Mathematics Notes
Note 55

25 May 1978

Application of Cauchy's Residue Theorem in Evaluating the Poles and Zeros of Complex Meromorphic Functions and Apposite Computer Programs

D. V. Giri
Science Applications, Inc., Berkeley, CA.
and
Carl E. Baum
Air Force Weapons Laboratory

Abstract

This note addresses itself to the problem of locating the zeros and poles of a complex meromorphic function M(s) in a specified rectangular or square region of the complex s-plane. It is assumed that M(s) has to be computed numerically as for example, i) a dispersion relation in plasma physics or ii) the system determinant of the matricized integral equation while employing the singularity expansion method (SEM)[1] to solve electromagnetic scattering problems. The procedure developed here eliminates the usual 2-dimensional search and replaces it with a direct constructive method for determining the poles of M(s) based on an application of Cauchy's residue theorem. The zeros of M(s) are easily found by applying the procedure to the reciprocal function 1/M(s). Two examples, i.e., 1) ratios of polynomials and 2) input impedance of a biconical antenna, are numerically illustrated.

TABLE OF CONTENTS

ACKNOWLEDGMENT

The authors are thankful to Dr. G. Scrivner of Computer Sciences Corporation, Albuquerque, and Dr. B.K. Singaraju of the Air Force Weapons laboratory for useful discussions.

List of Principal Symbols

1.	s	$\equiv$ Complex variable $= s_R + s_I = \Omega + j\omega$
2.*	$A(s)$	= Analytic function of s in a domain D
3.*	$M(s)$	= Meromorphic function of s in a domain D
	so, in general $M(s) = A_1(s)/A_2(s)$	
4.	$N_a(f,C)$	= Argument number of the function $f(s)$ in a prescribed counterclockwise or positive Jordan contour C $\equiv N_o(f,C) - N_p(f,C)$ (The arguments f and C may be omitted when obvious)
5.	$N_o(f,C)$	= Number of zeros of $f(s)$ in C
6.	$N_p(f,C)$	= Number of poles of $f(s)$ in C
7.	Argument number	$\equiv$ Excess number of zeros over poles
8.	$(z_1, z_2, \ldots.)$ and $(p_1, p_2, \ldots.)$	Locations of zeros and poles of a meromorphic function
9.	$M^{nor}(s)$	= Normalized version of $M(s)$

* In addition to being analytic on the contour C which encloses the domain D, $A(s)$ and $M(s)$ are required not to vanish on C.

List of Figures

I. Introduction

The problem of locating the poles and zeros of complex functions in a finite domain of the complex plane, occurs in many scientific disciplines e.g., dispersion relations in plasma physics, the singularity expansion method [1] in electromagnetic scattering or antenna problems. In an earlier note [2] Singaraju, et al. described a technique of locating the zeros of analytic functions in a given region of the complex plane. This note also included relevant computer programs and illustrative examples by way of i) polynomials ii) product of polynomials and exponentials and iii) determination of natural frequencies of a thin straight wire.

A sequential extension of the above mentioned work [2] is, of course, a method of locating poles and zeros of meromorphic functions when they coexist in a given contour. It is interesting to note that the word "meromorphic" is derived [3] from the Greek μερος = fraction and μορφή = form, and means "like a fraction." In keeping with the origin of the word "meromorphic," the complex function $M(s)$ considered in this note will be a ratio of two entire functions of the complex variable s. The problem at hand can now be defined in terms of given and required quantities, as follows:

<u>Given</u>:

i) A numerical way of evaluating a meromorphic complex function $M(s)$ of a complex variable s,

ii) A rectangular or a square region in the finite complex s-plane.

<u>Find</u>:

i) All the zeros and poles of $M(s)$ in the given region.

<u>Remark</u>:

i) Typically, evaluation of $M(s)$ is expensive in terms of computer time and hence it is desirable to optimize the number of $M(s)$ computations.

II. A Review of SGB Technique for Finding the Zeros of Analytic Functions

In a recent note [2], Singaraju, Giri and Baum described a technique of locating the zeros of an analytic function $A(s)$ in a finite domain D of the complex s-plane. This work, referred to as the SGB Technique also includes a family of computer programs titled SEARCH. This technique is based on the "principle of the argument" and a generalization thereof. The principle of argument for an analytic function is given by [4]

$$\frac{1}{2\pi i} \oint_C \frac{A'(s)}{A(s)}\, ds = N_o(A,C) \tag{2.1}$$

where

$A(s)$ = Analytic function* of s in a domain D enclosed by a simple contour C,

$N_o(A,C)$ = Number of zeros of the analytic function $A(s)$ inside the contour C.

Equation (2.1) is a special case of

$$\frac{1}{2\pi i} \oint_C A_m(s) \frac{A'(s)}{A(s)}\, ds = \sum_{\alpha=1}^{N_o} A_m(s_\alpha) \tag{2.2}$$

obtained by setting $A_m(s) = 1$. In equation (2.2) $A_m(s)$ is an analytic (at least in and on C) multiplier function and s_α are the zeros of $A(s)$ in C. If we choose $A_m(s) = s^n$ and consider n to take integer values ranging from 0 to N_o, we have

$$C_n = \frac{1}{2\pi i} \oint_C s^n \frac{A'(s)}{A(s)}\, ds; \quad \text{for } n = 0, 1, 2, \ldots N_o \tag{2.3}$$

which leads to

---

* $A(s)$ is required not to vanish on the contour C.

$$C_0 = 1 + 1 + 1 + 1 \ldots\ldots + 1 = N_o \qquad (2.4.0)$$

$$C_1 = s_1 + s_2 + s_3 + s_4 \ldots + s_{N_o} \qquad (2.4.1)$$

$$C_2 = s_1^2 + s_2^2 + s_3^2 + s_4^2 \ldots + s_{N_o}^2 \qquad (2.4.2)$$

$$\vdots$$

$$C_{N_o} = s_1^{No} + s_2^{No} + s_3^{No} + s_4^{No} + s_{N_o}^{No} \qquad (2.4.N_o)$$

After determining the moments (C_n) SEARCH proceeds to locate the zeros in the given contour C (if any) by solving the above system of equations. By way of an interesting example, SEARCH has been used in easing the chase for those "elusive and ubiquitous" [5] SEM poles of a thin wire.

