = r_{n+1} + b_n w_n, \tag{66}$$

where the scalars a_n, b_n are computed in such a way that at each step the residual r_{n+1} is precisely the residual obtained from the generalized Chebyshev iteration described earlier.

There are two main reasons why such a new form of Algorithm 3 might be sought. The first reason is that the above version is slightly more economical than Algorithm 3; specifically (64) to (66) can be performed with N fewer multiplications provided we rearrange slightly the computation.[1] The second reason is that in the above form, the residual vector is available at each step of the iteration.

Some theoretical background is necessary for the understanding of the remainder of this subsection. The approximation x_n provided by a polynomial iteration as described in § 2 is of the form $x_n = x_0 + y$, where y is some vector belonging to the space K_n spanned by $r_0, Ar_0, \cdots, A^{n-1}r_0$. In other words the approximate solution belongs to the affine space $x_0 + K_n$. The residual vector r_n of such an approximation is of the form $p_n(A)r_0$, where p_n is a polynomial of degree $\leqq n$. Therefore, the residual r_n belongs to the Krylov space K_{n+1}. An important observation is that there is a one-to-one mapping between the subspace K_{n+1} and the space $\mathbf{P}_n$ of all polynomials of degree not exceeding n. In effect, for any element u of K_{n+1} of the form $u = \mu_0 r_0 + \mu_1 A r_0 + \cdots + \mu_n A^n r_0$, we can associate the polynomial $p_u(\lambda) = \mu_0 + \mu_1 \lambda + \cdots + \mu_n \lambda^n$. We will now assume that the degree of the minimal polynomial for r_0 is not less than $n+1$. Under this assumption, it is clear that the transformation mapping u into p_u is an isomorphism between K_{n+1} and $\mathbf{P}_n$. This allows us to introduce a new inner product on K_{n+1}.

DEFINITION 5. We will call the generalized Chebyshev inner product on K_{n+1}, and will denote by $(\cdot, \cdot)_w$, the bilinear form defined by

$$(u, v)_w = \langle p_u, p_v \rangle \tag{67}$$

[1] More precisely, we need to use the vectors $y_n = a_n w_n$ in place of w_n to save N multiplications, just as suggested in Algorithm 4 for the vectors u_n.

where $\langle \cdot, \cdot \rangle$ is the inner product on $\mathbf{P}_n$ defined by (8), (14) and (15). That this constitutes an inner product is an immediate consequence of the isomorphism introduced above. There is no ambiguity in denoting again by $\|\cdot\|_w$ the norm on K_{n+1} induced by this inner product. We then have the following immediate result.

PROPOSITION 6. *At each step of the generalized Chebyshev iteration, the approximate solution x_n minimizes the generalized Chebyshev norms $\|f - Ax\|_w$ of the residual vectors $f - Ax$ over all vectors x of the affine space $x_0 + K_n$.*
In other words we have:

$$\|f - Ax_n\|_w \leqq \|f - Ax\|_w \quad \forall x \in x_0 + K_n.$$

The proof of the proposition is straightforward.

The conjugate residual method is known to minimize the *Euclidean norm* of the residual vector on the affine space $x_0 + K_n$ [2, 23]. If we replace in that algorithm the Euclidean inner product by our new inner product, we will clearly realize the same goal, i.e. we will minimize the generalized Chebyshev inner product norm on $x_0 + K_n$. Thus in order to derive the desired algorithm it suffices to rewrite the conjugate residual algorithm with the new inner product as described below.

ALGORITHM 5

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

2. Iterate

$$a_n := \langle \lambda p_{r_n}, p_{r_n} \rangle / \langle \lambda p_{w_n}, \lambda p_{w_n} \rangle \tag{68}$$

$$x_{n+1} := x_n + a_n w_n \tag{69}$$

$$r_{n+1} := r_n - a_n A w_n \tag{70}$$

$$b_n := \langle \lambda p_{r_{n+1}}, p_{r_{n+1}} \rangle / \langle \lambda p_{r_n}, p_{r_n} \rangle \tag{71}$$

$$w_{n+1} := r_{n+1} + b_n w_n \tag{72}$$

$$p_{w_{n+1}} := p_{r_{n+1}} + b_n p_{w_n} \tag{73}$$

The computation of the inner products involved in (68) and (71) can be carried out as before by use of Proposition 1 together with expansions similar to (20), (21) for each of the polynomials p_{r_n} and p_{w_n} but we omit the details.

As with the conjugate residual method in the indefinite case, the above Algorithm may breakdown in equation (71) because $\langle \lambda p_{r_n}, p_{r_n} \rangle$ can be equal to zero. However, this difficulty can easily be overcome by an appropriate strategy which switches to Algorithm 4 when a breakdown occurs, and switches back to the present version immediately after. This procedure was presented by Chandra [2] for the Euclidean inner product and was named the modified conjugate residual (MCR) method. Its adaptation to the Chebyshev inner product defined above is straightforward.

