▼Journal of Research (Science), Bahauddin Zakariya University, Multan, Pakistan, 2000, Vol. 11, No. 1, pp.18-25 ISSN 1021-1012

SIMULTANEOUS DETERMINATION OF ALL THE ZEROS OF CHEBYSHEV POLYNOMIAL

Nazir Ahmad Mir*, Zahida Akram
Centre for Advanced Studies in Pure & Applied Mathematics,
Bahauddin Zakariya University, Multan, Pakistan.

Abstract: An algorithm based on modified improved Ehrlich method is developed which finds simultaneously all the zeros of Chebyshev polynomials of first and second kind, the Chebyshev polynomials are generated by a three term recurrence relation. The only information required is degree of the polynomial. The same algorithm works for finding simultaneously all the zeros of real polynomials requiring information about its degree and coefficients.

***Keywords**: Zeros, Polynomials, Generalised Basis, Simultaneous Methods.*

1. INTRODUCTION

An algorithm[1,2] was designed for finding simultaneously all the zeros of a real as well as complex polynomials, expressed in monomial basis. The two algorithms required information only about degree and coefficients of the polynomial. Six methods belonging to the Durand-Kerner and Ehrlich families[3,4,5,6,7] were compared numerically in the above mentioned papers. It was found that the modified improved Ehrlich method was the best amongst the six methods considered. Here an algorithm based on modified improved Ehrlich method is developed for finding simultaneously all the zeros of Chebyshev polynomials of first and second kind, the only requirement being that the Chebyshev polynomial should be generated by a three term recurrence relation[8] which is described in Section 2.

The generalised Chebyshev polynomial of degree n would be a real polynomial of the form

$$p_n(z) = a_0 z^n + a_1 z^{n-1} + \dots + a_{n-1} z + a_n \qquad (1.1)$$

, where a_i, $i = 0,1,\dots n$, are real. The implemented algorithm also works for finding simultaneously all the zeros of a real polynomial, and has already been tested on over 250 polynomials of varying degrees[1]. The modified improved Ehrlich method is given in an algorithm form in Section 3. In Section 4, a way of generating initial estimates is described where as termination criterion and polynomial evaluation is discussed in Sections 5 and 6 respectively.

Section 7 contains some discussion on the results of Chebyshev polynomials of first and second kind of varying degrees. Section 8 contains a description of the form of the implemented algorithm, which is given as a FORTRAN procedure in the appendix.

2. THE BASIS POLNOMIAL

The basis polynomial[8] considered will form polynomials $\{\phi_r(z)\}$, where $\phi_r(z)$ is of exact degree r in z, $r = 0,1,2,\dots$ The family will be generated by

$$\phi_{-1}(z) = 0,\ \phi_0(z) = g_{1,0},$$

$$\phi_{k+1}(z) = (z g_{1,k+1} + g_{2,k+1})\phi_k(z) + g_{3,k+1}\phi_{k-1}(z) \qquad (2.1)$$

$$k \ge 0,\ g_{1,k+1} \ne 0.$$

Then $\{\phi_r(z)\}$, $r = 0,1,\dots,n$, form a linearly independent set and hence provides a basis for the representation of any polynomial of degree n. Many basis can be generated in this way, but we consider here the following two basis:

*Author to receive correspondence

(i) The Chebyshev polynomials of the first kind

$$\phi_r(z) = T_r(z),$$
$$g_{1,0} = 0, g_{1,1} = 1, g_{2,1} = 0, g_{1,k} = 2, g_{2,k} = 0,$$
$$g_{3,k} = -1,$$
$$k \geq 2,$$

(ii) The Chebyshev polynomials of the second kind

$$\phi_r(z) = U_r(z),$$
$$g_{1,0} = 1, g_{1,k} = 2, g_{2,k} = 0, g_{3,k} = -1, k \geq 1$$

These two bases are the examples of orthogonal basis and, of course any orthogonal basis is generated three-term recurrence of the form (2.1). The intervals of orthogonality of the two bases are [-1,1]. These two bases will be covered in the implementation of the algorithm considered.

3. ALGORITHM FOR MODIFIED IMPROVED EHRLICH METHOD

We give the algorithm for updating a set of zero estimates $z_1, z_2, \ldots, z_n$ to obtain improved estimates $\tilde{z}_1, \tilde{z}_2, \ldots, \tilde{z}_n$. Within the algorithm, the polynomial is denoted by $p(z)$ and its derivative by $p'(z)$.

