Mathematics Notes

Note 42

March 1976

Further Developments in the Application of Contour Integration to the Evaluation of the Zeros of Analytic Functions and Relevant Computer Programs

Bharadwaja K. Singaraju
D. V. Giri
and
Carl E. Baum

Air Force Weapons Laboratory

Abstract

This report discusses a procedure for finding the zeros of an analytic function. Cauchy's theorem is used to find the number of zeros in a contour and a simple extention of this theorem yields relationships involving the locations of the zeros. A simple procedure involving integration by parts is shown to simplify the equations involved yielding those that can be more accurately numerically integrated. Three automated computer subroutines are discussed for numerical evaluation of the locations of the zeros. Numerical examples involving the roots of a 20th order polynomial, 20th order polynomial times an exponential and the natural frequencies of a thin wire are also discussed.

I. Introduction

Since the advent of the SEM (Singularity Expansion Method)[1] various techniques have been investigated to find the zeros of a complex function. Most of these techniques have been based on iterative methods such as Muller and Newton-Raphson.[2] The major difficulties of using such iterative techniques are twofold: (1) they require knowledge of the approximate zero of the complex function; (2) they are unreliable if the complex function has an exponential behavior. Although the general problem of SEM is very complicated, general principles of complex analysis apply to the SEM. Since our motivation in developing the present method has been SEM, we will briefly discuss this method.

For objects of general interest, the electromagnetic response can be written as the solution of an integral equation[3] written as

$$\left\langle \tilde{\vec{\vec{\Gamma}}}(s) \,;\, \tilde{\vec{U}}(s) \right\rangle = \tilde{\vec{I}}(s) \tag{1.1}$$

where ~ (tilde) denotes a bilateral Laplace transformed quantity. $\tilde{\vec{\vec{\Gamma}}}$ is the dyadic kernel, $\tilde{\vec{U}}(s)$ the normalized delta function response of the object, $\tilde{\vec{I}}(s)$ the forcing function appropriately normalized, and the notation $\langle ; \rangle$ indicates a dot product between dyadic-vector quantities along with the integration over common coordinates. Natural frequencies s_α of the scatterer are solutions of the homogeneous equation

$$\left\langle \tilde{\vec{\vec{\Gamma}}}(s) \,;\, \tilde{\vec{\nu}}_\alpha \right\rangle = \vec{0} \tag{1.2}$$

or

$$\left\langle \tilde{\vec{\mu}}_\alpha \,;\, \tilde{\vec{\vec{\Gamma}}}(s) \right\rangle = \vec{0} \tag{1.3}$$

where $\tilde{\vec{\nu}}_\alpha$ and $\tilde{\vec{\mu}}_\alpha$ are the natural mode and coupling vector, respectively, at the natural frequency. Natural frequencies can then be defined as the complex frequencies at which the response $\tilde{\vec{U}}(s)$ has a pole assuming that the forcing function $\tilde{\vec{I}}(s)$ does not have a pole at the same frequency.

Since in most cases of interest no branch points have been observed in the complex s plane we will not deal with this occurrence. Branch points in the complex plane do not cause any problems as long as these are not encircled by the contour.

There are some simple structures such as thin wires, spheres and prolate spheroids for which the natural frequencies and natural modes can be computed analytically.[4,5,6] However, for most cases of interest, one resorts to the well-known Moment Method (MoM). In the MoM formulation (1.1) can be written as

$$(\tilde{\Gamma}_{n,m}(s)) \cdot (\tilde{U}_m(s)) = (\tilde{I}_n(s)) \qquad n,m = 1,2,3,\ldots,N \tag{1.4}$$

and the response $(\tilde{U}_m(s))$ can be written as

$$(\tilde{U}_m(s)) = (\tilde{\Gamma}_{n,m}(s))^{-1} \cdot (\tilde{I}_n(s)) \tag{1.5}$$

In (1.5), the response function has a pole in the complex frequency plane at $s = s_\alpha$ whenever the determinant of $\tilde{\Gamma}_{n,m}(s_\alpha)$ $(\det \tilde{\Gamma}_{n,m}(s_\alpha))$ is zero subject to the condition that the forcing function does not have a pole or a zero at $s = s_\alpha$. In general, one can then find the natural frequencies of a given scatterer by matricizing the corresponding integral operator and finding the points (s_α) in the complex s plane where the determinant is zero. In this respect, the problem of finding the zero of a complex function or that of a determinant become the same problem; as such the development in this report can be used in finding the zeros of any analytic function.

In principle, our method resembles that used by Delves and Lyness[7,8] and Beasley and Meier.[9] However, the differences are too numerous to enumerate here. In some respects the subroutines described here may not be as sophisticated as that of Delves and Lyness. It will become clear in later sections why such elaborate criteria are not necessary in our method.

We have in general assumed that those who will use these subroutines will use a normalized complex plane. In the case of SEM, suitable normalizations may be of the form $S = s\ell/\pi c$ or $S = s\ell/c$ where s is the complex radian frequency, c the velocity of light in free space while ℓ is proportional (proportionality constant of one being perfectly acceptable) to the long dimension of the scatterer.

II. General Theory

If a function f(z) is meromorphic in a simply connected domain D containing a Jordan contour C, there are well-known methods to find the number of zeros minus the number of poles in this contour C and relationships involving their locations can also be found.[10] In this section we will discuss these techniques in detail and catalogue various formulae.

A. Principle of the argument

Consider a function f(z) which is meromorphic in a domain D. We draw a Jordan contour C in D such that no zeros or poles of f(z) lie on the contour. In simple problems involving analytic solutions, it is easy to draw these contours. However, in an automated numerical technique, some test criteria may have to be found which detect a zero or a pole on the contour. For reasons which become clear later, we will not treat the case of a zero or a pole on the contour. Assume that the contour C has been drawn properly and let N_o represent the number of zeros of f(z) in C while N_p represents the poles of f(z) in C counted according to their multiplicity. Then

$$\frac{1}{2\pi i}\oint_C \frac{f'(z)}{f(z)}\,dz = N_o - N_p \tag{2.1}$$

where a prime superscript denotes a derivative with respect to z, and the counterclockwise direction of travel around the contour is important. We can also write (2.1) as

$$\frac{1}{2\pi i}\oint_C \frac{d}{dz}(\ln f(z))\,dz = N_o - N_p \tag{2.2}$$

or taking the principal branch of $\ln[f(z)]$,

$$\frac{1}{2\pi} \arg(f(z)) \oint_C = N_o - N_p \equiv N_a \qquad (2.3)$$

This is known as the principle of the argument and N_a the argument number. The principle of the argument suggests that the argument number is equal to the number of times the argument of f(z) winds around some reference axis in the complex plane. This axis corresponds to some chosen point on the contour.

If the contour C contains only zeros of f(z) or alternatively only poles of f(z), (2.1) through (2.3) can be simplified by dropping the term N_p or N_o depending upon whether the contour contains zeros or poles. Since poles of f(z) can be considered to be zeros of $F(z) = f^{-1}(z)$, we will confine our discussion to the zeros of an analytic function f(z). If a meromorphic function G(z) is given, we assume that the singularities of G(z) have been suitably determined and G(z) has been converted into an analytic function of z.

B. Location of the zeros

Consider a complex function g(z) analytic in the domain D. If the zeros (poles) of f(z) described in section A occur at z_{o_i}, using the residue theorem we can write

$$\frac{1}{2\pi i} \oint \frac{f'(z)}{f(z)} g(z)\,dz = \pm \sum_{i=1}^{n} g(z_{o_i}) \qquad (2.4)$$

where z_{o_i} is the location of the ith zero (pole) of f(z) in C, and n is the total number of zeros (poles) in C. Here zeros (poles) are counted according to their multiplicity. The + sign in (2.4) is appropriate if f(z) has only zeros in C while the - sign is applicable if f(z) has only poles in C. Remembering that a pole can be treated as the negative of the zero, we need to consider the case of zeros of f(z) alone.

In (2.4) we are free to choose any analytic function g(z) and we select

$$g(z) = z^k \qquad k = 0, 1, 2, \ldots, N_o \tag{2.5}$$

N_o being the number of zeros of f(z) in C. Representing the left-hand side of (2.4) for each k by I_k, we have

$$\begin{aligned}
&z_{o_1} + z_{o_2} + \cdots + z_{o_{N_o}} = I_1 \\
&z_{o_1}^2 + z_{o_2}^2 + \cdots + z_{o_{N_o}}^2 = I_2 \\
&\vdots \\
&z_{o_1}^{N_o} + z_{o_2}^{N_o} + \cdots + z_{o_{N_o}}^{N_o} = I_{N_o}
\end{aligned} \tag{2.6}$$

where

$$I_k \equiv \frac{1}{2\pi i} \oint_C z^k \frac{f'(z)}{f(z)}\, dz \qquad k = 0, 1, 2, \ldots, N_o \tag{2.7}$$

We note that if we set k = 0 in (2.7) we obtain (2.1). Assuming that I_k can be evaluated for a given analytic function, the system of nonlinear equations given by (2.6) can be solved to obtain the locations of the zeros z_{o_i} for each contour.

The procedure described above, however, assumes that the number of zeros in a contour are known accurately. This knowledge of number of zeros is critical since this determines the number of equations in the system given by (2.6). As a consequence, the procedure used for determining the number of zeros in a contour should be very reliable.

Returning to (2.7), there are various ways of expressing (2.7). It can be written as

$$I_k = \frac{1}{2\pi i} \oint_C z^k \frac{1}{f(z)} \frac{d}{dz} f(z)\, dz \tag{2.8}$$

$$= \frac{1}{2\pi i} \oint_C z^k \frac{1}{f(z)}\, d(f(z)) \tag{2.9}$$

$$= \frac{1}{2\pi i} \oint_C z^k \frac{d}{dz} (\ln f(z))\, dz \tag{2.10}$$

$$= \frac{1}{2\pi i} \oint_C z^k\, d(\ln f(z)) \tag{2.11}$$

where I_k can be considered as the kth moment of the logarithmic derivative. There are several ways of evaluating integrals of the form I_k. We will discuss two such procedures here and the readers are referred to earlier works for others.[7,8,9]

C. Evaluation of I_k by an approximation of the logarithmic derivative

In this procedure the integral representation of the form

$$I_k = \frac{1}{2\pi i} \oint_C z^k \frac{1}{f(z)} \frac{d}{dz} f(z)\, dz \tag{2.12}$$

is used. Following the procedure used by Baum,[11] we can write (2.12) as

$$I_k \simeq \frac{1}{2\pi i} \sum_C z^k \frac{\Delta(f(z))}{f(z)} \tag{2.13}$$

In the first choice of approximation we can write

$$\Delta(f(z)) = f(z_{m+1}) - f(z_m) \tag{2.14}$$

$$z^{-k} f(z) = \frac{1}{2}\left[z_{m+1}^{-k} f(z_{m+1}) + z_m^{-k} f(z_m)\right] \tag{2.15}$$

and

$$\frac{1}{\pi i}\sum_{m=1}^{M}\frac{f(z_{m+1}) - f(z_m)}{z_{m+1}^{-k} f(z_{m+1}) + z_m^{-k} f(z_m)} = \sum_{j=1}^{N_o} z_{o_j}^{k} \qquad k = 0,1,2,\ldots,N_o \tag{2.16}$$

Here M represents the number of points around the contour with $z_{m+1} = z_1$ and $f(z_{m+1}) = f(z_1)$.

The second choice of approximations is

$$\Delta(f(z)) = \frac{1}{2}\left[f(z_{m+1}) - f(z_{m-1})\right] \tag{2.17}$$

$$f(z) = f(z_m) \tag{2.18}$$

and

$$\frac{1}{4\pi i}\sum_{m=1}^{M} z_m^{k}\frac{f(z_{m+1}) - f(z_{m-1})}{f(z_m)} = \sum_{j=1}^{N_o} z_{o_j}^{k} \qquad k = 0,1,2,\ldots,N_o \tag{2.19}$$

If k = 0 the real part of (2.16) or (2.19) rounded off to the nearest integer represents the number of zeros in C while other values of k from 1 to N_o yield a system of equations relating the position of the zeros in the contour C. In the representations in (2.16) and (2.19), the derivative is approximated and as a consequence, the accuracy of this procedure is strongly dependent upon the number of points M. For numerical implementation this procedure was found to be inefficient.

D. Evaluation of I_k by integration by parts

The integral of the form (2.10) is the mainstay in this procedure and as a consequence we will discuss it here in detail. Rewriting it here for convenience,

$$I_k = \frac{1}{2\pi i}\oint_C z^k \frac{d}{dz}\left(\ln(f(z))\right) dz \tag{2.20}$$

$$\frac{d}{dz}\ln\left(f(z)\right) = \frac{d}{dz}\left[\ln\left(|f(z)|\right) + i\arg\left(f(z)\right)\right] \tag{2.21}$$

If in a given domain D we draw a Jordan contour C, and if f(z) has no zeros in that contour, $\ln(f(z))$ is a continuous function of z on that contour. Hence the kth moment of the logarithmic derivative of f(z) around a Jordan contour, where the contour does not contain any zeros of f(z) is zero. However, if the contour does contain zeros, the logarithm of f(z) contains a branch point at each of the zeros of f(z) in that contour. We recall from basic complex variable theory that $\ln(|f(z)|)$ is a continuous function of z while $\arg(f(z))$ is discontinuous as one traverses around the contour, provided f(z) has zeros in the contour. It is clear from Rieman surfaces that $\arg(f(z))$ can be made into a continuous function of z except at the end points of the contour. Hence we can make $\arg(f(z))$ a continuous function and represent it by $\text{adj}(\arg(f(z))$ where the symbol adj stands for adjusted to mean that the argument of f(z) has been adjusted such that it is a continuous function of z around the contour. We will discuss the process of adjusting the argument of f(z) in the numerical section.

Rewriting (2.20) via (2.21) and the adjusted argument of f(z)

$$I_k = \frac{1}{2\pi i} \oint_C z^k \left[\frac{d}{dz} \ln(|f(z)|) + i \frac{d}{dz} \left\{ \text{adj}\left(\arg(f(z))\right) \right\} \right] dz \qquad (2.22)$$

where the $\text{adj}(\arg f(z))$ is a continuous function except at the end points of the contour. Integrating (2.22) by parts and using the fact that $\ln|f(z)|$ is a continuous function yields

$$I_k = \frac{z^k}{2\pi} \left\{ \text{adj}\left(\arg(f(z))\right) \right\} \oint_{C_{ini}} - \frac{k}{2\pi i} \oint_{C_{ini}} z^{k-1} \left[\ln(|f(z)|) + i\,\text{adj}\left(\arg(f(z))\right) \right] dz \qquad (2.23)$$

were C_{ini} represents that the contour starting and end points are taken as z_{ini}. We can simplify (2.23) as

$$I_k = N_o z_{ini}^k - \frac{k}{2\pi i} \oint_{C_{ini}} z^{k-1} \left[\ln|f(z)| + i\,\mathrm{adj}\left(\arg\left(f(z)\right)\right) \right] dz \qquad (2.24)$$

where N_o is the number of zeros in the contour C.

The procedure described above is very different from those the authors are familiar with and does not involve any approximations such as polynomial fit or approximations in representing the logarithmic derivative. If the complex function f(z) or its value is known around the contour, we can evaluate (2.24) and obtain the system of equations given by

$$\sum_{j=1}^{N_o} z_{o_j}^k = N_o z_{ini}^k - \frac{k}{2\pi i} \oint_{C_{ini}} z^{k-1} \left[\ln\left(|f(z)|\right) + i\,\mathrm{adj}\left(\arg\left(f(z)\right)\right) \right] dz \qquad (2.25)$$

$$k = 1, 2, \ldots, N_o$$

where

$$N_o = \frac{1}{2\pi} \left\{ \mathrm{adj}\left(\arg\left(f(z)\right)\right) \right\} \oint_{C_{ini}} = \text{number of zeros in the contour} \qquad (2.26)$$

Under the present formulation, it is necessary that this initial point be the same for the remaining integral in (2.25). As can be seen from (2.25) and (2.26) if we know the logarithmic magnitude and the adjusted argument of f(z) around the contour, we can determine the number of zeros and the locations of the zeros of the analytic function f(z) within the contour.

III. Numerical Evaluation of the Integrals

If the analytic function f(z) is given, the number of zeros of this function and their locations within a given contour can be determined by (2.25) and (2.26). Determination of the number of zeros of f(z) in a given contour is made purely by the phase information while the relationships involving the locations of the zeros contain integrations.

A. Finding the number of zeros in a contour

The number of zeros of an analytic function f(z) in a contour C is determined by the phase information. Let us consider an arbitrary contour C as shown in figure 3.1. The arrow on the contour shows the positive direction of travel around the contour. The contour is divided into M not necessarily equal parts and the dividing points are numbered 1 through M in an increasing order with the counterclockwise direction being the positive direction of travel around the contour. The complex function f(z) is evaluated at these n points and its argument arg (f(z)) is found such that it is between 0 and 2π. We now perform a phase test to adjust the argument f(z). This test is performed as follows: consider the ith and i+1st points on the contour. Suppose that arg (f(z)) ($0 \leq \arg(f(z_i)) < 2\pi$) is in the first quadrant while $\arg(f(z_{i+1}))$ is in the fourth quadrant, the phase is taken to have changed by -2π between these two points and $\arg(f(z_{i+1}))$ is adjusted such that

$$\text{adj}\left(\arg\left(f(z_{i+1})\right)\right) = -2\pi + \arg\left(f(z_{i+1})\right) \tag{3.1}$$

Alternatively, if $\arg(f(z_i))$ is in the fourth quadrant while $\arg(f(z_{i+1}))$ is in the first quadrant, the phase is taken to have changed by $+2\pi$ between the points and

$$\text{adj}\left(\arg\left(f(z_{i+1})\right)\right) = 2\pi + \arg\left(f(z_{i+1})\right) \tag{3.2}$$

Adjustments of the form (3.1) or (3.2) are performed on all points starting with i+1 up to i-1 along with any other adjustments necessary because of

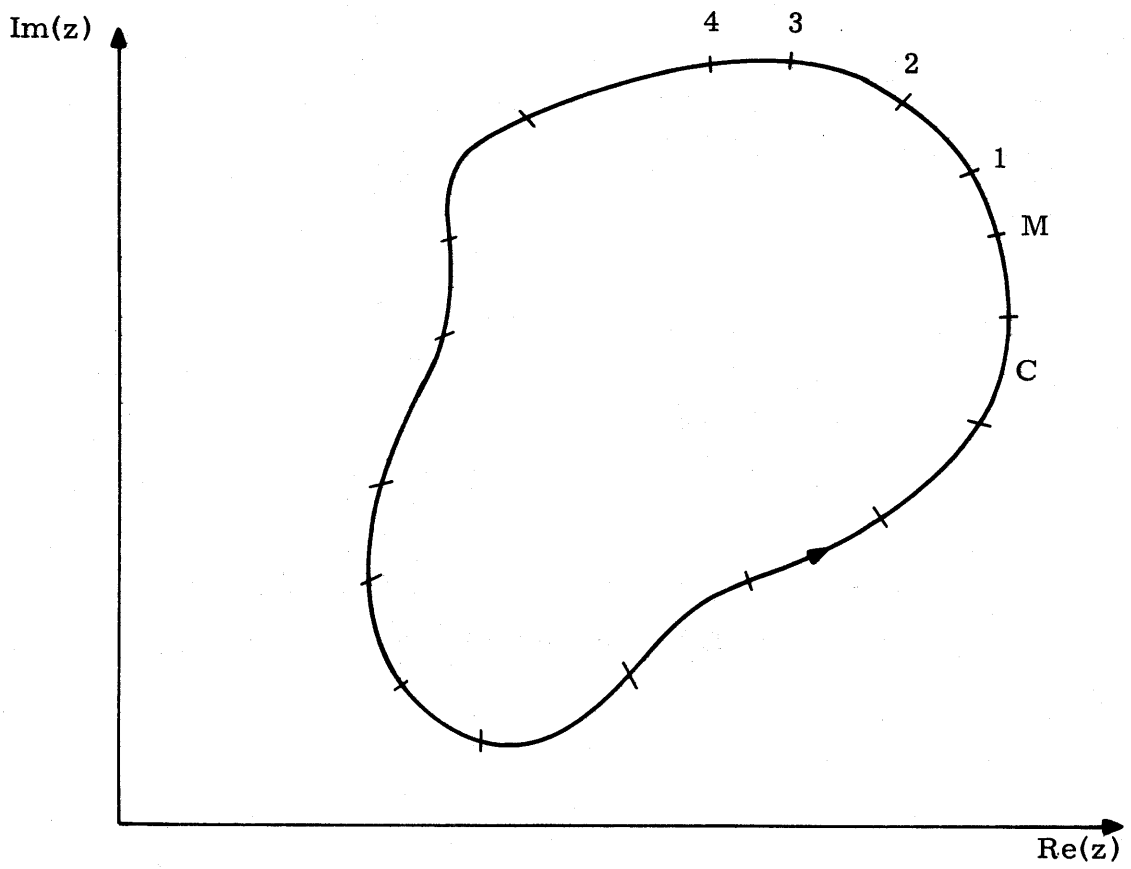

Figure 3.1. Arbitrary closed contour in the complex plane

the tests made on subsequent points on the contour. If for a given contour N_+ represents the number of times the phase crossed the 2π line going in the positive direction and N_- the negative direction, $(N_+ - N_-)$, if greater than zero denotes the number of zeros in the contour while, if negative, the poles in the contour, assuming that f(z) has only poles or zeros in C. If $(N_+ - N_-)$ is zero no zeros or poles of f(z) exist in C. The test procedure as described here assumes that between two adjacent points, the phase difference is no greater than $\pi/2$. Although this might appear to be a bad assumption, in extensive tests performed no problems have been observed. As will be seen later, the procedure has performed well even when a zero is on the contour.