Finally, note that it is also possible to write a conjugate gradient type method which would correspond to the approach outlined in § 3.3.

5.3. 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 Stiefel iteration proposed in [21].

As shown in Fig. 5.1, if an eigenvalue μ_i lies inside the interval $[b, c]$ then $p^*(\mu_i)$ is large in comparison with the other $p^*(\mu_j)$, with $j \neq i$.

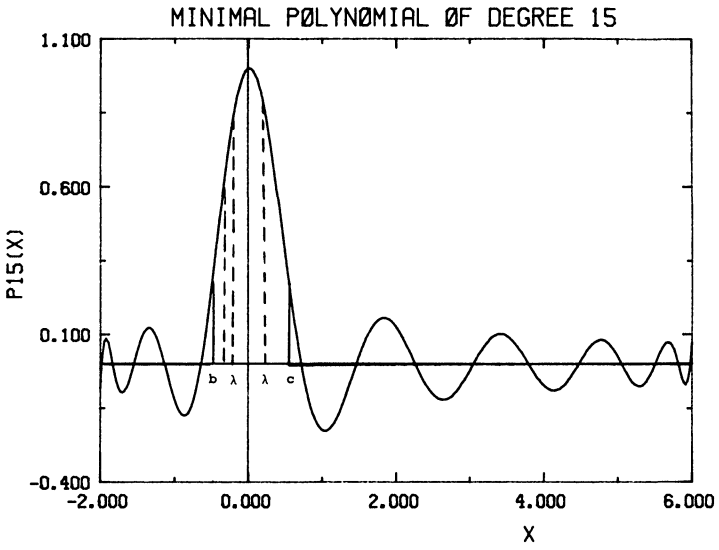

FIG. 5.1. *Least squares polynomial with some eigenvalues in* $[b, c]$.

As a consequence the residual $r_n = p_n^*(A)r_0$ will be mainly in the direction of the eigenvector z_i associated with μ_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 μ_i, we had m eigenvalues $\mu_i, \mu_{i+1}, \cdots, \mu_{i+m-1}$ inside $[b, c]$, and that we are performing m different generalized Chebyshev iterations, using m different starting vectors $x_{0,1}, x_{0,2}, \cdots, x_{0,m}$. We can again make the same argument: the residuals $r_{n,k}$ at the nth step will have their dominant components in the eigenvectors $z_i, z_{i+1}, \cdots, z_{i+m-1}$ and we can eliminate these components by projecting onto the subspace spanned by $\{r_{n,j}\}_{j=1,2,\cdots,m}$. It was shown in [21] 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 can be performed simultaneously. Ignoring the matrix by vector multiplication, intercommunication between the processors is needed only when the projection process takes place, which is infrequent.

5.4. A priori error bounds. We now establish an a priori error bound which shows that the generalized Chebyshev iteration is a converging process. We should emphasize, however, that the bound given below is proposed for the sole purpose of demonstrating the global convergence of the process and may not constitute a sharp error bound. The result is based on the following estimate of the residual norm which is an immediate consequence of Lemma 3.

LEMMA 7. *At the nth step of the generalized Chebyshev iteration the residual norm* r_n *of the approximate solution* x_n *satisfies*

$$\|r_n\| \leqq (n+1)^{1/2} \|p_n^*\|_w \|r_0\|. \tag{74}$$

Proof. From (6) we have

$$\|r_n\| \leqq \|r_0\| \max_{i=1,\cdots,N} |p_n^*(\mu_i)|.$$

Therefore

$$\|r_n\| \leqq \|r_0\| \|p_n^*\|_\infty.$$

The result follows from inequality (49) of Lemma 3. □

THEOREM 8. *Suppose that the spectrum of A is contained in $[a, b] \cup [c, d]$ and let*

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

Then at the nth step of the generalized Chebyshev iteration the residual vector r_n satisfies the inequality:

$$\|r_n\| \leqq 2(n+1)^{1/2} \frac{\|r_0\|}{T_{n'}[(M^2+m^2)/(M^2-m^2)]}, \tag{75}$$

where $n' = [n/2]$, 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 (74) we have

$$\|r_n\| \leqq (n+1)^{1/2} \|p_n^*\|_w \|r_0\| \leqq (n+1)^{1/2} \|p\|_w \|r_0\| \tag{76}$$

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

$$t_n(\lambda) = T_{n'}[h(\lambda)]/T_{n'}[h(0)]$$

with

$$h(\lambda) = [M^2 + m^2 - \lambda^2]/[M^2 - m^2].$$

The degree of t_n is $\leqq n$ and we have $t_n(0) = 1$. Therefore from (76) and Lemma 3 we get

$$\|r_n\| \leqq (n+1)^{1/2} \|t_n\|_w \|r_0\| \leqq 2(n+1)^{1/2} \|t_n\|_\infty \|r_0\|.$$

To complete the proof, we observe that t_n has been chosen in such a way that $\|t_n\|_\infty = [T_m(h(0))]^{-1}$ because for $\lambda \in [a, b] \cup [c, d]$, $|t_n(\lambda)| \leqq 1$, the value 1 being reached for some λ. □

Note that ideally M^2 and m^2 are the largest and the smallest eigenvalues of A^2. Hence an interpretation of (75) 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 $A^2 x = Af$. This result is, however, a pessimistic estimate as the numerical experiments indicate.