Above is a rather brief description of the underlying basis of the SGB technique and the interested reader is referred to Mathematics Note 42 [2] for all of the details regarding the working, limitations and use of the relevant computer programs. A logical extension of this work is of course a method of locating the zeros and poles of a meromorphic function in a finite region of the complex plane. In the following section, a method is developed which determines only the poles of a meromorphic function in a region regardless of whether or not there are zeros in that region. By applying the pole finding method of section III to the given function as well as its reciprocal, the zeros and poles of the given function in the given region are successfully located.

III. Poles and Zeros of Meromorphic Functions

A. Pole finder

In this section we shall develop a procedure to determine the number of poles $[N_p(M,C)]$ and their locations of a meromorphic function $M(s)$ in a given contour C. This procedure is independent of the presence or absence of zeros in C and also the actual shape of the contour C itself. However, for purposes of illustration and numerical ease, we shall consider the contour C to be a square as in figure 3.1 which in some special cases may be rectangular. We will also stipulate that the side of the square is equal to or not very different from unit length in the normalized s plane of figure 3.1.

The meromorphic function is representable by a ratio of two entire functions $E_1(s)$ and $E_2(s)$ as

$$M(s) = \frac{E_1(s)}{E_2(s)}$$

$$= \frac{A_{1_{int}}(s)\ A_{1_{ext}}(s)}{A_{2_{int}}(s)\ A_{2_{ext}}(s)} \qquad (3.1)$$

$E_1(s)$ and $E_2(s)$ are in turn written as a product of an interior and an exterior analytic function. The subscripts "interior" and "exterior" are with reference to the contour C of figure 3.1. The "exterior" functions are required to be analytic in and on C, and not vanish on C. Our procedure of finding poles inside contour C allows for other types of singularities like essential or branch point to occur outside and sufficiently away from the contour C. The poles of $M(s)$ within C are of course the zeros of the following equation

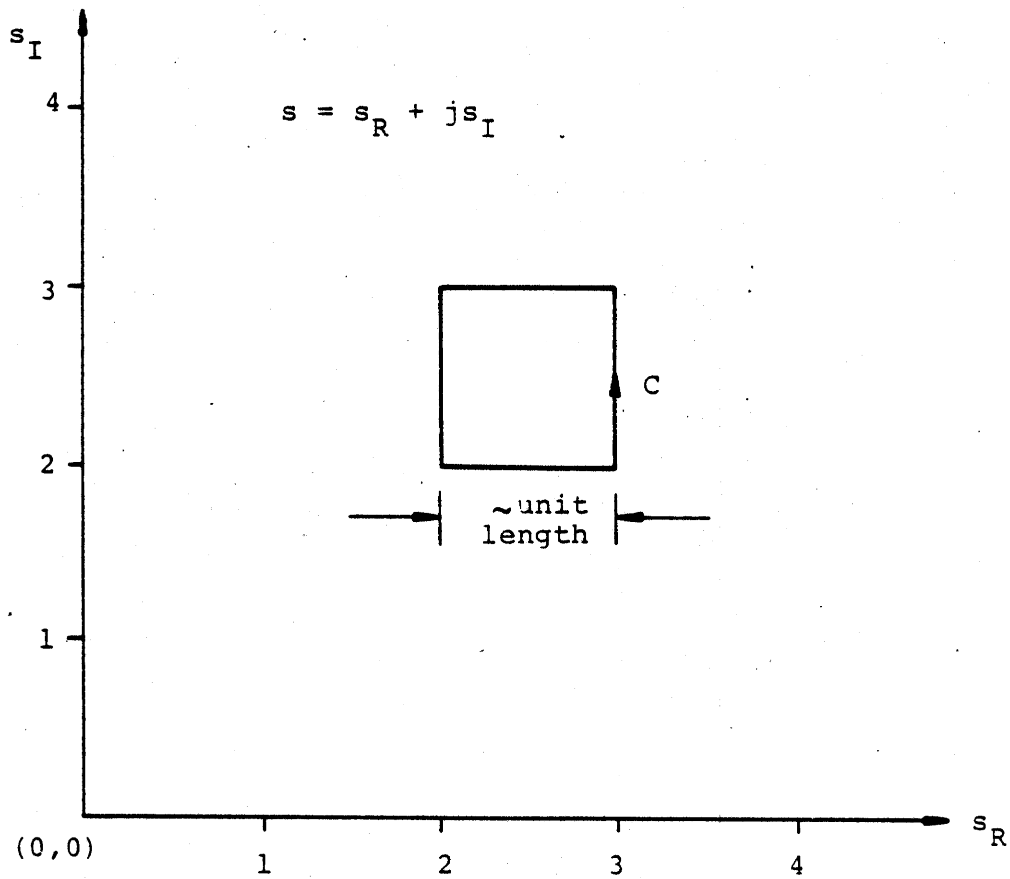

Figure 3.1 Normalized s-plane showing a simple square contour C.