One could construct an alternative test procedure as follows: consider $\arg(f(z_i))$, $\arg(f(z_{i+1}))$ and $\arg(f(z_{i-1}))$. A phase test is performed between $\arg(f(z_{i+1}))$ and $\arg(f(z_{i-1})$ as described above. Phase information of $\arg(f(z_i))$ is used to observe the direction of rotation of the phasor between $f(z_{i-1})$ and $f(z_{i+1})$ and hence the argument number. Another alternative procedure could be to pick a point between two adjacent points, if the phase difference exceeds $\pi/2$ between the original points. One could use the phase information of this new point to obtain the direction of rotation.

B. Finding the locations of the zeros (poles)

Let us assume for now that f(z) is an analytic function and hence contains only zeros within the contour C. The relationships involving the locations of the zeros is given by

$$\sum_{j=1}^{N_o} z_{o_j}^k = N_o z_{ini}^k - \frac{k}{2\pi i} \oint_C z^{k-1}\left[\ln|f(z)| + i\,\mathrm{adj}\left(\arg\left(f(z)\right)\right)\right]dz \qquad (3.3)$$

Since $\ln\left(|f(z)|\right)$ and $\mathrm{adj}\left(\arg(f(z))\right)$ are continuous functions of z, the integral in (3.3) is well behaved and the numerical integration is easy to perform. Although trapezoidal or Simpson's method may be used to

perform the integration, because of better convergent properties Gaussian quadrature formula was found to be better suited. In this procedure, the contour C can be broken up into M number of not necessarily equal segments as shown in figure 3.2. Each of the segments of this contour is broken up according to the order of Gaussian quadrature formula used. If we assume this order to be q, (3.3) can be written as[12]

$$\sum_{j=1}^{N_o} z_{o_j}^k \simeq N_o z_{ini}^k - \frac{k}{2\pi i}\left[\sum_{m=1}^{M} \frac{z_m - z_{m-1}}{2} \sum_{\ell=1}^{q} z_\ell^{k-1} w_\ell p(z_\ell)\right] \tag{3.4}$$

where

$$p(z_\ell) = \left[\ln|f(z_\ell)| + i\,\mathrm{adj}\left(\arg\left(f(z_\ell)\right)\right)\right] \tag{3.5}$$

$$z_\ell = \left(\frac{z_n - z_{n-1}}{2}\right) x_\ell + \left(\frac{z_n + z_{n-1}}{2}\right) \tag{3.6}$$

with x_ℓ being the location of the ℓth zero of qth order Legendre function while w_ℓ is the corresponding weight function. One could impose a convergence criterion and increase M until that criterion is met. We will discuss this procedure later.

As discussed earlier, if the contour C contains N_o number of zeros, one would obtain N_o equations relating the locations of the zeros which can be manipulated into an equation of the form

$$z_o^{N_o} + c_1 z_o^{N_o-1} + c_2 z_o^{N_o-2} + \cdots + c_{N_o} = 0 \tag{3.7}$$

whose roots are the locations of the zeros of f(z) in C. In general c_i in (3.7) is complex and analytical methods are known only up to a 4th order equation.[13] Computer routines discussed in later sections cannot find the locations of the zeros if more than three (3) zeros are in a contour.

If the contour C contains poles of f(z) alone, by virtue of (2.4) one can obtain an equation similar to (3.7) whose roots yield the pole locations.

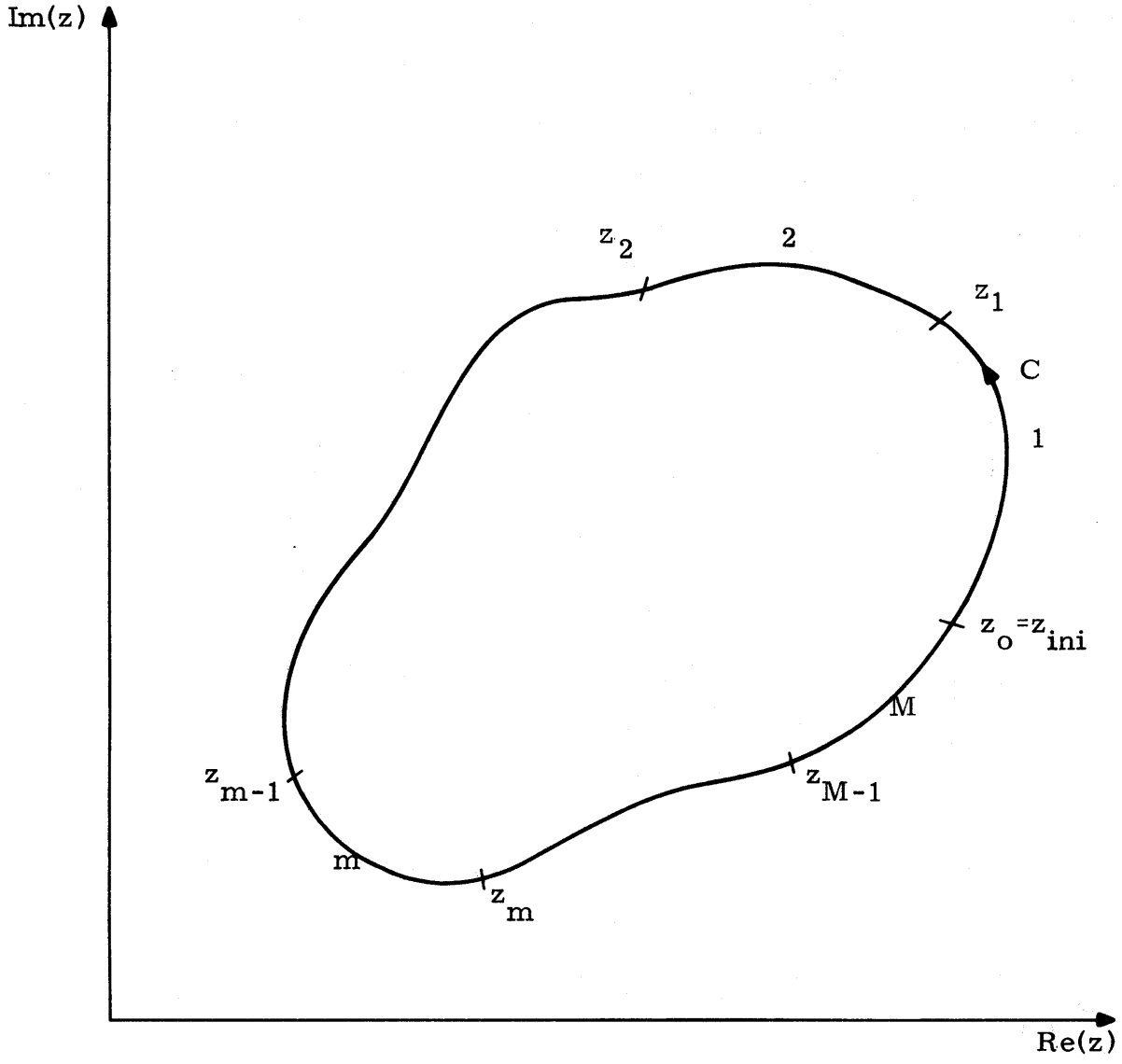

Figure 3.2. Division of a closed contour into segments

Alternatively, poles of f(z) are the zeros of $f^{-1}(z)$, hence, one can use (2.25) and (2.26) by substituting $f^{-1}(z)$ for f(z). If f(z) contains both poles and zeros in C, one would have to evaluate either the pole or zero locations analytically and remove the same from f(z); using the above discussed procedure one could then evaluate the pole or zero locations by using (2.25) and (2.26).

C. Exponential normalization procedure

If the complex function of interest f(z) has a complex exponential associated with it, the phase of f(z) would be varying very rapidly and the adjustment of the argument of f(z) becomes increasingly difficult. One way to counter this problem is to take increasingly larger number of points around the contour until the phase difference between two adjacent points on the contour is small. This procedure may not always be successful because of large magnitude variations and for problems involving large matrices such as moment method this becomes cost prohibitive; as a consequence, some other method has to be used.

Consider a general analytic function f(z) which has exponential behavior around the contour. We now normalize f(z) such that

$$F(z) = \frac{f(z)}{h(z)} \tag{3.8}$$

where h(z) is an entire function without any zeros within C. We choose h(z) to be of the form

$$h(z) = c\,e^{az} \tag{3.9}$$

where c and a are real constants so that h(z) is conjugate symmetric in the z plane ($\overline{h(z)} = h(\bar{z})$ where $-$ indicates conjugation). Now let us consider a rectangular contour C as shown in figure 3.3. On portions of the contour denoted by C_a and C_b the magnitude of h(z) is constant. We now chose c and a such that the average magnitude of h(z) is the same as that of f(z)

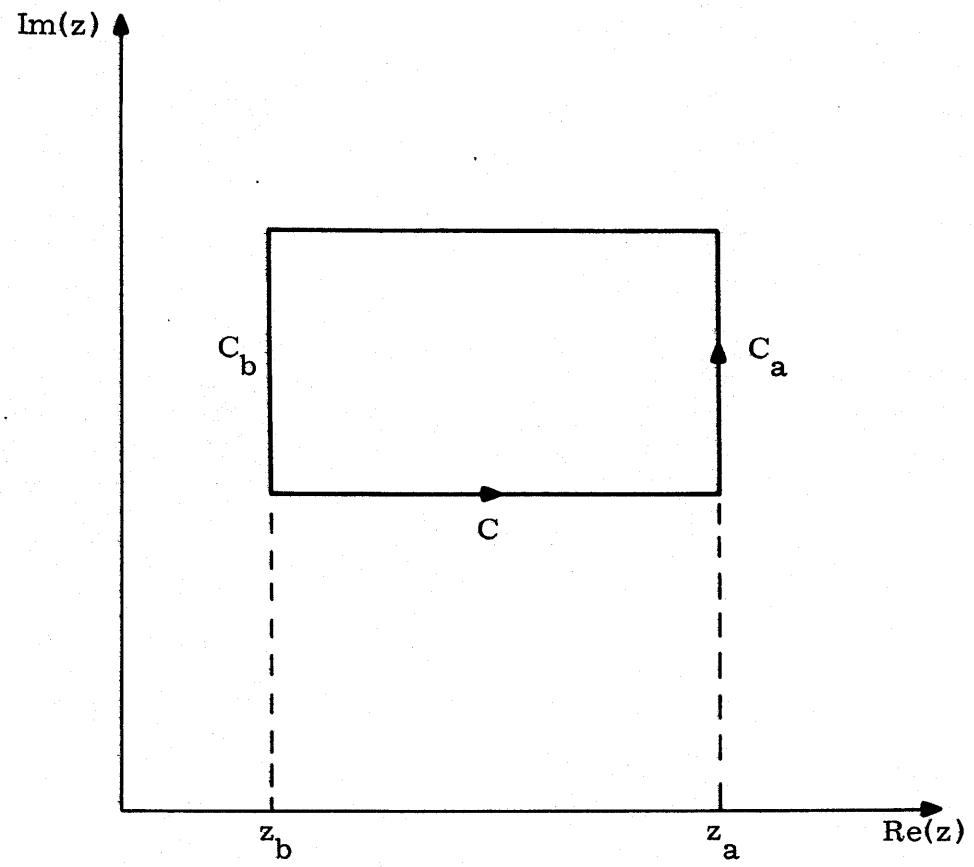

Figure 3.3. Rectangular contour in a complex plane

on C_a and C_b, the reason being, large phase variations, in general, accompany large variations in the magnitude. As such if the variations in the magnitudes are reduced, so are the phase variations. Hence we obtain

$$a = \frac{1}{z_a - z_b} \ln\left(\frac{\text{avg}\left(|f(z)|_{C_a}\right)}{\text{avg}\left(|f(z)|_{C_b}\right)}\right) \tag{3.10}$$

$$\begin{aligned} c &= e^{-az_a}\left[\text{avg}\left(|f(z)|_{C_a}\right)\right] \\ &= e^{-az_b}\left[\text{avg}\left(|f(z)|_{C_b}\right)\right] \end{aligned} \tag{3.11}$$

where avg represents average while the subscript C_a or C_b indicates the portion of the contour over which the average is taken. Using a and c given by (3.10) and (3.11), it is easy to show that the average magnitude of F(z) on C_a and C_b is 1. The argument of F(z) is given by

$$\begin{aligned} \arg\left(F(z)\right) &= \arg\left(f(z)\right) - \arg\left(e^{az}\right) \\ &= \arg\left(f(z)\right) - a\,\text{Im}(z) \end{aligned} \tag{3.12}$$

It has been observed that in general the average magnitude of F(z) around the contour is approximately 1, while rapid phase variations of f(z) do not appear in F(z). Since h(z) is an entire function without any zeros within the finite complex plane, the zeros (poles) of f(z) are undisturbed.

If the contour is circular as shown in figure 3.4, a and c are calculated using

$$a = \frac{1}{z_a - z_b} \ln\left(\frac{|f(z)|_{z_a}}{|f(z)|_{z_b}}\right) \tag{3.13}$$

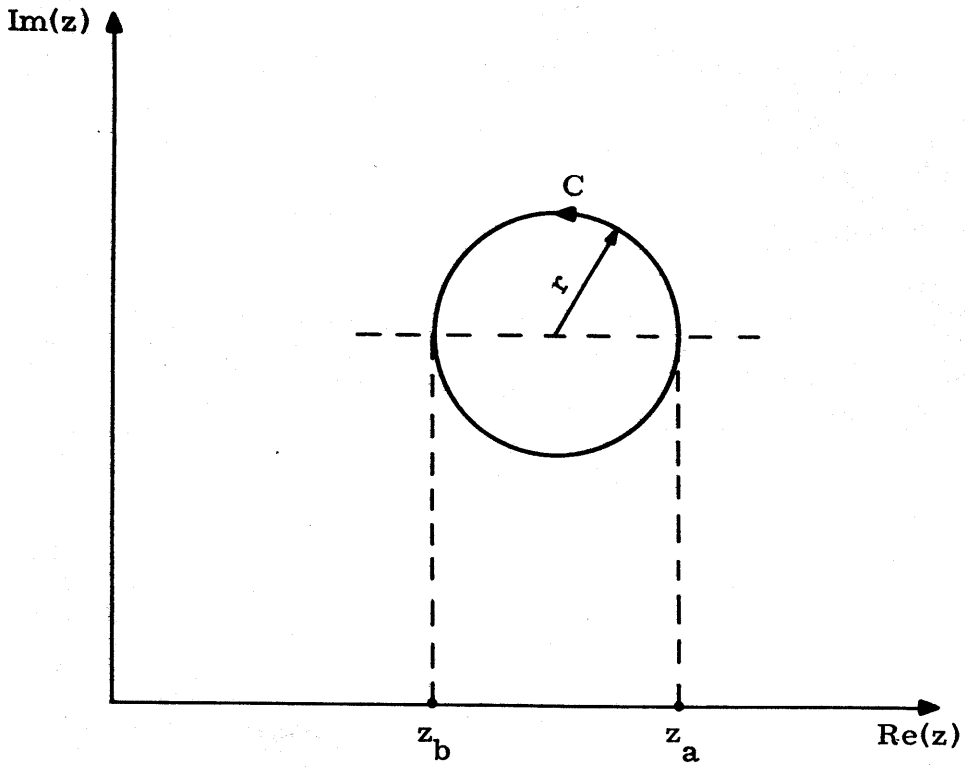

Figure 3.4. Circular contour in a complex plane

$$c = e^{-az_a} \frac{\left[|f(z)|_{z_a}\right]}{\left[|f(z)|_{z_b}\right]} \qquad (3.14)$$

where the subscript z_a or z_b indicates that the magnitude of f(z) is evaluated at those points with real parts z_a and z_b.

We have discussed here normalization procedures for rectangular and circular contours only because of ease of implementing these in numerical evaluation. However, the normalization procedure might be used for other contours also by using the method described for a circular contour.

IV. Description of the Computer Programs

Three computer subroutines have been written to facilitate numerical evaluation of the locations of the zeros of an analytic function f(z) in a given area of the complex plane. These subroutines have characteristics which make them suitable for different purposes. A brief description of these subroutines is given below and the listing for the subroutines is given in the appendix.

A. Subroutine SEARCH

If the analytic function f(z) is complicated and computationally expensive, SEARCH is a well suited subroutine for finding the zeros of f(z). Suppose that we wish to find the zeros of f(z) in some rectangular area of the complex plane called the scan area. In general, this scan area is divided into smaller rectangles and the zero locations of f(z), if any, are found in each of these rectangular areas called contours. Consider a rectangular scan area A enclosed by the contour C as shown in figure 4.1, this scan area can be uniquely described by the complex coordinates of the lower right corner (CSF) and those of the upper left corner (CSL). Let this scan area be divided into N_c smaller contours such that N_c = NR x NI where NR is the number of divisions c_1 is subdivided into while NI is the number of divisions c_2 is subdivided into as shown in figure 4.1. We note that two adjacent contours have one common side. As such, if the function values calculated on these common sides are stored, they can be used for its adjacent contour. Subroutine SEARCH stores these function values in two large arrays known as CV and CH.

