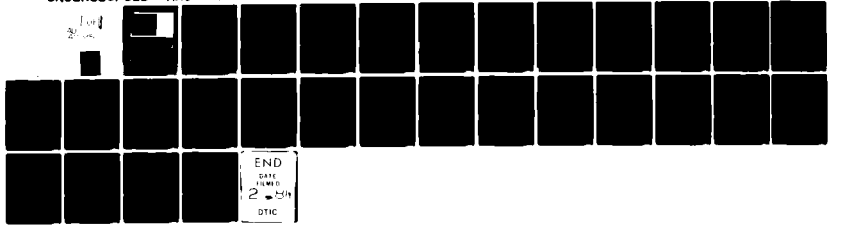
AD-A093 568 WISCONSIN UNIV-MADISON MATHEMATICS RESEARCH CENTER F/G 12/1
EXTREMAL POLYNOMIALS WITH APPLICATION TO RICHARDSON ITERATION F--ETC(U)
AUG 80 C DE BOOR, J R RICE DAAG29-80-C-0041
UNCLASSIFIED MRC-TSR-2107 NL



MRC Technical Summary Report # 2107

EXTREMAL POLYNOMIALS WITH APPLICATION TO RICHARDSON ITERATION FOR INDEFINITE LINEAR SYSTEMS

Carl de Boor and John R. Rice

LEVEL II

Mathematics Research Center
University of Wisconsin—Madison
610 Walnut Street
Madison, Wisconsin 53706

August 1980

Received June 6, 1980

Approved for public release
Distribution unlimited

Sponsored by

U. S. Army Research Office
P. O. Box 12211
Research Triangle Park
North Carolina, 27709

National Science Foundation
Washington, D. C. 20550

UNIVERSITY OF WISCONSIN - MADISON

MATHEMATICS RESEARCH CENTER

EXTREMAL POLYNOMIALS WITH APPLICATION TO RICHARDSON ITERATION FOR INDEFINITE LINEAR SYSTEMS

(10) Carl de Boor and John R. Rice

(9) Technical Summary Report #2107
August 1980

ABSTRACT

The application of Richardson iteration to a symmetric, but indefinite linear system requires certain parameters which can be determined from the zeros in the error of a certain best polynomial approximant on some set S known to contain the spectrum of the coefficient matrix. It is pointed out that this error can also be obtained as a multiple of the extremal polynomial for the linear functional $p \mapsto p(0)$, and this leads to an efficient Remes type algorithm for its determination. A program incorporating this algorithm for the case that S consists of two intervals bracketing zero is also given.

AMS(MOS) Subject Classification - 65F10, 65D15, 41A10

Key Words - Richardson iteration, symmetric indefinite, Remes, norm preserving extension, norm calculation for linear functional, Chebyshev polynomial

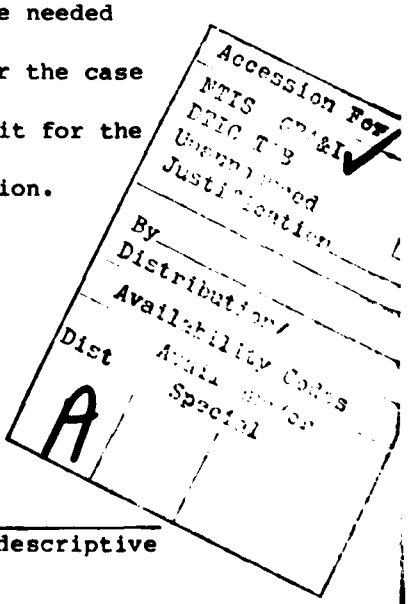
Work Unit Number 3 - Numerical Analysis and Computer Science

Sponsored by the United States Army under contract No. DAAG29-80-C-0041. The second author was also partially supported by National Science Foundation Grant MCS 77-01408.

SIGNFICANCE AND EXPLANATION

Sparse symmetric (positive) definite linear systems occur in various problems, chiefly in the construction of least-squares approximations and in the solution of elliptic partial differential equations. They are most advantageously solved by the conjugate gradient method. But this method may break down if the system fails to be definite, as it might be if constraints are added to the least-squares problem or standard conditions on the lower order terms in the elliptic PDE are relaxed. In such a case, Richardson iteration offers a possibly attractive alternative. This iteration method requires knowledge of some set S certain to contain the spectrum of the coefficient matrix of the linear system and this can often be supplied. In addition, it requires knowledge of the polynomial of smallest size on S which has a given degree and takes the value 1 at 0. If S lies to one side of 0, then this polynomial is just the Chebyshev polynomial for S (normalized to have value 1 at 0). But, in the indefinite case, the Chebyshev polynomial will not do.

The present report presents a novel characterization of the needed polynomial, an efficient algorithm for its construction and, for the case that S consists of two intervals, a Fortran program based on it for the determination of the iteration parameters for Richardson iteration.



EXTREMAL POLYNOMIALS WITH APPLICATION TO RICHARDSON ITERATION FOR INDEFINITE LINEAR SYSTEMS

Carl de Boor and John R. Rice

1. The iteration problem Consider the linear system of equations $Ax = b$ and the iteration

$$x^{n+1} = x^n - \alpha_n(Ax^n - b)$$

With $e^n := x^n - x$ the error in the n-th iterate, we have

$$e^n = (1-\alpha_{n-1}A)e^{n-1} = \ldots = \prod_{j=1}^{n}(1-\alpha_{j-1}A)\, e^0 = Q_n(A)e^0$$

where Q_n is the polynomial of degree n which vanishes at $1/\alpha_0, \ldots, 1/\alpha_{n-1}$ and is 1 at 0. This is Richardson's (first order) iteration, with iteration parameters α_j. If the spectrum of A is known to lie in some compact set S, then a standard analysis suggests that one should choose the parameters α_j so as to minimize

$$\|Q_n\|_S := \max_{s\in S}|Q_n(s)|$$

The resulting polynomial P_n is then the error in the best Chebyshev approximation on S to 1 from $\{\sum_{j=1}^{n}\beta_j t^j\}$. If S is an interval not containing the origin (hence A is known to be definite), then it is well known that a renormalization of P_n to make the coefficient of t^n equal to 1 gives T_n, the Chebyshev polynomial for the interval S. For this case, the three-term recurrence relation for the Chebyshev polynomials may be employed to build up x^n without the use of the zeros of P_n. This has the advantage that the iterates x^i so generated along the way are themselves using P_i. This method is known as the Chebyshev semi-iterative method. This variation requires some more memory (3 vectors rather than 2 are used) and more computation per step (since more vectors are combined per step). The conjugate gradient method is a further variation which, with some

Sponsored by the United States Army under Contract No. DAAG29-80-C-0041.
The second author was also partially supported by National Science Foundation Grant MCS 77-01408.

more work per iteration, removes the dependence on the interval S ; the mere knowledge that such an interval exists suffices to show that the error produced at the n-th step is of the form $P_n e^0$ with P_n the error in a best approximation to 1 on the spectrum of A itself.

The conjugate gradient method may run into difficulties when A , though symmetric and invertible, is not definite. See Paige and Saunders [1975] for a detailed discussion and some remedies. For this reason, Richardson iteration becomes an attractive alternative in this case. We now have the spectrum of A contained in two intervals, with the origin between them. Akhiezer [1928] determined the Chebyshev polynomials for two such intervals of equal length and Lebedev [1969] extended this technique to a set S consisting of an arbitrary number of intervals of equal length and applied his result to iteration. See Anderssen and Golub [1972] for a translation of Lebedev's paper and further discussions, particularly on the important subject of the order in which best to use the α_i's .

Specifically, let $S = [a,b] \cup [c,d]$. For certain values of a, b, c and d, Atlestam [1977] has obtained a representation of the Chebyshev polynomials for S , of the following form: Let

$$Q(t) := \cos\left(m(\pi+I(t))\right)$$

with

$$I(t) := \int_a^t (u-r)p(u)du , \quad r := \int_b^c up(u)du \,/\, \int_b^c p(u)du$$

and

$$p(u) := \left((u-a)(u-b)(u-c)(d-u)\right)^{-1/2}$$

If there are integers m and k so that $I(b) = k\pi/m$, then Q is a polynomial of degree m proportional to the Chebyshev polynomial for S . Atlestam further shows that, for any interval pair S, the Chebyshev polynomial is of this form but for a slightly different pair of intervals, and this difference goes to zero as the degree goes to infinity. Her arguments can be used to show that in the same way, for any interval pair S bracketing the origin, the best polynomial P_n is of the above form, but for a slightly different interval pair. These results can be used to obtain sharp asymptotic results on

the degree of convergence of the iteration method, but it is not clear how useful the representation is for obtaining the necessary iteration parameters.