6. Application to the computation of eigenvectors associated with interior eigenvalues. Consider the eigenvalue problem

$$Az = \mu z \tag{77}$$

where A is of order N and symmetric. When the desired eigenvalue μ is one of the few extreme eigenvalues of A, there are a host of methods for solving (77), among which the Lanczos algorithm appears to be one of the most successful. When μ is in the interior of the spectrum then problem (77) becomes more difficult to solve numerically. We would like to mention, however, that it is not often the case that one requires eigenvalues that are deeply inside the spectrum. Most commonly encountered eigenvalue problems are exterior ones, even though they may deal with a number of eigenvalues as high as 30 or 50, out of 1000 or 2000 (say).

A good approach is to factor the matrix $A - \tilde{\mu} I$ for some approximation $\tilde{\mu}$ of μ and use the Lanczos algorithm on $(A - \tilde{\mu} I)^{-1}$ [14], [15], [18]. This transforms the

interior eigenvalues into exterior ones thus allowing an efficient use of the Lanczos algorithm. Sparse matrix codes for performing efficient factorizations of the matrix A are now widely available. Although this solves most problems of this nature, we would like to outline an alternative based on the least squares polynomials of § 4, which may be useful in some particular situations such as the following:

- Only one eigenvalue and the corresponding eigenvector are needed.
- A low accuracy is required.
- The factorization of $A-\mu I$ is too expensive, or impossible with the computational environment available.

Consider an approximation to the eigenvector z of (77) of the form

$$v_n = p_n(A)v_0 \tag{78}$$

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

$$v_n = \sum_{i=1}^{N} \theta_i p_n(\mu_i) z_i \tag{79}$$

which shows that if v_n is to be a good approximation of the eigenvector z_i then every $p_n(\mu_j)$, with $j \neq i$, must be small in comparison with $p_n(\mu_i)$. This leads us to seek for a polynomial which is small in $[\mu_1, \mu_{i-1}] \cup [\mu_{i+1}, \mu_N]$ and which satisfies $p_n(\mu_i) = 1$. In other words we need to find a polynomial of the form $p(\lambda) = 1 - (\lambda - \mu_i)t_{n-1}(\lambda - \mu_i)$, with $t_{n-1} \in \mathbf{P}_{n-1}$, which is small in the least squares sense on those intervals. The simple change of variables $\xi = \lambda - \mu_i$ transforms this problem into the one of § 4, thus providing an algorithm for computing an approximation to the desired eigenvector associated with μ_i. Note that, in general, μ_i is not available and is replaced by some approximation $\tilde{\mu}$ which is improved during the process. We will thus find a polynomial in the variable ξ which corresponds to the operator $A - \tilde{\mu}I$, i.e. the approximate eigenvector v_n will have the form

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

where p_n^* is the least squares polynomial obtained from Algorithm 3 with

$$a = \mu_1 - \tilde{\mu}, \quad b = \mu^- - \tilde{\mu}, \quad c = \mu^+ - \tilde{\mu}, \quad d = \mu_N - \tilde{\mu},$$

where μ_{i-1} and μ_{i+1} are replaced by μ^- and μ^+ respectively.

A problem not yet discussed is how to estimate the interior parameters b and c, or more precisely to refine dynamically a given pair of starting parameters b, c. This is considered in the next section.

An obvious extension of this algorithm is the subspace iteration otherwise known as simultaneous iteration method (see e.g. [15]) in which one iterates with a block of vectors simultaneously and uses the Rayleigh–Ritz projection approximations from the subspace spanned by the blocks. Also, an interesting alternative to the polynomial method proposed above is a hybrid algorithm consisting in using the Lanczos algorithm to solve the auxiliary eigenvalue problem:

$$Bz = \nu z \quad \text{with } B = p_n(A), \quad \nu = p_n(\mu),$$

where p_n is a polynomial of moderate degree defined as above. An interior eigenvalue μ of A is thus transformed into an exterior one for B, without having to factor A.