Since the integration is done by Gaussian quadrature formula, once the order of Gauss' formula to be used is chosen, choices being 12, 20 or 40 order, the function f(z) is evaluated at proper points on the vertical and horizontal lines in C which subdivide the scan area and is stored in the vertical array CV and the horizontal array CH, respectively. Using the procedure described in section IV, the zeros of f(z), if any, and their

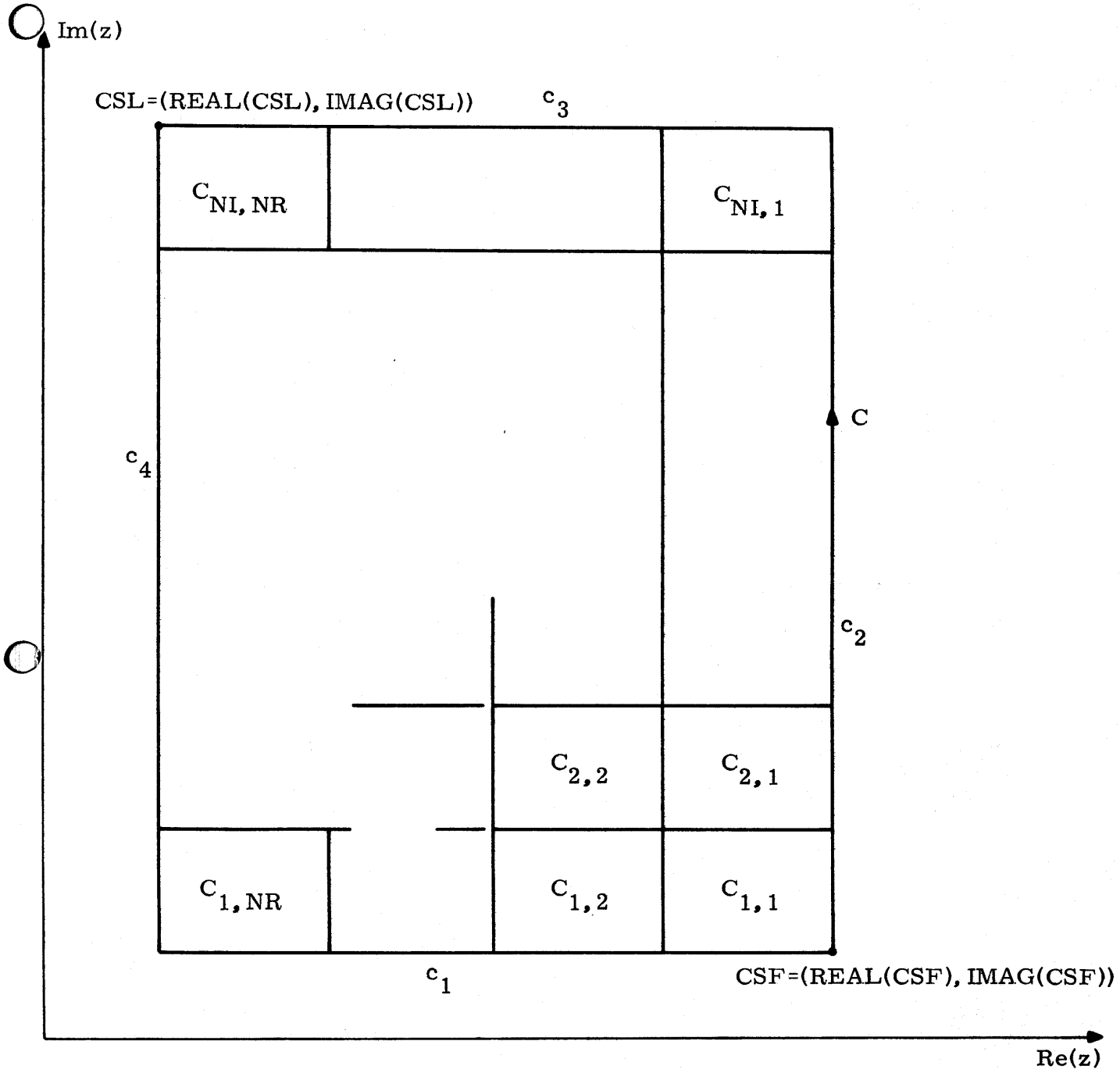

Figure 4.1. Division of a rectangular scan area into small rectangular contours

locations in each contour are found. If no zeros of f(z) exist in a particular contour, a message indicating that no zeros of f(z) exist in that contour is printed. If a pole of f(z) is detected by SEARCH in a contour, a message indicating that a pole of f(z) exists in that contour is printed; however, the location of the pole is not found. If more than three (3) zeros of f(z) are detected by SEARCH in a contour, a message indicating that the capabilities of SEARCH have been exceeded is printed. For each contour, the coordinates of the lower right-hand corner and the upper left corner are printed along with the number of poles or zeros in that contour detected by SEARCH are printed. If less than four (4) zeros of f(z) are detected in a contour, the locations of these zeros are found. A check is made to see if the zero location is inside the contour; if it is not, a warning message indicating that the zero is outside the contour is printed. If this occurs for a contour, it can be corrected by increasing the order of the Gaussian quadrature integration or by using subroutine SEEK (see section C). The average value of the magnitude of the function around the contour and the value of the magnitude of the function at the locations of the zeros is printed for each contour. A summary of results is printed for each scan area which includes the locations of the zeros, average value of the magnitude of the function around the contour in which the zero is located, magnitude of the function value at the zero and the ratio of the magnitude of the function at the zero to the magnitude of the average value of the function around the contour. The value of this ratio indicates the quality of the zero; the smaller the ratio the better is the zero location. If one desires to improve the results found by SEARCH, subroutine HOMEIN can be called by setting the proper flag. This subroutine is discussed in the next section. If the HOMEIN option is to be utilized, one would set NH = 1, otherwise NH = 0.

The header card of subroutine SEARCH reads as follows:

SUBROUTINE SEARCH (CFCTS, CSL, CSF, NR, NI, ND, NH, FACTOR)

where

CFCTS:	Complex function whose zero locations are to be found.
CSL:	Coordinates of the upper left corner of the scan area.
CSF:	Coordinates of the lower right corner of the scan area.
NR:	Number of major divisions of the real axis within the scan area.
NI:	Number of major divisions of the imaginary axis within the scan area.
ND:	Order of the Gaussian Quadrature formula to be used. Choices are 12, 20 and 40.
NH:	HOMEIN option indicator. If NH=0, HOMEIN subroutine is not called, if NH=1 it is called.
FACTOR:	This is factor by which you would expect the function value at the zero found by HOMEIN to be smaller than that found by SEARCH.
AVE:	Array in which the average value of the magnitude of the function around a contour is stored.
CH:	Array, used for storing common points (horizontal lines) between contours.
CV:	Array, used for storing common points (vertical lines) between contours.
CSA:	Array, used to store the zero locations found by SEARCH.
FUN:	Array, used to store the magnitude of the function values at the zero locations found by SEARCH.
X1, W1:	Arrays, used to store data for 12 point Gaussian Quadrature.
X2, W2:	Arrays, used to store data for 20 point Gaussian Quadrature.
X3, W3:	Arrays, used to store data for 40 point Gaussian Quadrature.
X, W:	Arrays, used to store the data for proper Gaussian Quadrature order specified.
ARG:	Array, used to store the argument of the function evaluated at proper points around the contour.
C:	Array, used to store the integrals.
CF:	Array, used to store the complex function values evaluated at proper points around the contour.

CS: Array in which the coordinates of the points around the contour are stored.

CSH: Array, used to store the zero location found by HOMEIN.

FZH: Array, used to store the magnitude of the function value at the zero location.

TMAG: Array, used to store the magnitude of the function at proper points around the contour.

PI: $= \pi \simeq 3.14159265$

A listing of this subroutine is provided in the appendix and numerical results are presented in section V.

B. Subroutine HOMEIN

This subroutine evaluates the location of the zero much more accurately than SEARCH. This subroutine can either be used in conjunction with SEARCH or by itself. This subroutine is very useful if the approximate locations of the zeros of an analytic function are known or if one wishes to improve the precision of the zero location obtained by SEARCH.

This subroutine takes the approximate location of the zero and the absolute function value at the zero as inputs. Approximate location of the zero is taken as the center of a circle and a circular contour of radius R is drawn in the normalized complex plane, where R is chosen according to the location of the center. For instance, if the center is within a radius R1 of 3, the initial radius of the contour is chosen as .08*R1 where R1 is the distance from the origin of the complex plane to the center. If $3 < R1 \leq 5$, R=.06*R1 and if $5 < R1 \leq 8$, R=.04*R1 and if R1>8, R=.03*R1. These radii have been chosen such that the contour will be small, motivation being, the smaller the contour, the more accurate the results.

Initially KDM*ND points are chosen around the contour C, where KDM=2 and ND=40 in this routine. As in the case of SEARCH, a phase test is performed to find if any zeros are enclosed in C. If no zero is detected, the radius of the contour is increased in multiples of 2 until zeros are detected in C or until the radius reaches ten times the initial

radius whichever occurs first. If no zero is detected within this final radius, a message stating this fact is printed. If zeros are detected in C, a test is made to see if there are less than 4 zeros in C. If it is found that more than 3 zeros are enclosed by C, the radius is decreased in steps of .9 until three zeros are enclosed or until the radius decreases to the minimum radius RMIN=.7*Initial radius, whichever occurs first. If at the minimum radius more than 3 zeros are detected in C, a message stating the number of zeros detected, the center of the contour, and the radius within which they are detected is printed. Experience indicates that this is a multiple zero of order greater than 3 and the user may choose to treat it as such.

If less than 3 zeros are detected in C, the locations of the zeros are evaluated as in SEARCH and the absolute value of the function is evaluated at the zero locations. The smallest of these function values (in the case of more than one zero in C) or the function value at the location of the zero found by HOMEIN (in the case of one zero in C) is compared to the function value at the approximate zero given to this subroutine. If this new function value is smaller than the original function value, the zero locations found by HOMEIN and the function values at these locations are printed. If this subroutine is used in conjunction with SEARCH, a summary of results is printed in SEARCH.

If the new function value is greater than the original function value inputted, the number of points taken around the contour is increased in multiples of 2 until a better zero is found or until the number of points are greater than KDMX*ND. If this limit has been exceeded, a message stating that HOMEIN could not find a better zero within the number of points specified is printed. Although the subroutine uses KDMX to be 10, the user may change this number by proportionally increasing the array sizes of TMAG, ARG, CS and CF.

The header card of subroutine HOMEIN reads as follows:

SUBROUTINE HOMEIN (IHH, CENTER, FUN, FACTOR)

If subroutine HOMEIN is used in conjunction with SEARCH, it suffices to specify the HOMEIN option as described in section A. If HOMEIN is used as an independent subroutine with the intention of improving the known locations of the zeros of the complex function CFCTS, the user may call HOMEIN in a do loop by feeding the approximate location of the zero as the center and the absolute value of the function at this location as FUN. IHH is a dummy variable which is equal to zero when HOMEIN is called for the first time. This variable is used as a counter to store the zero locations and the function values at the zero locations for purposes of summarizing in SEARCH. FACTOR is as described in SEARCH. This has to be inputted by the user. The larger this number, the better will be the zero found by HOMEIN.

IHH: Set equal to zero the first time HOMEIN is called from a do loop.

CENTER: Approximate location of the zero.

FUN: Absolute value of the function at the approximate zero.

CFCTS: Analytic function whose zero is to be improved.

PI=$\pi \approx 3.14159265$

TMAG: Array in which the magnitude of the complex function is stored.

ARG: Array in which the adjusted argument of the complex function is stored.

CS: Array in which the coordinates of the points around the contour are stored.

CF: Array in which the complex function values are stored.

C: Array in which various integrals are stored.

CSH: Array in which the zero locations found by HOMEIN are stored.

FZH: Array in which absolute values of the complex function values at the zero locations are stored.

X: Array in which Gaussian Quadrature integration data is stored.

W: Array in which Gaussian Quadrature integration data is stored.

A listing of this subroutine is provided in the appendix and numerical examples are discussed in section V.

C. Subroutine SEEK

The basic principle behind this subroutine is the same as SEARCH. Given the coordinates of the lower right corner and the upper left corner of a rectangular contour, this subroutine finds the zeros, if any, of a complex function. An added feature in this subroutine is a ratio test. It is clear that at the location of the zero, the function value is zero. However, numerically this will be a small number compared to the values of the function away from the location of the zero. We define a real number called ratio as the ratio of the absolute value of the function at the zero to the average of the absolute value of the function around the contour. Clearly a lower ratio at the zero location indicates a better zero. The user inputs the ratio criterion to be met, typical suggested values are 10^{-5} to 10^{-7}.

Given the coordinates of the contour and the ratio, SEEK, working the same way as SEARCH or HOMEIN, does a phase test to find if any zeros are enclosed. This is initially done by taking 160 points around the contour. If no zero is detected, a message indicating this fact is printed and the control is returned to the calling routine.

If less than four zeros are detected in the contour, their locations are found as in SEARCH and the absolute value of the function is evaluated at the zero locations. The ratio of the highest of these function values to the average of the absolute value of the function around the contour is calculated and compared to the ratio specified by the user. If the calculated ratio is smaller than the specified ratio, the zero location along with the magnitude of the function at the zero location and average value of the function around the contour are printed and the control returns to the calling program. If the ratio condition is not met, the number of points taken around the contour is increased in steps of 160 around the contour until the ratio condition is met or until the number of points around the

contour reaches KDMX*80. In the event the ratio condition could not be met, the last computation of the locations of the zeros, function values at these locations, etc. is printed along with a message indicating that the ratio condition could not be met.

If more than three zeros in a contour are detected by SEEK, the contour is subdivided into smaller contours according to the formula $N_c = 2\left(\frac{N_o}{3}+1\right)$ where N_o is the number of zeros detected in a contour and N_c is the integer part rounded off to the lower integer. For instance, if N_o is 4, the original contour is divided into 4 equal contours and the procedure described earlier is used to find the zeros in these smaller contours. If one or more of these smaller contours contain more than 3 zeros, a message is printed giving the coordinates of the contour, number of zeros detected and that the contour cannot be broken up any further.

The header card of SEEK reads as follows:

SUBROUTINE SEEK (CFCTS, CSM, CSMI, RATIO)

CFCTS: Complex function whose zeros are to be found.

CSM: Complex coordinates of the upper left corner of the contour.

CSMI: Complex coordinates of the lower right corner of the contour.

RATIO: Ratio condition which is to be satisfied.

PI=$\pi\approx$3.14159265

CSA: Array in which the locations of the zeros found are stored.

FUN: Array in which the magnitude of the function values at the zero locations is stored.

AVE: Array in which the average value of magnitude of the function around the contour is stored.

RAT: Array in which the ratio of the magnitude of the function at the zero location to the average value of the magnitude of the function around the contour is stored.

NOZ: Total number of zeros found.

ARG: Array, used to store the argument of the function evaluated at the proper points around the contour.

C: Array, used to store the integrals.

CF: Array, used to store the complex function values evaluated at proper points around the contour.

CS: Array in which the coordinates of the points around the contour are stored.

TMAG: Array, used to store the magnitude of the function at proper points around the contour.

X,W: Arrays, used to store data for Gaussian Quadrature.

Although subroutine SEEK can be used independently, it is primarily designed to be used with subroutine CONTOUR. Subroutine CONTOUR gives SEEK the capability of dividing a scan area into small contours as SEARCH is capable of doing. Although SEEK does not store common points between the contours, it has the capability of meeting certain crieria set by the user; in addition, if more than 3 zeros are found in a contour, SEEK has the capability of breaking the small contour into still smaller contours. The header card of subroutine CONTOUR reads as

SUBROUTINE CONTOUR (CSL, CSF, NR, NI, RATIO)

CSL, CSF, NR and NI are described in section A, while RATIO was described earlier in this section. This subroutine also summarizes the results found by SEEK.

Listings of CONTOUR and SEEK are provided in the appendix while numerical results are discussed in section V.

D. Function subroutine ANGLER

Subroutines SEARCH, HOMEIN and SEEK use ANGLER to calculate the angle in a continuous sense between 0 and 2π. This routine takes the real and imaginary parts of a complex function and returns the angle in radians between 0 and 2π. A listing of this subroutine is provided in the appendix.

V. Numerical Examples

In this section we will discuss two classes of examples: one concerning the roots of a polynomial and the other involves the natural frequencies of a thin wire.

A. Roots of a polynomial

A twentieth order polynomial f(s) as given by (5.1) is chosen for this numerical example

$$
\begin{aligned}
f(s) = {} & (s-(-.5+j1.0))(s-(-.4+j1.4))(s-(-.1+j2.6)) \\
& (s-(-.5+j2.7))(s-(-1.3+j1.8))(s-(-1.1+j3.95)) \\
& (s-(-2.4+j.1))(s-(-2.5+j.2))(s-(-2.6+j.3)) \\
& (s-(-2.1+j2.1))(s-(-2.1+j2.1))(s-(-2.2+j4.2)) \\
& (s-(-2.7+j4.8))(s-(-3.3+j1.6))(s-(-3.7+j3.8)) \\
& (s-(-4.25+j.35))(s-(-4.3+j2.3))(s-(-4.75+j2.75)) \\
& (s-(-4.3+j4.5))(s-(-4.95+j4.95)) \qquad (5.1)
\end{aligned}
$$

For purposes of numerical evaluation the scan area is taken as shown in figure 5.1. Smaller contours are drawn in the scan area as shown and the location of the roots are as marked. Notice that the root location at $s = -.5+j1.$ is on the side of the contour and some of the others are within .05 from the sides of the contour. Subroutine SEARCH along with HOMEIN option and SEEK with ratio condition of 10^{-10} were used to evaluate the locations. Results presented in table 5.1 were obtained using 48 points around the contour (12 point Gauss quadrature per side) while tables 5.2 and 5.3 were obtained by using 80 and 160 points respectively around the contour for SEARCH.

In order to introduce large oscillations in f(s), it has been multiplied by an entire function of the form e^{AS}. Define F(s) as

$$F(s) = e^{AS} f(s) \qquad (5.2)$$

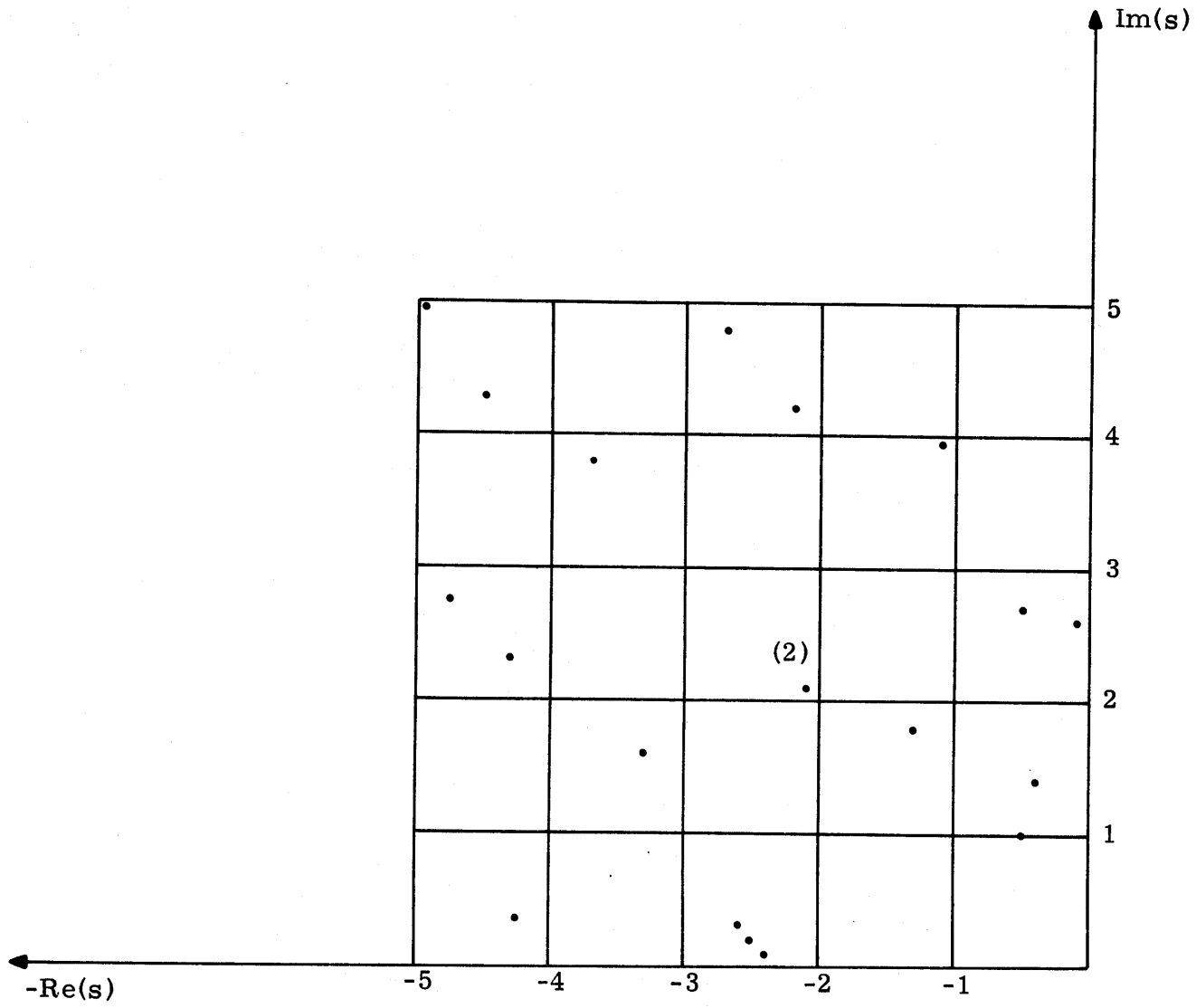

Figure 5.1. Scan area showing the subdivisions along with the locations of the roots. Root locations are shown by dots with their order indicated in parenthesis.