In the present paper, we give what we feel is a more useful formulation of the mathematical problem underlying the determination of the parameters; well known results then establish existence and uniqueness of the solution of this problem and characterize it. In particular, we are led to a Remes type algorithm for the determination of P_n, whose zeros can then be determined efficiently by the Modified Regula Falsi. For the particular case that S is an interval pair, we present some numerical results to illustrate the nature of the parameters and the convergence rate of the corresponding iteration. In an appendix, we list a Fortran program (written by Frederick Sauer) which produces the iteration parameters when supplied with the two intervals and the desired polynomial degree.

2. The extremal polynomial The papers mentioned above all use Chebyshev polynomials in some essential way, so we first note that, in general, the required polynomial P_n is unrelated to the Chebyshev polynomial T_n for S. This is seen in the analysis of Atlestam [1977] or, more directly, from the fact shown below that P_n alternates one less time on S than does T_n.

To recall, the Chebyshev polynomial T_n for the compact set S is the polynomial of the form $t^n + \Sigma_{j=0}^{n-1} \beta_j t^j$ which is as small as possible on S. In other words, T_n is the error in the best approximation on S to t^n from $\{\Sigma_0^{n-1} \beta_j t^j\}$. By contrast, we are interested in the polynomial P_n which is the error in the best approximation on S to 1 from $\{\Sigma_1^n \beta_j t^j\}$.

We now reformulate this problem as follows. Let λ be the linear functional on π_n (:= the polynomials of degree n or less) whose value at p is $p(0)$. In symbols,

$$\lambda : \pi_n \longrightarrow \mathbf{R} : p \longmapsto p(0)$$

An **extremal for** λ is any polynomial of norm 1 at which λ takes on its norm, i.e., any $p \in \pi_n$ with $\|p\|_S = 1$ and $\lambda p = \|\lambda\|$. Here

$$\|p\| = \|p\|_S := \max_{s \in S} |p(s)|$$

and

$$\|\lambda\| := \max_{p\in\pi_n} \frac{\lambda p}{\|p\|_S} = 1/\min\{\|p\|_S : p\in\pi_n,\ p(0) = 1\}$$

This shows that the polynomial P_n which is of minimum norm on S and satisfies $P_n(0) = 1$ is a multiple of an extremal p^* for λ, i.e., $P_n = p^*/p^*(0)$.

The standard approach to the construction of extremals is via norm preserving extensions, i.e., via a socalled canonical representation for λ (see, e.g., Rivlin [1974; pp.82ff]). Such a **canonical representation for** λ consists of $n+1$ points $t_1 < t_2 < \ldots < t_{n+1}$ in S and corresponding coefficients $\alpha_1, \alpha_2, \ldots, \alpha_{n+1}$ so that

$$\lambda p = \sum_{j=1}^{n+1} \alpha_j\, p(t_j)\ ,\quad \text{all}\quad p\in\pi_n$$

and

$$\|\lambda\| = \sum_{j=1}^{n+1} |\alpha_j|$$

In other words, writing $[t]$ for the linear functional of evaluation at t, such a canonical representation provides us with an extension

$$\sum_{j=1}^{n+1} \alpha_j [t_j]$$

of λ from π_n to all of $C(S)$:= Banach space of continuous functions on S, and this extension has the same norm (on $C(S)$) as does λ (on π_n).

We will give a constructive proof later on of the existence of such a canonical representation for our particular λ. Taking this for granted (or referring for it to Rivlin [1974]), we note that, for the Lagrange polynomials ℓ_j given by

$$\ell_j(t) := \prod_{\substack{i=1\\ i\neq j}}^{n+1} \frac{t - t_i}{t_j - t_i}\ ,\quad j=1,\ldots,n+1$$

we must then have

$$\ell_j(0) = \lambda\ell_j = \sum_{i=1}^{n+1} \alpha_i\, \ell_j(t_i) = \alpha_j$$

This implies that

$$\alpha_j = \prod_{\substack{i=1\\ i\neq j}}^{n+1} \frac{t_i}{t_j - t_i}\ ,\quad \text{all } j$$

hence all coefficients are nonzero if, as we assume, 0 does not lie in S. Further,

$$\alpha_j \alpha_{j+1} > 0 \quad \text{iff} \quad t_j < 0 < t_{j+1}$$

If now p^* is an extremal for λ, then we have

$$\|\lambda\| = \lambda p^* = \sum_{j=1}^{n+1} \alpha_j p^*(t_j) < \sum_{j=1}^{n+1} |\alpha_j| |p^*(t_j)| < \left(\sum_{j=1}^{n+1} |\alpha_j|\right) \|p^*\| = \|\lambda\|$$

so equality must hold throughout this relationship. In particular,

$$\text{sign}(\alpha_j) p^*(t_j) = \|p^*\| = 1, \quad \text{all } j .$$

This pins down p^* uniquely once we know all the t_j's. Explicitly,

$$p^* = \sum_{j=1}^{n+1} \text{sign}(\ell_j(0))\, \ell_j$$

Thus, for $n > 1$, p^* is not just a constant, therefore Theorem 2.15 of Rivlin [1974] shows that the points $t_1, \ldots, t_{n+1}$ are uniquely determined.

If now 0 lies to one side of $[t_1, t_{n+1}]$, then it follows that p^* alternates on $t_1, \ldots, t_{n+1}$, hence p^* is necessarily a multiple of the Chebyshev polynomial for $S \cap [t_1, t_{n+1}]$. Further, $|p^*(t)| > 1$ for t not in $[t_1, t_{n+1}]$. We conclude that in the case of particular interest to us, namely when 0 is in the convex hull of S, there must be some k for which

$$t_k < 0 < t_{k+1}$$

This shows our assertion at the beginning of this section that, in general, P_n need only alternate on n points in S. Further, for $t_k < t < t_{k+1}$,

$$p^*(t) = \sum_{j=1}^{n+1} \left(\text{sign}(\ell_j(0))\right) \ell_j(t) = \sum \left(\text{sign}(\ell_j(t))\right) \ell_j(t)$$

$$= \sum |\ell_j(t)| > \left| \sum \ell_j(t) \right| = 1$$

if $n > 1$. Consequently, then

$$t_k = b := \max S \cap (-\infty, 0], \quad \text{and} \quad t_{k+1} = c := \min S \cap [0, \infty)$$

We gather these various facts in the following theorem, for the record.

Theorem 1 Assume that S is compact and does not contain 0, and $n > 1$. Then

(a) λ has a unique canonical representation, and $\alpha_j = \prod_{i \neq j} t_i / (t_j - t_i)$, $j = 1, \ldots, n+1$.

(b) Correspondingly, λ has a unique extremal, and this is given by

$$p^* = \sum_{j=1}^{n+1} \operatorname{sign}\big(\ell_j(0)\big)\, \ell_j \quad \text{with} \quad \ell_j(t) = \prod_{i\neq j} \frac{t - t_i}{t_j - t_i} \ , \quad \text{all } j.$$

If, in addition, 0 is in the convex hull of S, then

(c) $t_k < 0 < t_{k+1}$ for some k. Further

$$p^*(t_j) = \begin{cases} (-)^{k-j} & , \ j=1,\dots,k \\ (-)^{j+1-k} & , \ j=k+1,\dots,n+1 \end{cases}$$

and $p^*(t) > 1$ for $t_k < t < t_{k+1}$, hence $t_k = \max S \cap [-\infty, 0]$ and $t_{k+1} = \min S \cap [0, \infty)$.

We pointed out earlier that $P_n = p^*/p^*(0)$ could also be obtained as the error in the Chebyshev approximation to 1 from the subspace $\{\Sigma_1^n \beta_j t^j\}$. This subspace forms a Haar set on S as long as S does not contain 0. We could therefore have obtained the above characterization from general criteria such as Kolmogorov's criterion, but the derivation would not have been any simpler. We note that, while P_n is in general not (a multiple of) the Chebyshev polynomial for S, it is always a multiple of a **Zolotarev polynomial** since its alternations over n points characterize it as the error in a best approximation to $\beta_n t^n + \beta_{n-1} t^{n-1}$ from π_{n-2}.

3. Remes algorithm for the extremal polynomial We begin with a statement of the algorithm. In it, we use the abbreviations

$$b := \max S \cap (-\infty, 0] \ , \quad c := \min S \cap [0, \infty)$$

introduced earlier.

Remes algorithm for the extremal polynomial

1. Choose $\mathbf{t} = \{t_j\}_{j=1}^{n+1}$ in S, strictly increasing, and with $t_k = b$, $t_{k+1} = c$ for some k.
2. Set $p := \Sigma_{j=1}^{n+1} \operatorname{sign}\big(\ell_j(0)\big)\, \ell_j$, with $\ell_j(t) := \Pi_{i\neq j}(t-t_i)/(t_j-t_i)$, all j.
3. Set $t_0 := \min S$, $t_{n+2} := \max S$ and construct $\mathbf{s}$ by

$$s_j := \begin{cases} t_j & \text{for } j=k,k+1 \\ \text{the first of the possibly two maxima of } p(t_j)p \text{ in } [t_{j-1},t_{j+1}] \text{ on } S & \text{for } j=1,\ldots,k-1,k+2,\ldots,n+1 \end{cases}$$

4. Choose $\tilde{\mathbf{t}}$ from $\mathbf{s}$ as follows:

   (a) if $p(t_0)p(t_1) < -1$, then $\tilde{\mathbf{t}} := (t_0, s_1, \ldots, s_n)$, and increase k by 1.