We assume some initial ordering of the indices, say (1,2…n) and each stage of algorithm is extended for i running through these values.

(i) $\Delta_i = -p(z_i) / p'(z_i)$

(ii) $\overset{*}{z_i} = z_i + \Delta z_i$

(iii) $\Delta z_i = \Delta_i \Big/ \left[1 + \Delta_i \sum_{k=1,k\neq i}^{n} \dfrac{1}{z_i - \overset{*}{z_k}} \right]$

with $\overset{*}{z_k} = \begin{cases} z_k + \Delta z_k, & k < i \\ z_k + \Delta z_k, & k > i \end{cases}$

(iv) $\tilde{z}_i = z_i + \Delta z_i$

(v) The ordering of the indices is reversed before the next iteration and this alternation continues until convergence.

4. GENERATION OF INITIAL ESTIMATES

Consider the polynomial (1.1) having zeros $z_1, z_2, \ldots, z_n$. Henrici [9], states that these roots all lie inside a circle of radius Beta

$$2 \max_{1\leq k\leq n} \left| a_k / a_0 \right|^{1/k} \tag{4.1}$$

where Beta is a bound on the largest zero of a polynomial (1.1). It is a well known result[10] that if $p(z)$ has a zero inside the circle $|z| \leq \rho$ $p(\rho z)$ has a zero inside the unit circle.This result is true for both real and complex polynomial The polynomial (1.1) was therefore, scaled using the bound (4.1) to bring all of its zeros into the unit circle. The arbitrary initial estimates for the zeros of the polynomial (1.1) were taken to be the points uniformly spaced round the unit circle having the centre at the origin, that is , the points $\exp[2\pi(k-1)/n \; i]$ in the complex plane. 0.05 was added to the exponent argument to avoid symmetric distributions which may cause numerical difficulties.

5. CONVERGENCE CRITERION

This is provided by computing a Bound on the accumulated round off error in the computed value of the polynomial.When this indicates that the later value can be fully accounted for by rounding error, the iterative process is terminated, as there may be no useful information available from which to determine an improved estimate. Bounds of this type have been given by Adams[11], Peters and Wilkinson[12] and Grant and Hitchins [13]. In practice , they have been found to be extremely reliable and accurate, particularly, if one or two extra iterations are performed to allow for the conservative nature of the bound. Their major disadvantage is the cost of evaluating them.

6. POLNOMIAL EVALUATION

As ultimately, we are getting polynomials with real coefficients only, they and their derivatives can be evaluated at $\alpha + i\beta$ by executing the algorithm

$$p = -2\alpha, \; q = \alpha^2 + \beta^2,$$
$$b_0 = a_0, b_1 = a_1 - p\, b_0,$$
$$c_0 = b_0, c_1 = b_1 - p\, c_0,$$
$$b_k = a_k - p\, b_{k-1} - q\, b_{k-2}, \; k = 2(1)n-1,$$
$$c_k = b_k - p\, c_{k-1} - q\, c_{k-2}, \; k = 2(1)n-3,$$
$$b_n = a_n + b_{n-1} - q\, b_{n-2}$$
$$c_{n-2} = b_{n-2} - q\, c_{n-4}$$

when $p_n(\alpha + i\beta) = b_n + i\beta\, b_{n-1}$

and $p'_n(\alpha + i\beta) = (-2\beta^2 c_{n-3} + b_{n-1}) + i\,2\beta(\alpha\, c_{n-3} + c_{n-2})$.

Following Adams[2], a bound on the error can be found using

$$e_0 = 0.8|b_0|, \; e_k = \sqrt{q}\, e_{k-1} + |b_k|, \; k = 1(1)n,$$

when the process is terminated if

$$|p_n(\alpha + i\beta)| \le (2|\alpha\, b_{n-1}| - 8(|b_n| + \sqrt{q}|b_{n-1}|) + 10\, e_n)\rho$$

which is a machine constant, the smallest number which when added to one produces a change. In the numerical result ρ was taken to be 2^{-53}.