No.	Exact Location	Location Found by SEARCH	Location Found by HOMEIN	Location Found by SEEK
1	-.5+j1.0	-.500000007857+j.986119248895	-.5000000001969+j1.000000001890	-.499999999974+j.999940196864
2	-.4+j1.4	-.4000000040506+j1.41388789989	-.4000000004101+j1.400000002860	-.39999999998499+j1.40005980458
3	-.1+j2.6	-.100020243996+j2.600119280976	-.1000000006140+j2.600000004191	-.100000000177+j2.600000000164
4	-.5+j2.7	-.500002274223+j2.700005852813	-.5000000006153+j2.700000003527	-.4999999999737+j2.699999999517
5	-1.3+j1.8	-1.300000112328+j1.800000616575	-1.300000000537+j1.800000004345	-1.300000000065+j1.80000000817
6	-1.1+j3.95	-1.0099886484113+j3.950192243346	-1.100000000680+j3.950000005234	-1.099999999991+j3.949999999897
7	-2.4+j.1	-2.316482193266+j.2454476633878	-2.399558032358+j.1000411584392	-2.4000000001776+j.09999999944050
8	-2.5+j.2	Did Not Find This Zero	Did Not Find This Zero	-2.500000000835+j.1999999999511
9	-2.6+j.3	-2.683392729839+j.3545761221847	-2.599995734474+j.2999942711363	-2.599999999090+j.3000000009063
10	-2.1+j2.1	-2.099574882516+j2.099568616821	Could Not Improve The Location	-2.0999847643629+j2.100006355441
11	-2.1+j2.1	-2.100421589660+j2.100427826144	Could Not Improve The Location	-2.100015235495+j2.099993644031
12	-2.2+j4.2	-2.200000234552+j4.200000230275	-2.200000000726+j4.200000007857	-2.199999999981+j4.19999999967
13	-2.7+j4.8	-2.699999903485+j4.800000608303	-2.700000000727+j4.800000005178	-2.6999999999513+j4.799999999618
14	-3.3+j1.6	-3.300000001492+j1.600000026712	-3.300000000630+j1.600000005847	-3.300000000136+j1.600000000114
15	-3.7+j3.8	-3.699999904416+j3.800000618768	-3.700000000638+j3.800000005827	-3.699999999891+j3.7999999999185
16	-4.25+j.35	-4.250000094067+j3.500000344994	-4.250000000821+j3.500000049303	-4.2500000000601+j3.5000000025688
17	-4.3+j2.3	-4.300000049898+j2.300000018840	-4.300000000809+j2.300000005851	-4.299999999989+j2.3000000000050
18	-4.75+j2.75	-4.750000017428+j2.750000049225	-4.750000000628+j2.750000004923	-4.750000000010+j2.750000000028
19	-4.3+j4.5	-4.299999350548+j4.49999901608	-4.300000000760+j4.500000003920	-4.2999999998560+j4.499999999526
20	-4.95+j4.95	-4.949935650793+j4.949935981450	-4.950000000892+j4.950000004387	-4.950000000063+j4.949999999957

Table 5.1. Locations of zeros found by SEARCH using 48 points around the contour, using HOMEIN and those found by SEEK using ratio condition of 10^{-10} for the case of A = 0.

No.	Exact Location	Location Found by SEARCH	Location Found by HOMEIN	Location Found by SEEK
1	-.5+j1.0	-.499999999978+j.9915429543772	-.5000000002160+j1.000000001901	-.499999999974+j.999940196864
2	-.4+j1.4	-.399999999858+j1.40845704707	-.4000000003864+j1.400000002842	-.39999999998499+j1.40005980458
3	-.1+j2.6	-.0999994607547+j2.59999711055	-.1000000006136+j2.600000004191	-.100000000177+j2.600000000164
4	-.5+j2.7	-.499999965471+j2.69999991563	-.5000000006153+j2.70000003527	-.4999999999737+j2.699999999517
5	-1.3+j1.8	-1.300000000224+j1.800000001297	Could Not Improve The Location	-1.300000000065+j1.800000000817
6	-1.1+j3.95	-1.099994306796+j3.949991721801	-1.100000000679+j3.950000005238	-1.099999999991+j3.949999999897
7	-2.4+j.1	-2.400002657901+j.09999921087853	-2.400000019429+j.1000000068627	-2.4000000001776+j.09999999944050
8	-2.5+j.2	-2.500000421022+j.2000002722860	-2.499999979075+j2.000000095665	-2.500000000835+j.1999999999511
9	-2.6+j.3	-2.599999894778+j.2999999449622	-2.600000003289+j.2999999946891	-2.599999999090+j.3000000009063
10	-2.1+j2.1	-2.099979082275+j2.100017330964	Could Not Improve The Location	-2.0999847643629+j2.100006355441
11	-2.1+j2.1	-2.100020826266+j2.099982577274	Could Not Improve The Location	-2.100015235495+j2.099993644031
12	-2.2+j4.2	-2.199999999859+j4.199999999541	Could Not Improve The Location	-2.199999999981+j4.19999999967
13	-2.7+j4.8	-2.699999999731+j4.800000000088	Could Not Improve The Location	-2.6999999999513+j4.799999999618
14	-3.3+j1.6	-3.300000000099+j1.600000000273	Could Not Improve The Location	-3.300000000136+j1.600000000114
15	-3.7+j3.8	-3.699999999708+j3.800000000347	Could Not Improve The Location	-3.699999999891+j3.799999999185
16	-4.25+j.35	-4.250000000019+j3.500000003153	Could Not Improve The Location	-4.2500000000601+j3.5000000025688
17	-4.3+j2.3	-4.299999999972+j2.30000000000	Could Not Inprove The Location	-4.299999999989+j2.3000000000050
18	-4.75+j2.75	-4.749999999997+j2.750000000013	Could Not Improve The Location	-4.750000000010+j2.750000000028
19	-4.3+j4.5	-4.300000006598+j4.500000009150	-4.300000000760+j4.500000003920	-4.2999999998560+j4.499999999526
20	-4.95+j4.95	-4.949999071288+j4.949999068539	-4.950000000892+j4.950000004389	-4.950000000063+j4.949999999957

Table 5.2. Locations of zeros found by SEARCH using 80 points around the contour, using HOMEIN and those found by SEEK using ratio condition of 10^{-10} for the case of A = 0.

No.	Exact Location	Location Found by SEARCH	Location Found by HOMEIN	Location Found by SEEK
1	-.5+j1.0	-.4999999999831+j.9957207925762	-.5000000002284+j1.000000001913	-.499999999974+j.999940196864
2	-.4+j1.4	-.3999999998648+j1.404279208886	-.4000000003662+j1.400000002826	-.39999999998499+j1.40005980458
3	-.1+j2.6	-.9999999974780+j2.60000000096	Could Not Improve The Location	-.100000000177+j2.600000000164
4	-.5+j2.7	-.4999999996735+j2.699999999703	Could Not Improve The Location	-.4999999999737+j2.699999999517
5	-1.3+j1.8	-1.300000000064+j1.800000000881	Could Not Improve The Location	-1.300000000065+j1.800000000817
6	-1.1+j3.95	-1.099999992661+j3.950000002239	-1.100000000679+j3.950000005238	-1.099999999991+j3.949999999897
7	-2.4+j.1	-2.400000000125+j.09999999895124	Could Not Improve The Location	-2.4000000001776+j.09999999944050
8	-2.5+j.2	-2.500000000751+j.2000000000769	Could Not Improve The Location	-2.500000000835+j.1999999999511
9	-2.6+j.3	-2.599999999125+j.3000000008287	Could Not Improve The Location	-2.599999999090+j.3000000009063
10	-2.1+j2.1	-2.099984656873+j2.100009408153	Could Not Improve The Location	-2.0999847643629+j2.100006355441
11	-2.1+j2.1	-2.100015342891+j2.099990591262	Could Not Improve The Location	-2.199915235495+j2.099993644031
12	-2.2+j4.2	-2.199999999960+j4.199999999649	Could Not Improve The Location	-2.199999999981+j4.19999999967
13	-2.7+j4.8	-2.6999999999256+j4.799999999665	Could Not Improve The Location	-2.6999999999513+j4.799999999618
14	-3.3+j1.6	-3.300000000116+j1.600000000291	Could Not Improve The Location	-3.300000000136+j1.600000000114
15	-3.7+j3.8	-3.699999999877+j3.799999999928	Could Not Improve The Location	-3.699999999891+j3.7999999999185
16	-4.25+j.35	-4.250000000050+j3.500000003322	Could Not Improve The Location	-4.2500000000601+j3.5000000025688
17	-4.3+j2.3	-4.299999999985+j2.300000000001	Could Not Improve The Location	-4.299999999989+j2.3000000000050
18	-4.75+j2.75	-4.750000000028+j2.750000000034	Could Not Improve The Location	-4.750000000010+j2.750000000028
19	-4.3+j4.5	-4.299999999833+j4.499999999352	Could Not Improve The Location	-4.2999999998560+j4.499999999526
20	-4.95+j4.95	-4.949999999952+j4.950000000213	Could Not Improve The Location	-4.950000000063+j4.949999999957

Table 5.3. Locations of zeros found by SEARCH using 160 points around the contour, using HOMEIN and those found by SEEK using ratio condition of 10^{-10} for the case of A = 0.

where f(s) is given by (5.1). Exponential real constant A was varied from 0 to 70 and roots of F(s) were again evaluated. No appreciable change in the locations of the zeros evaluated is found. Tables 5.4 through 5.6 contain the results for A = 70 using SEARCH with HOMEIN option and SEEK with ratio condition of 10^{-10}. As can be seen, large variations in phase and magnitude do not affect the results of SEARCH, HOMEIN or SEEK. For numerical evaluation A could not be increased beyond 70 because of the underflow condition problem encountered in the computer.

As can be seen from the tables, no significant changes in root locations have been observed even when large oscillations were introduced in the function. Although the results presented here used single precision, using extended precision the results would be far superior. In general, for polynomial roots, good results were obtained if the order of Gaussian quadrature formula was of the order of the polynomial.

B. Natural frequencies of a thin wire

In this section, we consider for illustrative purposes, the problem of determining the natural frequencies of a straight thin wire whose diameter to length ratio (d/L) is .01. A more detailed discussion of the formulation of this problem may be found in Giri, Singaraju and Baum.[14] The Hallén form of the integral equation of a straight thin wire, in the absence of the incident field is given by

$$\int_0^L \tilde{I}(z')\tilde{K}(z - z')\,dz' = A\sinh(\gamma z) + B\cosh(\gamma z) \tag{5.3}$$

where the tilde signifies a two-sided Laplace form and the kernel function is given by

$$\tilde{K}(z - z') = \frac{1}{2\pi a}\int_0^{2\pi} \frac{e^{-\gamma R}}{4\pi R}\,a\,d\phi' \tag{5.4}$$

No.	Exact Location	Location Found by SEARCH	Location Found by HOMEIN	Location Found by SEEK
1	-.5+j1.0	-.500000007857+j.986119248895	-.5888888881969+j1.000000001890	-.4999999999779+j.999760783781
2	-.4+j1.4	-.400000004050+j1.41388078998	-.400000000101+j1.400000002860	-.3999999998428+j1.4002392176680
3	-.1+j2.6	-.100020243996+j2.60011928097	-.1000000006140+j2.600000004191	-.1000000001833+j2.600000000091
4	-.5+j2.7	-.500002274223+j2.70000585281	-.5000000006153+j2.700000003527	-.4999999997101+j2.6999999999559
5	-1.3+j1.8	-1.30000011232+j1.80000061657	-1.300000000537+j1.800000004345	-1.3000000000464+j1.800000000853
6	-1.1+j3.95	-1.09988648411+j3.950192243346	-1.100000000680+j3.950000005234	-1.1000000002575+j3.949999999838
7	-2.4+j.1	-2.316482193266+j.2454476633879	-2.399558032358+j.100041158439	-2.400000000113+j.09999999946671
8	-2.5+j.2	Did Not Find This Zero	Did Not Find This Zero	-2.500000001099+j.1999999999152
9	-2.6+j.3	-2.683392729838+j.3545761221846	-2.599995734474+j2.999942711363	-2.5999999989483+j.3000000008994
10	-2.1+j2.1	-2.099574882526+j2.099568616810	Could Not Improve This Location	-2.099984711179+j2.100007309278
11	-2.1+j2.1	-2.100421589649+j2.100427826155	Could Not Improve This Location	-2.1000152885500+j2.099992690080
12	-2.2+j4.2	-2.20000034552+j4.200000230275	-2.200000000726+j4.200000007857	-2.199999999982+j4.199999999676
13	-2.7+j4.8	-2.69999993485+j4.800000608303	-2.700000000727+j4.800000005178	-2.6999999999453+j4.799999999630
14	-3.3+j1.6	-3.300000001492+j1.600000026712	-3.300000000630+j1.600000005847	-3.300000000116+j1.600000000113
15	-3.7+j3.8	-3.699999904416+j3.800000618768	-3.700000000638+j3.800000005827	-3.699999999898+j3.7999999999154
16	-4.25+j.35	-4.250000094067+j3.500000344994	-4.250000000821+j3.500000049303	-4.250000000060+j3.500000002568
17	-4.3+j2.3	-4.300000049898+j2.300000018841	-4.300000000809+j2.300000005851	-4.2999999999802+j2.300000000021
18	-4.75+j2.75	-4.750000017428+j2.750000049225	-4.750000000628+j2.750000004923	-4.7500000000153+j2.7500000000385
19	-4.3+j4.5	-4.299999350548+j4.499999016018	-4.300000000760+j4.500000003920	-4.2999999998067+j4.499999999645
20	-4.95+j4.95	-4.949935650793+j4.949935981450	Could Not Find This Zero*	-4.9500000227758+j4.9500000227286

*This zero could not be found because of underflow condition.

Table 5.4. Locations of zeros found by SEARCH using 48 points around the contour, using HOMEIN and those found by SEEK using ratio condition of 10^{-10} for the case of A = 70.

No.	Exact Location	Location Found by SEARCH	Location Found by HOMEIN	Location Found by SEEK
1	-.5+j1.0	-.4999999999781+j.9915429543772	-.5000000002160+j1.1000000001901	-.4999999999779+j.999760783781
2	-.4+j1.4	-.3999999998582+j1.408457047079	-.4000000003684+j1.400000002842	-.3999999998428+j1.4002392176680
3	-.1+j2.6	-.09999946077944+j2.599997110515	-.1000000006136+j2.600000004191	-.1000000001833+j2.600000000091
4	-.5+j2.7	-.4999999654467+j2.699999915671	-.5000000006153+j2.700000003527	-.4999999997101+j2.699999999559
5	-1.3+j1.8	-1.300000000224+j1.800000001297	Could Not Improve The Location	-1.3000000000464+j1.800000000853
6	-1.1+j3.95	-1.099994306796+j3.949991721801	-1.1000000006792+j3.950000005238	-1.1000000002575+j3.949999999838
7	-2.4+j.1	-2.400002657801+j.09999921102107	Could Not Improve The Location	-2.400000000113+j.09999999946671
8	-2.5+j.2	-2.500000421222+j.2000002718615	Could Not Improve The Location	-2.500000001099+j.1999999999152
9	-2.6+j.3	-2.599999894678+j.2999999452441	Could Not Improve The Location	-2.5999999989483+j.3000000008994
10	-2.1+j2.1	-2.099979445188+j2.100017297542	Could Not Improve The Location	-2.099984711179+j2.100007309278
11	-2.1+j2.1	-2.100020463353+j2.099982610696	Could Not Improve The Location	-2.1000152885500+j2.099992690080
12	-2.2+j4.2	-2.199999999856+j4.199999999516	Could Not Improve The Location	-2.199999999982+j4.199999999676
13	-2.7+j4.8	-2.699999999733+j4.800000000113	Could Not Improve The Location	-2.6999999999453+j4.799999999630
14	-3.3+j1.6	-3.300000000099+j1.600000000273	Could Not Improve The Location	-3.300000000116+j1.600000000113
15	-3.7+j3.8	-3.699999999708+j3.8000000003153	Could Not Improve The Location	-3.699999999898+j3.7999999999154
16	-4.25+j.35	-4.250000000020+j3.500000003153	Could Not Improve The Location	-4.250000000060+j3.500000002568
17	-4.3+j2.3	-4.299999999972+j2.299999999969	Could Not Improve The Location	-4.2999999999802+j2.300000000021
18	-4.75+j2.75	-4.7499999999976+j2.750000000044	Could Not Improve The Location	-4.7500000000153+j2.7500000000385
19	-4.3+j4.5	-4.300000006602+j4.500000009125	-4.300000000760+j4.500000003920	-4.2999999998067+j4.499999999645
20	-4.95+j4.95	-4.949999071284+j4.949999068564	Could Not Find This Zero*	-4.900000227758+j4.9500000227286

*This zero could not be found because of underflow condition.

Table 5.5. Location of zeros found by SEARCH using 80 points around the contour, using HOMEIN and those found by SEEK using ratio condition of 10^{-10} for the case of A = 70.

No.	Exact Location	Location Found by SEARCH	Location Found by HOMEIN	Location Found by SEEK
1	-.5+j1.0	-.4999999999831+j.9957207925762	-.5000000002284+j1.00000000191	-.4999999999779+j.999760783781
2	-.4+j1.4	-.3999999998648+j1.404279208886	-.4000000003662+j1.400000002826	-.3999999998428+j1.4002392176680
3	-.1+j2.6	-.09999999974775+j2.600000000096	Could Not Improve The Location	-.1000000001833+j2.600000000091
4	-.5+j2.7	-.4999999996735+j2.699999999703	Could Not Improve The Location	-.499999999710+j2.699999999559
5	-1.3+j1.8	-1.300000000064+j1.800000000881	Could Not Improve The Location	-1.3000000000464+j1.800000000853
6	-1.1+j3.95	-1.099999992661+j3.950000002239	-1.1000000006793+j3.950000005238	-1.1000000002575+j3.949999999838
7	-2.4+j.1	-2.400000000121+j.09999999895116	Could Not Improve The Location	-2.400000000113+j.09999999946671
8	-2.5+j.2	-2.500000000757+j.2000000000769	Could Not Improve The Location	-2.500000001099+j.1999999999152
9	-2.6+j.3	-2.599999999122+j.3000000008288	Could Not Improve The Location	-2.5999999989483+j.3000000008994
10	-2.1+j2.1	-2.099984658372+j2.100009407219	Could Not Improve The Location	-2.099984711179+j2.100007309278
11	-2.1+j2.1	-2.100015341392+j2.099990592195	Could Not Improve The Location	-2.1000152885500+j2.099992690080
12	-2.2+j4.2	-2.199999999959+j4.199999999650	Could Not Improve The Location	-2.199999999982+j4.199999999676
13	-2.7+j4.8	-2.699999999925+j4.799999999665	Could Not Improve The Location	-2.6999999999453+j4.799999999630
14	-3.3+j1.6	-3.300000000116+j1.600000000291	Could Not Improve The Location	-3.300000000116+j1.600000000113
15	-3.7+j3.8	-3.699999999877+j3.799999999928	Could Not Improve The Location	-3.699999999898+j3.7999999999154
16	-4.25+j.35	-4.25000000050+j.3500000003322	Could Not Improve The Location	-4.250000000060+j3.500000002568
17	-4.3+j2.3	-4.299999999985+j2.300000000001	Could Not Improve The Location	-4.2999999999802+j2.300000000021
18	-4.75+j2.75	-4.750000000027+j2.750000000034	Could Not Improve The Location	-4.7500000000153+j2.7500000000385
19	-4.3+j4.5	-4.299999999833+j4.499999999352	Could Not Improve The Location	-4.2999999998067+j4.499999999645
20	-4.95+j4.95	-4.949999999952+j4.950000000213	Could Not Find The Zero*	-4.9500000227758+j4.9500000227286

*This zero could not be found because of underflow condition.

