CLM - R 48

CLM - R 48

United Kingdom Atomic Energy Authority

RESEARCH GROUP

Report

A COMPUTER PROGRAM FOR FINDING COMPLEX ZEROS OF AN ARBITRARY FUNCTION

B. McNAMARA

Culham Laboratory,

Culham, Abingdon, Berkshire

1966

Available from H. M. Stationery Office

FIVE SHILLINGS NET

U.D.C.
681.3.06 : 517.531
517.531

UNCLASSIFIED
(Approved for Sale)

CLM - R 48

A COMPUTER PROGRAM FOR FINDING COMPLEX ZEROS OF AN ARBITRARY FUNCTION

by

B. McNAMARA

A B S T R A C T

The first part of this report describes a computer programme which finds complex zeros of arbitrary functions by iteration from some given initial guesses. The basis of the iteration is to approximate the given function by simpler functions whose zeros can be simply calculated. The regions of the complex plane in which the iteration converges to a required zero are greatly extended by several ad hoc procedures developed in a study of helicon wave propagation. The programme is constructed in such a way as to allow indefinite extension of the facilities offered without reference to the original programming. The helicon wave problem is used as an illustrative example and simple methods by which more complicated eigenvalue problems can be solved are indicated.

The second part of the report is an operating manual for the programme.

U.K.A.E.A. Research Group,
Culham Laboratory,
Nr. Abingdon,
Berks.

September, 1965. (C/13 MEA)

C O N T E N T S

PART I

A COMPUTER PROGRAM FOR FINDING COMPLEX ZEROS OF ARBITRARY FUNCTIONS

INTRODUCTION

The programme described here is a by product of a study of the propagation of helicon waves in a plasma cylinder by Klozenburg, McNamara, and Thonemann[1]. During this work it was realised that the computer programs developed could provide a simple means of solving any complex eigenvalue problem and with some modifications to the helicon wave programs this has been achieved. The primary object of the programme is to reduce the solution of such problems to a specification of the parameters and functions determining the problems, the programme providing the solutions with a minimum knowledge of computer programming on the part of the user. At the same time, the programme is constructed in such a way as to permit indefinite extension of the facilities provided.

The method used is to improve guesses at the solutions by iteration. The most important feature of the iteration process is that it is limited to rectangular regions of the complex plane so that unwanted solutions are not found. This necessarily introduces some ad hoc techniques for producing convergence and the complete process as described in Section I has rarely been found to fail.

The programme structure and facilities are described in Section II. A program using some of the methods described here is discussed by W.L. Frank[2]. In Section III some applications of the program are discussed. A simplified example from the helicon wave study demonstrates the value and necessity for many of the programme facilities. It is then shown how eigenvalue problems defined by second order differential equations or by second order differential integral equations may be tackled by simple techniques. These only require that the user be able to specify the coefficients, kernels, and boundary conditions for the particular problem.

Part II of this report is an Operating Manual for the Program.

I. THE ITERATION PROCESS

The basic function of the computer program described in this paper is to find, by iteration, complex zeros of arbitrary functions of complex variable, Z and an arbitrary number of parameters, P_i, i.e. to solve equations of the form

$$F_K (Z, P_1, P_2, \ldots, P_N) = U_K + iV_K = 0 \qquad \ldots (1)$$

K is an integer variable which may be used for the selection of different functions.

Iteration is a process of systematic guesswork and its effectiveness is frequently no better than the initial guess. To supplement analytic procedures for making initial guesses at the solutions of equation (1) a graphical program, such as that of F.M. Larkin[3], can be devised to give excellent information about the structure of the F_K in a specified region of the complex Z-plane for particular parameter values.

Having decided on an initial guess at the location of a required root of equation (1), one can proceed to improve it by approximating the function in the region of the guess by a simpler function whose roots can be more easily calculated. The initial guess is replaced by an 'appropriate' root of this simple function and the process repeated until equation (1) is sufficiently well satisfied. An essential feature of the iteration process is that unwanted roots should not be found. The simplest way to do this is not to allow the iteration to proceed outside a given rectangular region in the Z-plane around the initial guess. This style of limitation, apart from being convenient, is appropriate to many physical problems where the real and imaginary parts of the roots of F_K are dominated by different physical effects and can be approximated separately.

To locate a zero of F_K in the given rectangle the value of F_K is calculated at the initial guess and at two other points on the sides of the given rectangle. This assumes, of course, that the user has chosen the rectangle to define a region of the plane enclosing and close to the required zero. Using these three values it is possible to evaluate any approximate function requiring only three parameters for its definition. A choice of two functions is offered by the program, a quadratic

$$F_K \simeq aZ^2 + bZ + C$$

and a bilinear function

$$F_K \simeq \frac{a + bZ}{cZ + 1}$$

the latter being appropriate when the required root is near a pole of F_K. The roots of these approximations are at

$$Z = -\frac{b}{2a} \pm \frac{\sqrt{(b^2 - 4ac)}}{2a}$$

and

$$Z = -\frac{a}{b}$$

respectively. If the modulus of F_K at one of these roots is less than the modulus of F_K at any of the three points first used then the point with the largest modulus is discorded and a new approximation to F_K calculated. Iteration is stopped when a point is found such that the real and imaginary parts of F_K are less than given values or when more than 100 iteration have been performed. The convergence of the process using the quadratic approximation has been investigated by D. Muller[4]. He has shown that in the region of convergence, where the distance, ε_j, between the j^{th} iterate and the required root is sufficiently small, the distance, ε_{j+1}, of the next iterate from the root, r, is given by

$$\varepsilon_{j+1} = \left(-\frac{6F'(r)}{F''(r)}\right)^{\frac{1}{2}} \varepsilon_j^{1.84}$$

There is no a priori method of determining the convergence region for any particular zero of an F_K and so several ad hoc operations have been incorporated in the complete process to ensure, as far as possible, that if a zero exists in the given rectangular region it will be found. No matter how good the initial guess, it is always possible to find that the root(s) of the approximation function lies outside the given region of the complex plane, or that the modulus of F_K at the root is larger than that at any of the three current points. In the first case iteration is allowed to proceed provided the approximate root is inside double the given rectangle to avoid the possibility that the required root is only marginally outside the rectangle. In the second case the new iterate is brought closer to the current iterate in an attempt to find a point where $|F_K|$ is smaller so that the process does not jump out of the local valley in $|F_K|$. If this fails then the auxiliary point of smaller modulus is rotated about the current iterate by 90^o, keeping it within the given rectangle. This deals with the possibility that the process has brought the three points into line, thereby giving a poor estimate of the shape of F_K perpendicular to this line.

The quadratic approximation has two roots and, when all three points in current use are sufficiently close to the zero, the nearer root provides the next iterate. However, if the nearer root provides a value of F_K which is larger than that at the current iterate

the second root is examined and frequently turns out to be the one to take.

If these measures fail then the auxiliary points are brought closer to the current iterate in case they are sampling F_K in a region in which it is varying too rapidly for these points to provide an accurate approximation to the function. As a last resort the iteration process is restarted from a new initial guess and auxiliaries in the given region. The entire process as described is nearly guaranteed to find a zero of F_K in the given region, if one exists.

The process is composed of two logically distinct sets of operations. The first is, given a set of values of F_K at several points find an approximation to the location of a zero of F_K. The second is to provide an estimate of the worth of this approximation and in fact comprises the bulk of the complete process. These two operations are represented by separate subroutines and it is a simple matter for the user to replace the routines for calculating the three parameter approximation functions and their roots by more problem oriented routines. The complete process, representing the product of these two operations is represented by a routine called CRITIQ

II. THE GENERAL PURPOSE PROGRAM

The basic operation, CRITIQ, has been incorporated in a program providing a wide range of input/output facilities and a selection of combinations of this basic operation. The user is required to provide an operand, a subroutine defining the functions whose roots are required. In the FORTRAN dialects used at Culham this is done by providing a routine called FUNXNJ as follows.

```
SUBROUTINE FUNXNJ(X,Y,U,V,K)
COMMON A,B ..., R, P1, P2, ..., PN, S,T, ..., Z
Fortran statements defining
F_K (X + iY; P1, P2, ..., PN) = U_K + iV_K
ending with ...
U = U_K
V = V_K
RETURN
END
```

The parameters P1, P2, ..., PN may appear in a block anywhere in COMMON, the advantages of using COMMON in this way being that the user can employ mnemonic names for the parameters, whilst the program need only know where this block lies.

The program provides four combinations of root finding operations. The first and simplest is intended as a test of FUNXNJ and finds the root of F_K nearest to the origin whilst printing out complete details of the progress of operations. The second combination is to find a root of F_K in a specified region of the complex plane. The third combination provided is to locate a set of N zeros of F_K in N given regions of the complex plane for N different values of a parameter and is effected through routine ZDSETQ.

The fourth and most powerful combination is to execute this process automatically when the only data required is the range of variation of the parameter and an initial guess at the location of the root at either end of the range. In this combination the root of F_K is first located at the given end points of the range. The first root located is then used as an initial guess at the root of F_K when the parameter has been incremented by an amount depending on the number of points required in the range. The actual location of the root is calculated by CRITIQ, the parameter further incremented, and the process repeated till the parameter range has been covered. This combination of operations is effected through routine CONTZQ.

There are, unfortunately, many ways in which these simple processes have been found to fail, some of which are illustrated in Fig.1. The initial guesses Z1, Z2 are shown in their given regions of iteration together with the actual positions of the root at parameter values in the range. Starting from Z1, the first mishap illustrated is where another feature of the F_K has moved along the line AB to interfere with the root finding procedures. If this other feature is a zero then close to C this other zero might be found and the computation deflected along the curve CB. If the feature is a pole then the program may not be able to find the required root when it is very close to the pole. Use of the bilinear approximation may overcome this difficulty and the program may then find difficulty following the root around the sharply curved portion DE of the trajectory. The program attempts to overcome this difficulty by making a linear-extrapolation from the known results. If this fails, the program will, if possible, continue from the other end of the parameter range at Z2. The third difficulty illustrated is that the root trajectory may not be continuous due to the presence of the branch cut FG. As the program uses the last found location of the root as a guess at the new location, the branch cut cannot be crossed by the process and is pushed in front of the trajectory.