7. NUMERICAL RESULTS

The zeros of Chebyshev polynomials of first and second kind are real and always lie in[-1,1]. A discussion on some estimated zeros of Chebyshev polynomials of first and second kind of varying degrees using the implemented algorithm is given below:

7.1 Estimated Zeros of Chebyshev Polynomials of First Kind

Polynomial of degree 15

Real Part	imaginary Part
0.7431448255E+00	-0.9358454147-113
-0.2079116908E+00	0.0000000000E+00
-0.8660254038E+00	0.1029511518E-83
-0.9510565163E+00	-0.1034073842-105
-0.4067366431E+00	-0.9936008577-112
0.5877852523e+00	0.1005382342E-86
0.9945218954E+00	-0.2010356545-110
0.9510565163E+00	0.0000000000E+00
0.4067366431E+00	-0.4735102032E-91
-0.5877852523E+00	-0.1066044944E-91
-0.9945218954E+00	-0.4272697189E-91
-0.7431448285E+00	-0.6380428220E-89
0.2079116908E+00	-0.1123622325E-93
0.8660454038E+00	0.0000000000E+00
0.0000000000E+00	0.0000000000E+00

Here, four zeros of polynomial are correct to the last decimal place and the largest error in the remaining zeros is $0.1\text{x}10^{-83}$.

Polynomial of Degree 20

Real Part	Imaginary Part
0.760405965E+00	-0.2081566379E-70
0.2334453639E+00	-0.4766448662E-87
-0.3826834324E+00	-0.5253732430E-84
-0.8526401644E+00	-0.1743652001E-79
-0.9969173337E+00	0.3111507639E-60
-0.7604059656E+00	-0.1602604814E-70
-0.2334453639E+00	-0.8433758355E-80
0.3826834324E+00	-0.4138123923E-85
0.8526401644E+00	-0.8542112537E-77
0.9723699204E+00	0.0000000000E+00
0.9238795325E+00	0.0000000000E+00
0.5224985647E+00	-0.2027144756E-75
-0.7845909573E-01	0.0000000000E+00
-0.6494480483E+00	0.0000000000E+00
-0.9723699204E+00	0.0000000000E+00
-0.9238795325E+00	-0.1982483825E-74
-0.5224985647E+00	0.0000000000E+00
0.7845909573E-01	0.0000000000E+00
0.6494480483E+00	-0.1244603056E-59
0.9969173337E+00	0.3262652234E-54

We observe that seven zeros of polynomial of degree 20 are correct to the last decimal place and the largest error in the remaining zeros is $0.3x10^{-54}$.

7.2 Estimated Zeros of Chebyshev Polynomials of Second Kind

Polynomial of degree 15

Real Part	imaginary Part
0.7071067812E+00	-0.9762249700E-99
-0.1950903220E+00	0.00000000000E+00
-0.8314696123E+00	0.1256727927E-87
-0.9238795325E+00	-0.4299069269E-93
-0.3826834324E+00	-0.1513758426-102
0.5555702330E+00	-0.9882804745E-99
0.9807852804E+00	-0.1563865384E-97
0.9238795325E+00	0.0000000000E+00
0.3826834324E+00	0.00000000000E+00
-0.5555702330E+00	0.9588073174E-93
-0.9807852804E+00	0.2897817305E-69
-0.7071067812E+00	0.1207497965E-91
0.1950903220E+00	0.2035037952E-97
0.8314696123E+00	0.1521544815E-94
0.00000000000E+00	0.00000000000E+00

We observe that four zeros of polynomial of degree 15 are correct to the last decimal place and the largest error in the remaining zeros is $0.3x10^{-69}$.

Polynomial of degree 20

Real Part	imaginary Part
0.7330518718E+00	0.00000000000E+00
0.2225209340E+00	0.00000000000E+00
-0.3653410244E+00	0.2900212312-100
-0.8262387743E+00	0.00000000000E+00
-0.9888308262E+00	-0.2197720938E-86
-0.7330518718E+00	0.00000000000E+00
-0.2225209340E+00	0.8701176395E-91
0.3653410244E+00	0.00000000000E+00
0.8262387743E+00	0.3341234145E-82
0.9555728058E+00	0.7053365926E-81
0.9009688679E+00	0.3306063417E-82
0.5000000000E+00	0.00000000000E+00
-0.7473009359E-01	0.00000000000E+00
-0.6234898019E+00	0.8820562125E-82
-0.9555728058E+00	0.1700442537E-80
-0.9009688679E+00	0.00000000000E+00
-0.5000000000E+00	0.5131358458E-84
0.7473009359E-01	-0.1379593765E-92
0.6234898019E+00	-0.1274847264E-82
0.9888308262E+00	-0.1483682460E-66