$$A_{2_{int}}(s) = 0 \tag{3.2}$$

Any analytic function can be represented by a suitable polynomial so that,

$$A_{1_{int}}(s) = \prod_{i=1}^{N_o} (s - z_i) \tag{3.3}$$

$$A_{1_{ext}}(s) = E_1(s)/A_{1_{int}}(s) \tag{3.4}$$

$$A_{2_{int}}(s) = \prod_{k=1}^{N_p} (s - p_k) \tag{3.5}$$

$$A_{2_{ext}}(s) = E_2(s)/A_{2_{int}}(s) \tag{3.6}$$

leading to

$$M(s) = \frac{\left[\prod_{i=1}^{N_o} (s - z_i)\right]\left[A_{1_{ext}}(s)\right]}{\left[\prod_{k=1}^{N_p} (s - p_k)\right]\left[A_{2_{ext}}(s)\right]} \tag{3.7}$$

where

where

z_i 's are zeros of $M(s)$ within C,

p_k 's are poles of $M(s)$ within C,

N_o = number of zeros of $M(s)$ within c,

and

N_p = number of poles of $M(s)$ within C.

Our pole finding scheme determines N_p and subsequently p_k for $k = 1, 2, \ldots\ldots N_p$.

i) Finding the number of poles N_p

Besides the function values, what distinguishes a pole from a zero is the concept of residue and Cauchy's residue theorem. If the poles (p_k 's) are simple then the corresponding residues are given by

$$R_m = \lim_{s \to p_m} \left[(s - p_m) M(s) \right]$$

$$= \lim_{s \to p_m} \left[(s - p_m) \frac{\prod_{i=1}^{N_o} (s - z_i) \quad A_{1_{ext}} (s)}{\prod_{k=1}^{N_p} (s - p_k) \quad A_{2_{ext}} (s)} \right]$$

$$= \left[\frac{\prod_{i=1}^{N_o} (p_m - z_i) \quad A_{1_{ext}} (p_m)}{\prod_{\substack{k=1 \\ k \neq m}}^{N_p} (p_m - p_k) \quad A_{2_{ext}} (p_m)} \right] \quad ; \quad \text{for } m = 1, 2 \ldots . N_p \tag{3.8}$$

However, for the present purpose, R_m's are of no interest and will eventually be eliminated.

We will now define residue moments by

$$D_n \triangleq \frac{1}{2\pi j} \oint_C s^n M(s) \, ds \quad ; \text{ for } n = 0, 1, 2, \ldots 2N_p \tag{3.9}$$

In terms of the residues of equation (3.8), the residue moments are given by

$$D_0 = R_1 + R_2 + \ldots + R_{N_p} = \sum_{q=1}^{N_p} R_q \tag{3.10.0}$$

$$D_1 = p_1 R_1 + p_2 R_2 + \ldots + p_{N_p} R_{N_p} = \sum_{q=1}^{N_p} p_q R_q \tag{3.10.1}$$

$$\vdots$$

$$D_{N_p} = p_1^{N_p} R_1 + p_2^{N_p} R_2 + \ldots + p_{N_p}^{N_p} R_{N_p} = \sum_{q=1}^{N_p} p_q^{N_p} R_q \tag{3.10.N_p}$$

$$\vdots$$

$$D_{2N_p} = p_1^{2N_p} R_1 + p_2^{2N_p} R_2 + \ldots + p_{N_p}^{2N_p} R_{N_p} = \sum_{q=1}^{N_p} p_q^{2N_p} R_q \tag{3.10.$2N_p$}$$

The above system of equations can also be written compactly as

$$D_n = \sum_{q=1}^{N_p} p_q^n R_q; \quad \text{for} \quad n = 0, 1, 2, \ldots, 2N_p \tag{3.11}$$

Let us recall that we are trying to determine the order and the zeros of the polynomial $A_{2_{int}}(s)$ which give the number and locations of the poles of $M(s)$ within C.

Let

$$A_{2_{int}}(s) = \prod_{k=1}^{N_p} (s - p_k) \equiv \sum_{k=0}^{N_p} a_k s^k \qquad (3.12)$$

where the coefficient a_{N_p} of the highest degree term may be set equal to 1 without any loss of generality. We shall now eliminate the residues (R_q's) from the system of equations (3.10) by making use of equation (3.12). To achieve this, consider the first (N_p+1) number of equations in (3.10) starting with (3.10.0) and ending with (3.10.N_p). Multiplying these equations respectively by the coefficients a_0 to a_{N_p}, would yield

$$a_0D_0 + a_1D_1 + a_2 D_2 \cdots + a_{N_p-1}D_{N_p-1} + a_{N_p}D_{N_p}$$

$$= R_1\left(a_0+a_1p_1+a_2p_1^2 + \cdots\cdots\cdots + a_{N_p-1}p_1^{N_p-1} + a_{N_p}p_1^{N_p}\right)$$

$$+ R_2\left(a_0+a_1p_2+a_2p_2^2 + \cdots\cdots\cdots + a_{N_p-1}p_2^{N_p-1} + a_{N_p}p_2^{N_p}\right)$$

$$\vdots$$

$$+ R_{N_p}\left(a_0+a_1p_{N_p}+a_2p_{N_p}^2 + \cdots\cdots + a_{N_p-1}p_{N_p}^{N_p-1} + a_{N_p}p_{N_p}^{N_p}\right)$$

$$= \sum_{q=1}^{N_p} R_q \left[\sum_{k=0}^{N_p} a_k \, p_q^k \right]$$

$$= \sum_{q=1}^{N_p} \left[R_q \, A_{2_{int}} (p_q) \right]$$

$$= 0 \tag{3.13}$$

because p_q for $q = 1, 2, \ldots N_p$ are the zeros of $A_{2_{int}}(s) = 0$. Thus we have

$$a_0 D_0 + a_1 D_1 + a_2 D_2 + \ldots\ldots\ldots\ldots + a_{N_p} D_{N_p} = 0 \tag{3.14}$$