The impossibility of dealing with these and other possible troubles made it necessary to provide a means by which the user could control the program completely without too much effort. The program therefore calls a dummy control routine, GPDUMQ, at every

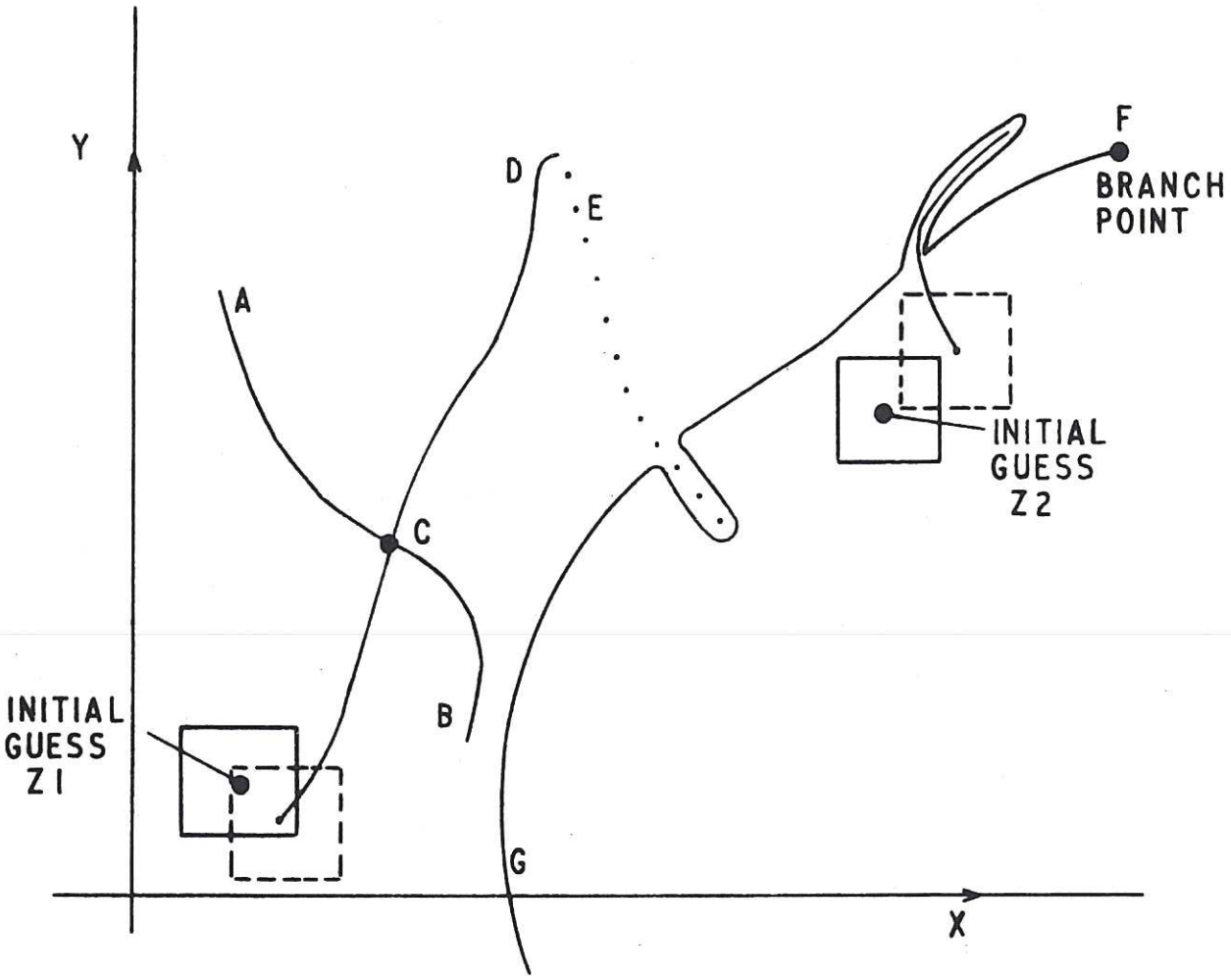

Fig. 1 Possible reasons for failure of CONTZQ (CLM-R48)

important point in the operations with an indication of the current state of affairs. Thus, GPDUMQ is called immediately before the beginning of each case or combination of operations, immediately before and after an attempt is made to locate a root, and immediately before any output operations are started. If the user wishes, all of these operations can be modified by provision of an appropriate GPDUMQ. The logical structure of the program is given by the flow diagram, Fig.2. The arrows labelled G1,2,3,4 indicate the various points at which GPDUMQ is called by the program.

Finally, to demonstrate the growth potential of the program, a special control subroutine has been written to enable the behaviour of a particular eigenvalue, λ, to be calculated automatically as a function of two parameters P_i, P_j. This routine uses CONTZQ to find the root λ, as a function of P_i at equally spaced values of P_j as illustrated by Fig.3. The only additional data required is the range of the second parameter and guesses at the location of the root, λ, at the corners of the rectangle in the P_i-P_j plane.

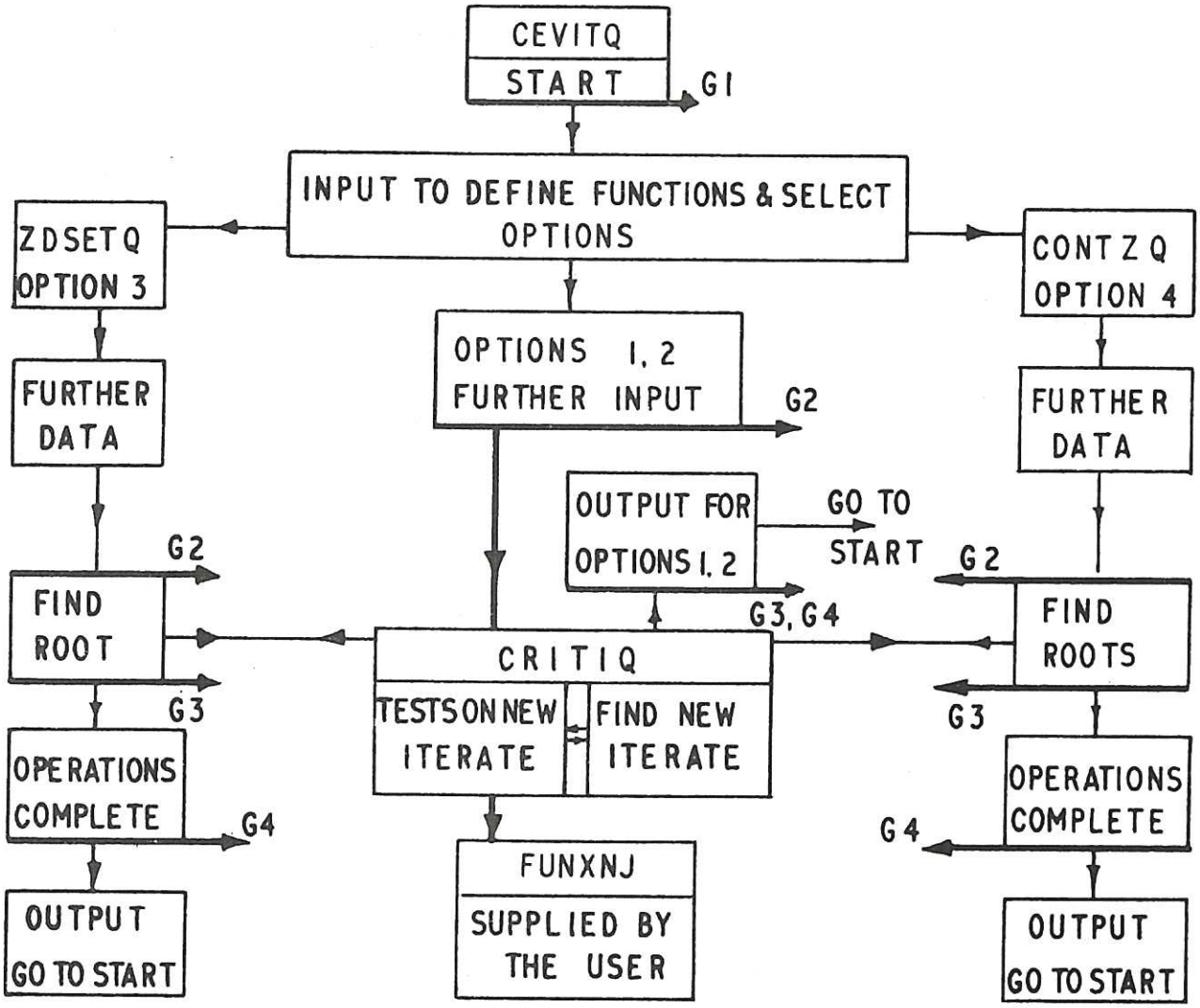

Fig. 2 (CLM-R 48)
Flow diagram for CEVITQ. G-arrows show points where GPDUMQ is called and state of program

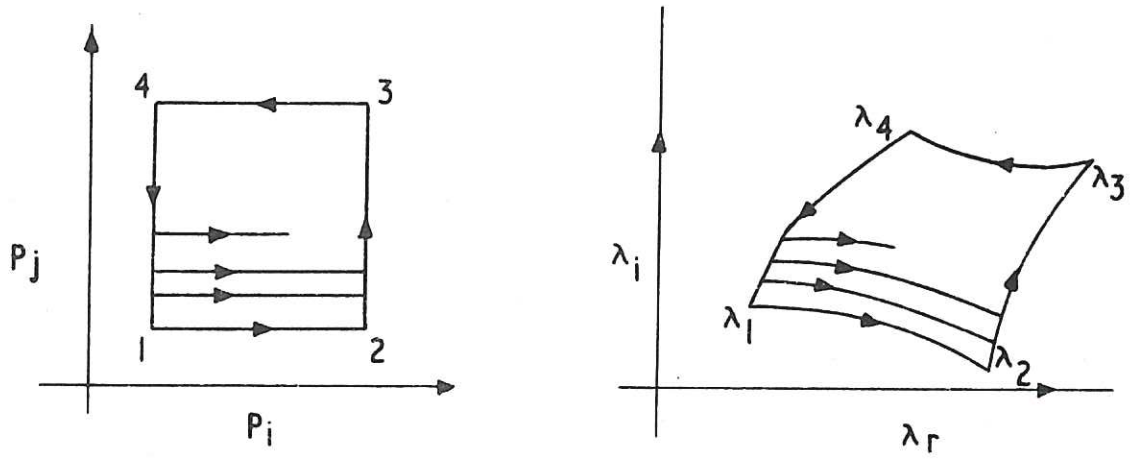

Fig. 3 (CLM-R 48)
Mapping of the p_i - p_j plane into the complex λ-plane

The probability that such complicated uses of the program will be devised shows the need for one other feature of the program. All the program facilities can be operated without the need for reading data cards and all normal forms of output provided in the program can be suppressed entirely, with the exception of failure messages, in favour of the user's requirements.

As is evident from the flow diagram it is possible to make substantial alterations to the program for particular purposes whilst retaining most of the facilities. Higher order iteration schemes, such as that of E. Frank[5], could be used by replacing the part of CRITIQ which determines the position of the new iterate. Still more complicated iterations can be devised for polynomials as discussed by W.D. Murro[6].