We observe that eight zeros of polynomial of degree 20 are correct to the last decimal place and the largest error in the remaining zeros is $-0.1x10^{-66}$

8. THE IMPLEMENTATION

The FORTRAN subroutine given in the Appendix is a direct implementation of the algorithm given in 3 Double-length real arithmetic is used throughout to allow for a termination criterion based on the use of Adams type error bound given in 5. If required, starting estimates are taken in the form

$$\exp[2\pi(k-1)/n + 0.05\, i], k = 1(1)n.$$

Polynomial is scaled using an upper bound on the largest zero of (1.1). After the zeros of scaled polynomial have been calculated, they are transferred back to give zero estimates for original polynomial evaluation.

Several different modes of entry are possible. Initial estimates may or may not be specified. If they are given as rlz(k) + i cmz (k),k =1(1)n, the character parameter ans should be ‘y’ or ‘Y’. The logical parameter con should be ‘true’ , if up-dating of an estimate is to cease once it has been detected as having converged; a value ‘false’ means that up-dating will continue until all estimates are indicated as having converged on the same sweep .

Chebyshev polynomials of odd degree have zero root which cause numerical difficulties. Therefore, a logical parameter zflag was introduced to separate zero root from the polynomial. This parameter is set to ‘false’ if it has zero root, i.e. if constant term of the polynomial is zero.

On exit, the coefficients are unaltered . A successful conclusion is indicated by icode having the value 1, when the computed root estimates are available as rlz(k) + i cmz(k) , k=1(1)n. The integer array itusd gives the number of iterations to convergence for the individual estimates. A value of -1 for icode indicates that not all the roots have converged within the permitted number of iterations ; those that have not converged being shown by the corresponding itusd entry 0.

There are some error exists from the subroutine indicated by the parameter iex, which normally has value 0. A value of –2 indicates that either the leading coefficient in the polynomial is 0 or the degree is less than 1. A value of-1 indicates that division by the complex number 0.0 + i 0.0 has been attempted within the subroutine.There are ten internally used arrays of fixed length allowing for the solution of polynomials of degrees not greater than 50.

REFERENCES

1. O. Aberth, "Iteration methods for finding all the zeros of a polynomial simultaneously", *Math. Comp.* , 1973, **27**,339-344.
2. D. A. Adams, "A stopping criterion for polynomial root finding", *Comm. ACM,* 1967, **10**, 655-658.
3. E. Durand, "Solutions Numeriques des Equations Algebriques", Tome I, Masson, Paris,1960.
4. L.W. Ehrilich,"A modified Newton method for Polynomials", *Comm., ACM,* 1967,**10**, 107-108.
5. J. A. Grant and N. A. Mir, "A numerical comparison of methods for finding simultaneously all the zeros of a real polynomial", Report No. 93-36, University of Bradford, U.K., 1993.
6. J.A. Grant and A.A. Rahman, "Determination of the zeros of a linear combination of generalised polynomials", *Journal of Computational and Applied Mathematics,* 1992, **42**, 269-278.
7. J. A. Grant, and G. D. Hitchins," Two algorithms for the solution of polynomial equations to limiting machine precision", *Comp. J.*, 1975, **18**, 258-264.
8. P. Henrici, and B.O. Watkins," Finding zeros of a polynomial by the Q-D algorithm", *Comm. ACM,* 1965,**8,** 570-574.
9. I. O. Kerner, I. O "Ein Gesamschrittverfahren zur Berechnung der Nullstellen eines Polynoms", *Num.Math*., 1966, **8**, 290-294.
10. N.A. Mir and Faisal Ali, "A comparison of methods for finding simultaneously all the zeros of a complex polynomial", *Sci. Int.,* (accepted)
11. A. W. M Nourein,., "An improvement on two iteration methods for simultaneous determination of the zeros of a polynomial", *Intern. J. Computer Math*., 1977, **6,** 241-252.
12. G. Peters, and J. H. Wilkinson, "Practical problems arising in the solution of polynomial equations", *J. Inst.Math Appl.,* 1971, **6**, 16-35.
13. A. Ralston, "A first course in numerical analysis", McGraw Hill ,1965.