7. Some practical considerations.

7.1. 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 figures significantly in its efficiency. Let us recall that a, d are ideally the smallest and largest eigenvalues μ_1 and μ_N of A, while b and c should be equal to the eigenvalues μ^- and μ^+ closest to the origin with $\mu^- < 0$ and $\mu^+ > 0$. The smallest and largest eigenvalues μ_1 and μ_N are easier to estimate and we have several ways of doing so, the simplest being perhaps the use of Gershgorin's theorem. The Lanczos algorithm can also be particularly useful since it provides at the same time a good estimate of the parameters a, d and a good starting vector for the iteration (see e.g. [12]). 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 μ_1 or d is less than μ_N then there is a high risk that the process will diverge. This is because outside the interval $[a, d]$ the residual polynomial can take huge values as is seen in Fig. 7.1. When this happens, however, nothing is lost because we can stop and use the (huge) residuals as approximate eigenvectors associated with μ_1 and μ_N thus providing more accurate estimates of both a and d, with the help of the Rayleigh quotient.

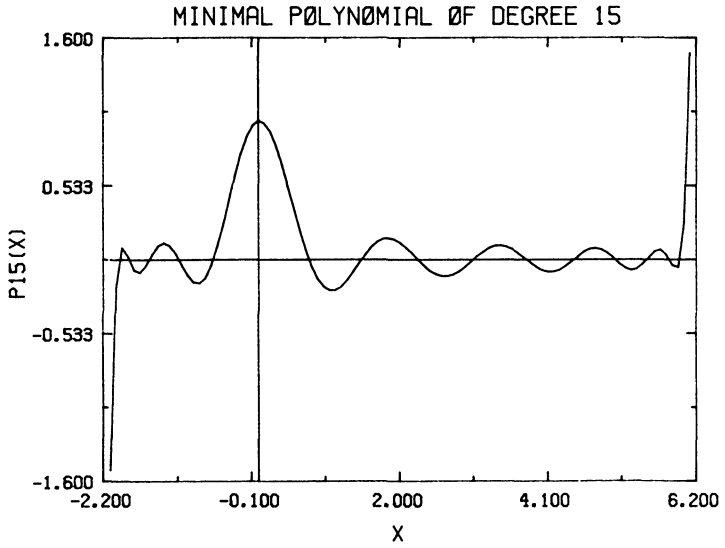

FIG. 7.1. *Least squares polynomial with some eigenvalues outside* $[a, d]$.

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 $b < \mu^-$ and $c > \mu^+$, then as is seen from Fig. 5.1 the residual vectors will have large components in the eigenvectors associated with the eigenvalues μ^- and μ^+ closest to the origin. A Rayleigh–Ritz approximation can therefore be applied to extract more accurate estimates of μ^- and μ^+ from two or more residual vectors.

A more complete procedure can be described as follows. Suppose that we start with an interval $[b, c]$ sufficiently wide to contain the eigenvalues μ^+, and μ^-. 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 $[b, c]$. Therefore, a projection process onto the subspace spanned by a small number of successive residual vectors will provide a good approximation of the eigenvalues μ^- and μ^+, and the Rayleigh–Ritz approximate solution will 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^TAQ and its eigenvalues (e.g. by 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 2 or 3 digits of accuracy). As soon as an eigenvalue which corresponds to either μ^-, or μ^+ converges, we replace b or c by that eigenvalue. When both of the eigenvalues have converged, we replace the current approximate solution x_n by the Rayleigh–Ritz approximation $\tilde{x} = x_n + \sum [\mu_j]^{-1} z_j^T r_n z_j$, where the sum is over the converged Ritz eigenvalues μ_j and the z_j are the associated approximate Ritz eigenvectors. Recall that z_j is defined by $z_j = Qs_j$ where s_j is an eigenvector of Q^TAQ associated with the Ritz value μ_j. The effect of this important projection step is to purify the residual vector from its large components in the direction of the converged eigenvectors. This is quite effective, as will be seen in some experiments described in § 8. Note that a similar technique was suggested by Rutishauser in conjunction with the conjugate gradient method in [19].

A useful variation here is to use the latest directions of search u_i instead of the latest residuals, thus avoiding to compute residual vectors. The reason this is effective is that, like r_n, $q_n(A)r_0$ 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 good approximations 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 μ_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 μ, μ^+, and μ^-. Then after a number of steps of the iteration described in § 6 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 μ, μ^- and μ^+. Also, the Ritz eigenvector will be a much better approximation than the current approximation. In practice we will orthonormalize the m latest approximations into the $N \times m$ orthonormal system Q. We then compute the Ritz matrix Q^TAQ and its eigenvalues and compare the Ritz eigenvalues obtained with those of the previous projection. As soon as an eigenvalue which corresponds to either μ, μ^-, or μ^+ starts converging, we replace μ, b or c by that eigenvalue. When the three of the eigenvalues have converged, we replace the approximate eigenvector by the Ritz vector $z = Qy$, where y is the eigenvector of Q^TAQ associated with μ. This process is tested in an experiment in § 8.