III. EXAMPLES

It is useful to consider a specific example of the application of this program before discussing wider applications. The problem, then, is to discover how a helicon electromagnetic wave will propagate down a uniform, resistive, magnetised plasma cylinder. The wave fields will be of the form:

$$\underline{B} = \underline{B}(r)\ e^{i(m\theta + kZ + \omega t)}$$

in cylindrical polar coordinates, (r,θ,Z), where m is an integer, k is the complex wave number, and ω is the frequency of the wave. For $m = 0$ the relation between k and ω is given by the solution of the equation[1]:

$$D\left(ak, \frac{\omega}{\omega_0}, \frac{\nu}{\Omega_c}\right) = \frac{\Upsilon_1}{q_1}\frac{J_0(\Upsilon_1)}{J_1(\Upsilon_1)} - \frac{\Upsilon_2}{q_2}\frac{J_0(\Upsilon_2)}{J_1(\Upsilon_2)} + \left(\frac{ak}{q_1} - \frac{ak}{q_2}\right)\frac{K_0(ak)}{K_1(ak)} = 0 \qquad \ldots (2)$$

$$\Upsilon_i^2 = q_i^2 - a^2k^2$$

where q_1, q_2 are the two solutions of the quadratic equation

$$\frac{i\nu}{\Omega_c}q^2 + akq + \frac{\omega}{\omega_0} = 0 \qquad \ldots (3)$$

and the J's and K's are Bessel functions, a = radius of cylinder, ν = electron collision frequency. Ω_c = electron cyclotron frequency, ω_0 = a frequency characteristic of a magnetised plasma cylinder. Corresponding to every solution of equation (2) is an electromagnetic wave whose structure may be indicated by the amplitude and phase of the Z component of the magnetic field as a function of radius:

$$B_Z - \frac{J_0(\Upsilon_1 r/a)}{J_1(\Upsilon_1)} - \frac{J_0(\Upsilon_2 r/a)}{J_1(\Upsilon_2)}$$

Some elementary analysis of the behaviour of the roots of equation (2) shows that for small ν/Ω_c

$$ak_{real}\ (\nu/\Omega_c = ak_{real}(0) + 0\ ((\nu/\Omega_c)^2)$$

$$ak_{imag}(\nu/\Omega_c) = ak_{imag}(0) + 0\ (\nu/\Omega_c)$$

results familiar from knowledge of elementary L-C-R circuits except for the boundary layer term $ak_{imag}(0)$. The structure of D in the complex P = ak plane for $\omega/\omega_o = 5$ and $\nu/\Omega_c = 0.05$ is shown in Fig.4. This diagram illustrates very well the need for many of the features of the computer program, especially the need for limiting the iteration process to a finite portion of the complex plane.

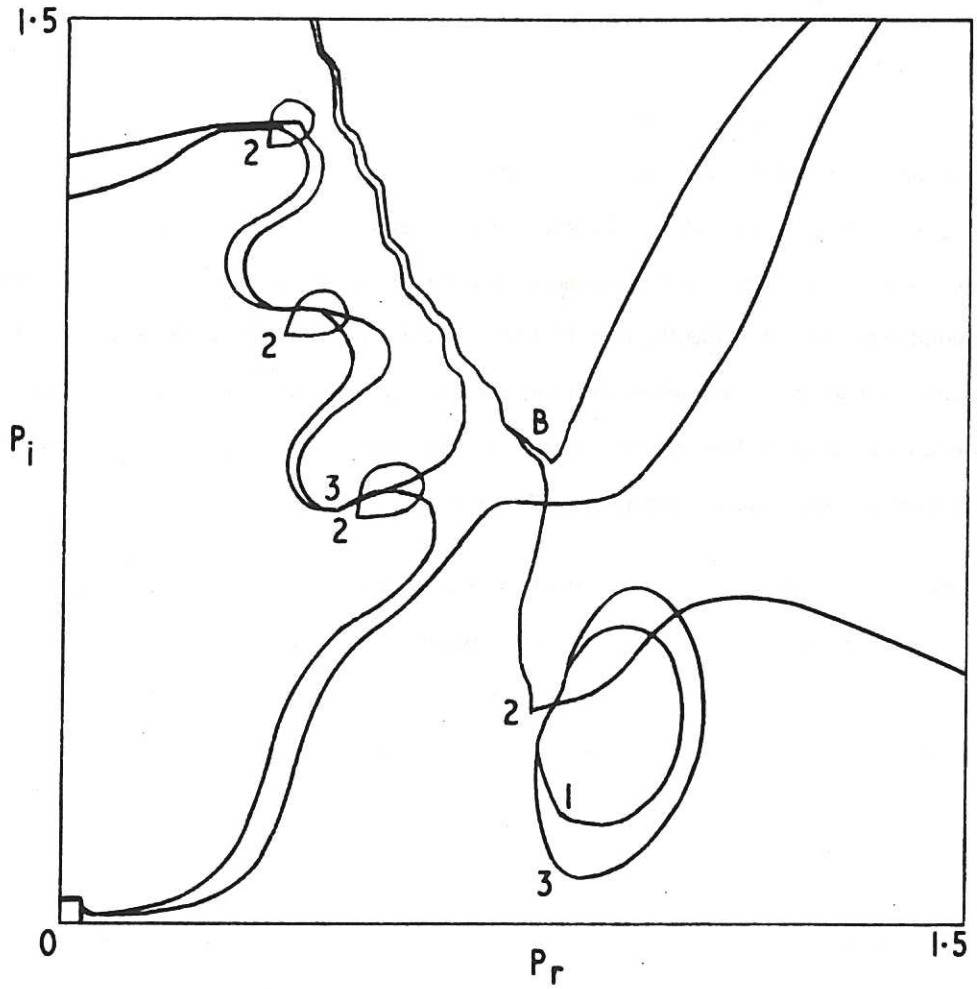

Fig. 4 (CLM-R48)
Contour map of D in the p-plane for $\frac{\omega}{\omega_o} = 5$, $\frac{\omega}{\omega_c} = .05$. Contour heights are:
1. DREAL = 0.2 2. DIMAG = 0.0 3. DREAL = 0.2
Intersection of 1, 2 and 3 are poles; intersection of 1 and 2 above are zeros;
B is a branch point of D

Using program CEVITQ it is now a simple matter to elucidate all the properties of the several roots shown in Fig.4 and the corresponding helicon waves. If, for example, it is required to know how the lowest root moves in the p-plane as ν/Ω_c varies and the corresponding B_Z magnetic wave-field it is only necessary to provide a routine FUNXNJ as follows:

```
SUBROUTINE FUNXNJ (AKR, AKI, DREAL, DIMAG, M)
COMMON WWO, PNU
Fortran statements defining
D (AKR + iAKI, WWO, PNU) = DREAL + iDIMAG
RETURN
END
```

To calculate the wave field at each root it is only necessary to provide a routine, GPDUMQ, which intercepts the program after each root is found, i.e. at points G3 of Fig.3, and performs the appropriate calculations. To complete the job CEVITQ is now given a guess at the lowest root, from Fig.4, at WWO = 5, PNU = 0.05 and another guess, from the analytic results, at the location of the root at some other value of PNU. The program is then asked to vary the second element of COMMON, namely PNU, and will proceed to do so. As it is only necessary for one of the given guesses to lead to the successful location of a root for the program to proceed it is possible to work away in any direction in parameter space from the relatively well known solutions of equation (3) shown in Fig.4.

Now, as might be supposed, the above example would not have yielded such a simple dispersion relation as equation (2) if the plasma properties had not been assumed uniform across the cylinder and the basic differential equations would not have been soluble in terms of elementary functions. This reveals the real power and value of a program such as CEVITQ, it is possible to tackle and frequently solve non-self-adjoint problems and to use experimentally measured variations of physical parameters.

As a simple approach to problems of this type consider the eigenvalue problems defined by the K second order differential equations

$$P_K\,(x,P_i,\omega)\,\frac{d^2\varphi}{dx^2} + Q_K\,(x,P_i,\omega)\,\frac{d\varphi}{dx} + R_K\,(x,P_i,\omega) = 0 \qquad \ldots\ (4)$$

and the corresponding boundary conditions

$$A_K\,(x_n,P_i,\omega)\,\varphi(x_n) + B_K\,(x_n,P_i,\omega)\,\frac{d\varphi}{dx}\Big|_{x\,=\,x_n} = 0 \quad n = 0,N \qquad \ldots\ (5)$$

to be applied at the two points x_o, x_N.

From these two equations it is possible to construct many functions where zeros correspond arbitrarily closely with the eigenvalues, ω. One possibility is to divide the given region, x_o to x_N, into N-1 intervals of width h and replace equations (4) and (5) with the difference equations

$$P_K(x_m)\varphi_{m+1} + (h^2R_K(x_m) - hQ_K(x_m) - 2P_K(x_m))\varphi_m + (P_K(x_m) - hQ_K(x_m))\varphi_{m-1} = 0$$
$$m = 1,2 \ldots, N-1$$

$$(hA_K(x_o) - B_K(x_o))\varphi_o + B_K(x_o)\varphi_1 = 0$$
$$(hA_K(x_N) - B_K(x_N)\varphi_N - B_K(x_N)\varphi_{N-1} = 0 \qquad \ldots (6)$$

These are a set of N linear algebraic equations in the φ_m which give increasingly better approximations to the values of the solutions φ of equation (4) at the points x_m as the value of N is increased. The condition that the equations (6) have a solution is that the determination, D_N of the coefficients be zero and this determinantal equation has roots which are approximations to the eigenvalues of equations (4) and (5). This determinant, then, is an appropriate function to use in solving the eigenvalue problem by CEVITQ. From equations (6) it is evidently only necessary to provide a means of calculating the functions P,Q,R,A,B to find the required eigenvalues. A routine, FUNXNJ, is available to set up and calculate the determinant D_N and only requires the user to provide routines to calculate P,Q,R,A,B. Exactly the same technique can be applied to the calculation of the eigenvalues of the differential-integral equations.

$$P_K \frac{d^2\varphi}{dx^2} + Q_K \frac{d\varphi}{dx} + R_K\varphi = \int_{x_o}^{x_N} G_K(x/y,\omega)\ \varphi(y)\ dy$$

A corresponding routine FUNXNJ is available and the user need only specify P,Q,R,G. Thus, with these routines and CEVITQ an immediate solution to a wide class of eigenvalue problems is possible.

IV. CONCLUSIONS

The major part of the effort required in scientific uses of computers is in the development of programs, rather than in setting up the mathematical basis of the program or in producing results. The program described uses basically sound numerical methods supported by several ad hoc procedures which greatly extend the range of application. This is an alternative to providing a wide variety of different numerical methods suited to different problems. Using this program many eigenvalue problems have been solved with an order of magnitude less effort than that required to solve the initial helicon wave problem. The

program has been constructed in such a way that it is completely compatible with any other program or subroutine and can be added to without reference to the original FORTRAN.

V. ACKNOWLEDGEMENTS

The author wishes to thank Dr. P.C. Thonemann for suggesting the original helicon wave problem and to the Theory Division Computing Section for advice on many points arising from the computer programming.

VI. REFERENCES

1. KLOZENBERG, J.P., McNAMARA, B., THONEMANN, P.C. The dispersion and attenuation of helicon waves in a uniform cylindrical plasma. J. Fluid Mech., vol.21, no.3, March, 1965. pp.545-563.

2. FRANK, W.L. Finding zeros of arbitrary functions. Ass. Comp. Mech. J., vol.5, no.2, 1958. pp.154-160.

3. LARKIN, F.M. A combined graphical and iterative approach to the problem of finding zeros of functions in the complex plane. Comp. J., vol.7, no.3, 1964. pp.212-219.

4. MULLER, D. A method for solving algebraic equations using an automatic computer. Math. Tab., Wash., vol.10, 1956. p.208-215.