Continuing this above procedure of successively multiplying a set of $(N_p + 1)$ equations from the system of equation (3.10) and using equation (3.12) will eliminate all of the residues (R_q; for $q = 1, 2, \ldots N_p$) and lead to the following matrix equation

$$\begin{bmatrix} D_0 & D_1 & D_2 & \cdots & D_{N_p} \\ D_1 & D_2 & D_3 & & D_{N_p+1} \\ D_2 & D_3 & D_4 & & D_{N_p+2} \\ \vdots & & & & \vdots \\ D_{N_p} & D_{N_p+1} & D_{N_p+2} & \cdots & D_{2N_p} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{N_p} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3.15a}$$

or

$$\sum_{n=0}^{N_p} D_{m+n} \, a_n = 0; \quad \text{for} \quad m = 0, 1, 2 \ldots N_p \tag{3.15b}$$

In this matrix equation, D's are the residue moments defined by equation (3.9) and a's are the coefficients of the polynomial $A_{2_{int}}(s)$, the zeros of which, are the poles of given meromorphic function M(s) within the contour C. From equation (3.15), we observe the following

a). If $N_p = 0$, then

$$D_i = 0 \;; \quad \text{for} \quad i = 0, 1, 2, \ldots \tag{3.16}$$

b) If $N_p = 1$

$$D_i + a_0 D_{i-1} = 0 \;; \quad \text{for} \quad i = 1, 2, 3, \ldots \tag{3.17}$$

c) If $N_p = 2$

$$D_i + a_1 D_{i-1} + a_0 D_{i-2} = 0 \;; \quad \text{for} \quad i = 2, 3, 4.. \tag{3.18}$$

d) If $N_p = 3$

$$D_i + a_2 D_{i-1} + a_1 D_{i-2} + a_0 D_{i-3} = 0 \;; \quad \text{for} \quad i = 3, 4, 5 \ldots \tag{3.19}$$

e) If $N_p = 4$

$$D_i + a_3 D_{i-1} + a_2 D_{i-2} + a_1 D_{i-3} + a_0 D_{i-4} = 0 \;; \quad \text{For} \quad i = 4, 5, 6, 7, \ldots \tag{3.20}$$

... etc.

Put differently, N_p will be $= R - 1$, where R = Rank of the infinite version of the D matrix of equation (3.15). However equations (3.16) thru (3.20 ...) are more useful in determining N_p, because as a by-product they yield the coefficients a_k for $k = 0, 1, 2, \ldots, N_p$, as well. This will be illustrated as follows, for example, if $N_p = 3$, equation (3.19) for $i = 3$, 4 and 5 will give

$$\begin{aligned} D_3 + a_2 D_2 + a_1 D_1 + a_0 D_0 &= 0 \\ D_4 + a_2 D_3 + a_1 D_2 + a_0 D_1 &= 0 \\ D_5 + a_2 D_4 + a_1 D_3 + a_0 D_2 &= 0 \end{aligned} \qquad (3.21)$$

which may be used in solving for a_0, a_1 and a_2. These a's may then be used in

$$D_i + a_2 D_{i-1} + a_1 D_{i-2} + a_0 D_{i-3} = 0 \; ; \quad \text{for } i = 6, 7, 8 \ldots \qquad (3.22)$$

to ensure that N_p is indeed 3. With the value of N_p and the coefficients of $A_{2_{int}}(s)$ polynomial known, it is a simple matter to solve for the locations p_k of the poles of the given meromorphic function $M(s)$ within the contour C being considered.

It is emphasized, at this stage that there are a few numerical pitfalls in implementing this scheme and section IV will address these problems specifically.

B. Zero finder

We still need to find the number N_o and locations z_i, $i = 1, 2, \ldots N_o$ of the zeros of the given meromorphic function $M(s)$ in the given contour C. This is a rather trivial numerical exercise by virtue of the fact that the pole finder described above is independent of the presence or absence of zeros. In view of this, if we worked with the reciprocal function $M^{-1}(s)$, the zeros which now become poles inside contour C, are easily determined by using the pole finding scheme.

IV. Numerical Implementation and Results

In this section, we deal with the numerical implementation of the pole and zero finding schemes described in the preceding section.