   (b) if $p(t_{n+2})p(t_{n+1}) < -1$, then $\tilde{\mathbf{t}} := (s_1, \ldots, s_{n+1}, t_{n+2})$, and decrease k by 1.

   (c) otherwise, $\tilde{\mathbf{t}} := \mathbf{s}$.

5. Set $\mathbf{t} := \tilde{\mathbf{t}}$.

6. Iterate steps 2 through 5.

Theorem 2 The sequence of polynomials produced by the above Remes algorithm converges to p^*.

Proof We first note that the algorithm is well defined at step 4 in that only one of the alternatives (a) and (b) is possible. Indeed, if both (a) and (b) were to occur, then p would alternate on $t_0, \ldots, t_k, t_{k+2}, \ldots, t_{n+2}$, and this is not possible for a polynomial of degree n or less.

As to the convergence, denote by $\tilde{p}$ the polynomial obtained from p after one iteration, i.e., the polynomial constructed from the sequence $\tilde{\mathbf{t}}$ obtained at step 4. We claim that $1 < \tilde{p}(t) \leq p(t)$ for any t in (b,c) and that strict inequality holds here unless $\tilde{\mathbf{t}} = \mathbf{t}$. The first inequality we already observed earlier (for P_n). As to the second, we have

$$\begin{aligned} (-)^{j-k}\,\tilde{p}(\tilde{t}_j) &= 1 \leq (-)^{j-k}p(\tilde{t}_j) \quad \text{for } j=1,2,\ldots,k \\ (-)^{j-k-1}\,\tilde{p}(\tilde{t}_j) &= 1 \leq (-)^{j-k-1}p(\tilde{t}_j) \quad \text{for } j=k+1,\ldots,n+1 \end{aligned}$$

by construction. This implies that, for $b < t < c$,

$$p(t) - \tilde{p}(t) = \sum_{j=1}^{n+1} \left(p(\tilde{t}_j) - \tilde{p}(\tilde{t}_j)\right)\tilde{\ell}_j(t) = -\sum |p(\tilde{t}_j)-\tilde{p}(\tilde{t}_j)||\tilde{\ell}_j(t)| \leq 0$$

and equality occurs only if $p = \tilde{p}$ on the $n+1$ points $\tilde{t}_1, \ldots, \tilde{t}_{n+1}$, i.e., only if $p = \tilde{p}$.

We conclude that the sequence generated by the Remes algorithm decreases monotonely on (b,c), yet is bounded below there. Hence it converges, uniformly on any finite interval, to some polynomial p^*. This polynomial is then a fixed point of the map T in π_n defined by

$$T : p \longrightarrow \tilde{p} ,$$

it being is straightforward to show that T is continuous. But this says that p^* is the desired extremal.

4. Efficient computation of the parameters We were led to study this problem by the work of Roloff [1979] where estimates of the parameters are provided. A result of Roloff's states that the zeros of P_n are approximately distributed in S in a proportion which is independent of n. One might hope that this proportion is determined by measure, i.e., a subinterval of S containing 1/10 the length of S contains about 1/10 of the zeros of P_n. We use this to obtain the initial guess (in step 1) for the Remes algorithm but we also note that this approximate distribution of zeros of P_n is not especially good. Rather, there is also a tendency for the zeros to be distributed equally among the intervals which make up S and the actual distribution resulting from these conflicting tendencies is not easily predicted.

The Lagrange basis for π_n is especially suited for the efficient and stable implementation of the Remes algorithm because one can obtain the polynomial p associated with the current point sequence $\mathbf{t}$ without any computation, because the basis is well conditioned near the optimal $\mathbf{t}$, and because, in the end, the zeros of $P_n = p^*/p^*(0)$ are particularly easily obtained from this form.

For efficiency in evaluating p away from $\mathbf{t}$ one should express p as

$$p(t) = \prod_{j=1}^{n+1} (t-t_j) \sum_{j=1}^{n+1} \alpha_j/(t-t_j)$$

where

$$\alpha_j := p(t_j) \Big/ \prod_{i\neq j} (t_j-t_i)$$

would be calculated once and for all. Also, the derivative of p at t_m is efficiently computed by

$$p'(t_m) = \prod_{j \neq m} (t_m - t_j) \sum_{j \neq m} \frac{\alpha_m + \alpha_j}{t_m - t_j}$$

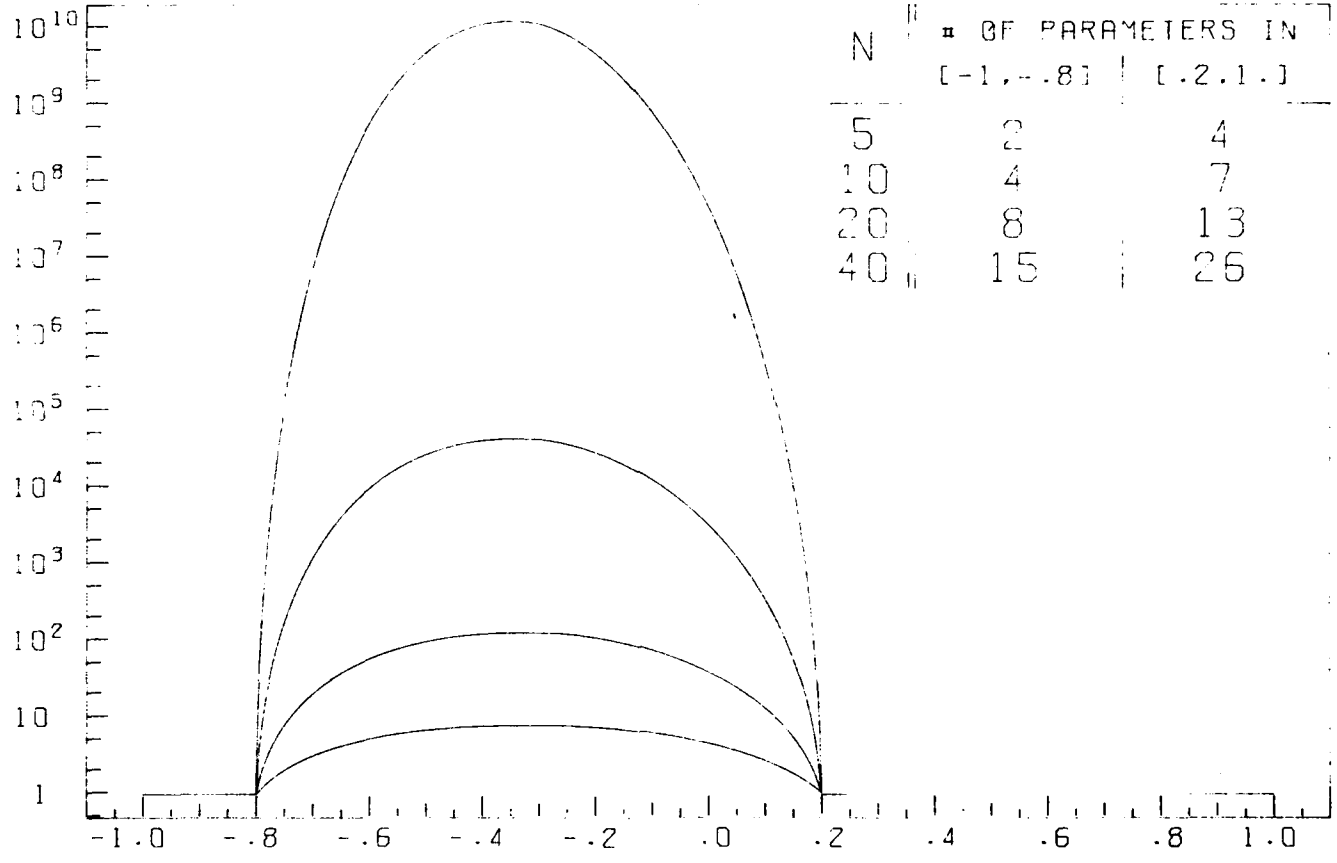
The interior local extrema of p are estimated by parabolic interpolation. As a first step, the unique extremum x^*, say, of the parabola matching p at t_j, t_j and x is found, with $x = t_{j-1}$ or t_{j+1} depending on the sign of $p'(t_j)$. The unique extremum of the parabola matching p at t_j, t_j, and x^* is then taken as the suitable approximation to the desired local extremum of p. This is a version of the standard technique for locating local extrema for use in the Remes algorithm; it is sufficiently accurate for quadratic convergence of the algorithm. Note that in our particular situation there is no need to make a global search for extrema as we know exactly where all the extrema must be.