7.2. Using high degree polynomials. It is our experience that using high degree polynomials is usually more effective than compounding several low degree polynomials. 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. Although we often encounter underflow situations with high degree polynomials, this is not too serious provided we simply replace the underflowing numbers by zeros. We open a parenthesis here to point out the following interesting observation. It has been observed during the numerical experiments that the *last* coefficients $\delta_i^{(n)}$, $\gamma_i^{(n)}$ become tiny as *i and n increase*. This is the cause of the underflowing conditions mentioned above. The practical consequence of this behavior 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 computational work and memory needs. In fact, a more important observation is that the coefficients α_n and β_n are cyclically converging. More precisely, it seems that α_{3k+j} is a converging sequence for fixed j, $j = 0, 1, 2$. The same phenomenon appears to be true for the β's. As a consequence, we believe that after a certain step it is possible to use the previous α_n's and β_n's, thus avoiding further calculations of these coefficients.

8. Numerical experiments. In this section we will describe a few numerical experiments and give a few additional details on the practical 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 Algorithm 4 rather than Algorithm 5.

8.1. Comparison with other methods. The SYMMLQ Algorithm described in [13] and the various versions of it such as MINRES [13], SYMMBK [2], MCR [2], all based on the Lanczos algorithm, are among the best known iterative methods for solving large indefinite linear systems. It is therefore important to compare the performances of our algorithm with this class of methods. We will mainly consider SYMMLQ although, as will be seen shortly, this is by no means the most effective method. It is difficult to make any definitive conclusion but we will see that there are many 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). Note that Algorithm 5 realizes GCI in $2N+NZ$ multiplications per step instead of $3N+NZ$.

TABLE 1

	Mult.'s	Storage
SYMMLQ	$9N+NZ$	$5N$
GCI	$3N+NZ$	$4N$

Let us discuss the above table under the assumption that $NZ = 5N$, which would correspond for example to using the inverse iteration method to compute an eigenvector of the Laplace operator. Then it is easily seen that our algorithm becomes competitive if the number of steps for GCI does not exceed 1.75 times that of 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 (μ_i), where the eigenvalues μ_i are distributed uniformly 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 μ_i are defined by:

$$\text{define } i_1 = [N(b-a)/(b-a+d-c)],$$

$$\text{for } i = 1, 2, \cdots, i_1: \qquad \mu_i = a + (i-1) \cdot h_i, \text{ with } h_1 = (b-a)/(i_1 - 1), \tag{80}$$

$$\text{for } i = i_1 + 1, \cdots, N: \qquad \mu_i = c + (i-1) \cdot h_2, \text{ with } h_2 = (d-c)/(N - i_1). \tag{81}$$

The right-hand side f is taken equal to $f = Ae$, where $e^T = (1, 1, 1, \cdots, 1)$. The initial vector is 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 polynomial of degree 25. The residual norms are plotted in Fig. 8.1 in logarithmic scale. This is done every 5 steps for a total of 75 steps. Observe that the performances of the two algorithms are almost identical.

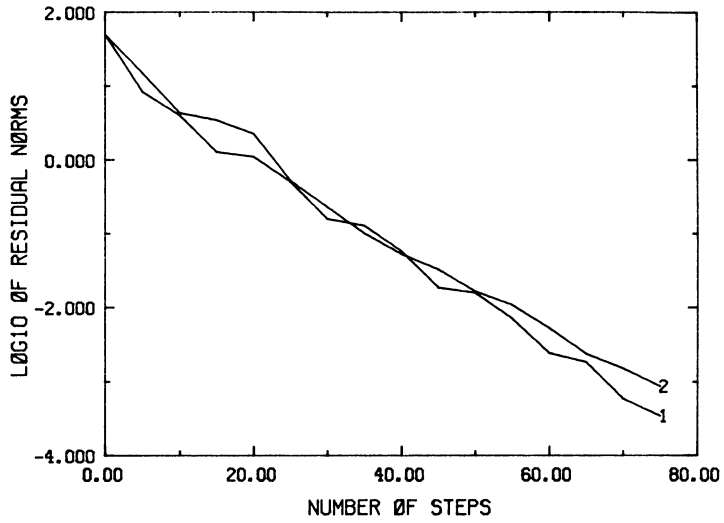

FIG. 8.1. *Comparison of* SYMMLQ *and* GCI. $N = 200$, $a = -2.0$, $b = -0.5$, $c = 0.5$, $d = 6.0$. 1: SYMMLQ; 2: GCI.

This example shows that when the origin is well separated from the spectrum, SYMMLQ and GCI will converge with nearly the same speed, but each step of SYMMLQ is more expensive as shown in Table 1.

