Computer Physics Communications 29 (1983) 245–259
North-Holland Publishing Company

COMPLEX ZEROS OF ANALYTIC FUNCTIONS

L.C. BOTTEN, M.S. CRAIG

School of Mathematical Sciences, The New South Wales Institute of Technology, P.O. Box 123, Broadway, NSW 2007, Australia

and R.C. McPHEDRAN

Department of Theoretical Physics, The University of Sydney, NSW 2006, Australia

Received 26 May 1982

PROGRAM SUMMARY

Title of program: CRTEST

Catalogue number: AAOO

Program obtainable from: CPC Program Library, Queen's University of Belfast, N. Ireland (see application form in this issue)

Computer: CDC Cyber 170/730; *Installation*: University of Sydney. (Also Honeywell Level 66/60; *Installation*: NSW Institute of Technology, Sydney)

Operating system: Nos 1.4 (GCOS)

Programming language used: FORTRAN 77

High speed storage required: 14336 words (on a Cyber 170)

No. of bits in a word: 60 (36)

Peripherals used: card reader, line printer

No. of cards in combined program and test deck: 2063

Card punching code: EBCDIC

Keywords: general purpose, complex zero, analytic function, Cauchy's integral

Nature of physical problem
The subroutine CROOT in CRTEST provides a versatile means of locating the zeros (within an annular region of the complex plane) of an analytic function specified by a user-supplied subroutine. CROOT has already been successfully applied in electromagnetic diffraction theory [1], in plasma astrophysics [2] and in solar energy [3].

Method of solution
The method of Delves and Lyness [4] is used to evaluate recursively all zeros of the user-supplied analytic function $f(z)$ lying within a specified annulus in the complex plane. The basis of the method is to employ Cauchy's integral [5] to determine the number of zeros of $f(z)$ lying within a particular region. If this is sufficiently small, Cauchy's integral and Newton's formulae [6] are used to construct a polynomial $p(z)$ having the same zeros as $f(z)$ within the region. The zeros of the low-order polynomial $p(z)$ are evaluated by Müller's method [7], and are refined iteratively by the same method.

If $f(z)$ has a large number of zeros within the initial annulus, radial bisection is used. This continues until all zeros have been found, or until successive radii become too close. In the latter case, angular division of the annulus into sectors is commenced.

Restrictions on the complexity of the problem
The function $f(z)$ must be analytic within the specified annulus. The variable z should be chosen so that the zeros are not too tightly clustered around a single point within the annulus. Both the function $f(z)$ and its derivative $f'(z)$ are needed, and their computer evaluation should be reasonably rapid.

Typical running time
This is highly problem-dependent. Some typical CPU times are given in section 5 of the long write-up.

Accuracy
The iterative refinement seeks to obtain zeros to a relative accuracy of 10^{-8} or so that the modulus of $f(z)$ is smaller than 10^{-12}.

References
[1] L.C. Botten, M.S. Craig and R.C. McPhedran, Opt. Acta 28 (1981) 1103.

[2] R.M. Winglee, Plasma Phys. (1983) in press.
[3] L.C. Botten, R.C. McPhedran, D.R. McKenzie and R.P. Netterfield, Appl. Opt. (submitted).
[4] L.M. Delves and J.N. Lyness, Math. Comput. 21 (1967) 543.
[5] E.T. Whittaker and G.N. Watson, A Course of Modern Analysis (Cambridge Univ. Press, London, 1965) p. 119.
[6] G.A. Korn and T.M. Korn, Mathematical Handbook for Scientists and Engineers (McGraw-Hill, New York, 1968) p 13.
[7] S.D. Conte and C. de Boor, Elementary Numerical Analysis – An Algorithmic Approach (McGraw-Hill, New York, 1980) p. 120.

LONG WRITE-UP

1. Introduction

Many practical problems require the finding of solutions of equations of the form

$$f(z)=0, \tag{1}$$

where f is an analytic function of the complex variable z. Iterative methods such as those based on the Regula Falsi rule [1] are often employed to solve (1), but these may prove unreliable in the determination of all required roots (see, for example, ref. [2]). In this communication, we present a powerful technique which we have found to be very reliable in locating complex zeros of analytic functions.

Our method is based on a technique suggested by Delves and Lyness [3]. Let C denote a closed contour in the complex z plane containing n zeros z_j $(j=1, 2,\dots, n)$ of the analytic function $f(z)$. Then from Cauchy's integral [4]

$$\frac{1}{2\pi i}\int_C z^p \frac{f'(z)}{f(z)}\mathrm{d}z=\sum_{j=1}^{n} z_j^p \equiv s_p, \tag{2}$$

where p is any non-negative integer. In particular, if $p=0$ then the integral (2) furnishes an estimate of the number, n, of zeros within C. The integral (2) is also evaluated with $p=1, 2,\dots, n$ to give sums s_p. If the number of zeros is small (say $n \leqslant 4$), these are used with Newton's formulae [5] to produce a polynomial of degree n (with leading coefficient one) having the same roots within C as $f(z)$. Specifically, the polynomial is written as

$$p_n(z)=\sum_{k=0}^{n} a_k z^k \tag{3}$$

with $a_n=1$. The coefficients a_k for $k=n-1, n-2,\dots, 1, 0$ are evaluated using the recurrence relations

$$(n-k)a_k+s_1 a_{k+1}+s_2 a_{k+2}+\dots+s_{n-k}a_n=0. \tag{4}$$

The low-order polynomial equation $p_n(z)=0$ is solved by a standard technique (in our case, Müller's method [6]) to give numerical estimates $\{z_i'\}$ of the actual set of zeros $\{z_i\}$. The differences $\{z_i - z_i'\}$ are due predominantly to quadrature errors. The estimates $\{z_i'\}$ are then refined by a further application of Müller's method, this time to the original function $f(z)$.

The principal difference between our method and that of Delves and Lyness [3] lies in the technique by which the contour C is subdivided when the number n of zeros is large. The latter method divided square regions up into four smaller squares, and circular regions into nine overlapping smaller circles. We have chosen for our basic region C an annulus. This is divided up by radial bisection into small annuli, and (where necessary) by angular bisection into annular sectors. Our subdivision technique leads to a more elegant and efficient program than do the techniques advocated by Delves and Lyness. (In particular, we divide the region C up recursively into non-overlapping subregions.)

The subroutines implementing our zero-finding method in FORTRAN 77 comprise 1765 cards of the program deck and require approximately 15 kwords of storage. The remainder of the deck comprises a driver program and a subroutine (FNDER) to evaluate $f(z)$ and $f'(z)$ for three test cases. In his application of the technique, the user will need to supply his own main program and subroutine FNDER.

2. Structure of the subroutine suite

The suite of subroutines for complex zero finding contains fifteen elements, and a block diagram of it is shown in fig. 1. Before detailing the operation of each subroutine, we will describe briefly the overall structure of the suite.

The main subroutine is CROOT. This sets up the initial parameters controlling the search for zeros, and inspects the results of the first search. If this has been unsuccessful, then parameters may be adjusted and the search recommenced, or the search may be terminated.