5. FRANK, E. On the calculation of roots of equations. J. Math. and Phys., vol.34, 1955. pp.187-197.

6. MURRO, W.D. Some iterative methods for determining zeros of functions of a complex variable. Pacific J. Math., vol.9, 1959. pp.555-566.

P A R T II

OPERATING MANUAL FOR CEVITQ

INTRODUCTION TO OPERATING MANUAL

Program CEVITQ is a standard program to find zeros of K functions of a complex variable, $Z = X + iY$, an arbitrary number of floating point parameters, and an arbitrary number of fixed point parameters. The program has been written in such a way that it can be operated successfully with no knowledge of FORTRAN - if the single essential subroutine FUNXNJ defining the K functions can be provided. For the user who must write his own, the first section of this manual shows how routine FUNXNJ should be written. The second section describes the data required by the program; the data layout is first summarised in Table I and subsequently described in full detail. The third section describes the card input to STRETCH and KDF9. These three sections contain all the information necessary for successful operation of the program.

The fourth and fifth sections describe FUNXNJ routines which have been written to solve the eigenvalue problem for second order differential and differential-integral operators. The user of these routines is required to supply further, simpler, routines to define the coefficients, kernels, and boundary conditions of the operators. From these definitions the FUNXNJ routines provided construct a function whose zeros are approximations to the eigenvalues of the corresponding operators. A simple worked example is given in each section.

Section VII describes a special control routine GPDUMQ which allows automatic variation of two independent parameters in an eigenvalue problem.

The next two sections of the manual, VII and VIII, give all the information which might be required by the more ambitious user who requires other combinations of the component operators in CEVITQ. Section VII gives the argument lists for the components of the root-finding operator CRITIQ and shows how the iteration process might be modified. Section VIII discusses the various means of user control. CEVITQ calls a dummy subroutine, GPDUMQ, at every important juncture in the program and suggestions are made for its use. Methods of operating the program facilities without the use of data cards and provision of alternative graphical output are also given.

Finally, Section IX provides a complete worked example with sample output.

I. SUBROUTINE FUNXNJ

This is the only routine the user must provide. In the case of second order differential-integral equations this obligation refers to routines for evaluating the coefficients of the differential operators and for the integral kernel (cf. Sections IV, V).

Subroutine FUNXNJ should evaluate the real and imaginary parts U, V of the K functions of a complex variable, $Z = X + iY$, whose roots are required

$$F_K\ (Z,\ P_1,\ P_2, \ldots,\ M_1,\ M_2\ \ldots.,\ M_L) = U_K + iV_K$$

The routine should be written as follows:

```
SUBROUTINE FUNXNJ (X,Y,U,V,K)
COMMON W,Z, ...., P1, P2, P3, .... PN,A,B, ....
COMMON M1, M2, ...., ML,F,G, ....
FORTRAN Statements ending with
U = Real part of F_K
V = Imaginary part of F_K
RETURN
END
```

FUNXNJ can CALL any other subroutines or programs provided by the user. The variables W,Z, ..., A,B, ..., F,G, ... are COMMON variables not required by FUNXNJ but which may be necessary for compatibility with other routines the user may need. Program CEVITQ does not itself require any COMMON storage although there must be at least one element in the COMMON list, in the KDF9 version, which is used only to provide a reference point in the machine.

CEVITQ reads integers ('fixed point variables') and numbers with a fractional part ('floating point variables') differently. The necessity for providing means for reading integers separately arises because integers are used by the computers differently from floating point numbers. The distinction is only important when integer is to be used as a DO-loop index, the index of a computed GO TO, or as an array subscript. Thus, the floating and fixed point numbers, p_i and m_i, required for the definition of the F_K may appear in two independent blocks in the COMMON list.

To understand the value of this arrangement it should be realised that the COMMON list is an ordered list, that the name used for a COMMON variable in a routine is a dummy and that only its position in the list matters. Thus, CEVITQ can refer to W and Z in the above example merely as the first and second elements of the list.

It must be remembered that STRETCH and KDF9 only deal with real numbers and so the real and imaginary parts of all expressions involved in F_K must be evaluated separately. Also, all the singular points of $F_K(Z)$ must be defined carefully, the best approach being to test Z and if F_K at a singular point is required set $U,V = 10^{20}$.

P A R T II

OPERATING MANUAL FOR CEVITQ

INTRODUCTION TO OPERATING MANUAL

Program CEVITQ is a standard program to find zeros of K functions of a complex variable, $Z = X + iY$, an arbitrary number of floating point parameters, and an arbitrary number of fixed point parameters. The program has been written in such a way that it can be operated successfully with no knowledge of FORTRAN - if the single essential subroutine FUNXNJ defining the K functions can be provided. For the user who must write his own, the first section of this manual shows how routine FUNXNJ should be written. The second section describes the data required by the program; the data layout is first summarised in Table I and subsequently described in full detail. The third section describes the card input to STRETCH and KDF9. These three sections contain all the information necessary for successful operation of the program.

The fourth and fifth sections describe FUNXNJ routines which have been written to solve the eigenvalue problem for second order differential and differential-integral operators. The user of these routines is required to supply further, simpler, routines to define the coefficients, kernels, and boundary conditions of the operators. From these definitions the FUNXNJ routines provided construct a function whose zeros are approximations to the eigenvalues of the corresponding operators. A simple worked example is given in each section.

Section VII describes a special control routine GPDUMQ which allows automatic variation of two independent parameters in an eigenvalue problem.

The next two sections of the manual, VII and VIII, give all the information which might be required by the more ambitious user who requires other combinations of the component operators in CEVITQ. Section VII gives the argument lists for the components of the root-finding operator CRITIQ and shows how the iteration process might be modified. Section VIII discusses the various means of user control. CEVITQ calls a dummy subroutine, GPDUMQ, at every important juncture in the program and suggestions are made for its use. Methods of operating the program facilities without the use of data cards and provision of alternative graphical output are also given.

Finally, Section IX provides a complete worked example with sample output.

I. SUBROUTINE FUNXNJ

This is the only routine the user must provide. In the case of second order differential-integral equations this obligation refers to routines for evaluating the coefficients of the differential operators and for the integral kernel (cf. Sections IV, V).

Subroutine FUNXNJ should evaluate the real and imaginary parts U, V of the K functions of a complex variable, $Z = X + iY$, whose roots are required

$$F_K\ (Z,\ P_1,\ P_2,\ldots,\ M_1,\ M_2\ \ldots.,\ M_L) = U_K + iV_K$$

The routine should be written as follows:

```
SUBROUTINE FUNXNJ (X,Y,U,V,K)
COMMON W,Z, ...., P1, P2, P3, .... PN,A,B, ....
COMMON M1, M2, ...., ML,F,G, ....
FORTRAN Statements ending with
U = Real part of F_K
V = Imaginary part of F_K
RETURN
END
```

FUNXNJ can CALL any other subroutines or programs provided by the user. The variables W,Z, ..., A,B, ..., F,G, ... are COMMON variables not required by FUNXNJ but which may be necessary for compatibility with other routines the user may need. Program CEVITQ does not itself require any COMMON storage although there must be at least one element in the COMMON list, in the KDF9 version, which is used only to provide a reference point in the machine.

CEVITQ reads integers ('fixed point variables') and numbers with a fractional part ('floating point variables') differently. The necessity for providing means for reading integers separately arises because integers are used by the computers differently from floating point numbers. The distinction is only important when integer is to be used as a DO-loop index, the index of a computed GO TO, or as an array subscript. Thus, the floating and fixed point numbers, p_i and m_i, required for the definition of the F_K may appear in two independent blocks in the COMMON list.

To understand the value of this arrangement it should be realised that the COMMON list is an ordered list, that the name used for a COMMON variable in a routine is a dummy and that only its position in the list matters. Thus, CEVITQ can refer to W and Z in the above example merely as the first and second elements of the list.

It must be remembered that STRETCH and KDF9 only deal with real numbers and so the real and imaginary parts of all expressions involved in F_K must be evaluated separately. Also, all the singular points of $F_K(Z)$ must be defined carefully, the best approach being to test Z and if F_K at a singular point is required set $U,V = 10^{20}$.

Finally, it should be noted that K is really somewhat redundant as it is probable that the zeros of only one function at a time will be required, as appears in the examples of other sections. However, program CEVITQ has been written to make reasonable provision for the unforeseeable uses of the program, and this type of redundancy has been found extremely useful. Indeed, the parameter K need not be a fixed point constant at all; it could equally well be a floating point array or any other kind of variable.

II. FACILITIES OFFERED AND DATA REQUIRED

The facilities offered and data required by CEVITQ are summarised in Table I below. Further details are explained in the remainder of this section. The data is to be punched in free format with two or more blanks between each number. Four numbers must be given on the first card to select the required facilities; subsequent data defines the function whose zeros are sought and the regions of the complex plane where CEVITQ must look. These blocks may be repeated for as many cases as are required.

TABLE I

First Data Card in Block

Selector	Selector Values	Meaning and Comment on Options
1	-1	Normal End of Job. No further data required.
Root-Finding Sequence	1	Test Option: Nearest root to origin with full print out using quadratic approximation.
	2	Isolated Root. Guess, iteration region, Max. Value of F_K, and parameter required as further data.
	3	Series of NR Roots. NR and NR quesses at NR parameter values required.
	4	Root at 51 values of a Parameter. Initial guesses at end points of range required.
	4 + NP	Roots found at NP values of Parameter. Initial Guesses Required.
2 Variable Parameter	0 - NC	Position in COMMON list of variable parameter. NC is number of elements in list. ($\geqslant$ 1 in KDF9) If selector = 0 no element will be set.
3	0	No Graphical Output — See also Section VI for
Graphical	1	Graphs with Standard Captions. — Two Parameter Problems.
Output	2	Graphs with captions given at end of data block
4	-1, - 2	Same as +1, +2 but printed output suppressed. Error print survives.
Iteration	1	Quadratic approximation to F_K. Results printed.
and	2	Bilinear approximation to F_K. Results printed.
Print	3	Quadratic approximation to F_K with detailed print out.
	4	Bilinear approximation to F_M with detailed print out.

Data should now be provided on further cards to define the function whose roots were required.

TABLE I
(continued)

Card	Selector Value	Numbers	Comments
1		K	Selects the K^{th} Function. Number may be punched on the first card of the block with the other selectors. K must have a value even if not used.
2		NFP	Number of floating point numbers to be read.
2A		MFP	If NFP≠0, MFP, the position in the COMMON list of the first of the NFP numbers, is given on a separate card, 2A.
2B		P_i	Now give the floating point numbers. The numbers may be close-packed in Cols. 1-72 of as many data cards as necessary with two blanks between each number.
3		JFX	Number of Integers to be read.
3A		LFX	If JFX≠0, LFX, the position in COMMON of the first of the JFX integers, is given on a separate card, 3A.
3B		M_i	The JFX integers.