Once the extremal polynomial p^* is found sufficiently accurately, then its zeros are found by the Modified Regula Falsi. The zeros are already bracketed by the t_j except for one which is outside the interval $[t_1, t_{n+1}]$. This one may be to the left or right of $[t_1, t_{n+1}]$ and may actually be at infinity. We make the transformation $x = 1/t$ and then apply the same method to $[x_1, x_{n+1}]$.

Maehly's second method (see Maehly [1963]) is an alternative method for computing p_n. Its attraction is that it operates directly on the representation of p as $\prod_{j=1}^{n+1}(1 - z_j t)$ and thus does not need the second phase, the computation of the α_j. We judge this approach to be less efficient overall because of the added work of solving for the parameters z_j of the new polynomial each time t is replaced by $\hat{t}$. This also involves the solution of $n+1$ simultaneous equations with a full coefficient matrix. We have not tried this approach; see Dunham [1966] for some remarks concerning improved convergence of this method.

5. Properties of the parameters and convergence rate of the iteration. An examination of particular examples of the extremal polynomials shows that they do not, in general, belong to orthogonal polynomial families and that they are not generated by a three-term

OPTIMAL POLYNOMIALS FOR B=-.8, C=.2
N=5,10,20,40



N	# OF PARAMETERS IN [-1,-.8]	[.2,1.]
5	2	4
10	4	7
20	8	13
40	15	26

FIGURE 1

recurrence relation. It follows from a classical result of Fekete (see Widom [1969]) that $\|P_n\|^{1/n}$ converges as n tends to infinity. Consequently, convergence of Richardson's first order method with these parameters is geometric. Of course, in practice, one is not likely to use the parameters from P_n, P_{n+1}, P_{n+2}, ... in sequence, but is likely to use the parameters for a fixed n cyclically. One still obtains geometric convergence, but examples (such as seen below) show that n should be rather large in order to exploit the method's potential fully.

In order to judge the convergence rates one could expect, we present in Figure 1 the graphs of p^* on the interval $[b,c]$ for the case $S = [-1,-.8] \cup [.2,1]$ and for various values of n. Recall that $P_n = p^*/p^*(0)$, hence $\|P_n\| = 1/p^*(0)$. Further, it is worthwhile at this point to realize that a linear change in the independent variable leaves P_n essentially unchanged. In other words, if this particular S is obtained from some interval pair $S^\# = [a^\#,b^\]] \cup [c^\#,d^\#]$ by the linear change $t = f(t^\#)$, so that $-1 = f(a^\#)$, $-.8 = f(b^\#)$, etc., then the polynomial $P_n^\#$ for $S^\#$ is simply $P_n \cdot f$. In particular, then $\|P_n^\#\| = 1/p^*(f(0))$, thus $1/p^*(t)$ runs through the possible values of such $\|P_n^\#\|$ as t runs between $-.8$ and $.2$. Thus as one moves from $b = -.8$ to $c = .2$, along one of the curves for fixed n, one sees the effect on the achievable error reduction of the location of the origin between the two intervals comprising an interval pair. Note that the rate of convergence becomes 1 and the linear system becomes (possibly) singular as $f(0)$ approaches b or c.

Figure 2 shows the dependence of the rate of convergence on the relative sizes of the two intervals which make up S. A contour plot is given of the maximum possible rate of convergence as b and c vary while $a = -1$ and $d = 1$ remain fixed. This maximum rate occurs at the point where p^* is at a maximum (between b and c) which depends on b and c. This rate approaches 1 as $c-b$ approaches 0 and becomes quite fast as b and c approach -1 and -1, respectively.

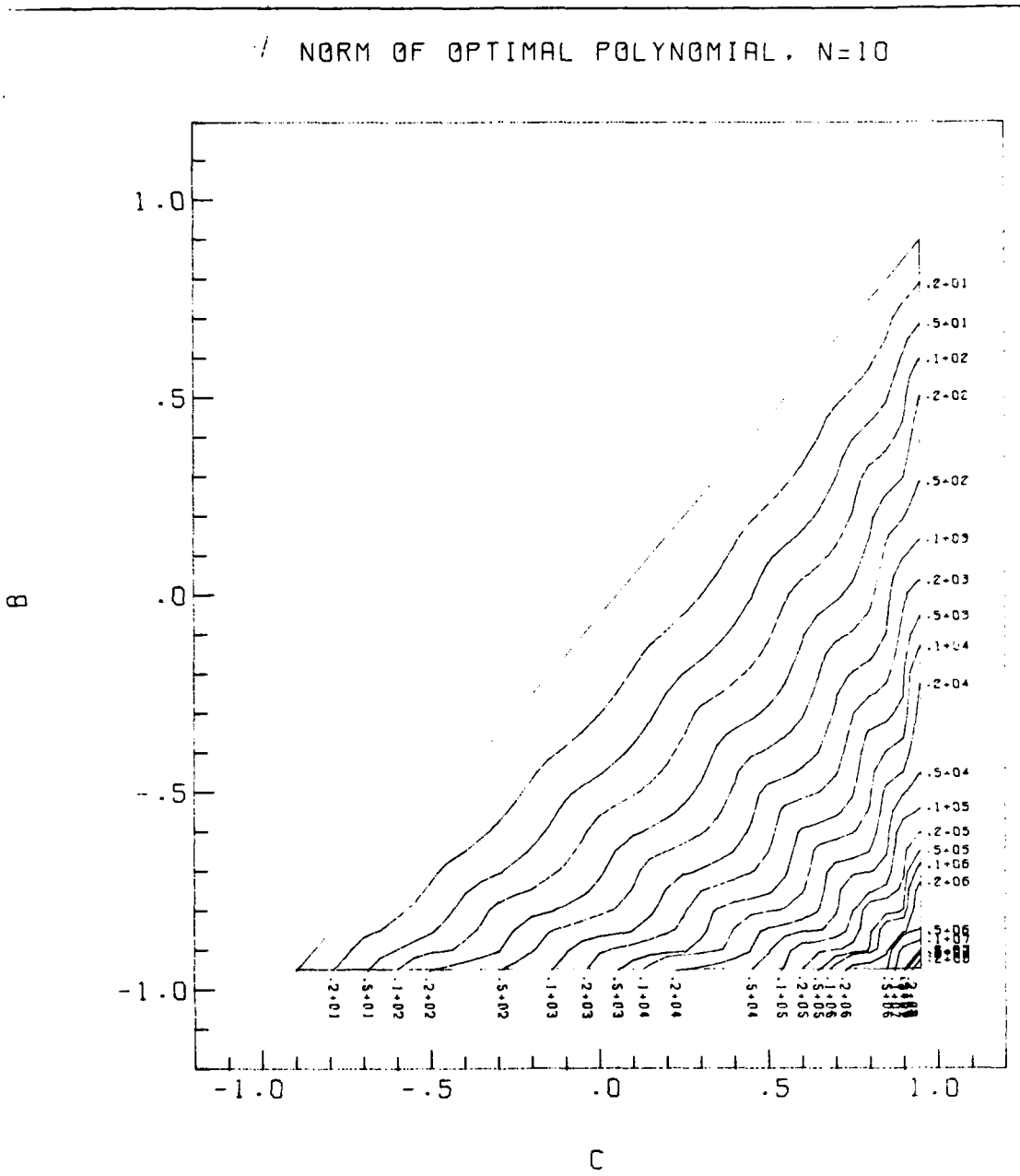

NORM OF OPTIMAL POLYNOMIAL, N=10
1.0
.5
.0
-.5
-1.0
B
-1.0
-.5
.0
.5
1.0
C
.2+01
.5+01
.1+02
.2+02
.5+02
.1+03
.2+03
.5+03
.1+04
.2+04
.5+04
.1+05
.2+05
.5+05
.1+06
.2+06