CROOT communicates with all other subroutines via RECUR. This latter subroutine controls the choice of annular regions in which zeros are to be located, and operates in a recursive fashion using two stacks containing the radii of circles (SR) and sums of powers of roots lying within the circles (SSR). If there are too many zeros lying within a given annulus, then that annulus is radially bisected, and the two stacks are lengthened. If a small number of zeros lies in an annulus, then these are extracted by the subroutine EXROOT and the two stacks are shortened.

If in the operation of RECUR two radii of SR are too close together, then the subroutine RECURT is called. RECURT is the recursion driver for extraction of zeros within annular sectors. It uses two stacks, one (SA) containing angles specifying the boundaries between sectors and the other (SSA) containing sums of powers of roots for each sector.

The subroutines RECUR and RECURT generate sums of powers of roots lying within various circles using subroutines ESIGR and ESIGT, respectively. These last two use the subroutine TRAP, which performs trapezoidal integration either along an arc of a circle or along a ray. TRAP fills all the elements in the stack (SSR or SST) for a given region simultaneously using a technique in which the number of integration points is repeatedly

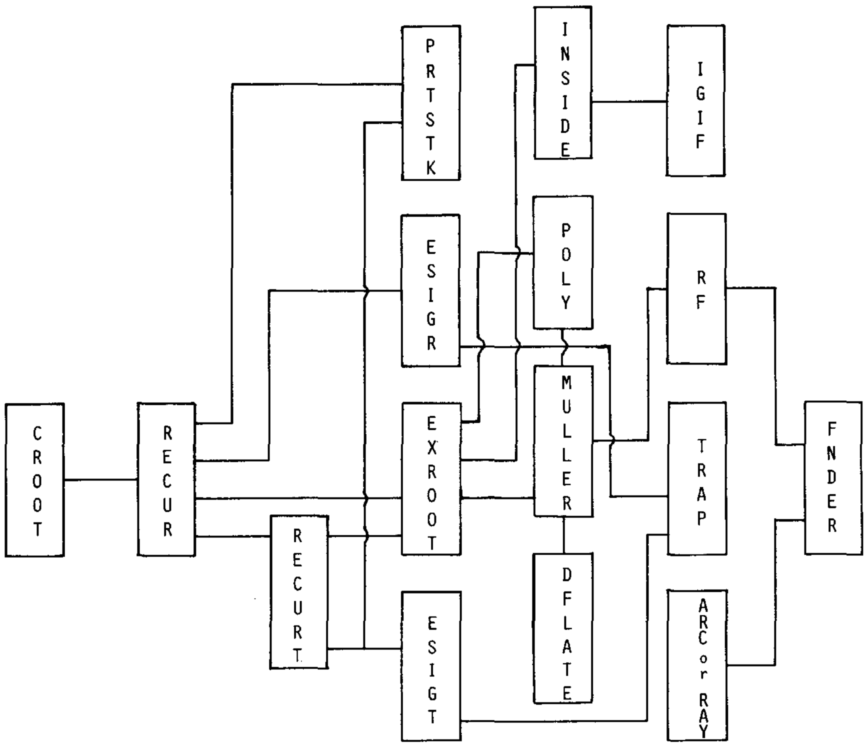

Fig. 1. Block diagram of the subroutines.

doubled until all integrals have converged to within an absolute or relative tolerance.

Both RECUR and RECURT use the subroutine EXROOT for finding all zeros lying within a given region. EXROOT employs Newton's formulae (4) to construct a polynomial of the form (3) having the same zeros within the given region as $f(z)$. These zeros are calculated using the subroutine MULLER, operating on the polynomial. A second use of MULLER operating on $f(z)$ serves to refine the zeros. EXROOT checks that all zeros actually lie inside the specified region using the subroutine INSIDE. If not, this is an indication that integration tolerances need to be reduced. RECUR and RECURT use the subroutine PRTSTK for printing summaries of their stacks, if these are required.

MULLER uses the subroutine POLY for polynomial evaluation, DFLATE for deflation of previously obtained zeros and RF for function evaluation alone, on the basis of FNDER. This last subroutine is supplied by the user, and is called by RF, ARC and RAY.

3. Functions and subroutines

3.1. Subroutine CROOT

This subroutine has formal parameters IR, IW, EVEN, RMIN, RMAX, CEN, NROOT, ZROOT and FAIL. CEN is complex and defines the centre of the annular search region, while RMIN and RMAX are the inner and outer radii of the annulus, respectively. (Note that RMIN can be zero.) After the completion of CROOT, NROOT zeros are stored in the complex vector ZROOT. FAIL is a logical variable, which if set true indicates an unsuccessful operation of CROOT. IR is the unit number from which data is read. If it is set negative, then the various method parameters discussed below take their default values. IW is the unit number to which a trace of the operation of CROOT is written. If this parameter is negative, the trace is suppressed but essential messages may be written to the unit $(-\mathrm{IW})$. It is strongly recommended that users consult the trace of CROOT whenever they are seeking zeros of a given function for the first time.

The logical variable EVEN is set true if the function specified in the subroutine FNDER has even symmetry about the point CEN – i.e., if

$$f(-z+\mathrm{CEN})=f(z-\mathrm{CEN}). \tag{5}$$

For a function having odd symmetry about CEN, the user may wish to divide his function by the appropriate power of $(z-\mathrm{CEN})$, so removing the zero at CEN, in order to obtain $f(z)$. Having done this, he would set EVEN to be true. In all but these two cases, EVEN should be set false.

CROOT assigns values to three variables using the PARAMETER statement. The use of these three variables (MBIS, MSIG and MTROUB) will be discussed below.

CROOT assigns values to eighteen method parameters, for communication to other subroutines using COMMON statements. The first of these is

COMMON /POWER/ IPOWER,

where the integer IPOWER is 2 for the case of an even function $f(z)$, and zero otherwise. The second is

COMMON /METHOD/ CRIT, MAXBIS.

Here CRIT is an upper bound on the modulus of the function to be integrated in TRAP. If the modulus exceeds CRIT, the integration is aborted and another contour is selected. This eliminates the possibility discussed by Delves and Lyness [3] of integrals being very slowly convergent because of a zero located very close to the path of integration. MAXBIS is the maximum number of bisections permitted in TRAP, so that no more than (2** MAXBIS) points may be placed along the path of integration. Next we consider

COMMON /ADJRAD/ ADJO, ADJI

and

COMMON /ADJANG/ ADJT.

These are relevant to ESIGR and ESIGT, respectively, and decide how the path of integration should be moved if a zero lies too close to it. ADJO is the multiplicative factor by which the initial outer radius is changed when a zero lies too close to the outer circumference of the annulus, while ADJI is the corresponding factor for the inner radius. In the case of integrals along rays, ADJT is subtracted from the angle specifying the ray if CRIT is exceeded.

For purposes of generality, the program finds zeros in an annulus of outer radius unity centred on the origin of the (program's) z-plane. The two parameters in

COMMON /RSCALE/ SCALE, CCEN

define a mapping from the z(PROG.) plane to the physical (z(PHYS.)) plane according to

$$z(\text{PHYS.}) = \text{CCEN} + \text{SCALE} * z(\text{PROG.}). \qquad (6)$$

In his interpretation of diagnostic stack listings and his programming of FNDER, the user should employ physical variables, and need not be concerned about the rescaling (6).