Further Data for each value of Selector 1

Card	Selector Value	Numbers	Comments
4	± 1	None	
	2	P X1,Y1,ΔX,ΔY ΔU,ΔV	If Selector 2≠0 the corresponding parameter is required cf. Fig.2 below. Initial Guess, iteration rectangle centred on guess, max. acceptable values of $U + iV = F_K$ at root. If ΔX = 0, ΔY,ΔU,ΔV not required. Program sets ΔX =ΔY = 1.0, ΔU = ΔV = 0.001
4	3	NR	Number of roots required.
5,6,...		P,XI,YI,ΔX, ΔY,ΔU,ΔV	Parameter and Initial guesses etc. for each root required. Each set of seven numbers on separate card or cards. ΔX = 0 treated as above. (Selector = 2).
4	4,4 + NP	P1,X11,Y11,	First parameter value and initial guess at this value.
		P2,X12,Y12,	Second parameter value and guess.
		ΔX,ΔY,ΔU,ΔV	Iteration region and max. values of U, V at root.

Finally, after all the data has been supplied, the program requires the caption cards, as described below, if Selector 3 = 2.

The above data layout may be repeated for as many cases as required.

THE FOUR ROOT FINDING SEQUENCES

Four root finding sequences are offered:

(i) The root of F_K nearest the origin will be found with complete print out of the operations performed.

(ii) A root of F_K in a specified region of the Z-plane will be found.

(iii) NR roots of F_K at NR given values of a specified parameter, P_i, will be found

(iv) NR roots of F_K at NR values of a specified parameter, P_i, in a given range, P1 to P2, will be found.

Thus, the first number to be given to CEVITQ, Selector 1, has values 1,2,3,4. If the Selector is 4, 51 roots will be found. If NR roots are required Selector 1 should be NR + 4.

THE VARIABLE PARAMETERS

The second number required by CEVITQ is the position in the COMMON list of the parameter to be varied. If the COMMON list in the example in Section I read

```
COMMON W,Z,P1,P2, ....
```

and P2 is to be varied, Selector 2 should be 4. Selector 2 must be set even if Selector 1 is 1 or 2.

GRAPHICAL OUTPUT

The third selector chooses the graphical output. The three possibilities are: no graphs, graphs with standard captions, graphs with captions provided by the user and the corresponding values of Selector 3 are 0,1,2. The output is provided by a single subroutine, GRAFBQ (cf. Section IX if different output style is required.) This routine utilizes the standard Culham graph-plotting facilities and output can be on either the Benson-Lehner or the Stromberg Carlson 4020 microfilm recorder.

The output is as follows: the real and imaginary parts of the roots are plotted as ordinate on separate graphs. Each graph is scaled to fit a 10 inch square on the Benson-Lehner, or a standard square on the SC 4020, in such a way that the scales are 1,2,4, or 5×10^N inches per unit variable, N integer. This enables results to be read off the graphs with a minimum of inconvenience. The minimum and maximum values along the edges of the square are printed on the graphs. The calculated points are printed on the graphs

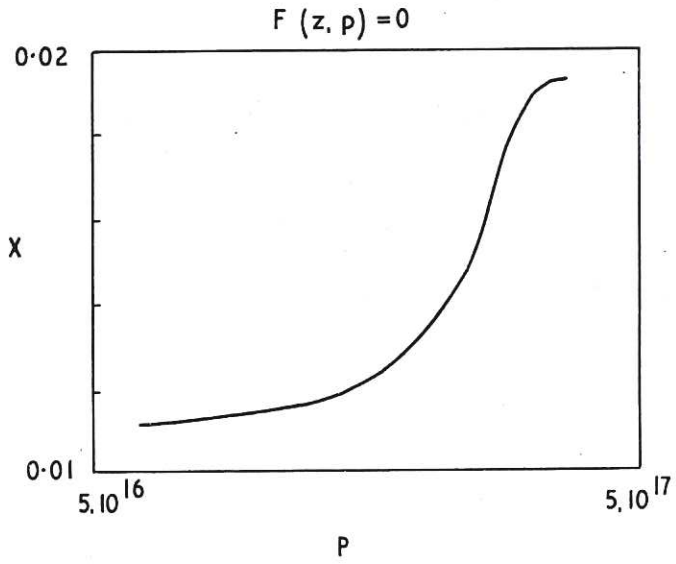

Fig. 5 Standard graphical output (CLM-R48)

and a smooth curve drawn between them. A typical output with standard captions is sketched in Fig.5. Selector 3 must be set even if Selector 1 is 1 or 2.

The user may provide special captions as follows (cf. Library File 1.10 CAPTBQ) after all other data has been given: a series of cards is punched according to the format

First Card:

Date in Cols.1-9, Format 3(1X,2I)
Initials, Cols.11-30
Description, Cols.31-70
Integer NC $\leqslant$ 3 Col.72

NC further cards punched in Cols.2-72 with a total of four captions, each followed by a $ sign. The first caption is the heading for the graphs and replaces 'F (Z,P) = 0'. The second caption labels the abscissae on each graph and the last two captions refer to the real and imaginary parts of the root.

ITERATION OPTIONS

The fourth selector decides the function used to approximate the F_K and controls the printing of the results. The F_K are approximated by a quadratic function of Z if Selector 4 is 1 or 3, and by a bilinear function of Z if Selector 4 is 2 or 4. The larger values in each case allow print out of complete details of the operations performed, unless Selector 1 is $\geqslant 4$. In this last case complete print out is only provided for the two given guesses at the end points of the range; the other roots are printed as they are found. Finally, a negative value of Selector 4 suppresses all print out except when the program fails.

All that remains in the way of data is to define the F_K and the guesses at the roots of the selected F_K.

DATA

It is now convenient to state how the data is to be punched. 'Formatless' reading is used in both versions of the program but the detailed operations of the corresponding routines are slightly different; the following instructions apply to either version of the

program and contain both sets of restrictions*. The required data must appear in the correct order on data cards in columns 1 to 72 and the numbers must be separated by at least two blank columns. As an example, the number 27 may be punched in any of the following forms:

27 27.0 2.7E1 270.0 E-1

If zero is intended to be read a zero must be punched on the data card.

The first data card or cards required by CEVITQ must give the four Selectors and the integer, K.

Floating point parameters, $P_i, i = 1,N$, defining FUNXNJ are then read into the COMMON list from position M; the integers N and M must be given on separate data cards in this order, followed by the P_i on as many cards as necessary.

Fixed point parameters, $m_i, i = 1,J$ are now read into COMMON from position L. The integers J and L must be given on separate cards followed by the m_i. If J or N are zero, zero(s) must be punched on one (or two) data card(s) and, of course, the corresponding position(s) M(L) need not be given, indeed must not be given.

Some of the four basic root finding sequences now require further data on a card, or cards, separate from the above. The first sequence of course requires no further data.

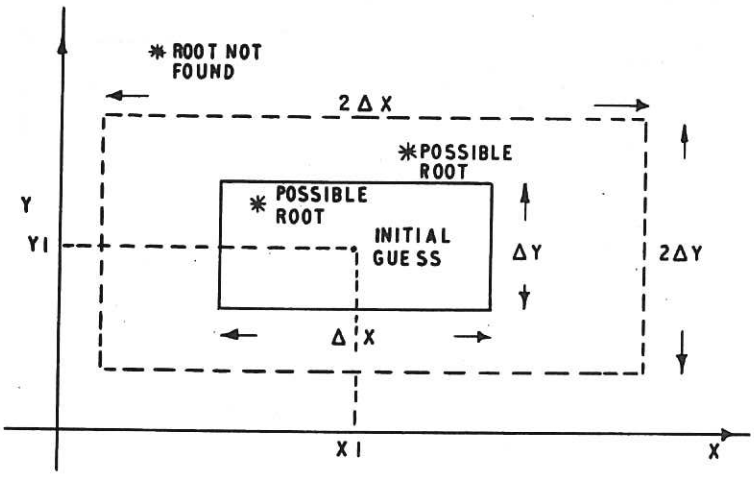

Fig. 6 (CLM-R 48)
Iteration region in x-y plane and possible roots of $F_K = 0$

The second option first requires a single floating point number, the parameter to be varied, if Selector $2 \neq 0$. Thereafter, it requires X1,Y1, the initial guess at the position of the required root of F_K (cf. Fig.6), ΔX, ΔY, the dimensions of the rectangular region of the X-Y plane, centred on X1,Y1, in which the iteration is to be confined. Finally, ΔU and ΔV are to be given, these being the maximum values of the real and imaginary parts of F_K which will be accepted as locating the root of F_K.

*The most troublesome restriction applies to the KDF9 version, namely the necessity for punching certain of the data on separate cards. This is not really necessary with the STRETCH version and all the data for a sufficiently simple case could appear on a single card.

If $\Delta X = 0$, then $\Delta Y, \Delta U, \Delta V$ need not be given and will be set to 1.0, 0.001 and 0.001 respectively; ΔX will be set to 1.0.

The third sequence requires much the same data but preceded by an integer NR, the number of roots to be found. NR sets of X1, Y1, $\Delta X, \Delta Y, \Delta U, \Delta V$ as described above must be given, each set being on a separate card or cards.

The fourth basic option requires $P_i(1)$, X1(1), Y1(1), $P_i(2)$, X1(2), Y1(2), ΔX, ΔY, ΔU, ΔV: The first six numbers are, the values of the variable parameter and a guess at the location of the required root at each parameter value. The last four numbers define iteration region and the maximum acceptable value of F_K at the roots. The rectangle is, of course, centred on the current initial guess during the calculations.

Finally, if Option 3=2 the caption cards must be given. These cards will always be read, even if the given case fails, so that the next case will not be affected by spurious data.

CEVITQ can deal with an indefinite sequence of independent cases with data for each case supplied as described.

III. INPUT TO THE COMPUTERS

STRETCH INPUT

The program is on the Culham Library Tape and the makeup of the input deck must be as follows: (LC means Library Card).

TABLE II

STRETCH Input

Card	Contents of Library File	Remarks
Job Card		Available from Computer Section
Time Card		Estimated time required for Job
SC-4020		Only inserted if graphs required on SC-4020
LC 172 or LC 215 or Any other PRELUDE supplied by the user.	General Purpose PRELUDE 400 Scalar PRELUDE	Only required if COMMON arrays required by user. Performs all essential functions of PRELUDE. Reads no data. Standard Prelude used when no COMMON arrays required
LC 400	Program CEVITQ: Includes dummy GPDUMQ, GRAFBQ and a 'MAIN program'.	GPDUMQ, GRAFBQ, and 'MAIN program' may be over-written by user by loading appropriate user routine after this card.
LC 214 LC 001	Graphical output: GRAFBQ Standard B-L Graph Plotting routines	Standard graphs. Overwrites dummy GRAFBQ on LC 400. Need only be inserted if graphs are required.
LC 179 LC 127 LC	SYMBOQ-Script Routines. Converts PLOTBQ to PLOTSC SC-4020 Routines	Only inserted if SC-4020 output required.
Now insert FUNXNJ and any other routines required by User.		
END 1 END 2 QUIT		Standard Control Cards

Now give PRELUDE data, if any, and then Option and Data Cards as listed in Section II.