The next test compares the two algorithms in presence of a 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 to 500. The eigenvalues μ_i are defined again by (80) and (81). We have plotted in Fig. 8.2 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 are taken). The initial vector is chosen randomly and has an initial residual norm of 0.719×10^2. Although SYMMLQ requires fewer steps here to reduce the residual norm by a factor of $eps = 10^{-6}$, observe that in the beginning GCI has a better performance than SYMMLQ. But as the number of steps approaches the dimension N of A, SYMMLQ becomes faster. Note that in exact arithmetic SYMMLQ would have converged exactly

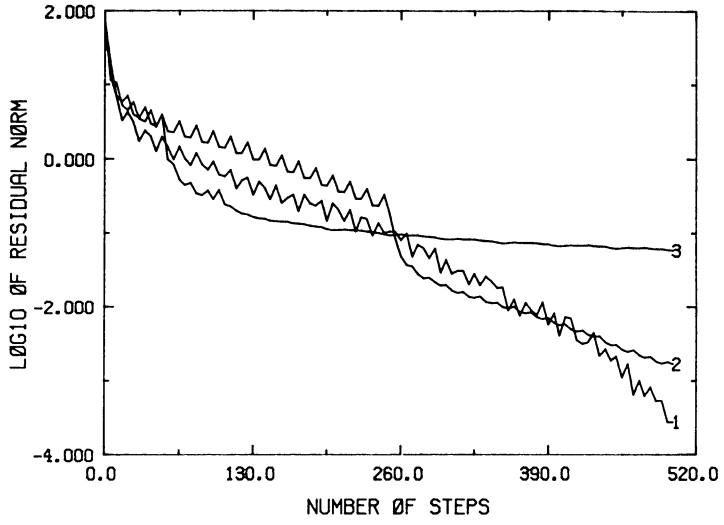

FIG. 8.2. *Comparison of* SYMMLQ *and* GCI. $N = 500$, $a = -2.0$, $b = -0.05$, $c = 0.05$, $d = 6.0$. 1: SYMMLQ; 2: GCI *with degree* 50; 3: GCI *with degree* 250.

at the 500th step. 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 advantage of using a higher degree polynomial when possible.

In the above experiments it was 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 § 7. In the same example of order 500 as above we have taken as exterior 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 § 7 was taken. The convergence of the Ritz eigenvalues is slow in this example because the relative gap between the eigenvalues of A is quite small. At the 9th iteration no Ritz value has converged to the demanded accuracy 10^{-3} but a projection step was taken anyway, with those eigenelements which have converged to a relative accuracy of 0.2. Then the algorithm gave as eigenvalues $\mu^- = -0.0504 \cdots$ and $\mu^+ = 0.0507 \cdots$ and the final (10th) iteration was pursued with these values for b and c. Fig. 8.3 is a plot of the residual norms obtained with this technique together with those previously obtained 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.

These comparisons cannot be complete without an assessment of some of the other competitive methods. Several algorithms equivalent to SYMMLQ and based on the Lanczos algorithm, have been derived in order to cut down cost [2], [20], [24]. Like SYMMLQ, most of the methods are based on some stable factorization of the tridiagonal matrix which represents the section of A in the Krylov subspace span $\{r_0, Ar_1, \cdots, A^{m-1}r_0\}$, with respect to the basis of Lanczos vectors (see e.g. [20]). Thus, the DIOM (2) algorithm [20] is based on the LU factorization with partial pivoting, known to be very stable for tridiagonal systems, while the SYMMBK

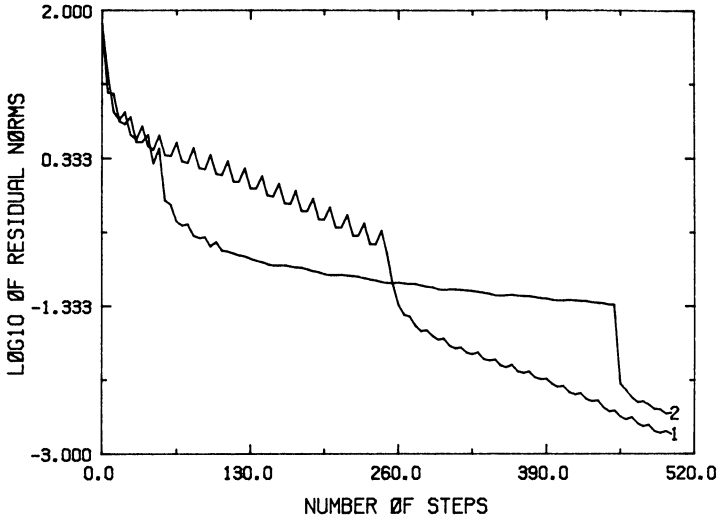

FIG. 8.3. GCI *with dynamical estimation of parameters and projection.* $N = 500$. 1: GCI, *degree* 250, *optimal parameters, no projection*; 2: GCI, *degree* 50, *estimated parameters and projection.*