COMMON /TOLE/ EP1, EP2

defines tolerances on the independent (z) and dependent ($f(z)$) variables for the extraction of zeros by EXROOT using MULLER. Note that EP1 should not be smaller than the relative precision (EP) of the computer being used. Here EP is the smallest number (in modulus) such that [(1. + EP) − 1.] still has a representable multiplicative inverse.

COMMON /TOLI/ TOLINT, TINT

defines, respectively, the tolerance to within which integrals in TRAP must converge, and a tolerance on the departure from an integer allowed for the estimate of the number of zeros within a given region. TINT is used by ESIGR and ESIGT.

COMMON /OUT/ LPRINT, ICW

contains a logical variable LPRINT, and an integer ICW. If IW is zero or negative, LPRINT is set false; otherwise it is set true. In the case where LPRINT is false, the trace of the operation of CROOT is suppressed, but if that operation is unsuccessful a message will be written to unit ICW(= −IW). If LPRINT is true, the trace is written to unit ICW = IW.

COMMON /RADSEP/ RSEP

contains a tolerance on the smallest separation of annuli permitted in RECUR. This tolerance is on the difference of normalized rather than physical radii, and where a radial difference becomes smaller than RSEP the annular region is sectored using RECURT.

COMMON /MDIFF/ TCROOT, TESIGR, TSIGT

contains three integers. The first of these specifies the number of times CROOT will tighten integration tolerances and re-enter RECUR in response to an abnormal return from that subroutine, TESIGR controls the number of times ESIGR will move a circular integration contour because it passes too close to a zero. TESIGT performs the corresponding function for ESIGT in connection with ray integrals.

If IR is not positive, the operation of TCROOT commences with the setting of default values for the method parameters NSIG, RSEP, CRIT, MAXBIS, EP1, EP2, TOLINT, TINT, ADJO, ADJI, ADJT, TCROOT, TESIGR and TESIGT. Of these, we have discussed the function of all save NSIG. This is the number of sums of non-zero powers of roots that are generated in ESIGR and ESIGT. It has been found that NSIG = 4 is a reasonable estimate. If NSIG is less than four, the extraction of zeros is slow, while if NSIG exceeds four the process may be unreliable. NSIG is in fact the maximum number of zeros that may be exracted from any subregion of the annulus at any one time.

If IR is positive, the method parameters are read in, from that unit, using list-directed format. The subroutine ensures that fairly reasonable values are given to the parameters by requiring that:

$0 \leqslant \text{NSIG} \leqslant \text{MSIG}$, $0 \leqslant \text{RSEP} \leqslant 1.0$,
$0 \leqslant \text{MAXBIS} \leqslant \text{MBIS}$, $0 \leqslant \text{CRIT}$, $0 \leqslant \text{EP1}$,
$0 \leqslant \text{EP2}$, $0 \leqslant \text{TOLINT}$, $0 \leqslant \text{TINT}$, $0 \leqslant \text{ADJO}$,
$0 \leqslant \text{ADJI}$, $0 \leqslant \text{TCROOT} \leqslant \text{MTROUB}$,
$0 \leqslant \text{TESIGR} \leqslant \text{MTROUB}$,
$0 \leqslant \text{TESIGT} \leqslant \text{MTROUB}$.

CROOT next calculates normalized radii for the annulus (RRMIN and RRMAX), while setting SCALE of (6) equal to RMAX. It then enters RECUR. If there is a normal return from RECUR (i.e. ABRET and FAIL are both false) then CROOT terminates. If ABRET is true and FAIL false, TOLINT is divided by ten and RECUR is re-entered. This process of re-entry can occur at most TCROOT times. If FAIL is true, then CROOT writes a message on unit ICW and returns to the main program.

3.2. Subroutine RECUR

This subroutine has formal parameters NSIG, RMIN, RMAX, ABRET, FAIL, NR and ZR. Of these, we have already encountered NSIG. RMIN and RMAX are scaled radii of the inner and outer circles between which zeros are to be found. On the first entry to RECUR, RMAX is one, and RMIN is the user's RMIN divided by SCALE. However, as we shall see below, both these quantities may be altered by RECUR. ABRET is a logical variable signalling an abnormal return to CROOT. This generally signifies that the integration tolerance TOLINT needs to be reduced. On the other hand, FAIL signals if it is true, that some fatal error has occurred within the search for zeros, and that recovery is impossible. When RECUR has run to completion, there are NR zeros to be found in the complex vector ZR.

RECUR introduces two new common statements. The first of these is:

```
COMMON /EXR/ IETYPE, BASE.
```

These variables are used in the subroutine INSIDE. IETYPE signals to INSIDE if it is one that zeros are to be checked to see whether they lie inside an annulus, and if it is two to see whether they lie inside a sector. BASE is used to locate the arguments of complex numbers in an appropriate fashion: they are made to lie between BASE and 2π + BASE. (RECUR sets IETYPE and BASE to the values 1 and 0, respectively.)

```
COMMON /RENTRY/ PREX
```

is used to indicate to ESIGR that it is dealing with a circular contour whose radius cannot be altered. The logical variable PREX is only set true if RECUR has successfully extracted some zeros in the annulus RMIN $\leqslant |z| \leqslant$ R (i.e. NR > 0) and if RECUR is then re-entered with a reduced value of TOLINT after some non-fatal error has occurred in the search for zeros. The search for further zeros then has to be conducted precisely in the region R $\leqslant |z| \leqslant$ RMAX – i.e., in the second and subsequent operations of RECUR, it cannot permit ESIGR to alter the inner radius of the search annulus.

We have mentioned above that RECUR operates two stacks, SR (containing radii of circles) and SSR (containing sums of powers of zeros lying within the circles). These stacks are initialized by calls to ESIGR with the radius chosen to be RMAX, and then RMIN. ESIGR is permitted 2*TESIGR attempts at finding suitable radii for the inner and outer circles, before FAIL is set true and RECUR returns to CROOT. (Here, of course, we are referring to the situation when PREX is false.)

If the initialization of the stacks has been successful, RECUR commences its main recursion loop. In this, the number of roots in the region of interest is examined. If the number of zeros is not in excess of NSIG, then EXROOT is called for the annular region in question. If EXROOT is successful in finding the zeros in the region, it adds them to the vector ZR, and RECUR lowers the stack by one. If EXROOT is unsuccessful, this is presumed to be due to inadequately tight integration tolerances, so that RECUR returns to CROOT for TOLINT to be reduced.

If the number of zeros exceeds NSIG, then RECUR inserts a radius halfway between the last two radii in SR, provided that the difference between these radii exceeds RSEP. (If it does not, RECURT is called.) ESIGR is used to generate the sums of powers for the new radius, and these are placed in SSR. (Of course, the call of ESIGR may be unsuccessful, thereby precipitating a return to CROOT.) Once this has been done, we examine the number of zeros in the new region of interest, thereby recommencing the recursion loop.