Figure 2

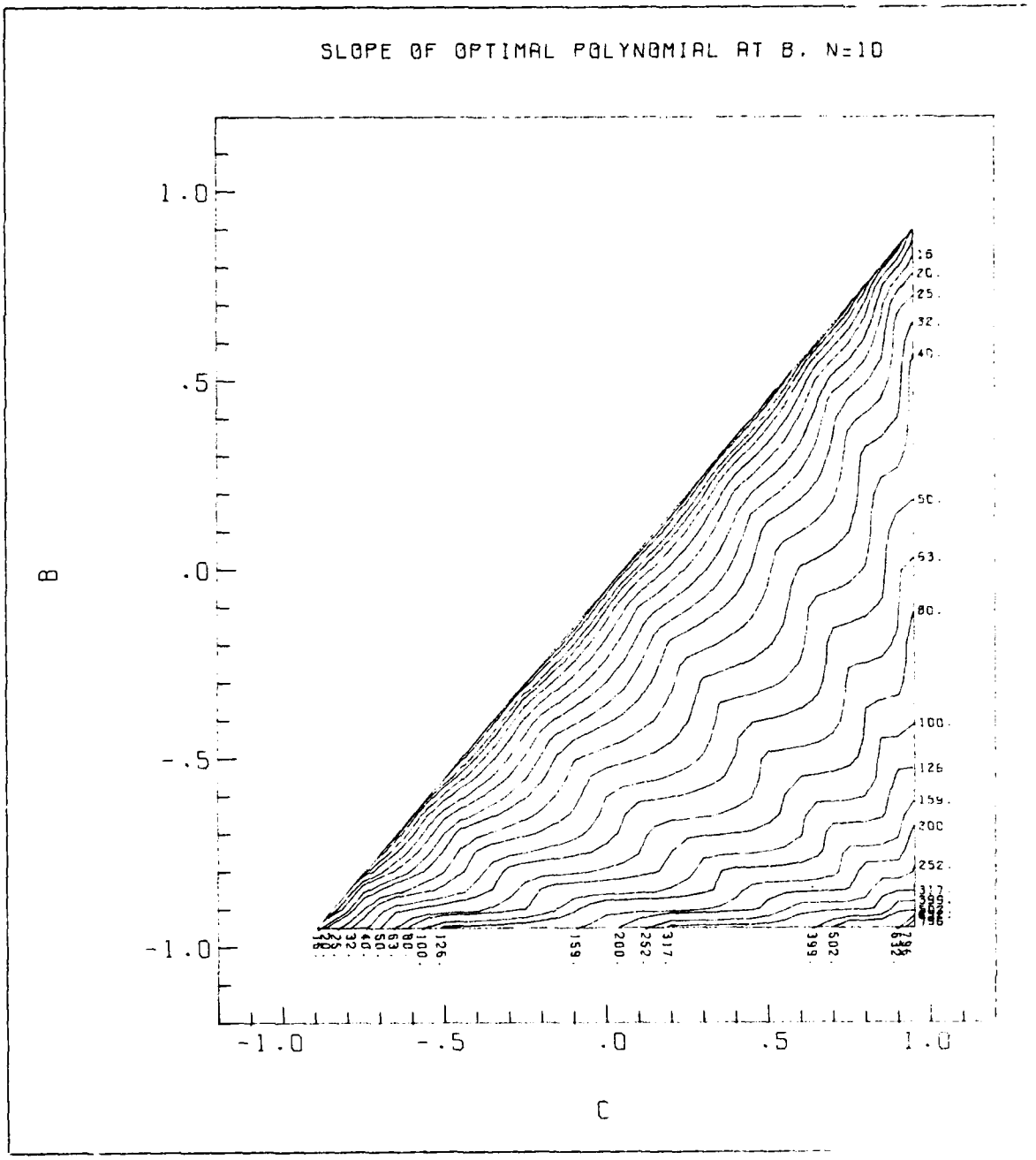

SLOPE OF OPTIMAL POLYNOMIAL AT B, N=10
1.0
.5
.0
-.5
-1.0
B
-1.0
-.5
.0
.5
1.0
C
16
20.
25.
32.
40.
50.
63.
80.
100.
126
159.
200
252.
317
399.
502.

Figure 3

Figure 3 indicates the effect of near singularity of the linear system on the rate of convergence. Again for S an interval pair and for 10 parameters, we plot contours of the logarithm of the slope of p^* at $t = c$. The larger this slope, the less the rate of convergence is degraded by the origin being close to c.

Both Figures 2 and 3 exhibit a somewhat erratic behavior due to the fact that the number of t_j's in each interval is a discrete function of b and c. This suggests that it would be quite difficult to obtain accurate and simple approximation formulae for the parameter distribution.

Acknowledgement We are pleased to thank Gene H. Golub for providing us with a copy of Atlestam's report and for several helpful discussions.