TABLE III

KDF9 INPUT

Mandatory Cards	Optional Cards	Comments
Job Card *XEQ	*IODTAPE 100/IBM/SAVE	Only required for graphical output
*DISC PROGRAM CEVITQ		
	*PRELUDE *DELETE SCALAR	See Section VII on Variation of 2 parameters.
*CHAIN 1 RLB or FORTRAN for FUNXNJ		
	*SELECT DIFEQN/FUNXNJ *SELECT DIFINT/FUNXNJ	2nd Order Differential equation. See IV Diff-Integral equation. See V
	*CHAIN 2 *SELECT TWOPAR/GRAFBQ	See VI
*DATA Data for Job *END JOB		

IV. EIGENVALUES OF 2nd ORDER LINEAR DIFFERENTIAL EQUATIONS

To find eigenvalues, Υ, of K 2nd Order Linear differential equations of the form

$$P_K(\Upsilon,p_i;x)\,\frac{d^2\varphi}{dx^2} + Q_K(\Upsilon,p_i;x)\,\frac{d\varphi}{dx} + R_K(\Upsilon,p_i,x)\,\varphi = 0 \qquad \ldots (1)$$

subject to boundary conditions at two points x_o, x_N of the form

$$A_K(\Upsilon,p_i,x_m)\,\varphi(x_m) + B(\Upsilon,p_i,x_m)\,\frac{d\varphi}{dx}\Big|_{x=x_m} = 0 \qquad m = 0,N \qquad \ldots (2)$$

where the p_i are a set of parameters required to define the above coefficients, a routine FUNXNJ has been provided for use with CEVITQ. The user is required to provide two subroutines. The first is,

SUBROUTINE BOUNDJ (AB, XO, XN, N, REGAM, GAMIM,K)

where N is the number of intervals to be used in the difference approximations to equations (1), (2). AB is a 4 × 2 array through which the real and imaginary parts of $A(x_o)$, $B(x_o)$ $A(x_N)$, $B(x_N)$ are returned to FUNXNJ. The user must also supply XO, XN, and N. The second routine required is

SUBROUTINE COEFFJ (PQR, X, REGAM, GAMIM,K)

PQR is a 3 × 2 array through which the real and imaginary parts of P_K, Q_K, R_K at X and Υ = (REGAM + i GAMIM) are returned to FUNXNJ.

Subroutine FUNXNJ and routine TRIDIQ, a routine required by FUNXNJ, is available on the Culham Library Tape on File 131 . For reference, TRIDIQ has arguments, (D,M,U,V) where D is an array of dimension M × 2 in which are stored the real and imaginary parts of the principal, upper, and lower diagonals of a tri-diagonal matrix. U and V are real and imaginary parts of the determinant of this matrix and represent the F_K whose roots correspond to the eigenvalues of the K equation (1).

The best method of using these routines is well described in terms of a simple example. Suppose one required the first few eigenvalues of

$$\frac{d^2\varphi}{dx^2} + \frac{1}{x}\frac{d\varphi}{dx} + (S^2\gamma^2 - \frac{m^2}{x^2})\ \varphi = 0 \qquad \ldots (3)$$

Subject to the boundary conditions

$$\begin{aligned} \varphi &= 0 \quad \text{at} \quad x = 0 \\ m\varphi + \frac{d\varphi}{dx} &= 0 \quad \text{at} \quad x = 1 \end{aligned} \qquad \ldots (4)$$

The first thing to do is multiply equation (3) by x^2 to eliminate the singularity at x = 0. The routines provided might be as follows

```
      SUBROUTINE BOUNDJ (AB, XO, XN, N, REGAM, GAMIM,M)
      COMMON NPT
      DIMENSION AB(4,2)
      AB(1,1) = 1.0
      AB(1,2) = 0.0
      AB(2,1) = 0.0
      AB(2,2) = 0.0
      AB(3,1) = M
      AB(3,2) = 0.0
      AB(4,1) = 1.0
      AB(4,2) = 0.0
      XO = 0.0
      N = NPT
      XN = 1.0
      RETURN
      END

      SUBROUTINE COEFFJ (PQR, X, REGAM, GAMIM,M)
      COMMON  NPT,S
      DIMENSION PQR(3,2)
      R = S*X
      PQR(1,1) = X*X
      PQR(1,2) = 0.0
      PQR(2,1) = X
      PQR(2,2) = 0.0
      PQR(3,1) = (REGAM**2-GAMIM**2)*R*R - M*M
      PQR(3,2) = 2.0*REGAM*GAMIM*R*R
      RETURN
      END
```

One must now find some estimate of the position of the required eigenvalues of equation (3) for some values of S and m. This estimate could then be given to CEVITQ with a request for the first element of COMMON to be varied from, say 10 to 30 and that 21 roots be found. From the results a suitable value of this first element, the number of points used in the difference scheme, can be chosen which will give the required per cent accuracy. It will be found that 10 points is sufficient for 1% accuracy and that 20 points gives 5 or 6 figures. An acceptable value of F_M at a root is of order ((XN-XO)/NPT) NPT which in the case of equation (3) which has already been normalised to a unit interval might be 10^{-25} over the range of NPT.

Having settled on a value of NPT one can now proceed to find the required eigenvalue over a range of the parameter, S.

V. EIGENVALUES OF DIFFERENTIAL-INTEGRAL EQUATIONS

To find eigenvalues, Υ, of K 2nd Order Linear differential-integral equations of the form

$$P_K(\Upsilon,p_i,x)\frac{d^2\varphi}{dx^2} + Q_K(\Upsilon,p_i;x)\frac{d\varphi}{dx} + R_K(\Upsilon,p_i,x) = \int_{x_o}^{x_N} G_K(\Upsilon,p_i;x|y)\varphi(y)dy \qquad \ldots (5)$$

where the p_i are parameters, a routine FUNXNJ has been provided for use with CEVITQ. The user is required to provide two subroutines to define P,Q,R,G,X_o,X_N and the number of points, N, to be used in the finite difference approximation to equation (5). The first is,

SUBROUTINE FREDIJ(X,Y,REGAM,GAMIM,K,GREAL,GIMAG,XO,XN,N)

where

$$G_K(x|y) = \text{GREAL}\ (X|Y) + i\ \text{GIMAG}\ (X|Y)$$

$$\Upsilon = \text{REGAM} + i\ \text{GAMIM}$$

The second routine required is

SUBROUTINE COEFFJ (PQR,X,REGAM,GAMIM,K)

PQR is a 3 × 2 array through which the real and imaginary parts of P_K,Q_K,R_K at X are returned to FUNXNJ.

Subroutine FUNXNJ and routine MDO2A, (cf. Library File 133), are available on the Culham Library Tape on File Routine MDO2A evaluates the determinant of a complex square matrix.

The best method of using these routines is illustrated by the example in the preceding section. The only point of difference in the treatment arises when the integrals

are singular. There are three possible ways in which the operators can be singular (1) they have discontinuities (2) they have singularities in the range of integration (3) the limits of integration extend to infinity. The first type of singularity should not affect the numerical results. The only point of interest about the second type is that the user should ensure that the value of G_K at the singularity is set to some suitably large value, not a machine ∞, which will not cause subsequent calculations to go outside the range of the machine. The last kind of singularity can be approximated by choosing large finite values for the limits XO and/or XN.

VI. VARIATION OF TWO PARAMETERS

A special control routine GPINTQ is available for calculating the dependence of a root, Z, of F_K on two parameters, PA,PB. The results of the calculations are stored in two PUBLIC matrices EIGREQ,EIGIMQ, storage for which must be allocated in Prelude. Thus, one must insert in the Job Deck cards *PRELUDE and *DELETE SCALAR immediately after the Disc Program card. The prelude so chosen reads the first Data card, on which should be punched the number of values, NP of parameter PA at which the root is required.

The special control routine GPINTQ is invoked by inserting a card, *SELECT TWOPAR/GPINTQ, after the *CHAIN 1 card. This routine allows the usual form of input, as detailed in Section II.

The meaning of some of this input is slightly different from that in Section II. Thus, the values of Selector 1 are 4,4 + NPB, where NPB is the number of values of parameter PB at which the root is to be found. Due to space limitations, NPB $\leqslant$ 51. The graphical Selector has completely different meanings, appropriate to the two-dimensional nature of the calculations, as follows:

0 No graphical output

1 Real Z is plotted against Imaginary Z as PA varies for each value of PB and, on a separate graph, as PB varies for each value of PA.

2 Real part of Z is plotted against PA for each value of PB and the Imaginary part is similarly plotted on a separate graph.

3 Contours of ReZ and Imaginary Z are drawn on separate graphs in the PA-PB plane.

The nature of these options can be better understood by referring to the example at the end of this section.

To save time and effort more than one of the above options may be taken with higher values of the Graph Selector as follows:

4	≡	1 and 3
5	≡	2 and 3
6	≡	1 and 2 and 3
7	≡	1 and 2

If graphical output is required then cards *CHAIN 2, *SELECT TWOPAR/GRAFBQ are required just before the *DATA card.

The Iteration/Print Selector has the following effects; the more positive the given value the more print out is produced, the more negative it is the less print out is produced. The smaller modulus number of each pair of numbers in the table below causes Muller's process to be used, the larger causes the Bilinear Approximation to be used.

3,4	Complete print out at every root is found together with collected tables of results. Matrices printed finally.
1,2	Tables of results only at each value of PB. Matrices Printed.
-1, -2	Tables of results only at the two ends of the PB range. Matrices Printed.
-3, -4	No print out of any kind, except failure print, is produced.

Having read the selector values the program now reads data to define the F_K and the usual data required when Selector 1 $\geqslant$ 4. The only further data required is

NPBCOM	Position in Common of Parameter PB. It is assumed that the initial value of PB has been supplied in the definition F_K.
PB2	Final value of PB.
XB1,YB1,XB2,YB2	The two guesses at the location of the required root at the upper corners of the rectangle in the PA-PB plane.

These last six numbers may appear on one card. Finally, if graphical output is in the form of contour maps then further data is required to determine the contour heights. On one card two integers NHR,NHI must be given to decide how many contour heights are required for the real and imaginary graphs. The program accepts the following numbers:

NHR or NHI zero:	11 equally spaced heights between the minimum and maximum of the calculated values of the real or imaginary part of the root will be used.
NHR or NHI negative:	|NHR| or |NHI| heights will be used.
Only NHR or only NHI positive:	Corresponding number of heights must be given on subsequent data cards.

NHR = NHI>0: NHR heights will be read and used for both graphs.

NHR>0,NHI>0,NHR≠NHI: NHR + NHI heights will be read. The first NHR will be used for contours of the real part of the root and the last NHI for the imaginary part.

AN EXAMPLE

The following dispersion relation arises in the study of electrostatic waves of the form exp $i(k_x X + k_z Z - \omega t)$ in a magnetised plasma slab interacting with an electron beam:-

$$k_x^2 \left(1 + \frac{\omega_b^2}{\Omega^2 - \mu_b^2}\right) + \frac{\omega_p^2}{\Omega^2 - \nu_p^2} + k_z^2 \left(1 - \frac{\omega_p^2}{\mu_b^2} - \frac{\omega_b^2}{\nu_p^2}\right) = 0$$

where ω_b, ω_p are the plasma frequencies of the beam and plasma respectively, Ω is the electron cyclotron frequency and

$$\mu_b = \omega - n\Omega - ikv_b \qquad n, \text{ integer}$$

$$\nu_p = \omega - ikv_p$$

Here v_b and v_p are the thermal velocities of the beam and the plasma. This dispersion relation has solutions with negative $\mathrm{Im}\omega$ for real k_x and k_z representing an unstable oscillation. An important feature of the dispersion relation is that the function $k_z = k_z(\omega)$ has a branch point in the negative half of the complex ω-plane showing that the instability can be convective[1].