The recursion loop continues until all zeros in the annulus have been found (which is indicated by the stack length ultimately being reduced to unity), or until there is an abnormal return from ESIGR or EXROOT, or until the number of entries in SR would be required to exceed its declared dimension by the radial bisection about to be performed. (The declared dimensions of SR and SSR are set by the quantities MSL, MSIG specified in a PARAMETER statement in RECUR.)

3.3. Subroutine PRTSTK

This has formal parameters L, S, SS, CALLER, TYPE, HEADER and PSCALE. Of these, CALLER, TYPE and HEADER are character strings, indicating, respectively, the subroutine which has called PRTSTK, whether the stack is of the radial or angular type, and whether the stack quantity is a radius or an angle. The integer L gives the number of elements of the arrays S and SS to be printed out, while PSCALE is used to rescale the quantities in S for printing out. PRTSTK is used by both RECUR and RECURT to print out their radial and angular stacks, respectively, provided that LPRINT has been set true. (The value of this logical quantity is communicated to PRTSTK via COMMON /OUT/.)

3.4. Subroutine ESIGR

This has formal parameters NSIG, ISL, SR, SSR, ADJPER, ABRET and FAIL. The subroutine generates sums of powers of zeros of the form (2), for $p = 0, 1, \ldots,$ NSIG, within the circle of radius SR(ISL). These sums are put onto the stack SSR at level ISL. If during the operation of ESIGR a problem occurs linked with insufficiently-small integration tolerances, ABRET is set true. More serius problems in ESIGR are indicated by FAIL being set true. ADJPER is a logical variable, set true if ESIGR is operating on the inner and outer circles of the annulus and false otherwise.

ESIGR shares its PARAMETER statement with RECUR. Its COMMON statements (/POWER/, /ADJRAD/, /RSCALE/, /TOLI/, /OUT/ and /MDIFF/) have already been discussed in connection with subroutine CROOT, apart from /RENTRY/ (see RECUR) and

COMMON /FCONST/ RAD, DUMMY.

This is used by the subroutine ARC, and communicates to it the radius RAD of the circular contour on which TRAP is carrying out its quadrature. (ARC has no need of the second element of /FCONST/, DUMMY. This common is also employed by ESIGT to communicate data to RAY.)

ESIGR uses the trapezoidal integration subroutine TRAP to evaluate the sums of powers of zeros (2). If TRAP indicates trouble arising from a zero lying too near to the circle of integration, then ESIGR changes its radius to be the average of the current radius and the next smallest radius in SR. This problem is permitted to occur TESIGR times, before FAIL is set true and ESIGR returns control to RECUR. (This strategy does not apply to the bounding circles of the annulus. Trouble in TRAP causes their radii to be multiplied by the appropriate factor of ADJO, ADJI.)

If there is a normal return from TRAP, ESIGR checks that the estimate for the number of zeros does not differ from an integer in its real part (or from zero in its imaginary part) by more than TINT. If it does, ABRET is set true.

3.5. Subroutine RECURT

This is the subroutine which is the recursion driver for the division of an annulus into sectors. Its formal parameters are NSIG, ISR, SR, SSR, ABRET, FAIL, NR and ZR. We have already encountered NSIG above. ISR is used to specify the annulus in question: its inner radius RA is SR(ISR), while its outer radius RB is SR(ISR-1). The logical formal parameters ABRET and FAIL have their usual interpretations, while NR is the number of zeros stored in the complex vector ZR (if RECURT runs to a successful termination).

Of the five common statements in RECURT, four have already been encountered (/POWER/, /RSCALE/ and /OUT/ in CROOT, and /EXR/ in ESIGR).

COMMON /ESTRAD/ RA, RB

is used to communicate the radii, RA, RB to ESIGT.

RECURT operates two stacks, SA and SSA. SA contains angles specifying the edge-rays of sectors, while SSA contains sums of powers of zeros for the sector running from the specified angle back to the initial angle (called BASE). BASE is fixed by an initial call of ESIGT, and the terminal angle SA(1) is set equal to 2π + BASE for general functions, and π + BASE for even functions. RECURT then commences the main recursion loop, starting off with the sector lying between SA(2) or BASE and SA(1).

By subtraction of the appropriate entries in SSA, the number of zeros in the appropriate sector is obtained. If this number is smaller than or equal to NSIG, EXROOT is called to extract the zeros, and the stack lengths are reduced by one. (If the call of EXROOT is unsuccessful, FAIL is set false, and RECURT returns control to RECUR.) If this number exceeds NSIG, then the sector is bisected by the addition of another angle to SA. ESIGT is then called to supply the sums of powers of zeros for the sector between the new ray and BASE to SSA. (ESIGT may well have to move the new ray away from a nearby zero, or it may fail in its operation, in which case RECURT returns control to RECUR.) RECURT then goes round the loop once more.

This process can end with an unsuccessful call of EXROOT or ESIGT, with the stacks SA and SSA overflowing due to a very large number of zeros in the annulus, or with stack lengths being reduced to unity (signalling a successful termination of RECURT).

3.6. Subroutine ESIGT

This subroutine has formal parameters NSIG, ISL, SA, SSA, INIT, ABRET and FAIL. It generates sums of powers of zeros within the sector bounded by the rays of angle SA(ISL) and SA(ISL-1), as well as by the arcs of radii RA, RB (these being communicated to ESIGT through the common /ESTRAD/). ABRET and FAIL have their usual meanings, while INIT distinguishes the initial entry into ESIGT from RECURT, during which a starting ray is sought, from subsequent entries.

The eight commons in ESIGT have already been discussed in connection with CROOT (/IPOWER/, /ADJ/, /RSCALE/, /TOLI/, /OUT/ and /MDIFF/), RECURT(/ESTRAD/) and ESIGR(/FCONST/). Only the last needs further comment here. ESIGT has to concern itself with integrals around sectors composed of two arcs of circles and two rays. The quadratures for arcs are performed just as in ESIGR, with the radius being fed into the first location in /FCONST/. For quadratures along rays, the two locations in /FCONST/ are given the values $\cos\theta$ and $\sin\theta$, θ being the angle specifying the ray.

If INIT is true, ESIGT seeks to find a suitable initial ray. If TRAP encounters trouble due to a zero lying close to the initial ray of the sector, it adjusts the ray angle by subtracting off ADJT, and tries again, for up to 4∗TESIGT times.

If INIT is false, then we are dealing with a normal integral around a sector contour, requiring four calls of TRAP. The estimate for the number of zeros lying in the sector is compared with the closest integer, just as in ESIGR.

3.7. Subroutine TRAP

This subroutine has formal parameters TYPE, F, A, B, NSIG, S and ABRET. Here TYPE, a character variable, is either 'RAY' or 'ARC', corresponding to the two necessary quadrature paths. F is a complex function, and can be either ARC or RAY (see below). TRAP calculates integrals of the form

$$\mathrm{S}(p)=\int_{\mathrm{A}}^{\mathrm{B}}(z(x))^{p}f(x)\,\mathrm{d}x \tag{7}$$

for $p = 0, 1, \ldots,$ NSIG, where in the case of integrals along circular arcs x is the angle, and in the case of ray integrals x is the radial distance. The logical variable ABRET signifies an aborted return (in general, as a consequence of poorly-convergent integrals).