Mathematics Research Center
610 Walnut Street
Madison, WI 53706

Division of Mathematical Sciences
Purdue University
Lafayette, IN 47907

APPENDIX

We list here a Fortran program, written by Frederick Sauer, which realizes the Remes algorithm, given in Section 3, for the extremal polynomial, for the special case that S consists of two (nontrivial) intervals. A more general version which allows for S to consist of finitely many intervals, some or all of which may even be trivial, has also been written by Frederick Sauer who will submit it for publication separately.

```
      PARAMETER DEGREE=25,DEGP1=DEGREE+1,DEGP3=DEGREE+3
      REAL X(DEGP3),XTILDE(DEGP3),ALPHA(DEGP1),PROD(DEGP1)
      INTEGER N,R,IPRINT,MAXIT
      DATA IPRINT,MAXIT, N ,EPS2  ,EPS3  ,EPS4
     1     / 2   , 20  , 10,1.E-2 ,1.E-3 ,1.E-4/
      CALL SETUP(N,-1.,-.8,.2,1.,X,R)
      CALL EXTREM(X,N,R,IPRINT,MAXIT,EPS2,EPS3,EPS4,XTILDE,ALPHA,
     1 PROD,PZERO)
      STOP
      END



      SUBROUTINE EXTREM (X,N,R,IPRINT,MAXIT,EPS2,EPS3,EPS4,XTILDE,
     1 ALPHA,PROD,PZERO)
C     THIS SUBROUTINE FINDS A POLYNOMIAL WHICH IS AN EXTREMAL OF THE LINEAR
C     FUNCTIONAL WHICH IS THE POINT EVALUATION AT THE POINT ZERO.  THE
C     POLYNOMIAL IS CHOSEN FROM THE SPACE OF POLYNOMIALS OF DEGREE LESS
C     THAN OR EQUAL TO N, WHERE N IS A PARAMETER SET BY THE USER. THE NORM
C     ON THIS SPACE IS DEFINED TO BE:
C
C     NORM(P) = MAX(P(T) : (X(1) .LE. T .LE. X(R+1)) OR
C                                    (X(R+2) .LE. T .LE. X(N+3)))
C
C ***** INPUT *****
C   X    - AN ARRAY OF DIMENSION N+3 WHICH CONTAINS THE STARTING VALUES
C          OF THE ABSCISSAE IN X(2), X(3), ... , X(N+2). IN X(1) AND
C          X(N+3) SHOULD BE THE ENDPOINTS (I.E. A AND D RESPECTIVELY)
C
C   N    - THE DEGREE OF THE POLYNOMIAL.
C
C   R    - SUCH THAT X(R+1)=B AND X(R+2)=C IN THE PROVIDED ARRAY X.
C          X(1),X(R+1),X(R+2),X(N+3) DEFINE THE INTERVALS FOR WHICH THE
C          NORM OF THE POLYNOMIAL IS CALCULATED. SEE DEF. OF NORM ABOVE.
C
C   IPRINT  -  =-1 SUPPRESS ALL PRINTING. PZERO WILL NOT BE CALCULATED
C              =0 PZERO = P(0) WILL BE CALCULATED AND PRINTED.
C              =1 ALSO, THE ROOTS OF THE POLYNOMIAL WILL BE CALCULATED
C                 AND PRINTED
C              =2 ALSO, FOR EACH ITERATION A MESSAGE WILL BE PRINTED
C
C   MAXIT   -  THE MAXIMUM NUMBER OF ITERATIONS ALLOWED.
C
C   EPS2    -  A TOLERANCE USED TO CONTROL THE MAJOR ITERATION. THE
C              ROUTINE WILL STOP WHEN THE MAXIMUM CHANGE OF ANY
C              PARTICULAR ELEMENT OF X CHANGES BY LESS THAN EPS2.
C
```

```
C   EPS3,EPS4 - TOLERANCE PARAMETERS USED BY SUBROUTINE REGULA.
C               THE CALCULATED VALUE OF THE ROOT, XRT, WILL BE
C               SUCH THAT XRT WILL BE WITHIN EPS3 OF AN ACTUAL
C               ROOT OR ABS(P(XRT)) WILL BE LESS THAN EPS4.
C
C ***** WORKSPACE *****
C   XTILDE - AN ARRAY OF DIMENSION AT LEAST N+3.
C   ALPHA  - AN ARRAY OF DIMENSION AT LEAST N+1.
C   PROD   - AN ARRAY OF DIMENSION AT LEAST N+1.
C
C ***** OUTPUT *****
C   X    - THE FINAL VALUES OF THE ABSCISSAE OF THE EXTREME POINTS
C          CALCULATED BY THIS ROUTINE.
C   R    - THE FINAL VALUE OF R AS DESCRIBED ABOVE.
C         ALPHA, PROD - THE FINAL VALUES OF ALPHA AND PROD AS CALCULATED
C                BY SUBROUTINE ALCALC ARE RETURNED IN THESE ARRAYS IF
C                IPRINT IS NOT EQUAL TO -1.
C        PZERO - VALUE OF EXTREMAL POLYNOMIAL AT 0.
C         XTILDE - CONTAINS THE RECIPROCALS OF THE ROOTS OF THE POLYNOMIAL IF
C                  IPRINT=1 OR 2. XTILDE(1) CORRESPONDS TO THE ROOT LYING OUT-
C                  SIDE THE INTERVAL. IT WILL BE SET TO ZERO IF THE ADDITIONAL
C                  ROOT DOES NOT EXIST.
C
C       ***** NOTE: THIS SUBROUTINE EXPECTS A STARTING GUESS FOR THE VALUES
C          OF X, AND A VALUE FOR R. THIS CAN BE ACCOMPLISHED BY A CALL TO
C          SUBROUTINE SETUP.
C
            INTEGER R,N,NP1,ITER,IPRINT,MAXIT,ISET,NP3,NP2,RP1,RP2
            REAL X(1),XTILDE(1),ALPHA(1),PROD(1)
            INTEGER I,J
            REAL EPS2,SIGN,SGSL,ERR,EPS3,PZERO
            NP1 = N + 1
            NP2 = N + 2
            NP3=N+3
            ITER = 0
            IF (N .LE. 1) RETURN
         40 IF (IPRINT .EQ. 2) WRITE (6,640) ITER,(X(I),I=2,NP2)
        640 FORMAT('0AFTER ',I3,' ITERATIONS THE POINTS X ARE',/(1X,10E13.5))
            RP1=R+1
            RP2=R+2
      C   GET THE INTERPOLATING POLYNOMIAL
            CALL ALCALC(X,NP1,R,ALPHA,PROD,XTILDE)
            XTILDE(1)=X(1)
            SIGN=1.
            IF (MOD(R,2) .EQ. 1) SIGN=-1.
            XTILDE(NP3)=X(NP3)
            ISET = 0
      C  IF ISET IS CHANGED THE RESULTING X'S MUST BE SHIFTED.
            IF (X(1) .EQ. X(2)) GO TO 45
               CALL EVAL (X(1),X,ALPHA,NP1,VAL)
               IF (VAL*SIGN .GT. 1.) ISET=-1
         45 DO 80 I=2,NP2
               IF ((I .EQ. RP1) .OR. (I.EQ.RP2)) GO TO 70
               J=I-1
               CALL DERIV(X,J,PROD,ALPHA,NP1,SLOPE)
               SGSL=1.
```

```
      IF (SLOPE*SIGN .GT. 0.) SGSL=-1.
      IF (SGSL .EQ. 1) J=I+1
      IF (SLOPE .NE. 0.) GO TO 60
         XTILDE(I)=X(I)
         GO TO 80
   60 SLOPE=-ABS(SLOPE)
      CALL INTERP(I,J,SIGN,X,ALPHA,NP1,SLOPE,SGSL,XTILDE(I))
      SIGN=-SIGN
      GO TO 80
   70 XTILDE(I)=X(I)
      SIGN=1.
   80 CONTINUE
      IF (X(NP2) .EQ. X(NP3)) GO TO 85
         CALL EVAL(X(N+3),X,ALPHA,NP1,VAL)
         IF (VAL*SIGN .LT. -1.) ISET=1
C  SET UP X FOR NEXT ITERATION
   85 R = R - ISET
      ERR=0.
      DO 90 I = 2,NP2
         J = ISET + I
         DIFF = ABS( X(I) - XTILDE(J) )
         IF (DIFF .GT. ERR) ERR = DIFF
   90 X(I) = XTILDE(J)
      IF ( ERR .LT. EPS2) GO TO 100
         ITER = ITER + 1
         IF (ITER .LT. MAXIT) GO TO 40
         WRITE (6,650) MAXIT
  650 FORMAT('0MAXIMUM NUMBER OF ITERATIONS EXCEEDED, MAXIT=',I4)
  100 IF (IPRINT .EQ. 2) WRITE (6,660) (X(I),I=2,NP2)
  660 FORMAT('0AFTER FINAL ITERATION THE POINTS X ARE',/(1X,10E13.5))
      IF (IPRINT .EQ. -1) RETURN
C  EVALUATE EXTREMAL POLYNOMIAL AT 0 .
      CALL ALCALC(X,NP1,R,ALPHA,PROD,XTILDE)
      CALL EVAL(0.,X,ALPHA,NP1,PZERO)
      WRITE(6,670) PZERO
  670 FORMAT('0P(0) =',E20.8)
      IF (IPRINT .EQ. 0) RETURN
C  FIND THE RECIPROCALS OF THE ZEROS OF THE EXTREMAL POLYNOMIAL
      CALL FINDZR(X,ALPHA,NP1,R,EPS3,EPS4,XTILDE)
      RETURN
      END



      SUBROUTINE ALCALC(X,M,R,ALPHA,PROD,WORK)
C  THIS SUBROUTINE CALCULATES THE COEFFICIENTS ALPHA AND PROD I
C  USED TO EVALUATE THE LAGRANGE INTERPOLATING POLYNOMIAL AND ITS
C  DERIVATIVES
C
C ***** INPUT *****
C    X - THE ABSCISSAE OF THE POINTS TO WHICH THE POLYNOMIAL
C        FIT ARE STORED IN X(2), X(3), X(4), ..., X(M+1).
C    M - THE ORDER (DEGREE + 1) OF THE INTERPOLATING POLYNOMIAL .
C    R - IS SUCH THAT X(R+1) = 0, X(R+2) = 0
C
C ***** OUTPUT *****
C    ALPHA - AN ARRAY OF DIMENSION M WHICH IS USED BY SUBROUTINE
```

```
C                                        M
C          ALPHA(I) =  P(X(I+1))/ PRODUCT  (X(I+1)-X(J+1))
C                                      J=1
C                                     J.NE.I
C        WHERE  P(X) IS THE VALUE OF THE POLYNOMIAL AT X. IN PARTICULAR,
C                              (-1)**(I-R) IF I .LE. R
C             P(X(I+1))=
C                              (-1)**(I+1-R) IF I .GT. R
C     PROD - AN ARRAY OF DIMENSION M WHICH IS USED TO STORE THE VALUES
C                           M
C          PROD(I) = PRODUCT (X(I+1) - X(J+1))
C                         J=1
C                        J.NE.I
C
C ***** WORKSPACE *****
C     WORK - AN ARRAY OF DIMENSION M.
      INTEGER R,M,I,J,MM1
      REAL X(1),ALPHA(M),PROD(M),WORK(M),XI,P,SIGN
      MM1 = M-1
      SIGN = 1.
      IF(MOD(R,2) .EQ. 0) SIGN = -1.
      DO 10 I=1,MM1
   10    WORK(I) = X(I+2)
      DO 30 J=1,M
         P = 1.
         XI = X(J+1)
         DO 20 I=1,MM1
   20       P = P*(XI-WORK(I))
         PROD(J) = P
         WORK(J) = XI
         IF (J .EQ. R+1) SIGN = - SIGN
         ALPHA(J) = SIGN/P
   30    SIGN = - SIGN
      RETURN
      END



      SUBROUTINE EVAL(T,X,ALPHA,M,P)
C   THIS SUBROUTINE EVALUATES THE VALUE OF THE LAGRANGE INTERPOLATING
C   POLYNOMIAL AT THE POINT T.
C
C ***** INPUT *****
C    T - POINT AT WHICH POLYNOMIAL IS TO BE EVALUATED.
C    X - ARRAY CONTAINING POINTS AT WHICH THE POLYNOMIAL IS KNOWN.
C    ALPHA - ARRAY OF DIMENSION M WHICH HAS THE COEFFICIENTS CALCULATED
C            BY SUBROUTINE ALCALC.
C    M - ONE PLUS THE DEGREE OF THE POLYNOMIAL.
C
C ***** OUTPUT *****
C    P - THE VALUE OF THE POLYNOMIAL AT T.
C
C
      INTEGER M,I
      REAL X(1), ALPHA(M),T,P,C,DIFF,SGN
C
C EVALUATE THE POLYNOMIAL
```

```
C
      C = 1.
      P = 0.
      DO 10 I = 1,M
         DIFF = (T-X(I+1))
         IF (DIFF .EQ. 0.) GO TO 20
         C = C*DIFF
   10    P = P + ALPHA(I)/DIFF
      P = P*C
      RETURN
   20 SGN = 1.
      IF (MOD(M-I,2) .EQ. 1) SGN = -1.
      P = SIGN (1., ALPHA(I)*SGN)
      RETURN
      END



      SUBROUTINE DERIV(X,I,PROD,ALPHA,M,D)