Table 5.6. Locations of zeros found by SEARCH using 160 points around the contour, using HOMEIN and those found by SEEK using ratio condition of 10^{-10} for the case of A = 70.

and

$$\gamma = s/c = \text{complex propagation constant}$$
$$s = \text{complex frequency} \equiv s_r + i s_i$$
$$c = \text{speed of light in free space} \tag{5.5}$$
$$R = [(z - z')^2 + 4a^2 \sin^2(\phi'/2)]^{1/2}$$

Figure 5.2 shows the geometry of the problem as well as the zoning of the wire structure, which is useful in matricizing (5.3) to get

$$\left[Z_{p,q}\right]\left[I_p\right] = \left[f_p(z)\right] \tag{5.6}$$

or

$$Z_o\left[Z_{N_{p,q}}\right]\left[I_p\right] = \left[f_p(z)\right] \tag{5.7}$$

In above equation, Z_o is the characteristic impedance of free space and $\left[Z_{N_{p,q}}\right]$ is the normalized Hallen-system matrix whose elements are given by

$$Z_{N_{p,q}} = \frac{1}{2\pi a} \int_{z_q-(\Delta/2)}^{z_q+(\Delta/2)} dz' \int_0^{2\pi} \frac{e^{-\gamma R_p}}{4\pi R_p} a\,d\phi' \tag{5.8}$$

where R_p is given by (5.5) with z replaced by z_p, $[I_p]$ is the unknown column matrix made up of currents at the centers of each of the (n+1) zones in the thin wire and f_p

$$f_p(z) = Z_o A \sinh(\gamma z_p) + Z_o B \cosh(\gamma z_p) \tag{5.9}$$

Imposing the end conditions $I(z=0) = I_1 = 0$, $I(z=L) = I_{(n+1)} = 0$ and rearranging (5.7) we have,

$$Z_o\left[Z'_{N_{p,q}}\right]\left[I'_p\right] = \left[0\right] \tag{5.10}$$

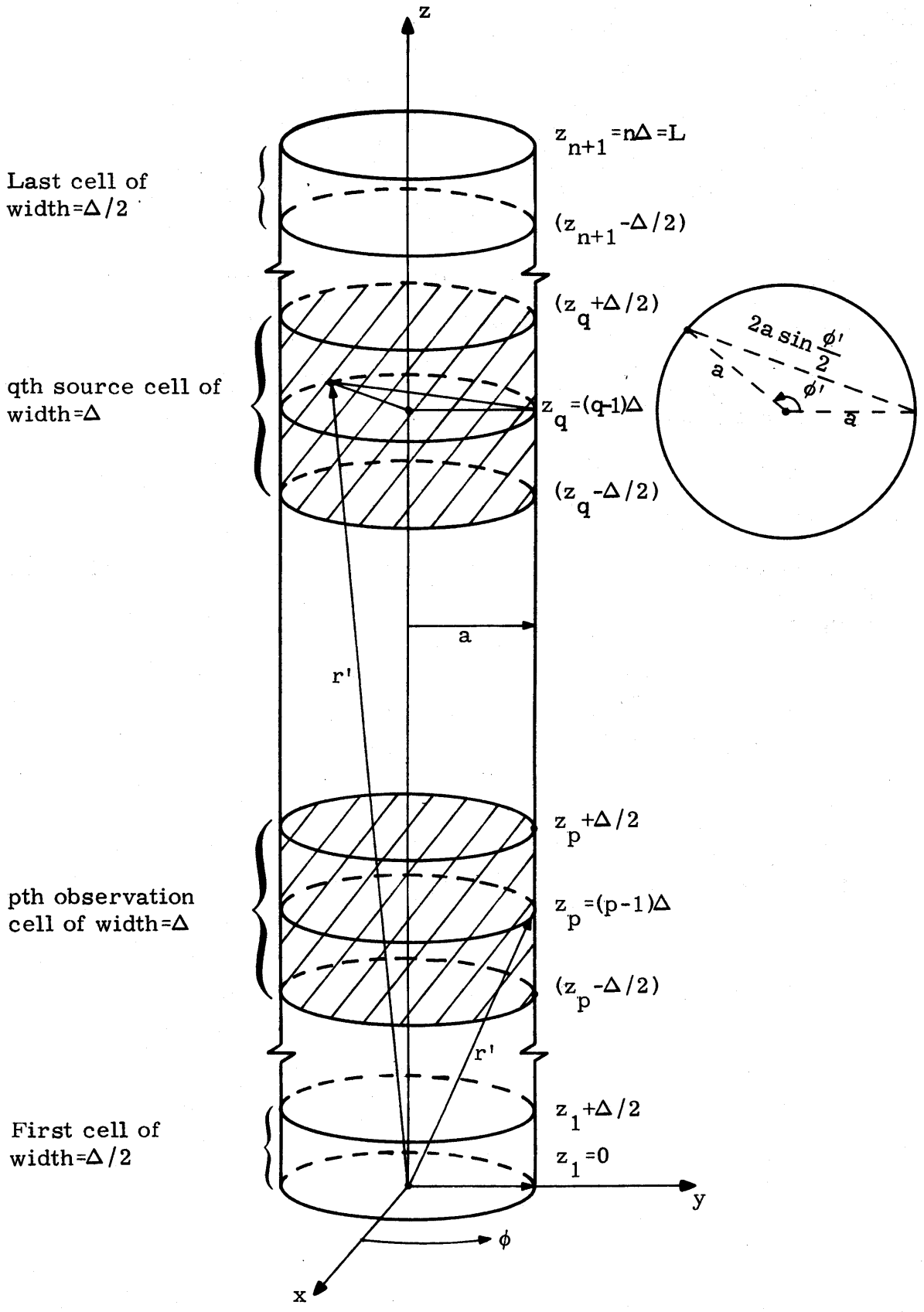

Figure 5.2. Geometry of the problem showing the zoning of the thin wire

where $\left[Z'_{N_{p,q}}\right]$ is a (n+1) x (n+1) matrix given by replacing the first and last columns of $\left[Z_{N_{p,q}}\right]$ via

$$Z'_{N_{p,1}} = -\sinh(\gamma z_p)$$
$$Z'_{N_{p,(n+1)}} = -\cosh(\gamma z_p) \tag{5.11}$$

The natural frequencies $[s_\alpha L/(\pi c)]$ are given by the zeros of the determinant of the Z'_N matrix, i.e.,

$$\left[\det Z'_N(s_\alpha)\right]_{(n+1),(n+1)} = 0 \tag{5.12}$$

We choose a sufficient number of cells so that the cell width Δ satisfies the conditions $|\gamma\Delta| \ll 1$ and $(\Delta/a) \geq 10$. For the singular diagonal element, however, the exponential in (5.8) may be approximated by the first two terms of its series expansion. Under such an approximation, the matrix elements are given by

$$Z'_{N_{p,q}}(s) \simeq \int_{-1}^{1} dx \left[\int_0^{2\pi} \frac{e^{-\gamma\Delta}}{16\pi^2 \sqrt{(|p-q|-x)^2 + \frac{4a^2}{\Delta^2}\sin^2\frac{\phi'}{2}}}\, d\phi' \right] \tag{5.13}$$

with $p = 1, 2, \ldots, n, (n+1)$

$q = 2, 3, \ldots, n$

and the diagonal element

$$Z'_{N_{p,p}}(s) \simeq \frac{\ln(2)}{2\pi} + \frac{1}{4\pi^2} \int_0^{2\pi} \ln\left(\frac{\Delta}{4a} + \sqrt{\left(\frac{\Delta}{4a}\right)^2 + \sin^2\frac{\phi'}{2}}\right) d\phi' - \frac{\gamma\Delta}{4\pi} \tag{5.14}$$

The matrix Z'_N is now completely specified by (5.11), (5.13) and (5.14). The double integral of (5.13) and the integral in (5.14) are all performed using a 12-point Gaussian Quadrature routine for complex (or real) integrands. For a prescribed frequency, the matrix elements are found and then the determinant, which serves as the complex function CFCTS(CS) used in conjunction with the subroutines CONTOUR and SEEK in the determination of the natural frequencies $\left[\frac{s_\alpha L}{\pi c}\right]$. The natural frequencies are known to occur in layers and as conjugate pairs in the left half plane. The rectangular (or square) contours used in determining the natural frequencies are described in table 5.7 where

$\left[\frac{s_{\ell,n} L}{\pi c}\right]$ = nth natural frequency on the ℓth layer in the normalized complex frequency plane

CSF = coordinates of the lower right corner of the rectangular/square contour

CSL = coordinates of the upper left corner of the rectangular/square contour

NR, NI = number of major divisions along the real and imaginary axes, respectively

The contours are uniquely specified by indicating CSF, CSL, NR and NI.

The results $\left[\frac{s_{\ell,n} L}{\pi c}\right]$ of the computations done on the CDC-7600 system, are found in table 5.8 and plotted in figure 5.3. In table 5.8, we also list for purposes of comparison, the natural frequencies on the layers 1 and 2 computed and tabulated by Tesche.[15] A good agreement is seen between the two and the minor discrepancies may be attributed to the ways of computing the system matrix and also the accuracies of the numerical zero-searching procedures. In this context, we point out that a 12-point sampling of the function on each side of the rectangular/square contour was employed.

Furthermore, as we move away from the origin into the left half plane, the elements of $Z_{p,q}$ defined by (5.6) grow without limit and consequently the average value of the magnitude of the function (i.e., the determinant)

	The two indices of natural frequency	Contour Parameters				Computer times in octal seconds
		CSF	CSL	NR	NI	
Layer 1	n = 1, 2	.5 + j.5	-.5 + j2.5	1	1	173*
	n = 3, 4	.5 + j2.5	-.5 + j4.5	1	1	420*
	n = 5,...,8	.5 + j4.5	-.5 + j8.5	1	2	350
	n = 9, 10	.5 + j8.5	-.5 + j10.5	1	1	513
Layer 2	n = 1	-1 -j.5	-3 + j.5	1	1	80
	n = 2, 3	-2 + j1	-4 + j3	1	1	120
	n = 4	-2 + j3	-4 + j4	1	1	150
	n = 5,...,10	-2 + j4	-4 + j10	1	3	890
Layer 3	n = 1	-3 - j.5	-5 + j.5	1	1	130
	n = 2, 3, 4	-4 + j1	-6 + j5	1	2	449
	n = 5,...,9	-4 + j4	-6 + j10	1	3	1049

Note: Computer times are total job times on the CDC-7600 system except for the starred (*) numbers which are on the CDC-6600 system.

Table 5.7. The contour descriptions and the computing times incurred in evaluating the natural frequencies of the thin wire (d/L = 0.01).

n	Layer 1	Layer 2	Layer 3
1	-.0828+j.9251 (-.082+j.926)	$-2.1687+j.349\times10^{-11}$ (-2.174+j0.0)	$-4.0993+j.394\times10^{-7}$
2	-.1212+j1.9117 (-.120+j1.897)	-2.500+j1.3329 (-2.506+j1.347)	-4.5142+j1.4979
3	-.1491+j2.8835 (-.147+j2.874)	-2.7342+j2.4680 (-2.725+j2.477)	-4.8285+j2.7472
4	-.1713+j3.8741 (-.169+j3.854)	-2.9146+j3.5334 (-2.890+j3.544)	-5.0693+j3.8894
5	-.1909+j4.8536 (-.188+j4.835)	-3.0454+j4.5757 (-3.025+j4.581)	-5.2851+j5.0070
6	-.2080+j5.8453 (-.205+j5.817)	-3.1640+j5.6097 (-.3139+j5.603)	-5.4647+j6.0811
7	-.2240+j6.8286 (-.220+j6.800)	-3.2659+j6.6221	-5.6277+j7.1478
8	-.2383+j7.8212 (-.234+j7.783)	-3.3562+j7.6405	-5.772+j8.1901
9	-.2522+j8.8068 (-.247+j8.767)	-3.4376+j8.6466	-5.9045+j9.2351
10	-.2648+j9.8001 (-.260+j9.752)	-3.5108+j9.6555	

Table 5.8. Pole locations $\left[\frac{s_\alpha L}{\pi c}\right]$ in the complex frequency plane for the thin wire of $d/L = .01$, determined by the contour integration method. The numbers in parenthesis in this table are reproduced for comparison from Tesche.[15]

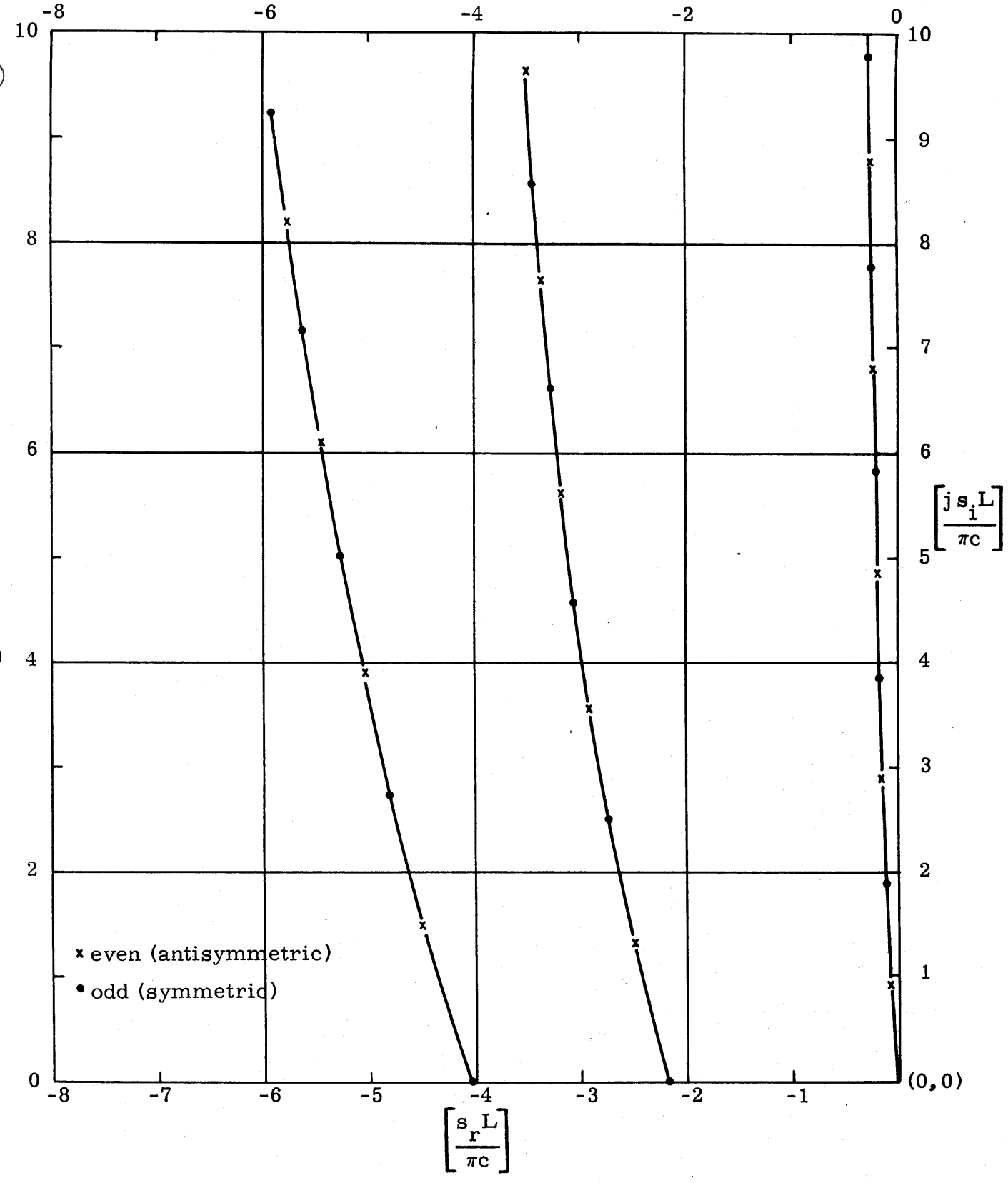

Figure 5.3. Plot of pole locations $\left[\frac{s_\alpha L}{\pi c}\right]$ in the normalized complex frequency plane for the thin wire of d/L = .01, found by contour integration method

also grows. However by virtue of normalizing every element of $Z_{p,q}$ with the characteristic impedance of free space Z_o, we have essentially a factor of $Z_o^{(n+1)}$ removed from the matrix according to

$$\left[\text{det. } Z_{p,q}\right] = Z_o^{(n+1)} \left[\text{det. } Z_{N_{p,q}}\right] \tag{5.15}$$

where the symbol det. stands for the determinant. It is once again emphasized that in order to meet the following restrictions on the zone width

$$\begin{aligned} &|\gamma\Delta| \ll 1 \\ &(\Delta/a) \geq 10 \end{aligned} \tag{5.16}$$

everywhere in the normalized complex s, i.e., $(sL/\pi c)$ plane, the number of zones were chosen according to

$$n = L/\Delta = 10 \times \text{integer part of } \left|sL/\pi c\right| \text{ at the center of the rectangular contour} \tag{5.17}$$

Since n depends on the frequency and increases with it, a contour which is located away from the origin has a larger factor removed from its function values because of $Z_o^{(n+1)}$ in (5.15). This, however, does not affect in any way the location of the natural frequencies because every contour described in table 5.7 is treated individually by subroutine SEEK and removing a constant (although this constant varies from contour to contour) from all the function values on a given contour does not affect the structure of zeros inside the contour. Because of this increasing normalization constant as we move away from the origin, in our actual computations, the magnitude of the determinant appears to drop significantly to values in the range of 10^{-60} to 10^{-70} near the top part of the third layer in figure 5.3. This is a drop by a factor of 10^{-40} to 10^{-50} from the determinant values encountered in evaluating for instance, the second zero on the first layer. However, irrespective of whether the function values grow or drop in magnitude, the

normalizing procedure used in subroutine SEEK works efficiently and helps in extracting the natural frequencies.

This example has reinforced our confidence in the numerical procedure developed in this note for the purpose of finding the zeros of complex functions by an application of contour integration methods with the a priori prescription about the absence of poles.

VI. Conclusions

In this report we have shown that the zeros of an analytic function can be found by a simple procedure. A new technique involving simple integration was discussed along with the computer subroutines. Two numerical examples, one involving the roots of a polynomial and the other the natural frequencies of a thin wire were discussed. These numerical examples can be considered to have encompassed both ends of the spectrum of complexity. These routines have also been used in conjunction with elliptic functions[16] and are being used to find the natural frequencies of a helical antenna.[17] As can be seen, one of the limitations, if any, of these subroutines is the imagination of the user.