The four common blocks in TRAP (/POWER/, /METHOD/, /TOLI/ and /OUT/) have already been encountered in subroutine CROOT. The two variables in the parameter statement (MBIS and MSIG) are used in declaring the array T, which holds the estimates for the S(p) arising from each cycle of the bisection process.

TRAP initially places two points on the interval [A, B], constituting one integration panel. It then places three points in two panels, five points in three panels, etc. It continues bisecting either until CRIT is exceeded (signifying a zero of $f(z)$ lying too close to the integration contour), or the number of bisections exceeds MAXBIS (signifying slowly-convergent integrals), or all the integrals $S(p)$ of (7) have converged. The criterion that the estimate $T(p, n)$ for $S(p)$ be converged is that (with $n \geqslant 4$) $T(p, n)$ and $T(p, n-1)$ differ relatively or absolutely by less than TOLINT.

3.8. *Complex function ARC*

This function evaluates for the argument t the quantities:

$$z = R \exp(it) \tag{8}$$

(where z is used by TRAP in calculating (7), and R comes from COMMON/FCONST/), and

$$ARC = zf'(z)/(2\pi f(z)). \tag{9}$$

Note that $f'(z)$ is the derivative of $f(z)$ with respect to the program's variable z, but that care has been taken to rescale the user's derivative (calculated with respect to his physical variable; see (6).

3.9. *Complex function RAY*

This performs a similar function to ARC. RAY has arguments R and Z, (8) is replaced by

$$z = R(EITR + i*EITI) \tag{10}$$

(where EITR, EITI come from COMMON/FCONST/), and (9) becomes

$$RAY = -izf'(z)/(2\pi Rf(z)). \tag{11}$$

3.10. *Subroutine EXROOT*

This subroutine performs the extraction of zeros within a region specified by (radial or angular) limits SR(ISL) and SR(ISL-1). It does this using the sums of powers of zeros contained in the complex stack SS (where the sums pertaining to the region in question are (SS(K, ISL − 1) − SS(K, ISL))), for K = 0, 1,..., N, N = (SS(0, ISL − 1) − SS(0, ISL)). The number of zeros actually found in the region is returned in the parameter NR, the zeros being contained in the complex vector ZR. The logical variable ABRET is set true if (due to computational errors) any of the elements to be inserted in ZR lie outside the search region.

The subroutine has four common blocks already encountered (/POWER/, /TOLE/, /OUT/ – see CROOT – and /RSCALE/ – see RECUR). It generates the coefficients of a polynomial having the same zeros within the search region as $f(z)$, and these are stored in a complex vector C. N and C form the elements of COMMON /EXPOLY/ N, C, which is used by the polynomial evaluation subroutine POLY. The elements of C are calculated using Newton's identities [5].

Having derived the equivalent polynomial, EXROOT finds its zeros using the subroutine MULLER. The estimates are then refined using MULLER, this time operating on the function $f(z)$ rather than the equivalent polynomial. All zeros are checked to see whether they lie within the search region. If any of the zeros lie outside it, ABRET is set true and EXROOT returns to its calling subroutine (either RECUR or RECURT). If all zeros lie inside the search region, the complex vector ZR is updated with the newly found roots.

3.11. *Subroutine INSIDE*

This subroutine tests to see whether a given complex zero Z lies within the search region specified by:

(a) in the case of the annulus (ITYPE = 1), the inner and outer radii are specified by the arguments A, B;
(b) in the case of a sector (ITYPE = 2), A and B specify the bounding angles, while the bounding radii, C, D come from COMMON /ESTRAD/ (see RECURT).

The modulus and argument of Z are returned to EXROOT using the parameters R and T. The logical parameter IN is true if Z lies inside the region, and false otherwise. INSIDE also uses the common EXR (see RECUR) to obtain ITYPE and BASE.

3.12. Integer function IGIF(X)

This calculates the greatest integer not exceeding the real X.

3.13. Subroutine MULLER

This determines up to N zeros of the function given by the external parameter F using Müller's parabolic method. The subroutine has been derived from one given by Conte and de Boor [6], with some minor modifications having been made. The complex array ZERO contains NPREV known zeros of the function in positions 1, 2,..., NPREV, and initial guesses for the locations of the zeros to be found in locations NPREV + 1,..., N. The real formal parameter HINIT is the initial step size used in the search for zeros. The integer parameter MAXIT is the maximum number of function evaluations allowed per zero. (If MAXIT is exceeded, MULLER returns control to EXROOT and a warning diagnostic is printed.) The iterative extraction of each zero is adjudged complete either when the ratio of the step size to the modulus of the estimated zero is smaller than EP1, or when the modulus of the corresponding function value is smaller than EP2. Upon successful completion, MULLER returns the values of the new zeros in the array ZERO, in locations (NPREV + 1),..., N.

3.14. Subroutine DFLATE

This subroutine is called by MULLER, and uses deflation on the values contained in ZERO(1), ZERO(2),..., ZERO(I − 1) to prevent these zeros interfering in the search for ZERO(I) (as may occur in the calculation of multiple zeros). See ref. [6] for a discussion of the technique of deflation.

3.15. Subroutine POLY

This subroutine calculates

$$\mathrm{FZ} = \sum_{j=0}^{n} \mathrm{C}(j)\mathrm{Z}^{j}, \tag{12}$$

the degree N and the coefficients C(j) of the polynomial being supplied to POLY through the COMMON/EXPOLY/ by EXROOT. Evaluation of (12) is performed using Horner's method.

3.16. Subroutine RF

This subroutine is used by EXROOT in the refinement of zeros by MULLER. Its arguments are the complex quantities Z and FZ, FZ being given the value $f(z)$ within RF by a call of the user-supplied subroutine FNDER.

4. User-supplied program elements

The user must write both a driving program and a subroutine FNDER. The latter must have three complex formal parameters, which we will label here Z, FZ and DZ. FNDER must give FZ and DZ the values of the function $f(z)$ and its derivative $f'(z)$ at the (physical) point $z = \mathrm{Z}$.

The driving program must assign values to the input parameters IR, IW, EVEN, RMIN, RMAX and CEN, and also to those quantities (if any) needed by FNDER to evaluate FZ and DZ, and fed to it through COMMON statements. After having called CROOT, we recommend that (if FAIL is true) the user should print out not only the NROOT zeros in ZROOT, but also the corresponding function and derivative values FZ and DZ. It may happen that |FZ| is quite large, even when the value of the zero (Z) has been accurately located (since, if ΔZ is the error in Z, $|\mathrm{FZ}| \approx |\mathrm{DZ} * \Delta \mathrm{Z}|$).

When running CROOT with a function he has not previously used, we would advice the user to obtain a trace of the operation of CROOT (i.e. have the output unit number IW > 0) and also to re-set the parameter TCROOT in CROOT to unity. If the user makes an error in FNDER, so that (FZ, DZ) do not correspond to an analytic function, then RECUR or RECURT will encounter non-integral numbers of zeros and CROOT will reduce TOLINT by a factor of ten and try (unsuccessfully) TCROOT times to overcom this problem.