To calculate solutions of this dispersion relation and find the location of this branch point one would use CEVITQ with the two-parameter facilities to map the k_z-plane into the ω-plane and vice versa.

Some typical results are shown in Fig.7. Here the real and imaginary parts of ω are contoured in a portion of the k_z-plane. The branch point is located at point B. The program was asked to follow the root as $\mathrm{Re}k_z$ was changed from 6.0 to 1.0 and as $\mathrm{Im}k_z$ was altered from 0.0 to -2.5. As the branch point was approached the program failed and worked back from $\mathrm{Re}k_z = 2.0$ to B. Thereafter, the branch cut was pushed before the iteration process to the edge of the region in the k_z-plane as shown.

In Fig.8 are shown the root trajectories as k_z was varied, effectively contours of constant $\mathrm{Im}k_z$ and $\mathrm{Re}k_z$ of the function $k_z(\omega)$ in the ω-plane. The branch point is shown at B and the real axis in the k_z-plane maps into the curve AC.

The input to KDF9 was as follows:

Cards	Comments
* JOB	
* IODTAPE 100/IBM/SAVE	
* XEQ	
* DISC PROGRAM CEVITQ	
* PRELUDE	
* DELETE SCALAR *SELECTS VERSION TWOPAR	
* CHAIN 1	
* SELECT TWOPAR/GPINTQ	
	Fortran defining the dispersion relation.
* CHAIN 2	
* SELECT TWOPAR/GRAFBQ	
* DATA	
21	Prelude Data. 21 points to be used in $R_e k_z$.
30 9 4 -1 1	(30-4) points in $Im k_z$, Vary $R_e k_z$- parameter 9, Graphical output 1 & 3, ω-arrays to be printed, n = 1.
10	Parameters defining dispersion relation to be
1	read in from position 1 in COMMON.
12.0 8.0 .09 .79 .237 3.0	0. 0. 0. 0.
0	No integers to be read
6.0 6.25 .16 1.0 7.7 .06	0.5 0.5 1.0E-4 1.0E-4
	$R_e k_z$ and guesses at required root ω etc.
10 -2.5 6.9 .3 8.3 1.3	Vary 10th element of COMMON, $Im k_z$, down to -2.5 where root is near 6.9, 0.3 and 8.3, 1.3 at $R_e k_z$ = 6.0,1.0
0 0	Use 11 contour heights for Re and $Im\,\omega = \omega(k_z)$.
* END JOB	

Location of the 546 roots, print out of the results and graph plotting took 8 minutes for the above job.

The author is indebted to Dr. J.G. Cordey who provided this example. (The effect of a Finite Temperature on the Electron Cyclotron Resonance Instability. CLM R-44 1965.)

VII. CRITIQ, THE ROOT FINDING OPERATOR

The fundamental routine used by CEVITQ to find complex zeros of $F_K(X + i\,Y) = U_K + i\,V_K$ is

SUBROUTINE CRITIQ (XV,YV,METHOD,K,MFAIL)

where XV,YV are two 5-vectors such that (cf. Fig.2):

Initial guess at required root is Z_i = XV(1) + i YV(1). The iteration is restricted to a rectangle in the complex Z-plane centred on the initial guess and with sides.

$$\Delta X = XV(2)$$
$$\Delta Y = YV(2)$$

If ΔX or ΔY is greater than 10^{10} no check is made on the wanderings of the iteration process in the corresponding direction(s). A point Z_r is accepted as a root of F_K if

$$|U_K| \leqslant XV(3)$$

and

$$|V_K| \leqslant YV(3)$$

The root, Z_r, is returned to the calling program in

$$Z_r = XV(4) + i\ YV(4)$$

and the value of F_K at this point is

$$F_K(Z_r) = XV(5) + i\ YV(5)$$

The approximation function to be used in the iteration process is specified by METHOD. This number also controls the print out provided by CRITIQ:

METHOD	= 1	Quadratic Approximation, no print out.
	= 2	Bilinear Approximation, no print out.
	= 3	Quadratic Approximation, full print out.
	= 4	Bilinear Approximation, full print out.

The success or failure of the iteration is indicated by the integer MFAIL,

MFAIL	= 1	Unqualified success.
	= 2	Success, but the dimensions of the given rectangle had to be doubled in one or both directions (cf. Fig.2).
	= 3	Failure.

If the routine fails the last three points calculated are printed out. The theoretical basis of the routine is discussed in Part I of this report.

The roots of the two approximation functions are calculated by two subroutines as follows:

SUBROUTINE QUADRQ (S,T,U,V,I1,I2,I3,A,B,R1,R2)

where U,V are three-vectors and S,T are four-vectors such that

$$F_K(S(I) + i\ T(I)) = U(I) + V(I), \qquad I = 1,2,3$$

I1, I2, I3 are permutations of 1,2,3 such that

$$|F_K(I1)| < |F_K(I2)| < |F_K(I3)|$$

The values of the coefficients of the quadratic $az^2 + bz + c$, generated by three numbers are returned in the five-vectors A,B:

$$a = A(1) + iB(1)$$

$$b = A(2) + iB(2)$$

$$c = A(3) + iB(3)$$

The roots of the quadratic are given in A(4) + iB(4), A(5) + iB(5). The equations between these two roots and the point S(1) + iT(1) are returned in R1 and R2 and S(4) + iT(4) is the nearer root.

The other routine is

SUBROUTINE BILINQ (S,T,UV)

UV is a 2 × 3 array such that

$$F_K(I) = UV(1,I) + i\ UV(2,I), \quad I = 1,2,3$$

S,T are as above but now S(4) + i T(4) is the single root of the Bilinear function.

The complete list of subroutines called by CRITIQ and provided in program CEVITQ is: QUADRQ, BILINQ, RECTAQ, CONSTQ, WTESTQ, CHANJQ, ORDERQ.

VIII. USER CONTROL

This section is perhaps the most important in the operating manual. It is hoped that the instructions given here will be sufficient to enable the user to devise any conceivable combination of root-finding and other operations with a minimum of effort.

The first point is that the program and all its facilities can be used at any time by writing

```
CALL CEVITQ
```

The main program loaded with Library Card 400 is a dummy which just calls CEVITQ and can be overwritten by the user if an alternative main program is loaded.

The second and major point of interest is the use of the subroutine GPDUMQ. This has arguments as follows

```
SUBROUTINE GPDUMQ (XV,YV,MFAIL,K,IG,XR,YR,PV,UR,VR)
```

where XV,YV are the two iteration vectors as described in the previous section and MFAIL indicates whether or not a root has been successfully found. IG is an integer which indicates from what point in the program GPDUMQ has been called. (See Fig.2 Part I. Flow Diagram for (CEVITQ)) At points G1, IG = - 1 and the user may for example take the opportunity of executing further input-output operations to define the job. Program CEVITQ uses CRITIQ with the following calling sequence, corresponding to points G2,G3 of Fig.2.

```
CALL GPDUMQ (XV,YV,MFAIL,K,0,XR,YR,PV,UR,VR)
CALL CRITIQ (XV,YV,METHOD,K,MFAIL)
CALL GPDUMQ (XV,YV,MFAIL,K,1,XR,YR,PV,UR,VR)
```

After this sequence the results of the iteration, if successful, are stored in the 300-vectors XR,YR, and the parameter values, if any, are stored in vector PV. The values of F_K at these roots and parameter values are stored in vectors UR,VR.

Thus, prior to any particular iteration, the user can completely redirect the process, changing the size of the iteration region, the acceptable limits on the F_K, etc. After

any particular iteration the user can perform other operations or, if the iteration was unsuccessful, would call CRITIQ to make a further attempt at locating the root, using a better initial guess etc. MFAIL,XU,YV should, of course, be changed appropriately.

Finally, when all iteration requested in a particular case have been performed, and all roots found or not found, GPDUMQ is called with IG = 2 (Points G4 in Fig.2). The results can now be output by the user in any form required or control may be passed back to CEVITQ when the standard forms of output will be given.

The four basic options provided by CEVITQ can all be operated by the user without using data cards. The first two options simply use CRITIQ directly, and the use of this routine has already been explained in Section VII. The third option is executed by Subroutine ZDSETQ:

SUBROUTINE ZDSETQ (NOPT2,-NOPT3,METHOD,K,XR,YR,PV,UR,VR,IC,XV,YV)

NOPT2 is Option 2, the position in the COMMON list of the parameter to be varied. -NOPT3 is the negative of the required graphical Option. This instructs ZDSETQ to look for the data required in the first eight elements of UR. Thus, UR(1) = NR, UR(2) = P, UR(3) = X1, UR(4) = Y1, UR(5) = ΔX, UR(6) = ΔY, RU(7) = ΔU, UR(8) = ΔV, as described in Section II. After each root has been found, the users GPDUMQ should provide data P,X1,Y1 in UR for the next required root. In the STRETCH version of the program IC = 100. In the KDF9 version (and the STRETCH version) IC can be obtained by writing

CALL CBASEQ (C1,IC)

where C1 is the first element of the COMMON list. A negative value of METHOD suppresses all printed output.

The fourth option is executed by

SUBROUTINE CONTZQ (NP,NOPT2,-NOPT3,METHOD,K,XR,YR,PV,UR,VR,IC,XV,YV)

The argument NP is the number of roots to be found. The negative value of the graph option NOPT3 instructs CONTZQ to look for the required data P1,X1,Y1,P2,X2,Y2,ΔX,ΔY,ΔU, ΔV in the first ten elements of UR.

It frequently happens that ZDSETQ or CONSTQ has to be CALLED by the users GPDUMQ after GPDUMQ has been already CALLED by the same routine. This is a recursive CALL and, whilst it works quite satisfactorily, control cannot then be returned to ZDSETQ or CONTZQ at the original point from which GPDUMQ was CALLED. Thus, in these circumstances, the user's GPDUMQ acts as the 'main program' and further operations can only be made with a 'CALL' statement. 'CALL CEVITQ' starts all operations afresh from square one!

Finally, graphical output is executed by

SUBROUTINE GRAFBQ (XR,YR,PV,NP,I)

I = 1 Standard Captions

I = 2 Captions read and printed by GRAFBQ

It is to be hoped that as problems are successfully solved with CEVITQ, users will be able to add to the facilities and techniques for solving complex eigenvalue problems, in a generally useful way.

The complete list of Subroutine names used by program CEVITQ is: CEVITQ, CBASEQ, GPINTQ, INDEXQ, CRITIQ and associated routines, ZDSETQ, CONTZQ, CMPLXQ, GPDUMQ, GRAFBQ. The STRETCH version also uses the Harwell Formatless Reading routines NAO1A/BS. If the GRAFBQ which draws graphs is loaded, then SKAPTQ, SYMBOQ and the general purpose graph plotting routines are also called.