algorithm [2] uses the Bunch and Kaufman factorization. Note that all these versions use the same basis of the Krylov subspace K_m, provided by the Lanczos algorithm with *no reorthogonalization.* Hence it is unlikely that the number of steps will differ significantly from one version to another and the savings that can be achieved are limited (for example: $7N$ multiplications $+NZ$ per step for DIOM (2) or SYMMBK versus $9N+NZ$ for SYMMLQ). However, the situation may change completely if one uses some sort of reorthogonalization in the Lanczos process. The Lanczos method has been reintroduced as a means for solving linear systems by Parlett [16]. The Lanczos vectors are generated and saved in secondary storage until convergence is known to be reached, and the solution is then computed from a combination of those vectors. The algorithm realizes the Lanczos method in $6N+NZ$ multiplications per step, apparently a little gain from the $7N+NZ$ of SYMMBK or DIOM (2), at the expense of using secondary storage. But this is only part of the truth. The Lanczos vectors, which are theoretically orthogonal, soon lose their linear dependence thus causing the SYMMLQ-type methods to slow down drastically if no extra measure is taken. A simple way of preventing loss of orthogonality, considered by Lanczos himself, was to reorthogonalize the vectors at every step. This is prohibitively expensive and several alternatives have been proposed by Parlett [16], and more recently by Simon [22] who shows conclusively that the resulting algorithms can be highly competitive, especially in some computational environments, where the I/O can be masked by computation. Partial reorthogonalization is also very beneficial when solving several linear systems with the same matrix.

Concerning the orthogonal polynomials of GCI, this raises the same question about the necessity of reorthogonalizing the sequence of polynomials, as it seems likely that the loss of orthogonality will also be severe if the degree is high. Note, however that the additional computational work incurred from the reorthogonalization will be a function of the degree of the polynomial rather than of the dimension N.

In conclusion, it is likely that the GCI method will be competitive in some particular situations, for example when the parameters are known, but the Lanczos-type methods will be superior in many other instances, such as when there are only a few negative eigenvalues or when the cost of the matrix by vector product is expensive.

8.2. Computing interior eigenelements. Consider the following $N \times N$ five-point discretization matrix:

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

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 μ 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 § 7 with $\tilde{\mu}$ equal to 2.5. Also, as the first estimates of the eigenvalues μ^+ and μ^-, the smallest eigenvalue at the right of μ and the largest eigenvalue at the left of μ respectively, we take 3.0 and 2.0. The initial vector v_0 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 § 7. When either of the eigenvalues corresponding to μ^+, μ, or μ^- has converged to a relative accuracy of 10^{-3} then b, $\tilde{\mu}$, or c is replaced accordingly. When the three eigenvalues have converged the approximate eigenvector is replaced by the Ritz vector. Fig. 8.4 shows 480 steps of this process (curve 1). Notice the sudden drop of the residual norm after the purification process. Clearly, such an improvement is not only due to the fact that we now use better parameters b and c, but mainly to the fact that the current approximate eigenvector has been purified from its largest components in the directions of the undesired eigenvectors.

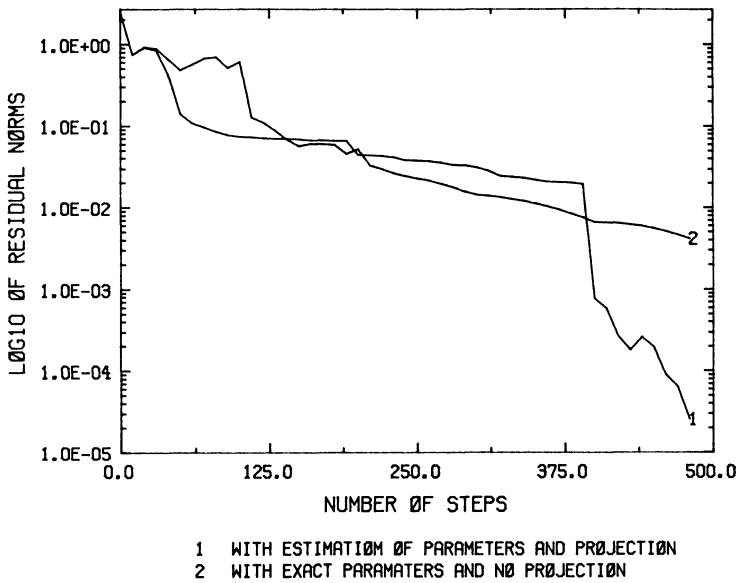

FIG. 8.4. *Computing an interior eigenpair with* GCI. $N = 150$, $\mu = 2.47119 \cdots$.

As a comparison the residual norms are plotted for the case where the exact parameters $\mu^- = 2.436872$, $\mu = 2.471196$, and $\mu^+ = 2.604246$ are used. In this case we iterated with a polynomial of degree 100. (Iteration with a polynomial of degree 40 gave much slower convergence.) Thus, the process suggested in § 7, using orthogonal projection, is more successful here than the algorithm using the optimal parameters and a reasonably high degree polynomial.