The choice of the complex parameter CEN may be important to the efficient operation of CROOT. If the zeros are clustered about a point P in the complex plane, then it is best to choose CEN to be this point. The consequence of having CEN far removed from P is to force CROOT into unnecessarily-tight sectoring, and possibly to cause it to fail (for example, through an overflow of the stacks in RECURT).

The values chosen for the method parameters in CROOT should not be regarded as anything more than a first guide to the user. His own experience, based on a study of the trace of the zero search in a few typical cases, will guide him to make choices of them more suited to his problem. We give here a few general principles to guide him in his choice of the more important parameters.

The choice of RSEP is controlled by the distribution of zeros of the user's function. If these zeros are clustered in radius, then sectoral extraction will be efficient and RSEP should be large. If they are clustered in angle and not in radius, RSEP should be small, to delay the introduction of sectoral searches.

If CRIT is set too small, wasted computational time will result from frequent shifts of the integration contour. If it is too large, very slowly convergent integrals will be attempted, again resulting in inefficient computations.

The parameters MAXBIS and TOLINT are linked. If TOLINT is decreased, then MAXBIS may need to be increased, to permit more integration points to be placed on the contour in response to the tighter integration tolerance. If TOLINT is too large, then the zeros of the equivalent polynomial may differ too significantly from those of $f(z)$, and this may cause trouble in the refinement of the zeros. On the other hand, too small a value of TOLINT results in unnecessarily large computational times.

The parameter EP1 should not be smaller than the relative precision of the computer in use. If the user wants to be certain of not overlooking any zeros in the specified search annulus, he should set ADJO greater than unity and ADJI less than unity. Movement of the inner and outer circles by ESIGR can then only result in an enlarged search area. The trouble parameters (TCROOT, TESIGR and TESIGT) should not be set too large unless the user is sure he has chosen the best possible search area, and has approriate values for RSEP, CRIT, TOLINT and MAXBIS.

It is only a difficult cases (e.g. where $f(z)$ has a large number of zeros concentrated in a small region) that the choice of method parameters is critical. The user is advised to gain some experience in the operation of CROOT with simpler problems before tackling such difficult cases.

5. Test examples

The test program for the suite of zero-finding subroutines has three modes of operation. The first of these is to find the zeros of a polynomial with given coefficients. The example we have chosen to illustrate this mode of operation is a badly ill-conditioned polynomial of degree 16, having real coefficients, discussed by Delves and Lyness [3]. The coefficients of this polynomial and the method parameters of CROOT are shown in the section test run output, together with the eight zeros having positive imaginary parts. Their extraction took around 2 s execution time on a CDC Cyber 170 computer. Comparing these zeros with those in ref. [3], we see that they are all correct in their real and imaginary parts to five or more decimal places. Greater accuracy would be provided if the zeros were refined using a double-precision representation of the polynomial.

The second test mode is to re-obtain the zeros of a polynomial whose coefficients have been calculated from a prescribed set of zeros. To illustrate this mode, we used a set of thirty zeros with randomly-chosen real and imaginary parts, all having modulus smaller than twenty. We show the data, method parameters, part of the trace of the search for zeros, and the zeros obtained in the test run output. The execution time for this zero search was about 470 s. All zeros were obtained correct to nine decimal places in their real and imaginary parts. The function values corresponding to some of the zeros are large in modulus, but this is a consequence of the extremely large modulus of the corresponding function derivatives.

The third test mode is to find the zeros of a transcendental function given in ref. [2]. The test case chosen is that of fig. 1 of ref. [2]. This figure shows the positions of those zeros having modulus smaller than 80. The test results here correspond to a zero search in the annulus of inner radius 40 and outer radius 80, and NSIG = 4. The fifteen zeros were obtained using four radii lying between the inner and outer radii. We have also run this problem with NSIG = 6, and found the zeros with only two intermediate radii being used. The run time for this example was around 30 s.

Acknowledgements

One of the authors (R.M.) acknowledges financial support for this work provided by the New South Wales State Government and by His Royal Highness Prince Nawaf Abdul Aziz of the Kingdom of Saudi Arabia through the Science Foundation for Physics within the University of Sydney.

References

[1] E. Clayton and G.H. Derrick, Austr. J. Phys. 30 (1977) 15.
[2] L.C. Botten, M.S. Craig and R.C. McPhedran, Opt. Acta 28 (1981) 1103.
[3] L.M. Delves and J.N. Lyness, Math. Comput. 21 (1967) 543.
[4] E.T. Whittaker and G.N. Watson, A Course of Modern Analysis (Cambridge Univ. Press, London, 1965) p. 119.
[5] A. Ralston and P. Rabinowitz, A First Course in Numerical Analysis (McGraw-Hill, Kogakusha, 1978) p. 408.
]6] S.D. Conte and C. de Boor, Elementary Numerical Analysis – An Algorithmic Approach (McGraw-Hill, New York, 1980) p. 120.