C  THIS SUBROUTINE EVALUATES THE DERIVATIVE OF THE LAGRANGE INTER-
C  POLATING POLYNOMIAL AT THE ITH POINT OF X.
C
C ***** INPUT *****
C    X - ARRAY CONTAINING POINTS AT WHICH THE POLYNOMIAL IS KNOWN.
C    I - THE DERIVATIVE IS TO BE EVALUATED AT X(I+1)
C    PROD - AN ARRAY OF DIMENSION M WHICH CONTAINS COEFFICIENTS CALCU-
C           LATED BY SUBROUTINE ALCALC.
C    ALPHA - AN ARRAY OF DIMENSION M WHICH CONTAINS COEFFICIENTS
C           CALCULATED BY SUBROUTINE ALCALC.
C    M - ONE PLUS THE DEGREE OF THE POLYNOMIAL.
C
C ***** OUTPUT *****
C    D - THE VALUE OF THE DERIVATIVE AT X(I+1).
C
      INTEGER I,M,J
      REAL X(1),PROD(M),D,XS,ALPHA(M),AS
       XS = X(I+1)
       AS = ALPHA(I)
       D = 0.
       DO 10 J=1,M
          IF (J .EQ. I) GO TO 10
          D = D + (AS + ALPHA(J))/(XS-X(J+1))
   10     CONTINUE
      D = D*PROD(I)
      RETURN
      END



      SUBROUTINE INTERP (I,J,SIGN,X,ALPHA,M,SLOPE,SGSL,XMIN)
C   THIS SUBROUTINE FINDS AN APPROXIMATION TO XMIN WHERE XMIN IS SUCH
C   THAT:
C
C     SIGN*P(XMIN)=MIN(SIGN*P(T):MIN(X(I),X(J)).LE.T.LE. MAX(X(I),X(J)))
C
C     WHERE P(T) IS THE INTERPOLATING POLYNOMIAL AT THE POINT T.
C
```

```
C ***** INPUT *****
C    I,J - INTEGERS SO THAT X(I) AND X(J) DEFINE THE ENDPOINTS OF THE
C        INTERVAL TO BE SEARCHED FOR XMIN.
C    SIGN - REAL VALUE, EITHER +1. OR -1. SO THAT SIGN*P(X(I)) = -1.
C    X - ARRAY OF DIMENSION M+2 WHICH CONTAINS THE ABSCISSAE OF THE
C        POINTS USED FOR LAGRANGE INTERPOLATION.
C    ALPHA - COEFFICIENTS FOR LAGRANGE POLYNOMIAL.
C    M - ORDER (DEGREE PLUS ONE) OF INTERPOLATING POLYNOMIAL.
C    SLOPE - -(ABS(P'(X(I))))
C    SGSL  - HAS THE FOLLOWING VALUE
C                +1.   IF   X(I) .LT. X(J)
C                -1.   IF   X(I) .GT. X(J)
C
C ***** OUTPUT *****
C    XMIN - THE APPROXIMATION OF XMIN CALCULATED BY THE ROUTINE
C
C ***** METHOD *****
C   LET P(T) BE THE LAGRANGE INTERPOLATING POLYNOMIAL AT THE POINT T.
C   S, THE FIRST APPROXIMATION OF XMIN, IS THE MINIMUM OF THE PARABOLA
C   DETERMINED BY P(X(I)), P(X(J)), AND P'(X(I)). XMIN IS THEN THE
C   THE POINT AT WHICH THE PARABOLA INTERPOLATING P(X(I)), P'(X(I)),
C   AND P(S) TAKES ON ITS MINIMUM.
C
      INTEGER I,J,M
      REAL SIGN,X(1),ALPHA(M),SLOPE,SGSL,XMIN
      REAL S,XMAX,F,XI,XT,DIFF2
      XI = X(I)
      IFIRST=1
      S= ABS(XI - X(J))
      XMAX = S
      F= 1.
C  CHECK IF END POINT
      IF ((J .NE. 1) .AND. (J .NE. M+2)) GO TO 10
         IF (X(J) .NE. XI) GO TO 5
            XMIN=XI
            RETURN
C   CALCULATE INTERPOLATING POLYNOMIAL AT X(J)
    5    XT=X(J)
         CALL EVAL(XT,X,ALPHA,M,F)
         F=F*SIGN
C  USE FOR FIRST STEP THE SLOPE AT X(I) AND THE POINT S.
   10 DIFF2= ((F+1.)/S - SLOPE)/S
      IF (DIFF2 .LE. 0) GO TO 999
      S= -SLOPE/DIFF2/2.
      IF (S .LE. XMAX) GO TO 30
         XMIN = SGSL*XMAX + XI
         RETURN
   30 XT= SGSL*S + XI
      CALL EVAL(XT,X,ALPHA,M,F)
      F= SIGN*F
      IF (IFIRST .NE. 1) GO TO 999
      IFIRST=2
      XMAX = S
      GO TO 10
  999 XMIN= XI + SGSL*S
      RETURN
      END
```

```
      SUBROUTINE FINDZR(X,ALPHA,M,R,EPS3,EPS4,WORK)
C   THIS SUBROUTINE FINDS THE ZEROS OF THE LAGRANGE INTERPOLATING POLY-
C   NOMIAL USING THE MODIFIED REGULA FALSI METHOD.  THE RESULTS ARE
C   PRINTED OUT.
C ***** INPUT *****
C    X - ARRAY CONTAINING THE ABSCISSAE OF THE DATA POINTS.
C    ALPHA - ARRAY CONTAINING COEFFICIENTS CALCULATED BY SUBROUTINE
C                ALCALC.
C    M - ORDER OF LAGRANGE POLYNOMIAL.
C    R - INTEGER SUCH THAT ZERO LIES IN THE INTERVAL X(R+1) TO X(R+2).
C    EPS3,EPS4 - TOLERANCE PARAMETERS USED BY SUBROUTINE REGULA. THE
C                CALCULATED VALUE OF THE ROOT, XRT, WILL BE SUCH THAT
C                XRT WILL BE WITHIN EPS3 OF AN ACTUAL ROOT OR
C                ABS(P(XRT)) WILL BE LESS THAN EPS4.
C
C ***** WORKSPACE *****
C    WORK - AN ARRAY OF DIMENSION AT LEAST M.
C
      INTEGER M,R,J,I,MM1
      REAL X(1),WORK(M),ALPHA(M),SIGN,EPS3,EPS4
      REAL A,B,FA,FB,SIGN1,SUM,XRT
      EXTERNAL EVAL,EVALI
      MM1=M-1
      SIGN = 1.
      IF (MOD(R,2) .EQ. 1) SIGN=-1.
      SIGN1 = SIGN
      DO 20 J=2,M
         IF (J .EQ. R+1) GO TO 20
         A = X(J)
         B = X(J+1)
         FA = -SIGN
         FB = SIGN
         CALL REGULA(EVAL,A,B,FA,FB,X,ALPHA,M,EPS3,EPS4,XRT)
         SIGN=-SIGN
         I=J
         IF (J .GT. R+1) I=J-1
         WORK(I) = XRT
   20    CONTINUE
      WRITE (6,620) (WORK(I),I=2,MM1)
  620 FORMAT('0THE ROOTS LYING IN THE INTERVAL (X(2),X(M+1)) ARE',
     1 /(1X,5E20.7))
      DO 21 J=2,M
   21       WORK(J) = 1./WORK(J)
C   FIND ROOT OUTSIDE OF INTERVAL IF IT EXISTS
      SUM = 0.
      ALMAX = 0.
      DO 30 I=1,M
         ABSAL = ABS(ALPHA(I))
         IF (ABSAL .GT. ALMAX) ALMAX = ABSAL
   30    SUM = SUM + ALPHA(I)
      IF (ABS(SUM) .GT. ALMAX * 1.E-4) GO TO 40
   35 WRITE(6,635)
  635 FORMAT('0THE ADDITIONAL ROOT CANNOT BE ACCURATELY CALCULATED',
     1 ' OR IS INFINITE')
```

```
      WORK(1)=0.
      RETURN
   40 IF (SUM*SIGN .GT. 0.) GO TO 70
C    THE ADDITIONAL ROOT IS TO THE LEFT OF X(2)
      A = 1./X(2)
      B = -1.E-4
      FA = -SIGN1
      CALL EVALI (B,X,ALPHA,M,FB)
      IF (FA*FB .GE. 0.) GO TO 35
      GO TO 90
C    THE ADDITIONAL ROOT IS TO THE RIGHT OF X(M+1)
   70 B = 1./X(M+1)
      A = 1.E-4
      FB=-SIGN
      CALL EVALI (A,X,ALPHA,M,FA)
      IF (FA*FB .GE. 0 ) GO TO 35
   90 CALL REGULA(EVALI, A, B, FA, FB, X, ALPHA, M, EPS3,EPS4, XRT)
      WRITE (6,690) XRT
      WORK(1)=XRT
  690 FORMAT('0THE RECIPROCAL OF THE ADDITIONAL ROOT IS A' X=',1PD20.7)
      RETURN
      END



      SUBROUTINE REGULA (FCT,A,B,FA,FB,X,ALPHA,M,EPS3,EPS4,XRT)
C  THIS SUBROUTINE FINDS THE ZERO OF THE LAGRANGE POLYNOMIAL WHICH IS
C  IN THE INTERVAL (A,B). THE MODIFIED REGULA FALSI METHOD IS USED.
C ***** INPUT *****
C    A - LEFT END POINT OF INTERVAL IN WHICH A ROOT LIES.
C    B - RIGHT END POINT OF INTERVAL IN WHICH A ROOT LIES.
C    FA - THE VALUE OF THE POLYNOMIAL AT THE POINT A.
C    FB - THE VALUE OF THE POLYNOMIAL AT THE POINT B.
C    X - ARRAY CONTAINING THE ABSCISSAE OF THE DATA POINTS.
C    ALPHA - ARRAY CONTAINING COEFFICIENTS CALCULATED BY SUBROUTINE
C                ALPHAS
C    M - ORDER OF LAGRANGE POLYNOMIAL.
C    EPS3,EPS4 - TOLERANCE PARAMETERS. THE CALCULATED VALUE OF XRT
C                WILL BE SUCH THAT XRT WILL BE WITHIN EPS3 OF AN
C                ACTUAL ROOT OR ABS(P(XRT)) WILL BE LESS THAN EPS4.
C
C ***** OUTPUT *****
C    XRT - THE CALCULATED VALUE OF THE ROOT.
C
      INTEGER M
      REAL X(1),ALPHA(M),A,B,FA,FB,EPS3,EPS4,FW,XRT,FXRT
      EXTERNAL FCT
      FW = FA
   10 XRT = (FB*A - FA*B)/(FB - FA)
      CALL FCT(XRT,X,ALPHA,M,FXRT)
      IF (ABS(FXRT) .LE. EPS4) RETURN
      IF ( SIGN(1.,FXRT)*FA .GT. 0. ) GO TO 12
      B = XRT
      FB = FXRT
      IF (SIGN(1.,FXRT) * FW .GT. 0.) FA=FA/2.
      GO TO 15
   12 A = XRT
```

```
      FA = FXRT
      IF (SIGN(1.,FXRT)*FW .GT. 0.) FB=FB/2.
   15 FW = FXRT
      IF (B-A .GT. EPS3) GO TO 10
      XRT= (A + B) / 2.
      RETURN
      END



      SUBROUTINE EVALI(T,X,ALPHA,M,P)
C     THIS SUBROUTINE EVALUATES (T**(M-1))*P(1/T) WHERE P(1/T) IS THE
C     LAGRANGE INTERPOLATING POLYNOMIAL EVALUATED AT 1/T
C
C ***** INPUT *****
C       T - POINT AT WHICH POLYNOMIAL IS TO BE EVALUATED.
C       X - ARRAY CONTAINING POINTS AT WHICH THE POLYNOMIAL IS KNOWN.
C       ALPHA - ARRAY OF DIMENSION M WHICH HAS THE COEFFICIENTS CALCULATED
C               BY SUBROUTINE ALCALC.
C       M - ONE PLUS THE DEGREE OF THE POLYNOMIAL.
C
C ***** OUTPUT *****
C       P - THE VALUE OF THE POLYNOMIAL AT T.
C
C
      INTEGER M,I
      REAL X(1), ALPHA(M),T,P,C,DIFF,SGN
C
C EVALUATE THE POLYNOMIAL
C
      C = 1.
      P = 0.
      DO 10 I = 1,M
         DIFF = (1.-T*X(I+1))
         IF (DIFF .EQ. 0.) GO TO 20
         C = C*DIFF
   10    P = P + ALPHA(I)/DIFF
      P = P*C
      RETURN
   20 SGN = 1.
      IF (MOD(M-I,2) .EQ. 1) SGN = -1.
      P = SIGN (1., ALPHA(I)*SGN)*T**(M-1)
      RETURN
      END



      SUBROUTINE SETUP (N, A, B, C, D, X, R)
C     THIS SUBROUTINE SETS UP THE ARRAY X IN PREPARATION FOR A CALL TO
C     SUBROUTINE EXTREM.
C
C ***** INPUT *****
C       N - THE DEGREE OF THE POLYNOMIAL TO BE USED.
C       A, B, C, D - INTERVAL ENDPOINTS. THE INTERVAL PAIR IS (A,B), (C,D)
C
C ***** OUTPUT *****
C       X - AN ARRAY OF DIMENSION N+3 CONTAINING THE CALCULATED VALUES.
```

```
C     R - INTEGER SUCH THAT X(R+1) = B  AND X(R+2) = C.
C
      INTEGER N,R,RM1,NMR
      REAL X(1), A,B,C,D,DEL1,DEL2
      IF (N .LE. 1) RETURN
      DEL1 = B - A
      DEL2 = D - C
      R = INT( FLOAT(N-1) * (DEL1/(DEL1 + DEL2))) + 1
      IF (R .LE. 1) GO TO 20
         RM1 = R - 1
         DEL1 = DEL1 / FLOAT(RM1)
         DO 10 I=2,R
   10       X(I) = A + FLOAT(I-2)*DEL1
   20 NMR = N-R
      IF (NMR .LT. 1) GO TO 40
         DEL2 = DEL2 / FLOAT (NMR)
         DO 30 I=1,NMR
   30       X(R+I+1) = C + FLOAT(I-1)*DEL2
   40 X(1) = A
      X(R+1) = B
      X(N+2) = D
      X(N+3) = D
      RETURN
      END
```