Given the function and a rectangular region C in the normalized complex plane, we initially divide the region C into a number of subcontours of approximately unit sized square regions (see Figure 4.1). Improved accuracy is obtained by centering each subcontour around the point $1 + j0$ in the complex plane, via a simple change of variable. The need for this change of variable is explained in detail, later in this section, while describing the subroutine RESIDUE in which all of the residue moments of equation (3.9) are computed. Function values are computed at locations on the subcontour, determined by a 40-point Gaussian quadrature integration scheme and these values are stored in a complex array. In a sequential fashion, the stored function values are recalled and normalized for each of the subcontours of approximate size unity per side. The exponential normalization of the function, intended to improve the quality and accuracy of the location of the singularities (poles), is well described in the previous work under section III C of reference [2]. It is noted that the normalization function is an entire function with no zeros or poles in or on the subcontour, so that the singularities of the original function $M(s)$ are undisturbed. With reference to the typical subcontour $C_{m,n}$ shown in Figure 4.1, the entire function $E(s)$ used in the process of normalization is given by

$$E(s) = u\, e^{vs} \tag{4.1}$$

where u and v are real constants given by

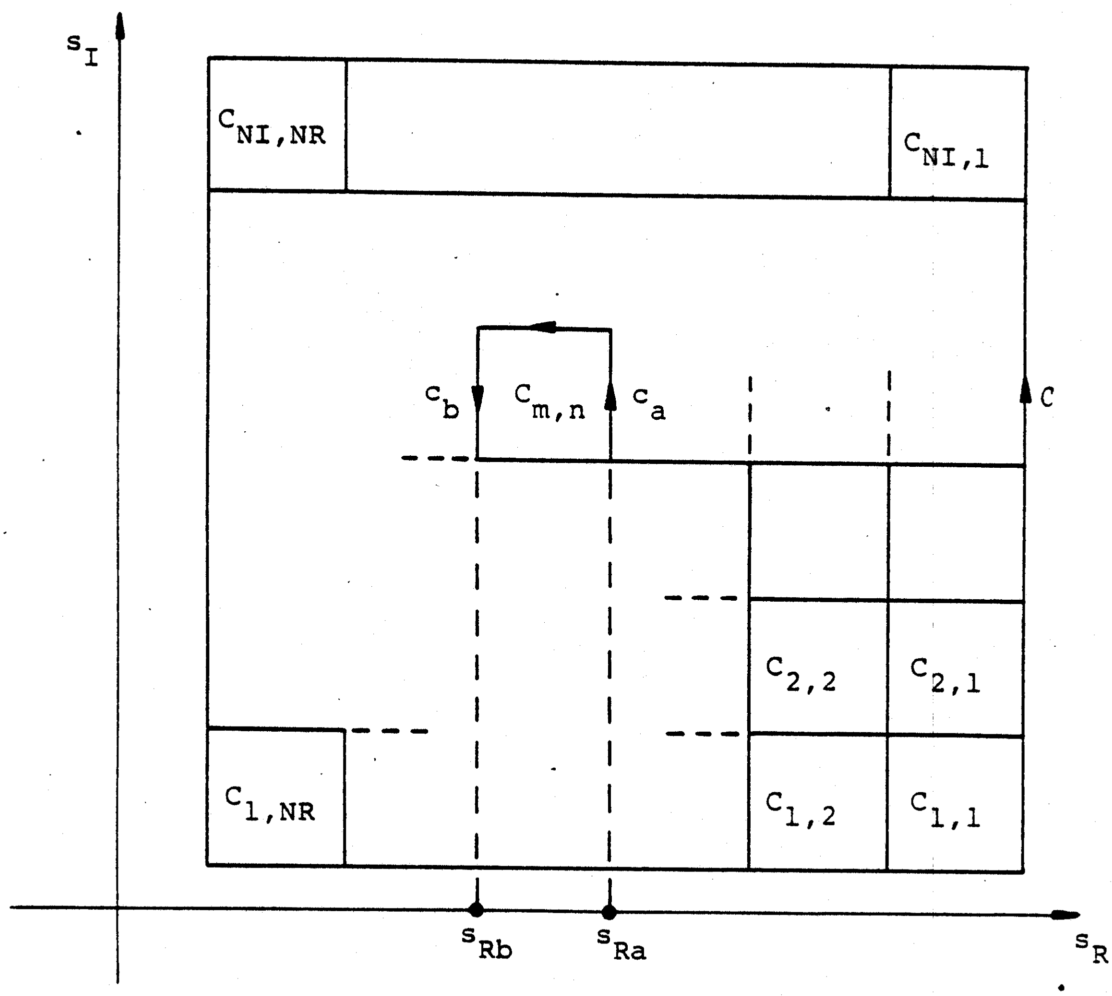

Figure 4.1 Division of the given rectangular domain into smaller rectangular (or square) subcontours

Note. $C_{m,n}$ is a typical subcontour and c_a and c_b are the parts of $C_{m,n}$ with constant real parts s_{Ra} and s_{Rb} respectively.

$$v = \frac{1}{(s_{Ra} - s_{Rb})} \ln \left(\frac{\text{average of } |M(s| \text{ on } c_a}{\text{average of } |M(s)| \text{ on } c_b} \right) \tag{4.2}$$

$$u = \exp(-vs_{Ra}) \times (\text{average of } |M(s)| \text{ on } c_a)$$

or (4.3)

$$= \exp(-vs_{Rb}) \times (\text{average of } |M(s)| \text{ on } c_b)$$

Using the above entire function, the normalized function $M^{nor}(s)$ is obtained by using

$$M^{nor}(s) = M(s)/E(s) \tag{4.4}$$

With this normalization scheme, the average magnitude of $M(s)$ on c_a and c_b is unity and in general, does not depart significantly from unity on the rest of the subcontour. Also, effects of large phase variations in $M(s)$ tend to become reduced when one works with $M^{nor}(s)$ instead. It is emphasized that, $E(s)$ being an entire function does nct disturb the singularities of the given function $M(s)$.