APPENDIX

```
c subroutine miehrl calculates  zero
c estimates using modified
c improved Ehrlich method
  subroutine miehrl(a,n,rlz,cmz,maxit,
+itusd,icode,iex,ans,con,kind,zflag)

c attempts  to find the  zeros  of a
c chebyshev polynomial equation
c a - double precision one-dimensional
c array of the coefficients
c rlz,cmz –double precision  array of
c initial  estimates  of   real  and
c imaginary  parts   of  zeros  on
c entry,  computed    estimates   on
c exit
c maxit -maximum  number  of  iterations
c allowed
c itusd - 1-dimensional  integer   array
c giving no.of iterations to   convergence
c for the  individual zero estimates
c icode - on successful    conclusion has
c value 1, otherwise -1
c iex - an integer    indicating  error
c conditions,-2 if leading coefficient is
c zero or  n<2, -1 for      overflow etc
c ans - character,     if 'y' or'Y'  then
c initial    estimates    are   available
c otherwise they are  generated internally
c con      –logical   variable    which
c determines whether estimates are
c up-dated  after  convergence  or   not,
c true means not
c zflag    -logical  variable    which
c separates the zero root from the
c chebyshev polynomial equation
```

```
c kind-integer,1if chebyshev poly of
c first kind,2 if of second kind
  implicit double precision (a-h,o-z)
  dimension a(n+1),b(51),rdelta(50),

+cdelta(50),rzstar(50),czstar(50),rp(5
+0),cp(50),rlz(n),cmz(n),iflag(50),
+itusd(n),rpd(51),cpd(51),d(50)
character ans
logical con,sat,zflag
if(.not.(ans.eq.'Y'.or.ans.eq.'y'))
then call chpol(a,n,kind)
endif

      if(a(n+1).eq.0.0) then
         n=n-1
         zflag = .true.
      endif

      iex=0
c if leading coefficient zero or  n<2
c force c exit
      if (a(1).eq.0.0.or.n.lt.2) then
              iex=-2
              return
      endif
c initialize parameters
      np1=n+1
      icode=1
      iroot=0
      do i=1,n
          itusd(i)=0
      enddo
      k1=1
      k2=n
      k3=1
c copy the coefficients
      do i=1,np1
          b(i)=a(i)
      enddo
      if (b(1).eq.1.0) goto 1
c making leading coefficient unity
      const=1.0/b(1)
      do i=1,np1
          b(i)=b(i)*const
      enddo

  1  if (ans.eq.'Y'.or.ans.eq.'y')
goto 2
c  find bound on the largest zero
      call bnd(b,np1,beta)
c  scale the polynomial to bring zeros
in c unit circle
      call scale(b,np1,beta)
c generate inital estimates round the
unit c circle
      call gstval(n,rlz,cmz)
c set convergence flags
  2 do i=1,n
          iflag(i)=1
      enddo
      do l=1,maxit

c calculate poly and deriv values for non-
c converged roots and store
c in arrays rp,cp rpd and cpd
      if (.not.con) iroot=0
      do 20 i=1,n
      if (con .and. iflag(i).eq.0) goto 20
   call evaluate(b,n,rlz(i),cmz(i),rp(i),
  +cp(i),rpd(i),cpd(i),sat)
          if (sat .and. l.gt.1) then
              iflag(i)=0
              iroot=iroot+1
              itusd(i)=l
          endif
          bf=rpd(i)**2+cpd(i)**2
          if (bf.eq.0.0) goto 12
rdelta(i)=(-rp(i)*rpd(i)-cp(i)*cpd(i))/bf
cdelta(i)=(-cp(i)*rpd(i)+rp(i)*cpd(i))/bf
c calculate zstar and store  inrzstar  c
and czstar
          rzstar(i)=rlz(i)+rdelta(i)
          czstar(i)=cmz(i)+cdelta(i)
20 continue
      do 500 k=k1,k2,k3
if (con .and. iflag(k).eq.0) goto 500
          sr=0.0
          sc=0.0
      do 50 j=1,n
              if (j.eq.k) goto 50
              ar=rlz(k)-rzstar(j)
              ac=cmz(k)-czstar(j)