IX. A WORKED EXAMPLE

The example chosen is the first one discussed at the end of Part I. To solve equations (2), (3) I, for ak over a range of PNU $\equiv \nu/\Omega_c$ between 0.05 and 0.25 for a value of WWO $\equiv \omega/\omega_o$ of 5.0 the computer input (KDF9) might be as follows.

```
            Job Card
            *XEQ
            *IODTAPE 100/IBM/SAVE
            *CHAIN 1
            *FORTRAN
            SUBROUTINE FUNXNJ (AKR,AKI,DR,DI,K)
            COMMON WWO, PNU
            Fortran to define the real and imaginary parts of the dispersion function.
            DR =
            DI =
            RETURN
            END
            *FORTRAN
            SUBROUTINE GPDUMQ (XV,YV,MFAIL,K,IG,XR,YR,PV,UR,VR)
            DIMENSION BZM(100), BZP(100), RAD(100)
            J = IG + 2
            GO TO (1,2,3,2), J
          1 PRINT 100
        100 FORMAT (1H1/20X, 13HJOB FOR DR.X /)
          2 RETURN
          3 GO TO (5,5,2), MFAIL
          5 CALL BZCALC (BFP,BZM,RAD, ...)
C               Calculates B_z field as a function of radius etc.
            RETURN
            END
*DATA
            4 2 1 3 1
            1                       1 parameter
            1
            5.0                     WWO
            0                       No Integers
            .05   1.2   0.5
           0.25   1.2   2.0
           0.2    0.2     1.0E-6    1.0E-6      Guesses etc.
    29      02    84       DR.X     HELICON DAMPING RATES.          1
         $ HELICONS   $ PNU     $ AKR     $ AKI
```

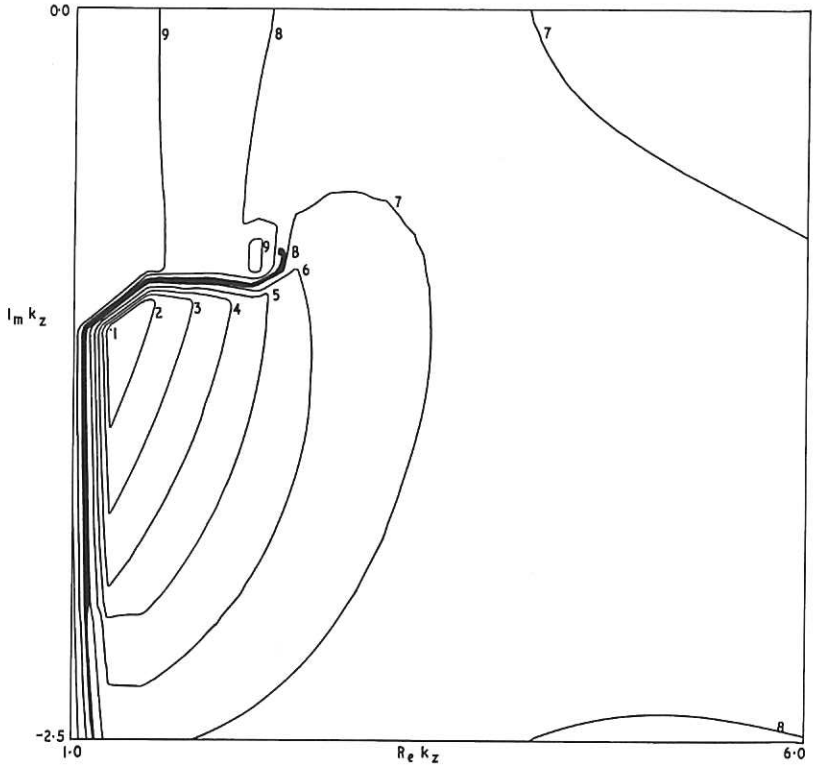

Fig. 7(a) (CLM-R 48)

ω in the k_z plane. $R_e\omega$ contours of $\omega(k_z)$. Equally spaced contours between 3.6 and 8.6

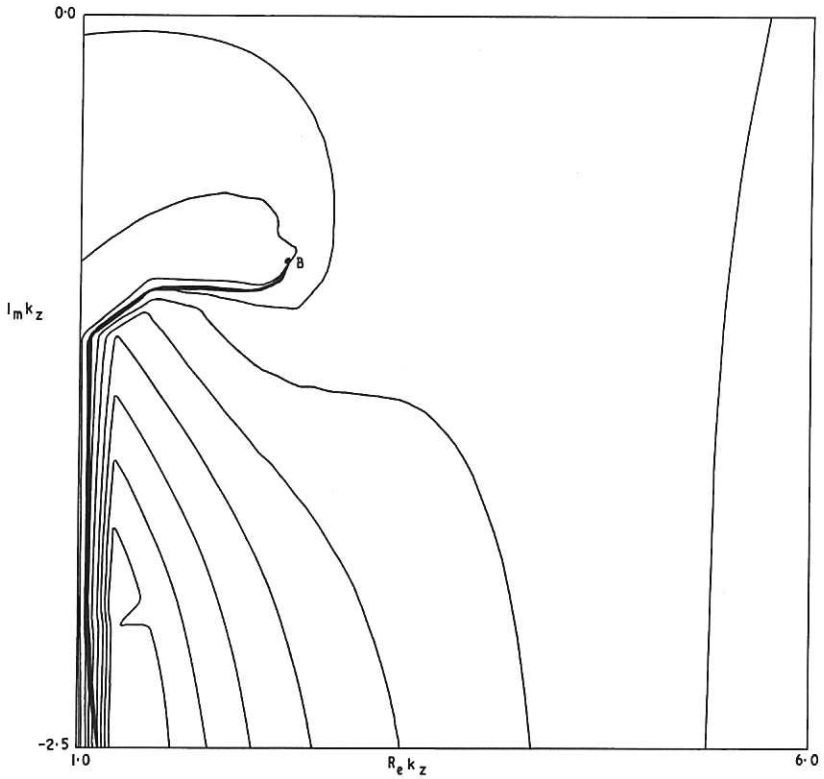

Fig. 7(b) (CLM-R 48)

ω in the k_z plane. $I_m\omega$ contours of $\omega(k_z)$. Equally spaced contours between 2.3 and -2.7

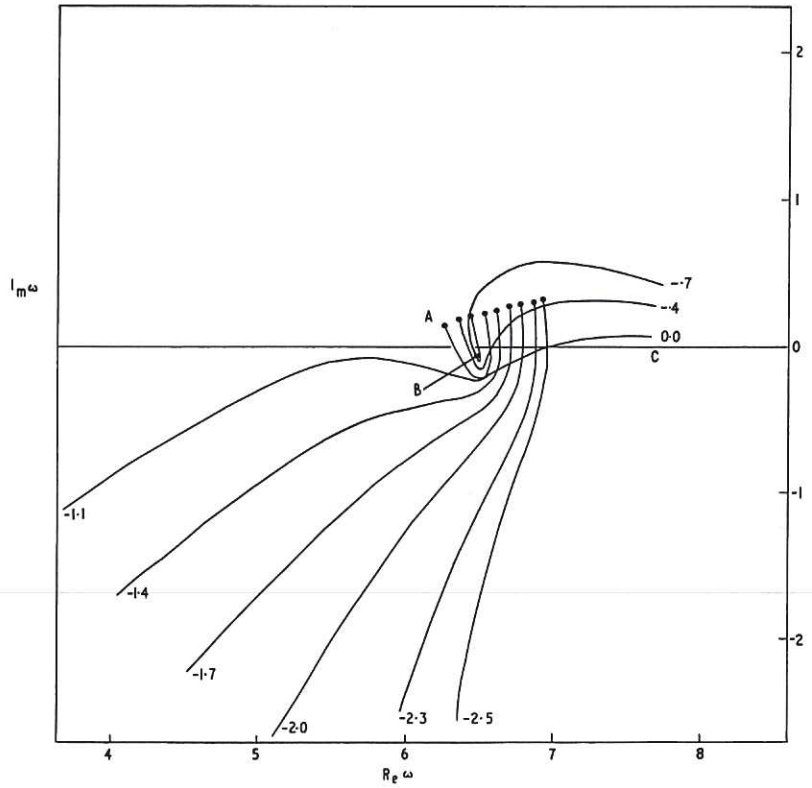

Fig. 8(a) (CLM-R 48)
k_z in the ω plane. $I_m k_z$ contours of $k_z(\omega)$

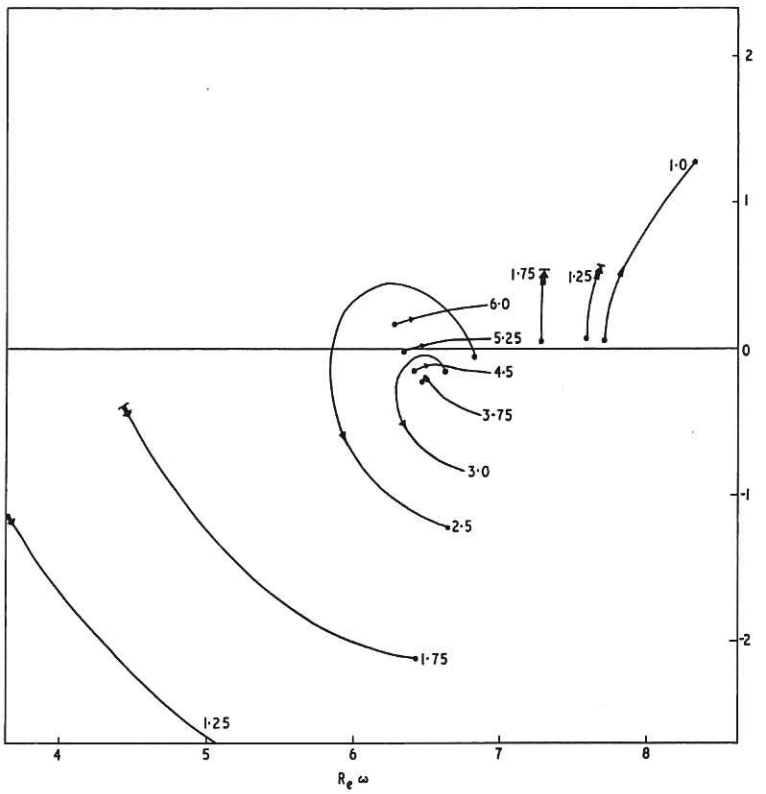

Fig. 8(b) (CLM-R 48)
k_z in the ω plane. $R_e k_z$ contours of $k_z(\omega)$

Available from

HER MAJESTY'S STATIONERY OFFICE

49 High Holborn, London, W.C.1
423 Oxford Street, London W.1
13a Castle Street, Edinburgh 2
109 St. Mary Street, Cardiff
Brazennose Street, Manchester 2
50 Fairfax Street, Bristol 1
35 Smallbrook, Ringway, Birmingham 5
80 Chichester Street, Belfast

or through any bookseller.

Printed in England

S. O. Code No. 91-3-18-71