TEST RUN OUTPUT

```
CRTEST - MODE=  1
COEFFICIENTS OF THE POLYNOMIAL OF DEGREE 16 ARE:


DEGREE           COEFFICIENT            DEGREE           COEFFICIENT

  16    1.25016256E+09  0.                 7    8.37860000E+05  0.
  15    3.85455882E+08  0.                 6    2.67232000E+05  0.
  14    8.45947696E+08  0.                 5    4.41840000E+04  0.
  13    2.40775148E+08  0.                 4    1.04160000E+04  0.
  12    2.47926664E+08  0.                 3    1.28800000E+03  0.
  11    6.42493560E+07  0.                 2    2.24000000E+02  0.
  10    4.10187520E+07  0.                 1    1.60000000E+01  0.
   9    9.49084000E+06  0.                 0    2.00000000E+00  0.
   8    4.17826000E+06  0.


CROOT  - ENTRY - METHOD PARAMETERS ARE:
RMIN  = 0.               RMAX  = 3.00000E-01     EVEN  = F
CENTRE= 0.             3.00000E-01
NSIG  = 5                RSEP  = 1.00000E-02
MAXBIS=12                CRIT  = 1.00000E+04
TOLINT= 1.00000E-03      TINT  = 1.00000E-02
EP1   = 1.00000E-10      EP2   = 1.00000E-14
ADJO  = 1.05000E+00      ADJI  = 9.50000E-01     ADJT  = 5.00000E-02
TCROOT= 3                TESIGP= 3               TESIGT= 3

CRTEST - NORMAL RETURN -   8 ROOTS LOCATED


 I               ROOT(I)                         FUNCTION(I)                          DERIVATIVE(I)

 1   -2.32094355E-03  2.92583742E-01   3.55271368E-13  2.85105273E-13    1.44143440E-04 -7.43563884E-05
 2   -1.46566748E-04  3.08611642E-01  -9.23705556E-13  5.91852956E-13    3.66946438E-07 -2.86553203E-08
 3   -4.91153803E-04  3.04182136E-01  -4.26325641E-14  1.48028811E-12   -3.88472631E-06  7.38977095E-07
 4   -3.93720108E-05  3.11707203E-01  -1.16529009E-12  1.08215520E-13    5.74478918E-08 -8.67312638E-10
 5   -1.66382692E-06  3.12208602E-01  -1.60582658E-12 -1.74473652E-12   -1.19216566E-07 -2.33249390E-09
 6   -3.03166795E-05  3.10652740E-01  -1.50635060E-12  1.11803189E-12   -9.46587306E-08 -1.19231604E-09
 7   -1.86949954E-02  2.53445682E-01   3.41060513E-13 -1.69642078E-13   -1.18649594E-02  3.88286394E-02
 8   -1.32447247E-01  1.36005508E-01   4.26325641E-14 -1.42108547E-14    9.38451282E+00 -1.98613424E+01
CRTEST - MODE=  2
THE  30 PRESCRIBED ROOTS ARE:


  I               ROOT(I)                  I              ROOT(I)

  1   -1.47404050E-01  3.45888941E-01  16    6.29830395E+00  4.19304037E+00
  2    1.37152828E+00 -4.08664206E-01  17   -6.43178534E+00  5.49435795E+00
  3   -1.35630319E+00 -7.02647116E-01  18   -8.87237820E+00  2.76331793E-01
  4    1.66700765E+00 -1.00914175E-01  19    9.49927145E-01  9.00247382E+00
  5    1.38562750E+00 -1.86375551E+00  20    9.06697553E+00  3.56923798E-01
  6    5.21033394E-01 -2.37709453E+00  21    8.67412046E-01  9.11581541E+00
  7    1.69825389E+00 -1.98594813E+00  22   -7.90530667E+00  6.01476285E+00
  8    1.30763032E+00 -2.31416391E+00  23    1.10459105E+01 -3.54973240E+00
  9   -2.73236842E+00 -2.80041436E-01  24    9.49165268E+00  6.77151032E+00
 10   -2.77941153E+00  2.40499898E+00  25   -2.12953540E+00  1.15856708E+01
 11   -9.85529792E-01  4.96159866E+00  26    2.98990401E+00  1.15282661E+01
 12   -1.88925378E+00  5.18101832E+00  27    9.64142008E+00 -7.28497975E+00
 13   -4.02127638E+00 -3.79749332E+00  28    4.17897386E+00 -1.17568738E+01
 14   -4.98042211E+00 -2.94530666E+00  29   -1.18634611E+01  4.12642513E+00
 15   -3.15037600E+00 -5.40254974E+00  30   -7.73241519E-01 -1.26485693E+01
CROOT  - ENTRY - METHOD PARAMETERS ARE:
RMIN  = 0.               RMAX  = 2.00000E+01     EVEN  = F
CENTRE= 0.             0.
NSIG  = 4                RSEP  = 1.00000E-01
MAXBIS=12                CRIT  = 1.00000E+03
TOLINT= 1.00000E-03      TINT  = 1.00000E-02
EP1   = 1.00000E-08      EP2   = 1.00000E-12
ADJO  = 1.05000E+00      ADJI  = 9.50000E-01     ADJT  = 5.00000E-02
TCROOT= 3                TESIGR= 3               TESIGT= 3
```

```
RECUR  - SUMMARY OF RADIAL STACK OF LENGTH  5
LEVEL    RADIUS           NUMBER CF RCOTS
  1    2.00000E+01   3.00000E+01 -5.05666E-07
  2    1.00000E+01   2.20000E+01 -4.58845E-09
  3    5.00000E+00   9.99999E+00 -1.82285E-06
  4    2.50000E+00   6.00000E+00 -5.07559E-07
  5    1.25000E+00   1.00000E+00  1.77541E-08
************************************************************************

RECURT - ENTRY - NSIG=  4 - SECTOR EXTRACTION
R( 5) =  1.25000E+00     R( 4) =  2.50000E+00
ESIGT  - ENTRY AT LEVEL   2
ESIGT  - CURRENT ANGLE= 0.
ESIGT  - CONTCUR INITIALIZATION
TRAP   - ENTRY FOR RAY INTEGRAL WITH NSIG= 4
TRAP   - INTEGRALS AT STEP  5.  SIGMA(I),I=0, 4 ARE:
 -9.05183E-01-6.66696E-01  -7.77541E-02-7.03752E-02  -6.85409E-03-7.44873E-03
                           -6.20786E-04-7.92948E-04  -5.78015E-05-8.50597E-05
TRAP   - SATISFACTORILY COMPLETED
ESIGT  - INITIALIZATION COMPLETE - SEARCH ANGLE= 0.
ESIGT  - RETURN TO  RECURT

************************************************************************
RECURT  - SUMMARY OF ANGULAP STACK OF LENGTH  2
LEVEL     ANGLE           NUMBER OF ROOTS
  1    6.28319E+00   5.00000E+00 -5.25314E-07
  2    0.            0.            0.
************************************************************************

RECURT - ANGULAR STACK TO BE RAISED FROM LEVEL  2 TO  3
ESIGT  - ENTRY AT LEVEL   2
ESIGT  - CURRENT ANGLE= 3.14159E+00
TRAP   - ENTRY FOR RAY INTEGRAL WITH NSIG= 4
TRAP   - INTEGRALS AT STEP  5.  SIGMA(I),I=0, 4 ARE:
 -9.05183E-01-6.66696E-01  -7.77541E-02-7.03752E-02  -6.85409E-03-7.44873E-03
                           -6.20786E-04-7.92948E-04  -5.78015E-05-8.50597E-05
TRAP   - SATISFACTORILY COMPLETED
TRAP   - ENTRY FOR RAY INTEGRAL WITH NSIG= 4
TRAP   - INTEGRALS AT STEP  5.  SIGMA(I),I=0, 4 ARE:
  2.66191E-01-2.18782E-01  -2.50560E-02 1.95540E-02   2.45764E-03-1.80454E-03
                           -2.50074E-04 1.71683E-04   2.62570E-05-1.68010E-05
TRAP   - SATISFACTORILY COMPLETED
TRAP   - ENTRY FCF ARC INTEGRAL WITH NSIG= 4
TRAP   - INTEGRALS AT STEP  7.  SIGMA(I),I=0, 4 ARE:
  2.45708E+00 1.94822E-01   5.50646E-02 1.11700E-01   9.95031E-03 4.88772E-03
                            3.99375E-04 9.65610E-04   8.63296E-05 6.70591E-05
TRAP   - SATISFACTORILY COMPLETED