Acknowledgments. The ideas of this work have been developed during a visit at the University of California at Berkeley. I would like to express my gratitude to Prof. Beresford N. Parlett for his hospitality at Berkeley and for valuable comments about this paper. I also benefited from helpful discussions with Walter Gautschi and Stanley C. Eisenstat.

REFERENCES

[1] R. S. ANDERSON AND G. H. GOLUB, *Richardson's non stationary matrix iterative procedure*, Technical Report STAN-CS-72-304, Stanford University, Stanford, CA, 1972.

[2] R. CHANDRA, *Conjugate gradient methods for partial differential equations*, PhD thesis, Tech. Rep. 129, Yale University, New Haven, CT, 1981.

[3] C. C. CHENEY, *Introduction to Approximation Theory*, McGraw-Hill, New York, 1966.

[4] P. H. DAVIS, *Interpolation and Approximation*, Dover, New York, 1963.

[5] C. DE BOOR AND J. R. RICE, *Extremal polynomials with applications to Richardson iteration for indefinite systems*, Tech. Rep. 2107, Math. Res. Center, Univ. of Wisconsin, Madison, 1980.

[6] W. GAUTSCHI, *On generating orthogonal polynomials*, SIAM J. Sci. Statist. Comp., 3 (1982), pp. 289–317.

[7] ———, *Construction of the Gauss–Christoffel quadrature formula*, Math. Comp., 22 (1968), pp. 251–270.

[8] G. N. GOLUB AND R. S. VARGA, *Chebyshev semi iterative methods, successive overrelaxation iterative methods and second order Richardson iterative methods*, Numer. Math., 3 (1961), pp. 147–168.

[9] J. F. GRCAR, *Analysis of the Lanczos algorithm and of the approximation problem in Richardson's method*, PhD thesis, Univ. Illinois, Urbana-Champaign, 1981.

[10] O. G. JOHNSON, C. A. MICCHELLI AND G. PAUL, *Polynomial preconditionings for conjugate gradient calculations*, Technical Report, IBM Watson Res. Center, Yorktown Heights, NY, 1982.

[11] V. I. LEBEDEV, *Iterative methods for solving operator equations with a spectrum contained in several intervals*, USSR Comp. Math. Math. Phys., 9(6) (1969), pp. 17–24.

[12] D. A. O'LEARY, *Hybrid conjugate gradient algorithms*, PhD dissertation, Computer Science Dept., Stanford University, Stanford, CA, 1976.

[13] C. C. PAIGE AND M. A. SAUNDERS, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–624.

[14] B. N. PARLETT, *How to solve* $(K - \lambda M)z = 0$ *for large K and M*, In E. Asbii et al., ed., Proc. 2nd International Congress on Numerical Methods for Engineering (GAMNI 2), Dunod, Paris, 1980, pp. 97–106.

[15] ———, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

[16] ———, *A new look at the Lanczos algorithm for solving symmetric systems of linear equations*, Lin. Alg. and Appl., 29 (1980), pp. 323–346.

[17] R. R. ROLOFF, *Iterative solution of matrix equations for symmetric matrices possessing positive and negative eigenvalues*, Tech. Rep. UIUCDCS R-79-1018, Univ. Illinois, Urbana-Champaign, 1979.

[18] A. RUHE AND T. ERICSSON, *The spectral transformation Lanczos method in the numerical solution of large sparse generalized symmetric eigenvalue problems*, Math. Comp., 35 (1980), pp. 1251–1268.

[19] H. RUTISHAUSER, *Theory of gradient methods*, In Refined Iterative Methods for Computation of the Solution and the Eigenvalues of Self-Adjoint Boundary Value Problems, Birkhauser Verlag, Basel-Stuttgart, 1981, pp. 24–49.

[20] Y. SAAD, *Practical use of some Krylov subspace methods for solving indefinite and unsymmetric linear systems*, Technical Report 214, Yale University, New Haven, Connecticut, 1982, SIAM J. Sci. Statist. Comp., 5 (1984), to appear.

[21] Y. SAAD AND A. SAMEH, *A parallel Block Stiefel method for solving positive definite systems*, Proceedings of the Elliptic Problem Solver Conference, M.H. Schultz, ed., Los Alamos Scientific Lab., Academic Press, New York, 1980, pp. 405–412.

[22] H. D. SIMON, *The Lanczos algorithm for solving symmetric linear systems*, PhD thesis, Univ. California, Berkeley, 1982.

[23] E. L. STIEFEL, *Kernel polynomials in linear algebra and their applications*, U.S. NBS Applied Math. Series 49, 1958, pp. 1–24.

[24] J. STOER AND R. FREUND, *On the solution of large indefinite systems of linear equations by conjugate gradient algorithms*, Tech Rep., Inst. für Ang. Math. und Stat., Univ. Würzburg, W. Germany, 1981.