A. Description of the Computer Programs

The following subroutines and function subprograms were written and they are useful in numerically evaluating the locations of poles and zeros of the given function M(s) which is meromorphic in a given region of the finite complex s-plane.

CONTOUR*, RESIDE, POLECHK, DETER*, ANGLER*,

POLY1, POLY2, POLY3, CFCTS

(* from Reference 2)

In what follows, we shall briefly describe each of these subroutines. Their listings are included in Appendix A.

a) Subroutine CONTOUR (CSL, CSF, NR, NI, KDM)

This program divides up the given scan area enclosed by the contour into a specified number of rectangular/square subcontours and calls subroutine RESIDUE once for each of the contours. The arguments appearing in this subroutine are as follows.

- CSL : Coordinates of the upper left corner of C,
- CSF : Coordinates of the lower right corner of C,
- NR : Number of major divisions of the real axis within C,
- NI : Number of major divisions of the imaginary axis within C,
- KDM : A multiplicative factor for Gaussian integration, i.e., the number of points in the integration is given by $40 \times$ KDM per side of the subcontour. Although the largest allowable value is 4 because of the present dimensioning of the arrays in RESIDUE, KDM = 1 should be adequate in most cases.

This subroutine also summarizes the results by listing the location of all the poles and zeros found as well as the function values at the zeros and the reciprocal of the function value at the poles. These function values may be used by the user in judging the quality of pole-zero locations which are numerically determined.

b) Subroutine RESIDUE (CFCTS, CSM, CSMI, KDM)

This is the core subroutine in the entire package and essentially does the following:

1) makes a change of variable so that the subcontour is now centered around the point $(1 + j0)$ in the complex plane

2) computes the function value on this new sub-contour at locations required by a 40-point Gaussian integration procedure, (the locations and the function values are stored in complex arrays CS and CF respectively),

3) normalizes the function values and computes the argument number and nine residue moments D_n for $n = 0, 1, \ldots, 8$, of equation (3.9),

4) calls subroutine POLECHK which determines a potential value of the number of poles N_p, the coefficients of the denominator polynomial and the pole locations as well,

and 5) goes through a similar procedure, working with the reciprocal function, to determine the location of zeros.

Various arguments of this subroutine are described below

CFCTS : User supplied function subprogram,

CSM : Coordinates of the upper left corner of the subcontour $C_{m,n}$,

CSMI : Coordinates of the lower right corner of the subcontour $C_{m,n}$,

KDM : Same as in subroutine CONTOUR.

In computing the residue moments, given by

$$D_n = \frac{1}{2\pi j} \oint_{C_{m,n}} s^n \, M(s) \, ds \qquad (4.5)$$

for $n = 0, 1, 2, \ldots, 8$

a change of variable of the following form was found to improve the accuracy of the above integration,

$$z = s - s_c - 1 \qquad (4.6)$$

where,

s_c = coordinates of the center point of the subcontour $C_{m,n}$.

The reason for this change of variable is that when the subcontour $C_{m,n}$ is located away from the origin, the numerical value of the factor s^n in the integrand can become quite large compared with the average magnitude of unity for the normalized $M(s)$ around the subcontour $C_{m,n}$. With the change of variable given by equation (4.6), the new subcontour in the z-plane is centered around the point $1 + j0$ so that the entire integrand will now have an average magnitude of unity resulting in improved accuracy for the numerical evaluation of the residue moments.

c) Subroutine POLECHK (DO, D, DAO, DA, NP)

This subroutine accepts the nine residue moments as input and determines a potential value for the number of poles N_p and the coefficients of the denominator polynomial as well. The various arguments of this subroutine are:

- DO : Zeroth residue moment D_0 which is simply the sum of the residues,
- D : A complex array which contains the residue moments D_n for $n = 1, 2, \ldots, 8$,
- DAO : Constant term in the denominator polynomial (a_0 of equation (3.12)),
- DA : A complex array which contains the remaining coefficients of the denominator polynomial,
- NP : Most likely value of the number of poles N_p in the subcontour.

d) Subroutine DETER (CM, CB, CW, CV, MS, CD)

This subroutine finds the determinant of a given square matrix. A description of the various arguments of this subroutine is given below.

CM	:	Complex array continaing the matrix elements,
CB, CW and CV	:	Complex working arrays, local to the subroutine,
MS	:	Size of the input square matrix,
CD	:	Computed value of the determinant.

e) FUNCTION ANGLER (X, Y)

This function subprogram computes the phase ϕ of a complex number in radians such that $0 \leq \phi \leq 2\pi$. The arguments are:

X	:	Real part of the complex number,
Y	:	Imaginary part of the complex number,
ANGLER	:	Phase ϕ of the complex number $X + j\,Y$ such that $0 \leq \phi \leq 2\pi$.

It was important to determine the phase in this range of $0 \leq \phi \leq 2\pi$ for obtaining the argument number, rather than the commonly available range of $-\pi \leq \phi \leq \pi$.

f) Subroutines POLY1 (C0, C1, CLIN)
POLY2 (C0, C1, C2, CQUAD)
POLY3 (C0, C1, C2, C3, CUBE)

These three subroutines respectively solve for 1, 2 and 3 roots of the following linear, quadratic and cubic polynomials

$$C_0 + C_1 s = 0 \tag{4.7a}$$

$$C_0 + C_1 s + C_2 s^2 = 0 \tag{4.7b}$$

$$C_0 + C_1 s + C_2 s^2 + C_3 s^3 = 0 \tag{4.7c}$$

when the appropriate coefficients are fed in. The arguments appearing in the three subroutines are:

C0, C1, C2, C3	:	Coefficients of the polynomial,
CLIN	:	Root of the linear equation (4.7a),
CQUAD	:	Two roots of the quadratic equation (4.7b),
CUBE	:	Three roots of the cubic equation (4.7c).

g) COMPLEX FUNCTION CFCTS (CS, CSHIFT)

This complex function subprogram numerically evaluates the meromorphic function M(s) for a prescribed s. The two arguments and the result of the function subprogram are

CS	:	Complex value of the variable s,
CSHIFT	:	Complex constant to facilitate a change of variable (can be set equal to zero),
CFCTS	:	Function value.