TRAP   - ENTRY FOR ARC INTEGRAL WITH NSIG= 4
TRAP   - INTEGRALS AT STEP  7.  SIGMA(I),I=0, 4 ARE:
  1.28590E+00-2.53174E-01   2.27711E-03 2.17907E-02   6.46774E-04-7.54633E-04
                            2.72515E-05 1.80195E-06   2.46912E-06-1.12692E-06
TRAP   - SATISFACTORILY COMPLETED
```

```
CRTEST - NORMAL RETURN -  30 ROOTS LOCATED

 I               ROOT(I)                              FUNCTION(I)                          DERIVATIVE(I)

 1   -1.47404050E-01  3.45088941E-01   0.              0.                   2.67984788E+22 -1.16369750E+21
 2   -1.35630319E+00 -7.02647116E-01   0.              0.                  -1.10943944E+22 -4.70164182E+22
 3    1.37152828E+00 -4.08664206E-01   0.              0.                   1.92235811E+20  8.40181856E+21
 4    1.38562750E+00 -1.86375551E+00   0.              0.                  -3.51600428E+21 -9.90125534E+20
 5    1.66700765E+00 -1.00914175E-01   9.64645486E+06  1.51950178E+06       3.42161382E+21 -2.17218853E+22
 6    5.21033394E-01 -2.37709453E+00   0.              0.                   3.05902090E+22  1.16302520E+22
 7    1.69825389E+00 -1.98594813E+00   1.39661170E+08 -1.11510236E+07       9.82778114E+21 -7.84683524E+20
 8    1.30763032E+00 -2.31416391E+00  -1.52274153E+08 -3.28271554E+07       1.07153410E+22  2.31000570E+21
 9   -2.77941153E+00  2.40499898E+00  -3.97850567E+10  2.54735520E+10       5.03541276E+23 -2.29608320E+24
10   -2.73236842E+00 -2.80041436E-01   0.              0.                   4.26445916E+23  8.04278512E+22
11   -9.85529792E-01  4.96159868E+00   1.72358565E+11  2.21272860E+11       6.91902005E+24 -6.93061276E+24
12   -1.88925378E+00  5.18101832E+00  -2.60380686E+11  4.17767213E+11      -1.46988771E+25 -9.16133094E+24
13   -4.02127638E+00 -3.79749332E+00   0.              0.                   4.20566661E+24  1.07465517E+26
14   -4.98042211E+00 -2.94530866E+00  -3.14957612E+12 -3.22063070E+11      -2.26631738E+25  2.21631716E+26
15   -3.15037600E+00 -5.40254974E+00   0.              0.                  -9.33346713E+26  1.24675571E+27
16    6.29830395E+00  4.19304037E+00   4.39326472E+14 -5.73475030E+14       1.54574261E+28 -2.01773588E+28
17   -6.43178534E+00  5.49435795E+00   0.              0.                   8.66640406E+27  4.00821500E+27
18    9.49927145E-01  9.00247392E+00   0.              0.                  -5.96300296E+26  1.17470715E+27
19    9.06697553E+00  3.56923798E-01   0.              0.                   8.07297071E+29  3.54826372E+29
20    8.67412046E-01  9.11581541E+00   6.96526250E+13  5.34900557E+13       9.41007012E+26 -1.22534194E+27
21   -7.90530667E+00  6.01476285E+00   0.              0.                  -4.71689373E+28 -2.05866463E+29
22   -8.87237820E+00  2.76331793E-01  -4.98841785E+15 -1.29663461E+16       8.77571748E+28  2.28106373E+29
23    9.49165268E+00  6.77161032E+00  -3.58025069E+19 -1.39395310E+19      -6.40504443E+31 -2.13201601E+31
24    2.98990401E+00  1.15282661E+01  -2.05877181E+16  1.49624436E+17       1.44873187E+30 -1.05288837E+31
25   -2.12953540E+00  1.15866708E+01   6.65667242E+18  9.98357857E+18      -5.07415582E+29  7.16126844E+30
26   -1.18634611E+01  4.12642513E+00   3.57860017E+18  3.85505466E+18      -1.35637678E+32  1.25910800E+32
27   -7.73241519E-01 -1.26485693E+01   1.57749558E+20  1.47614582E+19      -2.59686319E+32  2.77515958E+33
28    4.17897386E+00 -1.17568738E+01  -8.13048564E+19  8.44515981E+18       1.48568823E+32  1.43033016E+33
29    1.10459105E+01 -3.54973240E+00  -6.90631269E+18  7.00109205E+17      -1.21497138E+32  1.23164514E+31
30    9.64142008E+00 -7.28497975E+00   0.              0.                   1.40399803E+32 -4.01798046E+32


CRTEST - MODE=   3
DATA FOR LOSSY LAMELLAR GRATING EXAMPLE


D=    1.00000000E+00             C=    4.00000000E-01
WL=   5.50000000E-01             PHI=  5.00000000E+00
R1=   1.00000000E+00   0.
R2=   7.50000000E-01   2.46200000E+00


CROOT  - ENTRY - METHOD PARAMETERS ARE:
RMIN  = 4.00000E+01     RMAX  = 8.00000E+01      EVEN  = T
CENTRE= 0.           0.
NSIG  = 4               RSEP  = 1.00000E-01
MAXBIS=12               CRIT  = 1.00000E+03
TOLINT= 1.00000E-03     TINT  = 1.00000E-02
EP1   = 1.00000E-08     EP2   = 1.00000E-12
ADJO  = 1.05000E+00     ADJI  = 9.50000E-01      ADJT  = 5.00000E-02
TCROOT= 3               TESIGP= 3                TESIGT= 3

CRTEST - NORMAL RETURN -  15 ROOTS LOCATED


 I               ROOT(I)                              FUNCTION(I)                          DERIVATIVE(I)

 1    4.88147465E+01 -3.12210261E+00  -1.42108547E-14  4.26325641E-14       9.22926325E-01 -1.48579320E-01
 2    4.53174367E+01 -3.64136996E+00   1.70530257E-12  2.55795385E-13      -1.18503241E+00  1.34631584E-01
 3    4.40794696E+01 -3.51478322E+00  -3.69482223E-13 -1.08002496E-12      -1.29675808E+00 -4.50271504E-01
 4    4.32321932E+01 -3.84883698E+00   4.12114787E-13  1.73372428E-12       1.15673238E+00 -2.67625188E-01
 5    5.61453591E+01 -2.72473138E+00   5.68434189E-14  1.35003120E-13      -9.04695734E-01  3.68667301E-02
 6    5.04989545E+01 -3.18329406E+00  -5.32907052E-14  2.84217094E-14      -9.22391388E-01  1.58527413E-01
 7    5.44085996E+01 -2.87746906E+00   4.97379915E-13 -5.04485342E-13       9.19828833E-01 -3.78883807E-02
 8    6.00693440E+01 -2.57170201E+00   7.10542736E-14  7.10542736E-15       9.27224116E-01 -4.39231904E-02
 9    6.19420261E+01 -2.46420966E+00  -4.26325641E-14 -1.42108547E-14      -9.20261816E-01  3.60148601E-02
10    7.18304194E+01 -2.09368688E+00  -1.31450406E-13  0.                   8.83462722E-01 -3.26901256E-02
11    6.59024811E+01 -2.29561931E+00  -2.13162821E-14  3.55271368E-15       9.01835761E-01 -4.59081136E-02
12    6.77867412E+01 -2.24972001E+00  -3.05533376E-13 -3.55271368E-15      -9.00181066E-01  4.27018313E-02
13    7.97021310E+01 -1.87754616E+00  -4.97379915E-13 -2.30926389E-14      -8.75124858E-01  2.14673912E-02
14    7.37086776E+01 -2.04776864E+00   1.77635684E-13 -7.10542736E-15      -8.81146534E-01  3.20178121E-02
15    7.78037302E+01 -1.93157233E+00   1.27897692E-13 -1.06581410E-14       8.77839668E-01 -2.25207835E-02
```