References

N.I. Akhieser [1928] Über einige Funktionen, die in gegebenen Intervallen am wenigsten von Null abweichen, Izvestia Kazanskovo Fiz.-Mat.Obshchestva.

R.S. Anderssen and G.H. Golub [1972] Richardson's non-stationary matrix iterative procedure, Report STAN-CS-72-304, Comp.Sci.Dept., Stanford U., Stanford,CA.

B. Atlestam [1977] Tschebycheff-polynomials for sets consisting of two disjoint intervals with application to convergence estimates for the conjugate gradient method, Research Report 77.06R, Dept.Computer Sciences, Chalmers University of Technology and the University of Göteborg, Sweden .

C.B. Dunham [1966] Convergence problems in Maehly's second method.II, J.Assoc.Comp.Mach. 13, pp.108-113.

V.I. Lebedev [1969] An iteration method for the solution of operator equations with their spectrum lying on several intervals, Zhur.Vych.Mat. i Fiz. 9, pp.1247-1252; an English transl.has appeared in R.S. Anderssen and G.H. Golub [1972] .

H.J. Maehly [1963] Methods for fitting rational approximations. Parts II and III, J.Assoc.Comp.Mach. 10, pp.257-277.

C.C. Paige and M.A. Saunders [1975] Solution of sparse indefinite systems of linear equations, SIAM J.Numer.Anal. 12, pp.617-629.

T.J. Rivlin [1974] The Chebyshev polynomials, John Wiley & Sons, New York.

R.R. Roloff [1979] Iterative solutions of matrix equations for symmetric matrices possessing positive and negative eigenvalues, Ph.D.Thesis, Dept.Comp.Sci., University of Illinois at Urbana-Champaign, Urbana IL 61801 .

H. Widom [1969] Extremal polynomials associated with a system of curves in the complex plane, Adv.Math. 3, pp.127-232.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: 2107	2. GOVT ACCESSION NO.: AD-A093 568
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE (and Subtitle): EXTREMAL POLYNOMIALS WITH APPLICATION TO RICHARDSON ITERATION FOR INDEFINITE LINEAR SYSTEMS	5. TYPE OF REPORT & PERIOD COVERED: Summary Report - no specific reporting period
	6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): Carl de Boor and John R. Rice	8. CONTRACT OR GRANT NUMBER(s): DAAG29-80-C-0041 MCS 77-01408
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Mathematics Research Center, University of Wisconsin 610 Walnut Street Madison, Wisconsin 53706	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 3 - Numerical Anlaysis and Computer Science
11. CONTROLLING OFFICE NAME AND ADDRESS: See Item 18	12. REPORT DATE: August 1980
	13. NUMBER OF PAGES: 26
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office):	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

U. S. Army Research Office
P.O. Box 12211
Research Triangle Park
North Carolina 27709

National Science Foundation
Washington, D. C. 20550

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Richardson iteration, symmetric indefinite, Remes, norm preserving extension, norm calculation for linear functional, Chebyshev polynomial

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The application of Richardson iteration to a symmetric, but indefinite linear system requires certain parameters which can be determined from the zeros in the error of a certain best polynomial approximant on some set S known to contain the spectrum of the coefficient matrix. It is pointed out that this error can also be obtained as a multiple of the extremal polynomial for the linear functional $p \mapsto p(0)$, and this leads to an efficient Remes type algorithm for its determination. A program incorporating this algorithm for the case that S consists of two intervals bracketing zero is also given.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)