This completes a brief description of the various subroutines and function subprograms in this package. It is recalled that, in the subroutine RESIDUE, we computed the nine residue moments D_n for $n = 0, 1, \ldots, 8$, which introduces a limitation of no more than 3 poles in a subcontour which is of approximate size unity square. Also when the number of zeros and poles $(N_0 + N_p)$ in any subcontour is more than 4, because of large changes in the function value in a small region, the residue moment calculations may not be sufficiently accurate to determine the exact value of N_p. We have encountered no difficulty when $(N_0 + N_p) \le 4$ per subcontour and some modifications to improve the accuracy may be required for the rare occurrence in physical problems, when the poles and zeros are more closely bunched together. When $(N_0 + N_p) > 4$ per subcontour, another possibility is to divide the subcontour of unit size into smaller subcontours. This has not been automated in the present family of computer programs.

In the following two subsections, we consider two examples that validate the application of this pole finding scheme to determine the poles and zeros of given complex functions that are meromorphic in a specified region of the finite complex plane.

B. Ratio of Polynomials (Example 1)

Consider a meromorphic function $M_1(s)$ given by,

$$M_1(s) = \frac{\prod_{p=1}^{16} (s-z_p)}{\prod_{q=1}^{10} (s-p_q)} \qquad (4.8)$$

which is readily seen to have 16 zeros and 10 poles in the complex s-plane. The locations of zeros indicated by z, and poles indicated by p are shown in Figure 4.2. The zeros are given by;

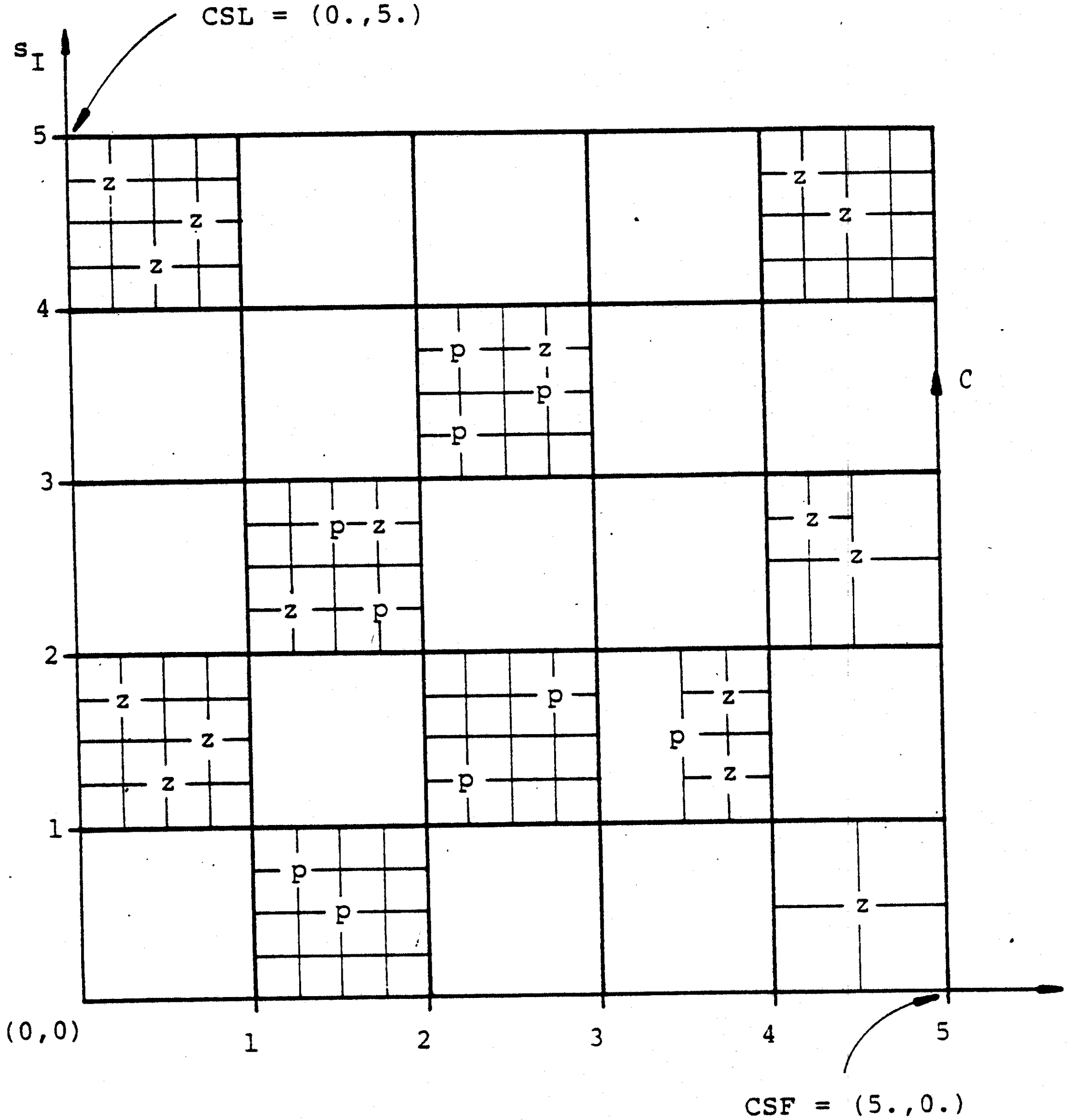

Figure 4.2. The pole-zero configuration of the given ratio of polynomials (example 1).