Appendix

```
      SUBROUTINE SEARCH (CFCTS,CSL,CSF,NR,NI,ND,NH,FACTOR)
      IMPLICIT COMPLEX (C)
      COMMON PI,TMAG(401),ARG(401),CS(401),CF(401),C(4),CSH(25),FZH(25)
      DIMENSION AVE(25), CH(1500), CV(1500), CSA(25), FUN(25), X1(12), X
     12(20), X3(40), W1(12), W2(20), W3(40), X(40), W(40)
      DATA N1/12/,X1/-.981560634246719,-.904117256370475,-.7699026741943
     105,-.587317954286617,-.367831498998180,-.125233408511469,.12523340
     28511469,.367831498998180,.587317954286617,.769902674194305,.904117
     3256370475,.981560634246719/
      DATA W1/.047175336386512,.106939325995318,.160078328543346,.203167
     1426723066,.233492536538355,.249147045813403,.249147045813403,.2334
     292536538355,.203167426723066,.160078388543346,.106939325995318,.04
     37175336386512/
      DATA N2/20/,X2/-.993128599185094924786,-.963971927277913791268,-.9
     1122344282851325905868,-.839116971822218823395,-.746331906460150792
     6214,-.636053680726515025453,-.510867001750827098002 4,-.373706088715
     3419560637,-.227785851141645078080,-.076526521133497333755,.0765265
     421133497333755,.227785851141645078080,.373706088715419560637,.5108
     567001750827098002 4,.636053680726515025453,.746331906460150792614,.
     683911697188233 95,.912234428251325905868,.963971927277913791268,.99
     731285991850949247 86/
      DATA W2/.017614007139152118312,.040601429800386941331,.06267204833
     14109063570,.083276741576704748725,.101930119817240435073,.11819453
     2196158417312,.131688638449176268 98,.142096109318382051329,.149172
     3986472603746788,.152753387130725850698,.152753387130725850698,.149
     4172986472603746788,.142096109318382051329,.131688638449176268 98,.
     5118194531961518417312,.101930119817240435037,.083276741576704748 72
     65,.062672048334109063570,.040601429800386941331,.017614007139152 11
     78312/
      DATA N3/40/,X3/-.998237709710559,-.990726238699457,-.9772599499837
     174,-.957916819213792,-.932812808278677,-.902098806968874,-.8659595
     203212260,-.824612230833312,-.778305651426519,-.727318255189927,-.6
     371956684614180,-.612553889667980,-.549467125095128,-.4830758016861
     479,-.413779204371605,-.341994090825758,-.268152185007254,-.1926975
     580701371,-.116084070675255,-.387724175060508E-1,.387724175060508E-
     61,.116084070675255,.192697580701371,.268152185007254,.341994090825
     7758,.413779204371605,.483075801686179,.549467125095128,.6125538896
     867980,.671956684614180,.727318255189927,.778305651426519,.82461223
     90833312,.865959503212260,.902098806968874,.932812808278677,.957916
     $819213792,.977259949983774,.990726238699457,.998237709710559/
      DATA W3/.452127709853 32E-2,.104982845311528E-1,.164210583819079E-1
     1,.222458491941670E-1,.279370069800234E-1,.334601952825478E-1,.3878
     221679744720E-1,.438709081856733E-1,.486958076350722E-1,.5322784698
     339368E-1,.574397690993916E-1,.613062424929289E-1,.648040134566010E
     4-1,.679120458152339E-1,.706116473912868E-1,.728865823958041E-1,.74
     5723169057968 3E-1,.761103619006262E-1,.770398181642480E-1,.77505947
     69784248E-1,.775059479784248E-1,.770398181642480E-1,.761103619006 26
     72E-1,.747231690579683E-1,.728865823958041E-1,.706116473912868E-1,.
     8679120458152339E-1,.648040134566010E-1,.613062424929289E-1,.574397
     9690993916E-1,.532278469839368E-1,.486958076350722E-1,.438709081856
     $733E-1,.387821679744720E-1,.334601952825478E-1,.279370069800234E-1
     $,.222458491941670E-1,.164210583819079E-1,.104982845311528E-1,.4521
     $27709853319E-2/
       IF (ND-20) 10,30,50
```

```
10 CONTINUE
   ND=N1
   DO 20 I=1,ND
   X(I)=X1(I)
   W(I)=W1(I)
20 CONTINUE
   GO TO 70
30 CONTINUE
   ND=N2
   DO 40 I=1,ND
   X(I)=X2(I)
   W(I)=W2(I)
40 CONTINUE
   GO TO 70
50 CONTINUE
   ND=N3
   DO 60 I=1,ND
   X(I)=X3(I)
   W(I)=W3(I)
60 CONTINUE
70 CONTINUE
   RCSF=REAL(CSF)
   SCIF=AIMAG(CSF)
   RCSL=REAL(CSL)
   SCIL=AIMAG(CSL)
   RINC=(RCSL-RCSF)/FLOAT(NR)
   SCIINC=(SCIL-SCIF)/FLOAT(NI)
   NRTR=NR*ND
   NITC=NI*ND
   NRT=NRTR*NI
   NIT=NITC*NR
   NR1=NR+1
   NI1=NI+1
   PI3=1.5*PI
   TPI=2.*PI
   HPI=.5*PI
   ND1=ND+1
   ND2=2*ND
   ND3=3*ND
   ND4=ND*4
   NM1=ND4+1
   NMM1=ND4-1
   NDP=ND2+1
   PRINT 810, RCSF,SCIF,RCSL,SCIL,ND4,NR,NI
   DO 100 I=1,NI
   II=I-1
   IIND=II*ND
   AIM=FLOAT(II)*SCIINC+SCIF
   AIMM=FLOAT(I)*SCIINC+SCIF
   DELI=(AIMM-AIM)*.5
   DELIP=(AIMM+AIM)*.5
   DO 90 K=1,ND
   Y=DELI*X(K)+DELIP
   DO 80 J=1,NR1
```

```
      JJ=J-1
      JJN=JJ*NITC
      KP=JJN+IIND+K
      REA=FLOAT(JJ)*RINC+RCSF
      CA=CMPLX(REA,Y)
      CV(KP)=CFCTS(CA)
   80 CONTINUE
   90 CONTINUE
  100 CONTINUE
      DO 130 I=1,NR
      II=I-1
      IIND=II*ND
      RMI=FLOAT(II)*RINC+RCSF
      RMM=FLOAT(I)*RINC+RCSF
      DELR=(RMM-RMI)*.5
      DELRP=(RMM+RMI)*.5
      DO 120 K=1,ND
      R=DELR*X(K)+DELRP
      DO 110 J=1,NI1
      JJ=J-1
      JJN=JJ*NRTR
      IP=JJN+IIND+K
      AIMMA=FLOAT(JJ)*SCIINC+SCIF
      CA=CMPLX(R,AIMMA)
      CH(IP)=CFCTS(CA)
  110 CONTINUE
  120 CONTINUE
  130 CONTINUE
      NOZ=0
      DO 720 J=1,NR
      J1=J+1
      JJ=J-1
      SRMI=RCSF+RINC*FLOAT(J)
      SRM=RCSF+RINC*FLOAT(JJ)
      DELX=(SRM-SRMI)*.5
      PLX=(SRM+SRMI)*.5
      DO 710 I=1,NI
      I1=I+1
      II=I-1
      SIM=SCIF+SCIINC*FLOAT(I)
      SIMI=SCIF+SCIINC*FLOAT(II)
      DELY=(SIM-SIMI)*.5
      PLY=(SIM+SIMI)*.5
      DO 250 K=1,4
      KK=K-1
      KKND=KK*ND+1
      KPND=K*ND
      IF (K-2) 160,170,140
  140 IF (K-3) 170,190,150
  150 XU=SRM
      XL=SRMI
      YC=SIMI
      GO TO 230
  160 YU=SIM
```

```
      YL=SIMI
      XC=SRM
      GO TO 210
  170 XU=SRM
      XL=SRMI
      YC=SIM
      DO 180 L=KKND,KPND
      CS(L)=CMPLX(-DELX*X(L-KK*ND)+PLX,YC)
  180 CONTINUE
      GO TO 250
  190 YU=SIM
      YL=SIMI
      XC=SRMI
      DO 200 L=KKND,KPND
      CS(L)=CMPLX(XC,-DELY*X(L-KK*ND)+PLY)
  200 CONTINUE
      GO TO 250
  210 CONTINUE
      DO 220 L=KKND,KPND
      CS(L)=CMPLX(XC,DELY*X(L-KK*ND)+PLY)
  220 CONTINUE
      GO TO 250
  230 CONTINUE
      DO 240 L=KKND,KPND
      CS(L)=CMPLX(DELX*X(L-KK*ND)+PLX,YC)
  240 CONTINUE
  250 CONTINUE
      CS(NM1)=CS(1)
      PRINT 820, SRM,SIMI,SRMI,SIM
      NY1=(J-1)*NITC+(I-1)*ND
      NX11=I*NRTR+(J-1)*ND
      NY22=J*NITC+I*ND+1
      NX2=(I-1)*NRTR+J*ND+1
      DO 260 K=1,ND
      KK=NY1+K
      CF(K)=CV(KK)
  260 CONTINUE
      DO 270 K=1,ND
      KK=NX11+K
      KKK=ND+K
      CF(KKK)=CH(KK)
  270 CONTINUE
      DO 280 K=1,ND
      KK=NY22-K
      KKK=ND2+K
      CF(KKK)=CV(KK)
  280 CONTINUE
      DO 290 K=1,ND
      KK=NX2-K
      KKK=ND3+K
      CF(KKK)=CH(KK)
  290 CONTINUE
      ASUM=0.
      DO 300 K=1,ND
```

```
      ASUM=ASUM+CABS(CF(K))
  300 CONTINUE
      A1=ASUM/FLOAT(ND)
      ASUM=0.
      DO 310 K=NDP,ND3
      ASUM=ASUM+CABS(CF(K))
  310 CONTINUE
      A2=ASUM/FLOAT(ND)
      A3=ALOG(A1/A2)/(SRM-SRMI)
      A4=A1*EXP(-A3*SRM)
      CSUM=(0.,0.)
      DO 320 K=1,ND4
      CSUM=CSUM+CF(K)
      CF(K)=CF(K)/(A4*CEXP(A3*CS(K)))
  320 CONTINUE
      CAVE=CSUM/FLOAT(ND4)
      AVEE=CABS(CAVE)
      DO 330 L=1,ND4
      RF=REAL(CF(L))
      Z=AIMAG(CF(L))
      ARG(L)=ANGLER(RF,Z)
      TMAG(L)=ALOG(CABS(CF(L)))
  330 CONTINUE
      IO=0
      OVF=ARG(1)
      OV=ARG(1)
      OVL=ARG(ND4)
      DO 390 K=1,NMM1
      IF (OV.GT.PI3.AND.OV.LT.TPI) GO TO 340
      IF (OV.GT.0..AND.OV.LT.HPI) GO TO 350
      GO TO 380
  340 IF (ARG(K+1).GT.0..AND.ARG(K+1).LT.HPI) GO TO 360
      GO TO 380
  350 IF (ARG(K+1).GT.PI3.AND.ARG(K+1).LT.TPI) GO TO 370
      GO TO 380
  360 IO=IO+1
      GO TO 380
  370 IO=IO-1
  380 OV=ARG(K+1)
      ARG(K+1)=IO*TPI+ARG(K+1)
  390 CONTINUE
      IF (OVL.GT.PI3.AND.OVL.LT.TPI) GO TO 400
      IF (OVL.GT.0..AND.OVL.LT.HPI) GO TO 410
      GO TO 440
  400 IF (OVF.GT.0..AND.OVF.LT.HPI) GO TO 420
      GO TO 440
  410 IF (OVF.GT.PI3.AND.OVF.LT.TPI) GO TO 430
      GO TO 440
  420 IO=IO+1
      GO TO 440
  430 IO=IO-1
  440 CONTINUE
      ARG(NM1)=FLOAT(IO)*TPI+ARG(1)
      IF (IO) 580,590,450
```

```
450 CONTINUE
    IO1=IO+1
    IF (IO.GE.4) GO TO 600
    DO 510 L=1,IO1
    CON1=(0.,0.)
    CON2=(0.,0.)
    CON3=(0.,0.)
    CON4=(0.,0.)
    LM1=L-1
    LL=LM1-1
    DO 500 K=1,ND4
    IF (K.GT.ND.AND.K.LE.ND2) GO TO 460
    IF (K.GT.ND2.AND.K.LE.ND3) GO TO 470
    IF (K.GT.ND3.AND.K.LE.ND4) GO TO 480
    CON1=CON1+(CS(K)**LL)*W(K)*CMPLX(TMAG(K),ARG(K))
    GO TO 490
460 CON2=CON2+(CS(K)**LL)*W(K-ND)*CMPLX(-ARG(K),TMAG(K))
    GO TO 490
470 CON3=CON3+(CS(K)**LL)*W(K-ND2)*CMPLX(TMAG(K),ARG(K))
    GO TO 490
480 CON4=CON4+(CS(K)**LL)*W(K-ND3)*CMPLX(-ARG(K),TMAG(K))
490 CONTINUE
500 CONTINUE
    CON1=CON1*DELY
    CON2=CON2*DELX
    CON3=CON3*DELY
    CON4=CON4*DELX
    CON5=((CMPLX(SRM,SIMI))**LM1)*FLOAT(IO)
    C(L)=CON5-FLOAT(LM1)*(CON1+CON2-CON3-CON4)/TPI
510 CONTINUE
    IF (IO-2) 520,540,610
520 PRINT 830, C(2)
    IF (REAL(C(2)).GT.SRM.OR.REAL(C(2)).LT.SRMI) GO TO 530
    IF (AIMAG(C(2)).GT.SIM.OR.AIMAG(C(2)).LT.SIMI) GO TO 530
    NOZ=NOZ+1
    CSA(NOZ)=C(2)
    CFUN=CFCTS(C(2))
    FUN(NOZ)=CABS(CFUN)
    AVE(NOZ)=AVEE
    PRINT 840, AVEE
    PRINT 850, FUN(NOZ)
    GO TO 700
530 PRINT 860
    GO TO 700
540 CSQ=CSQRT(-C(2)**2+2.*C(3))
    CR1=(C(2)+CSQ)/2.
    CR2=(C(2)-CSQ)/2.
    PRINT 870, CR1,CR2
    PRINT 840, AVEE
    IF (REAL(CR1).GT.SRM.OR.REAL(CR1).LT.SRMI) GO TO 560
    IF (AIMAG(CR1).GT.SIM.OR.AIMAG(CR1).LT.SIMI) GO TO 560
    NOZ=NOZ+1
    CSA(NOZ)=CR1
    CFUN=CFCTS(CR1)
```

```
      FUN(NOZ)=CABS(CFUN)
      AVE(NOZ)=CABS(CAVE)
      PRINT 880, FUN(NOZ)
  550 IF (REAL(CR2).GT.SRM.OR.REAL(CR2).LT.SRMI) GO TO 570
      IF (AIMAG(CR2).GT.SIM.OR.AIMAG(CR2).LT.SIMI) GO TO 570
      NOZ=NOZ+1
      CSA(NOZ)=CR2
      CFUN=CFCTS(CR2)
      FUN(NOZ)=CABS(CFUN)
      AVE(NOZ)=CABS(CAVE)
      PRINT 880, FUN(NOZ)
      GO TO 700
  560 PRINT 940
      GO TO 550
  570 PRINT 950
      GO TO 700
  580 IO=-IO
      PRINT 890, IO
      GO TO 700
  590 PRINT 900
      GO TO 700
  600 CONTINUE
      PRINT 910, IO
      GO TO 700
  610 CONTINUE
      CALF=((C(2)*C(2))-3.*C(3))/6.
      CBET=(-2.*(C(2)*C(2)*C(2))+9.*C(2)*C(3)-9.*C(4))/27.
      CQR=CSQRT((CBET**2)/4.+(CALF**3)/27.)
      CAS=CEXP((1./3.)*CLOG(-CBET/2.+CQR))
      CBS=CEXP((1./3.)*CLOG(-CBET/2.-CQR))
      EPS=1.E-5
      CQTY=-CALF/3.
      CI=(0.,1.)
      DO 630 IC=1,3
      TIC=FLOAT(IC)
      CF(IC)=CEXP(CI*TPI*TIC/3.)*CAS
      DO 620 JC=1,3
      TJC=FLOAT(JC)
      CS(JC)=CEXP(CI*TPI*TJC/3.)*CBS
      CP=CF(IC)*CS(JC)-CQTY
      DELR=ABS(REAL(CP))
      DELI=ABS(AIMAG(CP))
      IF (DELR.LE.EPS.AND.DELI.LE.EPS) GO TO 640
  620 CONTINUE
  630 CONTINUE
      PRINT 920
  640 CAS=CF(IC)
      CBS=CS(JC)
      CR1=CAS+CBS+C(2)/3.
      CR2=-(CAS+CBS)/2.+(CAS-CBS)*SQRT(3.)*(0.,1.)/2.+C(2)/3.
      CR3=-(CAS+CBS)/2.-(CAS-CBS)*SQRT(3.)*(0.,1.)/2.+C(2)/3.
      PRINT 930, IO,CR1,CR2,CR3
      PRINT 840, AVEE
      IF (REAL(CR1).GT.SRM.OR.REAL(CR1).LT.SRMI) GO TO 670
```

```
      IF (AIMAG(CR1).GT.SIM.OR.AIMAG(CR1).LT.SIMI) GO TO 670
      NOZ=NOZ+1
      CSA(NOZ)=CR1
      CFUN=CFCTS(CR1)
      FUN(NOZ)=CABS(CFUN)
      AVE(NOZ)=CABS(CAVE)
      PRINT 880, FUN(NOZ)
  650 IF (REAL(CR2).GT.SRM.OR.REAL(CR2).LT.SRMI) GO TO 680
      IF (AIMAG(CR2).GT.SIM.OR.AIMAG(CR2).LT.SIMI) GO TO 680
      NOZ=NOZ+1
      CSA(NOZ)=CR2
      CFUN=CFCTS(CR2)
      FUN(NOZ)=CABS(CFUN)
      AVE(NOZ)=CABS(CAVE)
      PRINT 880, FUN(NOZ)
  660 IF (REAL(CR3).GT.SRM.OR.REAL(CR3).LT.SRMI) GO TO 690
      IF (AIMAG(CR3).GT.SIM.OR.AIMAG(CR3).LT.SIMI) GO TO 690
      NOZ=NOZ+1
      CSA(NOZ)=CR3
      CFUN=CFCTS(CR3)
      FUN(NOZ)=CABS(CFUN)
      AVE(NOZ)=CABS(CAVE)
      PRINT 880, FUN(NOZ)
      GO TO 700
  670 PRINT 940
      GO TO 650
  680 PRINT 950
      GO TO 660
  690 PRINT 960
      GO TO 700
  700 PRINT 970
  710 CONTINUE
  720 CONTINUE
      PRINT 980
      IW=1
  730 RATIO=FUN(IW)/AVE(IW)
      PRINT 990, IW,CSA(IW),AVE(IW),FUN(IW),RATIO
      IW=IW+1
      IF (IW.LE.NOZ) GO TO 730
      IF (NOZ) 800,800,740
  740 IF (NH) 750,800,750
  750 CONTINUE
      PRINT 1000
      IHH=1
      DO 760 I=1,NOZ
      CALL HOMEIN (IHH,CSA(I),FUN(I),FACTOR)
  760 CONTINUE
      PRINT 1010
      IWH=1
  770 PRINT 1020, IWH,CSH(IWH),FZH(IWH)
      IWH=IWH+1
      IF (IWH.LT.IHH) GO TO 770
      IF ((IHH-1)-NOZ) 780,800,790
  780 PRINT 1030
```

```
    RETURN
790 PRINT 1040
800 RETURN
810 FORMAT (1H1,2X,56HCOORDINATES OF THE LOWER RIGHT CORNER OF SCAN AR
   1EA ARE (,1X,F12.8,1H,,F12.8,1X,1H),/,3X,56HCOORDINATES OF THE UPPE
   2R LEFT CORNER OF SCAN AREA ARE  (,1X,F12.8,1H,,F12.8,1X,1H),//,3X,
   343HTOTAL NUMBER OF POINTS USED PER CONTOUR ARE,2X,I4,//,3X,43HNUMB
   4ER OF DIVISIONS ALONG THE REAL AXIS ARE,2X,I2,5X,44HNUMBER OF DIVI
   5SIONS ALONG THE IMAG. AXIS ARE,2X,I4,//,3X,100H**********
   6**********
   7**********,//////)
820 FORMAT (1H0,2X,44HCOORDINATES OF THE LOWER RIGHT CORNER ARE  (,1X,
   1F12.8,1H,,F12.8,1X,1H),/,3X,44HCOORDINATES OF THE UPPER LEFT CORNE
   2R ARE   (,1X,F12.8,1H,,F12.8,1X,1H))
830 FORMAT (1H0,2X,37HTHERE IS ONE (1) ZERO IN THIS CONTOUR,/,3X,23HLO
   1CATION OF THE ZERO IS,2X,E24.14,3H+ J,E24.14,//)
840 FORMAT (1H0,2X,52HAVERAGE VALUE OF THE FUNCTION AROUND THIS CONTOU
   1R IS,2X,E24.14,/)
850 FORMAT (1H0,2X,40HMAGNITUDE OF THE FUNCTION AT THE ZERO IS,2X,E24.
   114,/)
860 FORMAT (1H0,2X,25HWARNING--WARNING--WARNING,/,3X,44HLOCATION OF TH
   1E ZERO IS OUTSIDE THE CONTOUR.,/,3X,109HFOR ACCURACY THIS CONTOUR
   2SHOULD BE REDONE WITH HIGHER NUMBER OF POINTS OR THE CONTOUR SIZE
   3SHOULD BE REDUCED,//)
870 FORMAT (1H0,2X,39HTHERE ARE TWO (2) ZEROS IN THIS CONTOUR,/,3X,26H
   1LOCATIONS OF THE ZEROS ARE,2X,3H(1),2X,E24.14,3H+ J,E24.14,/,31X,3
   2H(2),2X,E24.14,3H+ J,E24.14,//)
880 FORMAT (1H0,2X,66HABSOLUTE VALUE OF THE FUNCTION AT THE ZERO WITHI
   1N THE CONTOUR IS  ,2X,E24.14,/)
890 FORMAT (1H0,2X,9HTHERE ARE,1X,I2,1X,29HPOSSIBLE POLES IN THE CONTO
   1UR,////)
900 FORMAT (1H0,2X,34HTHERE ARE NO ZEROS IN THIS CONTOUR,////)
910 FORMAT (1H0,2X,10HTHERE ARE ,2X,I2,2X,21HZEROS IN THIS CONTOUR,/,3
   1X,70HTHIS SUBROUTINE CANNOT EVALUATE MORE THAN THREE (3) ZEROS IN
   2A CONTOUR,////)
920 FORMAT (1H0,2X,109HCOULD NOT MEET THE CRITERION IN SOLVING THE CUB
   1IC EQUATION. WILL TAKE THE PRINCIPLE VALUES FOR THE CUBE ROOTS,//)
930 FORMAT (1H0,2X,9HTHERE ARE,1X,I2,1X,21HZEROS IN THIS CONTOUR,/,3X,
   126HLOCATIONS OF THE ZEROS ARE,2X,3H(1),2X,E24.14,3H+ J,E24.14,/,31
   2X,3H(2),2X,E24.14,3H+ J,E24.14,/,31X,3H(3),2X,E24.14,3H+ J,E24.14,
   3//)
940 FORMAT (1H0,2X,25HWARNING--WARNING--WARNING,/,3X,34HFIRST  ZERO IS
   1 OUTSIDE THE CONTOUR,/)
950 FORMAT (1H0,2X,25HWARNING  WARNING  WARNING,/,3X,34HSECOND ZERO IS
   1 OUTSIDE THE CONTOUR,/)
960 FORMAT (1H0,2X,25HWARNING--WARNING--WARNING,/,3X,33HTHIRD ZERO IS
   1OUTSIDE THE CONTOUR,/)
970 FORMAT (1H0,2X,58H*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
   1*-*-*-*-* ,//)
980 FORMAT (1H1,53X,18HSUMMARY OF RESULTS,//,43X,39HLOCATIONS OF ZEROS
   1 CALCULATED BY SEARCH,//,2X,3HNO.,20X,21HZERO FROM SEARCH (ZS),18X
   2,17HAVERAGE MAGNITUDE,6X,13HMAG. OF F(ZS),13X,5HRATIO,///)
990 FORMAT (1H0,2X,I3,1X,E24.14,1X,3H+ J,1X,E24.14,3X,2(E18.10,3X),3X,
   1E18.10)
```

```
1000 FORMAT (1H1,2X,50H******* HOMEIN OPTION HAS BEEN EXCERCISED ******
    1**,/,3X,27HRESULTS ARE AS FOLLOWS.....,////)
1010 FORMAT (1H1,50X,18HSUMMARY OF RESULTS,//,48X,29HHOME-IN OPTION HAS
    1 BEEEN USED,///,2X,3HNO.,23X,22HZEROS FROM HOMEIN (ZH),20X,13HMAG.
    2 OF F(ZH),///)
1020 FORMAT (1H0,2X,I3,1X,E24.14,1X,3H+ J,1X,E24.14,3X,E15.8)
1030 FORMAT (1H0,2X,83HHOMEIN IMPROVED SOME OF THE RESULTS GIVEN BY SEA
    1RCH. FOR DETAILS SEE HOMEIN RESULTS)
1040 FORMAT (1H0,2X,45HHOMEIN FOUND MORE ZEROS THAN GIVEN BY SEARCH.,/,
    13X,26HIGNORE COMMON ZEROS IF ANY)
     END
```

```
      SUBROUTINE HOMEIN (IHH,CENTER,FUN,FACTOR)
      IMPLICIT COMPLEX (C)
      COMMON PI,TMAG(401),ARG(401),CS(401),CF(401),C(4),CSH(25),FZH(25)
      DIMENSION X(40), W(40)
      EXTERNAL CFCTS
      DATA ND/40/,X/-.998237709710559,-.990726238699457,-.97725994998377
     14,-.957916819213792,-.932812808278677,-.902098806968874,-.86595950
     23212260,-.824612230833312,-.778305651426519,-.727318255189927,-.67
     31956684614180,-.612553889667980,-.549467125095128,-.48307580168617
     49,-.413779204371605,-.341994090825758,-.268152185007254,-.19269758
     50701371,-.116084070675255,-.387724175060508E-1,.387724175060508E-1
     6,.116084070675255,.192697580701371,.268152185007254,.3419940908257
     758,.413779204371605,.483075801686179,.549467125095128,.61255388966
     87980,.671956684614180,.727318255189927,.778305651426519,.824612230
     9833312,.865959503212260,.902098806968874,.932812808278677,.9579168
     $19213792,.977259949983774,.990726238699457,.998237709710559/
      DATA W/.45212770985332E-2,.104982845311528E-1,.164210583819079E-1,
     1.222458491941670E-1,.279370069800234E-1,.334601952825478E-1,.38782
     21679744720E-1,.438709081856733E-1,.486958076350722E-1,.53227846983
     39368E-1,.574397690993916E-1,.613062424929289E-1,.648040134566010E-
     41,.679120458152339E-1,.706116473912868E-1,.728865823958041E-1,.747
     5231690579683E-1,.761103619006262E-1,.770398181642480E-1,.775059479
     6784248E-1,.775059479784248E-1,.770398181642480E-1,.761103619006262
     7E-1,.747231690579683E-1,.728865823958041E-1,.706116473912868E-1,.6
     879120458152339E-1,.648040134566010E-1,.613062424929289E-1,.5743976
     990993916E-1,.532278469839368E-1,.486958076350722E-1,.4387090818567
     $33E-1,.387821679744720E-1,.334601952825478E-1,.279370069800234E-1,
     $.222458491941670E-1,.164210583819079E-1,.104982845311528E-1,.45212
     $7709853319E-2/
      FUN=FUN/FACTOR
      R1=CABS(CENTER)
      IF (R1.LT.3.) RP=.08
      IF (R1.GE.3..AND.R1.LT.5.) RP=.06
      IF (R1.GE.5.AND.R1.LT.8.) RP=.04
      IF (R1.GE.8.) RP=.03
      RADIUS=RP*R1
      RAD=RADIUS
      RMIN=.7*RAD
      KDMX=10
      HPI=.5*PI
      PI3=1.5*PI
      TPI=2.*PI
   10 PRINT 340, CENTER,RADIUS
      KDM=2
   20 CONTINUE
      DT=TPI/FLOAT(KDM)
      DO 40 KD=1,KDM
      TL=DT*FLOAT(KD-1)
      TU=TL+DT
      DIF=(TU-TL)/2.
      DIFF=(TU+TL)/2.
      DO 30 I=1,ND
      II=I+(KD-1)*ND
      THETA=DIF*X(I)+DIFF
```

```
      CS(II)=CENTER+CMPLX(RADIUS*COS(THETA),RADIUS*SIN(THETA))
      CF(II)=CFCTS(CS(II))
   30 CONTINUE
   40 CONTINUE
      NDD=ND*KDM
      NM1=NDD+1
      NMM1=NDD-1
      A1=CABS(CF(1))
      A2=CABS(CF(KDM*ND/2+1))