              bf=ar**2+ac**2
              if (bf.eq.0.0) goto 12
              sr=sr+ar/bf
              sc=sc-ac/bf
50  continue
          ar=rdelta(k)
          ac=cdelta(k)
          br=ar*sr-ac*sc+1.0
          bc=ar*sc+ac*sr
          bf=br**2+bc**2
          if (bf.eq.0.0) goto 12
          rdelta(k)=(ar*br+ac*bc)/bf
          cdelta(k)=(ac*br-ar*bc)/bf
          rzstar(k)=rlz(k)+rdelta(k)
          czstar(k)=cmz(k)+cdelta(k)
500 continue
      do i=1,n
          rlz(i)=rzstar(i)
          cmz(i)=czstar(i)
      enddo
c reverse the order of updating
      k4=k1
      k1=k2
```

```
      k2=k4
      k3=-k3
      if (iroot.eq.n) goto 13
      enddo

      icode=-1
13 if(.not.(ans.eq.'Y'.or.ans.eq.'y'))
then
          do i=1,n
              rlz(i)=rlz(i)*beta
              cmz(i)=cmz(i)*beta
          enddo
   endif
      if (zflag) then
      n = n + 1
      rlz(n) = 0.
      cmz(n) = 0.
      itusd(n)=0
      endif
      return


c abnormal exit - overflow
12iex=-3
      return
      end
c subroutine to evaluate poly and
deriv
  subroutine
evaluate(a,n,x,y,rp,cp,rdp,
+cdp,sat)
 double precision
a(n+1),x,y,rp,cp,rdp,cdp
+,b1,b2,b3,a1,a2,a3,p,q,c,t,tol
      logical sat
      integer n,i
      tol=2.0**(-53)
      sat=.false.
      p=-2.0*x
      q=x*x+y*y
      t=dsqrt(q)
      b2=0.0
      a2=0.0
      b1=1.0
      a1=1.0
      c=0.8
      do i=1,n-2
          a3=a2
          a2=a1
          a1=a(i+1)-p*a2-q*a3
          c=t*c+dabs(a1)
          b3=b2
          b2=b1
          b1=a1-p*b2-q*b3
      enddo
      a3=a2
      a2=a1
      a1=a(n)-p*a2-q*a3
```

```
      rp=a(n+1)+x*a1-q*a2
      cp=a1*y
      rdp=a1-2.0*b2*y*y
      cdp=2.0*y*(b1-x*b2)
      c=t*(t*c+dabs(a1))+dabs(rp)

sat=dsqrt(rp*rp+cp*cp).lt.(2.0*dabs(x*a1)-
+8.0*(dabs(rp)+dabs(a1)*t)+10.0*c)*tol
print*,dsqrt(rp*rp+cp*cp),(2.0*dabs(x*a1)-
8.0*(dabs(rp)+dabs(a1)*t)+10.0*c)*tol,sat
c  pause
      return
      end

c subroutine to find bound on largest zero
      subroutine bnd(b,n,beta)
      double precision b(n),beta,xm,xm1
      integer n,i

      xm=abs(b(1))
      do i=2,n
          xm1=abs(b(i))**(1.0/i)
          xm=dmax1(xm,xm1)
      enddo
      beta=2.0*xm
      return
      end

c subroutine to bring  zeros   within c
unit circle
      subroutine scale(b,n,beta)
      double precision b(n),beta,t,t1
      integer n,i
      t=1.0/beta
      t1=1.0
      do i=2,n
          t1=t1*t
          b(i)=b(i)*t1
      enddo
      return
      end

c subroutine   to   generate   initial c
estimates
      subroutine gstval(n,r,c)
      double precision r(n),c(n),x,a
      integer n,i
      a=4.0*atan(1.0)/n
      do i=1,n
          x=2*(i-1)*a+0.05
          r(i)=cos(x)
          c(i)=sin(x)
      enddo
      return
      end
    subroutine chpol(a,n,kind)
    implicit double precision(a-h,o-z)
    dimension a(n+1),b(51),c(51),g(3)
```

```
      a(2)=0
      b(2)=0
      a(3)=-1
      g(1)=2
      g(2)=0
      g(3)=-1
      if(kind.eq.1)then
      b(1)=1
      a(1)=2
      elseif(kind.eq.2)then
      b(1)=2
      a(1)=4
      endif
      do j=3,n
      c(1)=g(1)*a(1)
      c(2)=g(1)*a(2)+g(2)*a(1)
       k=4
100   s=0
      do i=1,2
        s=s+g(I)*a(K-I)
      enddo
       c(k-1)=s+g(3)*b(k-3)
         k=k+1
       if(k.lt.(j+2)) goto 100
        c(k-1)=g(2)*a(k-2)+g(3)*b(k-3)
      do i=1,n+1
          b(i)=a(i)
          a(i)=c(i)
      enddo
      enddo
      return
      end
```