      SRMI=REAL(CS(KDM*ND/2+1))
      SRM=REAL(CS(1))
      A3=ALOG(A1/A2)/(SRM-SRMI)
      A4=A1*EXP(-A3*SRM)
      DO 50 I=1,NDD
      CF(I)=CF(I)/(A4*CEXP(A3*CS(I)))
   50 CONTINUE
      DO 60 I=1,NDD
      TMAG(I)=ALOG(CABS(CF(I)))
      RF=REAL(CF(I))
      Z=AIMAG(CF(I))
      ARG(I)=ANGLER(RF,Z)
   60 CONTINUE
      IO=0
      OVF=ARG(1)
      OV=ARG(1)
      OVL=ARG(NDD)
      DO 120 K=1,NMM1
      AIO=FLOAT(IO)*TPI
      IF (OV.GT.PI3.AND.OV.LT.TPI) GO TO 70
      IF (OV.GT.0..AND.OV.LT.HPI) GO TO 80
      GO TO 110
   70 IF (ARG(K+1).GT.0..AND.ARG(K+1).LT.HPI) GO TO 90
      GO TO 110
   80 IF (ARG(K+1).GT.PI3.AND.ARG(K+1).LT.TPI) GO TO 100
      GO TO 110
   90 IO=IO+1
      GO TO 110
  100 IO=IO-1
  110 OV=ARG(K+1)
      ARG(K+1)=IO*TPI+ARG(K+1)
  120 CONTINUE
      IF (OVL.GT.PI3.AND.OVL.LT.TPI) GO TO 130
      IF (OVL.GT.0..AND.OVL.LT.HPI) GO TO 140
      GO TO 170
  130 IF (OVF.GT.0..AND.OVF.LT.HPI) GO TO 150
      GO TO 170
  140 IF (OVF.GT.PI3.AND.OVF.LT.TPI) GO TO 160
      GO TO 170
  150 IO=IO+1
      GO TO 170
  160 IO=IO-1
  170 CONTINUE
      ARG(NM1)=FLOAT(IO)*TPI+ARG(1)
      IF (IO.EQ.0) GO TO 320
```

```
      IF (IO.GT.3) GO TO 310
      DO 200 I=1,IO
      C(I)=(0.,0.)
      DO 190 JP=1,KDM
      JJ=JP-1
      DO 180 K=1,ND
      J=K+JJ*ND
      CC=CS(J)-CENTER
      THET=ANGLER(REAL(CC),AIMAG(CC))
      CS1=CEXP(CMPLX(0.,FLOAT(I)*THET))
      C(I)=C(I)+CS1*CMPLX(TMAG(J),ARG(J))*W(K)
  180 CONTINUE
  190 CONTINUE
      C(I)=(RADIUS**I)*(FLOAT(IO)-(FLOAT(I)*C(I)/TPI)*DIF)
  200 CONTINUE
      IF (IO-2) 210,240,260
  210 CONTINUE
      C(1)=CENTER+C(1)
      CFUN=CFCTS(C(1))
      FH=CABS(CFUN)
      IF (FH.LT.FUN) GO TO 230
  220 KDM=KDM+2
      IF (KDM.LE.KDMX) GO TO 20
      PRINT 350, NDD
      GO TO 330
  230 PRINT 360, C(1),FH
      CSH(IHH)=C(1)
      FZH(IHH)=FH
      IHH=IHH+1
      GO TO 330
  240 CSQ=CSQRT(-C(1)**2+2.*C(2))
      CR1=(C(2)+CSQ)/2.+CENTER
      CR2=(C(2)-CSQ)/2.+CENTER
      CFUN1=CFCTS(CR1)
      FUN1=CABS(CFUN1)
      CFUN2=CFCTS(CR2)
      FUN2=CABS(CFUN2)
      IF (FUN1.LT.FUN2) RFUN=FUN2
      IF (FUN2.LT.FUN1) RFUN=FUN1
      IF (RFUN.LT.FUN) GO TO 250
      GO TO 220
  250 PRINT 370, CR1,CR2,FUN1,FUN2
      CSH(IHH)=CR1
      FZH(IHH)=FUN1
      IHH=IHH+1
      CSH(IHH)=CR2
      FZH(IHH)=FUN2
      IHH=IHH+1
      GO TO 330
  260 CONTINUE
      CALF=((C(1)*C(1))-3.*C(2))/6.
      CBET=(-2.*(C(1)*C(1)*C(1))+9.*C(1)*C(2)-9.*C(3))/27.
      CQR=CSQRT((CBET**2)/4.+(CALF**3)/27.)
      CAS=CEXP((1./3.)*CLOG(-CBET/2.+CQR))
```

```
      CBS=CEXP((1./3.)*CLOG(-CBET/2.-CQR))
      EPS=1.E-5
      CQTY=-CALF/3.
      CI=(0.,1.)
      DO 280 IC=1,3
      TIC=FLOAT(IC)
      CF(IC)=CEXP(CI*TPI*TIC/3.)*CAS
      DO 270 JC=1,3
      TJC=FLOAT(JC)
      CS(JC)=CEXP(CI*TPI*TJC/3.)*CBS
      CP=CF(IC)*CS(JC)-CQTY
      DELR=ABS(REAL(CP))
      DELI=ABS(AIMAG(CP))
      IF (DELR.LE.EPS.AND.DELI.LE.EPS) GO TO 290
  270 CONTINUE
  280 CONTINUE
      PRINT 380
  290 CAS=CF(IC)
      CBS=CS(JC)
      CR1=CAS+CBS+C(1)/3.
      CR2=-(CAS+CBS)/2.+(CAS-CBS)*SQRT(3.)*(0.,1.)/2.+C(1)/3.
      CR3=-(CAS+CBS)/2.-(CAS-CBS)*SQRT(3.)*(0.,1.)/2.+C(1)/3.
      CR1=CR1+CENTER
      CR2=CR2+CENTER
      CR3=CR3+CENTER
      CFUN1=CFCTS(CR1)
      FUN1=CABS(CFUN1)
      CFUN2=CFCTS(CR2)
      FUN2=CABS(CFUN2)
      CFUN3=CFCTS(CR3)
      FUN3=CABS(CFUN3)
      IF (FUN1.LT.FUN2) RFUN=FUN2
      IF (FUN2.LT.FUN1) RFUN=FUN1
      IF (RFUN.LT.FUN3) RFUN=FUN3
      IF (RFUN.LT.FUN) GO TO 300
      GO TO 220
  300 PRINT 390, CR1,CR2,CR3,FUN1,FUN2,FUN3
      CSH(IHH)=CR1
      FZH(IHH)=FUN1
      IHH=IHH+1
      CSH(IHH)=CR2
      FZH(IHH)=FUN2
      IHH=IHH+1
      CSH(IHH)=CR3
      FZH(IHH)=FUN3
      IHH=IHH+1
      GO TO 330
  310 PRINT 400, IO
      RADIUS=RADIUS*.9
      IF (RADIUS.GT.RMIN) GO TO 10
      PRINT 410, IO,RMIN
      GO TO 330
  320 PRINT 420
      RADIUS=RADIUS*2.
```

```
      IF (RADIUS.LE.10.*RAD) GO TO 10
      PRINT 430
  330 CONTINUE
      PRINT 440
      RETURN
  340 FORMAT (1H0,2X,27HCENTER OF THE CONTOUR IS AT,2X,F12.8,3H+ J,F12.8
     1,/,3X,24HRADIUS OF THE CONTOUR IS,2X,F12.8,/)
  350 FORMAT (1H0,2X,38HHOMEIN CANNOT FIND A BETTER ZERO WITH ,2X,I4,2X,
     125HPOINTS AROUND THE CONTOUR,/)
  360 FORMAT (1H0,2X,33HTHERE IS ONE ZERO IN THIS CONTOUR,/,3X,24HLOCATI
     1ON OF THIS ZERO IS,2X,E24.14,3H+ J,E24.14,//,3X,41HFUNCTION VALUE
     2AT THE ABOVE LOCATION IS  ,2X,E24.14,////)
  370 FORMAT (1H0,2X,35HTHERE ARE TWO ZEROS IN THIS CONTOUR,/,3X,26HLOCA
     1TIONS OF THE ZEROS ARE,2X,3H(1),2X,E24.14,3H+ J,E24.14,/,31X,3H(2)
     2,2X,E24.14,3H+ J,E24.14,//,3X,48HABSOLUTE VALUES OF THE FUNCTION A
     3T THE ZEROS ARE,2X,3H(1),2X,E24.14,/,53X,3H(2),2X,E24.14,////)
  380 FORMAT (1H0,2X,109HCOULD NOT MEET THE CRITERION IN SOLVING THE CUB
     1IC EQUATION. WILL TAKE THE PRINCIPLE VALUES FOR THE CUBE ROOTS,//)
  390 FORMAT (1H0,2X,37HTHERE ARE THREE ZEROS IN THIS CONTOUR,/,3X,26HLO
     1CATIONS OF THE ZEROS ARE,2X,3H(1),2X,E24.14,3H+ J,E24.14,/,31X,3H(
     22),2X,E24.14,3H+ J,E24.14,/,31X,3H(3),2X,E24.14,3H+ J,E24.14,//,3X
     3,48HABSOLUTE VALUES OF THE FUNCTION AT THE ZEROS ARE,2X,3H(1),2X,E
     424.14,/,53X,3H(2),2X,E24.14,/,53X,3H(3),2X,E24.14,////)
  400 FORMAT (1H0,2X,9HTHERE ARE,1X,I2,21HZEROS IN THIS CONTOUR,/,3X,35H
     1... THE RADIUS IS BEING REDUCED ...,/)
  410 FORMAT (1H0,2X,9HTHERE ARE,2X,I2,2X,26HZEROS WITH IN A RADIUS OF ,
     12X,E24.14,2X,/,3X,67HTHIS SUBROUTINE CANNOT SOLVE MORE THAN THREE
     2ZEROS WITHIN A CONTOUR,/)
  420 FORMAT (1H0,2X,59HTHERE IS NO ZERO IN THIS CONTOUR... RADIUS IS BE
     1ING CHANGED,/)
  430 FORMAT (1H0,2X,85HHOMEIN COULD NOT FIND A ZERO WITH IN 8 PERCENT R
     1ADIUS OF THE LOCATION GIVEN BY SEARCH,/)
  440 FORMAT (1H0,2X,58H*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
     1*-*-*-*-* ,//)
      END
```

```
      SUBROUTINE CONTOUR (CSL,CSF,NR,NI,RATIO)
      IMPLICIT COMPLEX (C)
      COMMON PI,CSA(25),FUN(25),RAT(25),AVE(25),NOZ
      EXTERNAL CFCTS
      NOZ=0
      RCSF=REAL(CSF)
      SCIF=AIMAG(CSF)
      RCSL=REAL(CSL)
      SCIL=AIMAG(CSL)
      RINC=(RCSL-RCSF)/FLOAT(NR)
      SCIINC=(SCIL-SCIF)/FLOAT(NI)
      PRINT 60, RCSF,SCIF,RCSL,SCIL,NR,NI
      DO 20 J=1,NR
      JJ=J-1
      SRMI=RCSF+RINC*FLOAT(J)
      SRM=RCSF+RINC*FLOAT(JJ)
      DO 10 I=1,NI
      II=I-1
      SIM=SCIF+SCIINC*FLOAT(I)
      SIMI=SCIF+SCIINC*FLOAT(II)
      CSM=CMPLX(SRMI,SIM)
      CSMI=CMPLX(SRM,SIMI)
      CALL SEEK (CFCTS,CSM,CSMI,RATIO)
   10 CONTINUE
   20 CONTINUE
      IF (NOZ) 30,30,40
   30 PRINT 70
      RETURN
   40 PRINT 80
      IW=1
   50 PRINT 90, IW,CSA(IW),AVE(IW),FUN(IW),RAT(IW)
      IW=IW+1
      IF (IW.LE.NOZ) GO TO 50
      RETURN
   60 FORMAT (1H1,2X,56HCOORDINATES OF THE LOWER RIGHT CORNER OF SCAN AR
     1EA ARE (,1X,F12.8,1H,,F12.8,1X,1H),/,3X,56HCOORDINATES OF THE UPPE
     2R LEFT CORNER OF SCAN AREA ARE  (,1X,F12.8,1H,,F12.8,1X,1H),//,3X,
     343HNUMBER OF DIVISIONS ALONG THE REAL AXIS ARE,2X,I3,5X,43HNUMBER
     40F DIVISIONS ALONG THE IMAG.AXIS ARE,2X,I4,//,3X,104H**********
     5**********************************************************************
     6**************************,/////)
   70 FORMAT (1H1,2X,46HSEEK COULD NOT FIND ANY ZEROS IN THE SCAN AREA)
   80 FORMAT (1H1,53X,18HSUMMARY OF RESULTS,//,44X,39HLOCATIONS OF ZEROS
     1 CALCULATED BY SEARCH,//,2X,3HNO.,20X,19HZERO FROM SEEK (ZS),20X,1
     27HAVERAGE MAGNITUDE,6X,13HMAG. OF F(ZS),13X,5HRATIO,///)
   90 FORMAT (1H0,2X,I3,1X,E24.16,3H+ J,E24.16,3X,2(E18.10,3X),3X,E18.10
     1)
      END
```

```
      SUBROUTINE SEEK (CFCTS,CSM,CSMI,RATIO)
      IMPLICIT COMPLEX (C)
      EXTERNAL CFCTS
      COMMON PI,CSA(25),FUN(25),RAT(25),AVE(25),NOZ
      DIMENSION ARG(801), C(4), CF(801), CS(801), TMAG(801), W(20), X(20
     1)
      DATA NP/20/,X/-.9931285991850949247 86,-.9639719272779137912 68,-.91
     1223442825132590586 8,-.8391169718222188233 95,-.7463319064601507926 1
     24,-.6360536807265150254 53,-.5108670017508270980024,-.3737060887154
     319560637,-.2277858511416450780 80,-.0765265211334973337 55,.07652652
     41133497333755,.2277858511416450780 80,.3737060887154195606 37,.51086
     570017508270980024,.6360536807265150254 53,.7463319064601507926 14,.8
     639116971882339 5,.9122344282513259058 68,.9639719272779137912 68,.993
     7128599185094924786/
      DATA W/.0176140071391521183 12,.0406014298003869413 31,.062672048334
     1109063570,.0832767415767047487 25,.1019301198172404350 73,.118194531
     296158417312,.1316886384491766268 98,.1420961093183820513 29,.1491729
     386472603746788,.1527533871307258506 98,.1527533871307258506 98,.1491
     472986472603746788,.1420961093183820513 29,.1316886384491766268 98,.1
     518194531961518417312,.1019301198172404350 37,.0832767415767047487 25
     6,.0626720483341090635 70,.0406014298003869413 31,.0176140071391521183
     7312/
      PI3=1.5*PI
      TPI=2.*PI
      HPI=.5*PI
      RCSF=REAL(CSMI)
      RCSL=REAL(CSM)
      SCIF=AIMAG(CSMI)
      SCIL=AIMAG(CSM)
      NR=1
      NI=1
      KDMX=8
      NBRK=0
   10 RINT=(RCSL-RCSF)/FLOAT(NR)
      SCINT=(SCIL-SCIF)/FLOAT(NI)
      DO 660 JT=1,NR
      JJT=JT-1
      SRMI=RCSF+RINT*FLOAT(JT)
      SRM=RCSF+RINT*FLOAT(JJT)
      DO 650 IT=1,NI
      IIT=IT-1
      SIMI=SCIF+SCINT*FLOAT(IIT)
      SIM=SCIF+SCINT*FLOAT(IT)
      PRINT 670, SRM,SIMI,SRMI,SIM
      KDM=2
   20 ND=KDM*NP
      ND1=ND+1
      ND2=2*ND
      ND3=3*ND
      ND4=4*ND
      NM1=ND4+1
      NMM1=ND4-1
      NDP=ND2+1
      DO 140 K=1,4
```

```
      KK=K-1
      KKK=KK*KDM*NP
      IF (K-2) 40,50,30
   30 IF (K-3) 50,60,70
   40 YU=SIM
      YL=SIMI
      XC=SRM
      GO TO 110
   50 XU=SRMI
      XL=SRM
      YC=SIM
      GO TO 80
   60 YU=SIMI
      YL=SIM
      XC=SRMI
      GO TO 110
   70 XU=SRM
      XL=SRMI
      YC=SIMI
   80 DL=(XU-XL)/FLOAT(KDM)
      DO 100 L=1,KDM
      LL=L-1
      LLL=LL*NP+KKK
      XLOW=XL+FLOAT(LL)*DL
      XUP=XLOW+DL
      DLX=(XUP-XLOW)/2.
      PLX=(XUP+XLOW)/2.
      DO 90 M=1,NP
      MM=M+LLL
      CS(MM)=CMPLX(DLX*X(M)+PLX,YC)
   90 CONTINUE
  100 CONTINUE
      GO TO 140
  110 DL=(YU-YL)/FLOAT(KDM)
      DO 130 L=1,KDM
      LL=L-1
      LLL=LL*NP+KKK
      YLOW=YL+FLOAT(LL)*DL
      YUP=YLOW+DL
      DLY=(YUP-YLOW)/2.
      PLY=(YUP+YLOW)/2.
      DO 120 M=1,NP
      MM=M+LLL
      CS(MM)=CMPLX(XC,DLY*X(M)+PLY)
  120 CONTINUE
  130 CONTINUE
  140 CONTINUE
      DO 150 K=1,ND4
      CF(K)=CFCTS(CS(K))
  150 CONTINUE
      CS(NM1)=CS(1)
      ASUM=0.
      DO 160 K=1,ND
      ASUM=ASUM+CABS(CF(K))
```

```
160 CONTINUE
    A1=ASUM/FLOAT(ND)
    ASUM=0.
    DO 170 K=NDP,ND3
    ASUM=ASUM+CABS(CF(K))
170 CONTINUE
    A2=ASUM/FLOAT(ND)
    A3=ALOG(A1/A2)/(SRM-SRMI)
    A4=A1*EXP(-A3*SRM)
    CSUM=(0.,0.)
    DO 180 K=1,ND4
    CSUM=CSUM+CF(K)
    CF(K)=CF(K)/(A4*CEXP(A3*CS(K)))
180 CONTINUE
    CAVE=CSUM/FLOAT(ND4)
    AVEE=CABS(CAVE)
    DO 190 L=1,ND4
    RF=REAL(CF(L))
    Z=AIMAG(CF(L))
    ARG(L)=ANGLER(RF,Z)
    TMAG(L)=ALOG(CABS(CF(L)))
190 CONTINUE
    IO=0
    OVF=ARG(1)
    OV=ARG(1)
    OVL=ARG(ND4)
    DO 250 K=1,NMM1
    IF (OV.GT.PI3.AND.OV.LT.TPI) GO TO 200
    IF (OV.GT.0..AND.OV.LT.HPI) GO TO 210
    GO TO 240
200 IF (ARG(K+1).GT.0..AND.ARG(K+1).LT.HPI) GO TO 220
    GO TO 240
210 IF (ARG(K+1).GT.PI3.AND.ARG(K+1).LT.TPI) GO TO 230
    GO TO 240
220 IO=IO+1
    GO TO 240
230 IO=IO-1
240 OV=ARG(K+1)
    ARG(K+1)=IO*TPI+ARG(K+1)
250 CONTINUE
    IF (OVL.GT.PI3.AND.OVL.LT.TPI) GO TO 260
    IF (OVL.GT.0..AND.OVL.LT.HPI) GO TO 270
    GO TO 300
260 IF (OVF.GT.0..AND.OVF.LT.HPI) GO TO 280
    GO TO 300
270 IF (OVF.GT.PI3.AND.OVF.LT.TPI) GO TO 290
    GO TO 300
280 IO=IO+1
    GO TO 300
290 IO=IO-1
300 CONTINUE
    ARG(NM1)=FLOAT(IO)*TPI+ARG(1)
    IF (IO) 530,540,310
310 CONTINUE
```

```
      IO1=IO+1
      IF (IO.GE.4) GO TO 550
      DO 470 L=1,IO1
      CON1=(0.,0.)
      CON2=(0.,0.)
      CON3=(0.,0.)
      CON4=(0.,0.)
      LM1=L-1
      LL=LM1-1
      DO 460 K=1,4
      KK=(K-1)*KDM*NP
      IF (K-2) 330,360,320
  320 IF (K-3) 360,390,420
  330 DO 350 M=1,KDM
      MM=(M-1)*NP
      DO 340 N=1,NP
      NN=KK+MM+N
      CON1=CON1+(CS(NN)**LL)*W(N)*CMPLX(TMAG(NN),ARG(NN))
  340 CONTINUE
  350 CONTINUE
      GO TO 450
  360 DO 380 M=1,KDM
      MM=(M-1)*NP
      DO 370 N=1,NP
      NN=KK+MM+N
      CON2=CON2+(CS(NN)**LL)*W(N)*CMPLX(-ARG(NN),TMAG(NN))
  370 CONTINUE
  380 CONTINUE
      GO TO 450
  390 DO 410 M=1,KDM
      MM=(M-1)*NP
      DO 400 N=1,NP
      NN=KK+MM+N
      CON3=CON3+(CS(NN)**LL)*W(N)*CMPLX(TMAG(NN),ARG(NN))
  400 CONTINUE
  410 CONTINUE
      GO TO 450
  420 DO 440 M=1,KDM
      MM=(M-1)*NP
      DO 430 N=1,NP
      NN=KK+MM+N
      CON4=CON4+(CS(NN)**LL)*W(N)*CMPLX(-ARG(NN),TMAG(NN))
  430 CONTINUE
  440 CONTINUE
  450 CONTINUE
  460 CONTINUE
      DELX=(SRM-SRMI)/(2.*FLOAT(KDM))
      DELY=(SIM-SIMI)/(2.*FLOAT(KDM))
      CON1=CON1*DELY
      CON2=CON2*DELX
      CON3=CON3*DELY
      CON4=CON4*DELX
      CON5=((CMPLX(SRM,SIMI))**LM1)*FLOAT(IO)
      C(L)=CON5-FLOAT(LM1)*(CON1+CON2-CON3-CON4)/TPI
```

```
470 CONTINUE
    IF (IO-2) 480,500,580
480 CFUN=CFCTS(C(2))
    FUN1=CABS(CFUN)
    RAT1=FUN1/AVEE
    IF (RAT1.LE.RATIO) GO TO 490
    KDM=KDM+2
    IF (KDM.LE.KDMX) GO TO 20
    PRINT 680, ND4
490 NOZ=NOZ+1
    CSA(NOZ)=C(2)
    FUN(NOZ)=FUN1
    RAT(NOZ)=RAT1
    AVE(NOZ)=AVEE
    PRINT 690, C(2)
    PRINT 700, AVEE
    PRINT 710, FUN(NOZ)
    GO TO 640
500 CSQ=CSQRT(-C(2)**2+2.*C(3))
    CR1=(C(2)+CSQ)/2.
    CR2=(C(2)-CSQ)/2.
    CFUN1=CFCTS(CR1)
    FUN1=CABS(CFUN1)
    RAT1=FUN1/AVEE
    CFUN2=CFCTS(CR2)
    FUN2=CABS(CFUN2)
    RAT2=FUN2/AVEE
    IF (RAT1.GT.RATIO) GO TO 520
    IF (RAT2.GT.RATIO) GO TO 520
510 NOZ=NOZ+1
    CSA(NOZ)=CR1
    FUN(NOZ)=FUN1
    RAT(NOZ)=RAT1
    AVE(NOZ)=AVEE
    NOZ=NOZ+1
    CSA(NOZ)=CR2
    FUN(NOZ)=FUN2
    RAT(NOZ)=RAT2
    AVE(NOZ)=AVEE
    PRINT 720, CR1,CR2
    PRINT 700, AVEE
    PRINT 730, FUN1,FUN2
    GO TO 640
520 KDM=KDM+2
    IF (KDM.LE.KDMX) GO TO 20
    PRINT 680, ND4
    GO TO 510
530 IO=-IO
    PRINT 740, IO
    GO TO 640
540 PRINT 750
    GO TO 640
550 IF (NBRK) 560,560,570
560 PRINT 760, IO
```

```
      NT=IO/3+1
      NBRK=1
      GO TO 640
  570 PRINT 770, IO
      GO TO 640
  580 CONTINUE
      CALF=((C(2)*C(2))-3.*C(3))/6.
      CBET=(-2.*(C(2)*C(2)*C(2))+9.*C(2)*C(3)-9.*C(4))/27.
      CQR=CSQRT((CBET**2)/4.+(CALF**3)/27.)
      CAS=CEXP((1./3.)*CLOG(-CBET/2.+CQR))
      CBS=CEXP((1./3.)*CLOG(-CBET/2.-CQR))
      EPS=1.E-5
      CQTY=-CALF/3.
      CI=(0.,1.)
      DO 600 IC=1,3
      TIC=FLOAT(IC)
      CF(IC)=CEXP(CI*TPI*TIC/3.)*CAS
      DO 590 JC=1,3
      TJC=FLOAT(JC)
      CS(JC)=CEXP(CI*TPI*TJC/3.)*CBS
      CP=CF(IC)*CS(JC)-CQTY
      DELR=ABS(REAL(CP))
      DELI=ABS(AIMAG(CP))
      IF (DELR.LE.EPS.AND.DELI.LE.EPS) GO TO 610
  590 CONTINUE
  600 CONTINUE
      PRINT 780
  610 CAS=CF(IC)
      CBS=CS(JC)
      CR1=CAS+CBS+C(2)/3.
      CR2=-(CAS+CBS)/2.+(CAS-CBS)*SQRT(3.)*(0.,1.)/2.+C(2)/3.
      CR3=-(CAS+CBS)/2.-(CAS-CBS)*SQRT(3.)*(0.,1.)/2.+C(2)/3.
      CFUN1=CFCTS(CR1)
      FUN1=CABS(CFUN1)
      RAT1=FUN1/AVEE
      CFUN2=CFCTS(CR2)
      FUN2=CABS(CFUN2)
      RAT2=FUN2/AVEE
      CFUN3=CFCTS(CR3)
      FUN3=CABS(CFUN3)
      RAT3=FUN3/AVEE
      IF (RAT1.GT.RATIO) GO TO 630
      IF (RAT2.GT.RATIO) GO TO 630
      IF (RAT3.GT.RATIO) GO TO 630
  620 NOZ=NOZ+1
      CSA(NOZ)=CR1
      FUN(NOZ)=FUN1
      RAT(NOZ)=RAT1
      AVE(NOZ)=AVEE
      NOZ=NOZ+1
      CSA(NOZ)=CR2
      FUN(NOZ)=FUN2
      RAT(NOZ)=RAT2
      AVE(NOZ)=AVEE
```

```
      NOZ=NOZ+1
      CSA(NOZ)=CR3
      FUN(NOZ)=FUN3
      RAT(NOZ)=RAT3
      AVE(NOZ)=AVEE
      PRINT 790, IO,CR1,CR2,CR3
      PRINT 700, AVEE
      PRINT 800, FUN1,FUN2,FUN3
      GO TO 640
  630 KDM=KDM+2
      IF (KDM.LE.KDMX) GO TO 20
      PRINT 680, ND4
      GO TO 620
  640 CONTINUE
  650 CONTINUE
  660 CONTINUE
      NR=NT
      NI=NT
      IF (NBRK.NE.0) GO TO 10
      PRINT 810
      RETURN
  670 FORMAT (1H0,2X,44HCOORDINATES OF THE LOWER RIGHT CORNER ARE  (,1X,
     1F12.8,1H,,F12.8,1X,1H),/,3X,44HCOORDINATES OF THE UPPER LEFT CORNE
     2R ARE   (,1X,F12.8,1H,,F12.8,1X,1H))
  680 FORMAT (1H0,2X,37HRATIO CONDITION COULD NOT BE MET WITH,2X,I5,2X,2
     15HPOINTS AROUND THE CONTOUR,/,3X,40HRESULTS OF THE LAST COMPUTATIO
     2N ARE ....,//)
  690 FORMAT (1H0,2X,37HTHERE IS ONE (1) ZERO IN THIS CONTOUR,/,3X,23HLO
     1CATION OF THE ZERO IS,2X,E24.16,3H+ J,E24.16,//)
  700 FORMAT (1H0,2X,52HAVERAGE VALUE OF THE FUNCTION AROUND THIS CONTOU
     1R IS,2X,E24.16,/)
  710 FORMAT (1H0,2X,40HMAGNITUDE OF THE FUNCTION AT THE ZERO IS,2X,E24.
     116,/)
  720 FORMAT (1H0,2X,39HTHERE ARE TWO (2) ZEROS IN THIS CONTOUR,/,3X,26H
     1LOCATIONS OF THE ZEROS ARE,2X,3H(1),2X,E24.16,3H+ J,E24.16,/,31X,3
     2H(2),2X,E24.16,3H+ J,E24.16,//)
  730 FORMAT (1H0,2X,33HFUNCTION VALUES AT THE ZEROS ARE ,3X,3H(1),3X,E2
     14.16,/,39X,3H(2),3X,E24.16,////)
  740 FORMAT (1H0,2X,9HTHERE ARE,1X,I2,1X,29HPOSSIBLE POLES IN THE CONTO
     1UR,////)
  750 FORMAT (1H0,2X,34HTHERE ARE NO ZEROS IN THIS CONTOUR,////)
  760 FORMAT (1H0,2X,9HTHERE ARE,2X,I4,2X,21HZEROS IN THIS CONTOUR,/,3X,
     153HTHIS CONTOUR WILL BE BROKEN UP INTO SMALLER CONTOURS.,/)
  770 FORMAT (1H0,2X,23HTHE SMALLER CONTOUR HAS,2X,I4,2X,5HZEROS,3X,47HT
     1HISSUBROUTINE CAN BREAK UP A CONTOUR ONLY ONCE,///)
  780 FORMAT (1H0,2X,109HCOULD NOT MEET THE CRITERION IN SOLVING THE CUB
     1IC EQUATION. WILL TAKE THE PRINCIPLE VALUES FOR THE CUBE ROOTS,//)
  790 FORMAT (1H0,2X,9HTHERE ARE,1X,I2,1X,21HZEROS IN THIS CONTOUR,/,3X,
     126HLOCATIONS OF THE ZEROS ARE,2X,3H(1),2X,E24.16,3H+ J,E24.16,/,31
     2X,3H(2),2X,E24.16,3H+ J,E24.16,/,31X,3H(3),2X,E24.16,3H+ J,E24.16,
     3//)
  800 FORMAT (1H0,2X,32HFUNCTION VALUES AT THE ZEROS ARE,3X,3H(1),3X,E24
     1.16,/,38X,3H(2),3X,E24.16,/,38X,3H(3),3X,E24.16,////)
  810 FORMAT (1H0,2X,58H*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
```

```
1*-*-*-*-* ,//)
 END
```

```
      FUNCTION ANGLER (X,Y)
      COMMON PI
      IF (X) 90,10,50
   10 IF (Y) 30,20,40
   20 ANGLER=0.
      RETURN
   30 ANGLER=1.5*PI
      RETURN
   40 ANGLER=PI*.5
      RETURN
   50 IF (Y) 80,60,70
   60 ANGLER=0.
      RETURN
   70 ANGLER=ATAN(Y/X)
      RETURN
   80 ANGLER=-ATAN(-Y/X)+2.*PI
      RETURN
   90 XN=-X
      IF (Y) 120,100,110
  100 ANGLER=PI
      RETURN
  110 ANGLER=PI-ATAN(Y/XN)
      RETURN
  120 ANGLER=PI+ATAN(-Y/XN)
      RETURN
      END
```

References

1. C. E. Baum, "On the Singularity Expansion Method for the Solution of Electromagnetic Interaction Problems," Interaction Notes, Note 88, December 11, 1971.

2. F. M. Tesche, "On the Singularity Expansion Method as Applied to Electromagnetic Scattering from Thin Wires," Interaction Notes, Note 102, April 1972.

3. C. E. Baum, "On the Singularity Expansion Method for the Case of First Order Poles," Interaction Notes, Note 129, October 24, 1972.

4. J. A. Stratton, Electromagnetic Theory, McGraw-Hill, 1941.

5. L. Marin, "Natural Modes of Certain Thin-Wire Structures," Interaction Notes, Note 186, August 1974.

6. E. Hallén, "Über die Elektrischen Schwingungen in Drahtförmigen Leitern," Uppsala Univ., Årsskrift, No. 1, pp. 1-102, 1930.

7. L. M. Delves and J. N. Lyness, "A Numerical Method for Locating the Zeros of an Analytic Function," Math. Comp., V. 21, 1967, pp. 543-560.

8. J. N. Lyness and L. M. Delves, "On Numerical Contour Integration Round a Closed Contour," Math. Comp., V. 21, 1967, pp. 561-577.

9. C. O. Beasley and H. K. Meier, "Subroutine Cauchy: Complex Roots of a Function Using a Cauchy Integral Technique," Mathematics Notes, Note 37, August 1974.

10. E. Hille, Analytic Function Theory, Vols. I, II, Ginn and Company, 1962.

11. C. E. Baum, "On the Use of Contour Integration for Finding Poles, Zeros, Saddles and Other Function Values in the Singularity Expansion Method," Mathematics Notes, Note 35, February 18, 1974.

12. Handbook of Mathematical Functions, edited by M. Abramowitz and I. A. Stegun, Dover, 1965.

13. Rektorys, Survey of Applicable Mathematics, M.I.T. Press, 1969.

14. D. V. Giri, B. K. Singaraju and C. E. Baum, "Accurate Evaluation of the Impedance Matrix Elements in the Hallén Integral Equation for Wire Antennas," Mathematics Notes, Note 40, December 1975.

15. F. M. Tesche, "Application of the Singularity Expansion Method to the Analysis of Impedance Loaded Linear Antennas," Sensor and Simulation Notes, Notes 177, May 1973.

16. C. E. Baum, D. V. Giri and R. D. González, "Electromagnetic Field Distribution of the TEM Mode in a Symmetrical Two-Parallel-Plate Transmisssion Line," Sensor and Simulation Notes, to be published.

17. B. K. Singaraju and R. L. Gardner, "SGEMP Induced Currents on a Helical Antenna on a Ground Plane," Theoretical Notes, to be published.