Note: 1) C is the given square scan area, uniquely specified by the points CSF and CSL

2) z → location of a zero

3) p → location of a pole

$z_1 = 4.50+j0.50$, $z_2 = 4.25+j4.75$, $z_3 = 3.75+j1.25$, $z_4 = 3.75+j1.75$
$z_5 = 4.50+j2.50$, $z_6 = 4.25+j2.75$, $z_7 = 2.75+j3.75$, $z_8 = 1.25+j2.25$
$z_9 = 1.75+j2.75$, $z_{10} = 4.50+j4.50$, $z_{11} = 0.75+j1.50$, $z_{12} = 0.50+j1.25$
$z_{13} = 0.25+j1.75$, $z_{14} = 0.50+j4.25$, $z_{15} = 0.75+j4.50$, $z_{16} = 0.25+j4.75$
(4.9)

The poles are given by:

$p_1 = 3.50+j1.50$, $p_2 = 1.75+j2.25$, $p_3 = 1.50+j2.75$, $p_4 = 2.25+j3.25$
$p_5 = 2.75+j3.50$, $p_6 = 2.25+j3.75$, $p_7 = 1.25+j0.75$, $p_8 = 2.75+j1.75$
$p_9 = 1.50+j0.50$, $p_{10} = 2.25+j1.25$
(4.10)

In Figure 4.2, the scan area which is a square of side 5 units with its lower left corner as the origin of the complex s-plane is indicated by the counterclockwise contour C. This contour is divided into 25 subcontours and the pole-zero locations are indicated in the various subcontours. The formal input variables CSL and CSF that uniquely specify the scan area are also shown in the Figure 4.2. The results of using this function in order to recover the poles and zeros are presented in Table 1, in the same format as the subroutine CONTOUR would summarize. Comparing the results of Table 1 with those of equations (4.9) and (4.10), it is seen that the poles and zeros of this ratio of polynomial given by $M_1(s)$ are recovered with a high degree of accuracy. With the view of introducing large phase variations in $M_1(s)$, it was multiplied by an entire function of the form e^{AS}. Define a new function $N_1(s)$ as

$$N_1(s) = e^{AS} M_1(s) \tag{4.11}$$

where $M_1(s)$ is given by equation (4.8). The real

SUMMARY OF RESULTS

	Real part	Imag. part		Real part	Imag. part
1 POLE AT	.35000000E+01	.15000000E+01	1/F(POLE) =	-.51481437E-16	-.79124576E-16
2 POLE AT	.27500000E+01	.17500000E+01	1/F(POLE) =	-.16026682E-15	-.52103539E-16
3 POLE AT	.22500000E+01	.12500000E+01	1/F(POLE) =	-.27757148E-16	-.70485840E-17
4 POLE AT	.22500000E+01	.37499999E+01	1/F(POLE) =	.34166970E-10	.48627420E-10
5 POLE AT	.22500000E+01	.32500000E+01	1/F(POLE) =	.25440087E-10	.64503667E-11
6 POLE AT	.27499999E+01	.35000000E+01	1/F(POLE) =	.63817493E-10	-.92700647E-10
7 POLE AT	.15000000E+01	.50000000E+00	1/F(POLE) =	.85008163E-17	-.32501785E-17
8 POLE AT	.12500000E+01	.75000000E+00	1/F(POLE) =	.15115588E-16	-.46924316E-17
9 POLE AT	.17500000E+01	.22500000E+01	1/F(POLE) =	-.18510407E-14	.46241384E-15
10 POLE AT	.15000000E+01	.27500000E+01	1/F(POLE) =	-.94138859E-14	.24127978E-14
1 ZERO AT	.45000000E+01	.50000000E+00	F(ZERO) =	-.33974973E-10	.37801892E-10
2 ZERO AT	.45000000E+01	.25000000E+01	F(ZERO) =	.71209024E-11	-.91285320E-11
3 ZERO AT	.42500000E+01	.27500000E+01	F(ZERO) =	.14853889E-11	-.10834960E-10
4 ZERO AT	.45000000E+01	.45000000E+01	F(ZERO) =	.60244057E-11	.38265879E-11
5 ZERO AT	.42500000E+01	.47500000E+01	F(ZERO) =	.31923752E-11	.52167555E-11
6 ZERO AT	.37500000E+01	.17500000E+01	F(ZERO) =	-.34168506E-11	.23797403E-09
7 ZERO AT	.37500000E+01	.12500000E+01	F(ZERO) =	-.27022936E-09	.28549864E-09
8 ZERO AT	.27500000E+01	.37500000E+01	F(ZERO) =	-.10640196E-09	-.31899668E-09
9 ZERO AT	.17500000E+01	.27500000E+01	F(ZERO) =	-.32643289E-08	-.54749864E-09
10 ZERO AT	.12500000E+01	.22500000E+01	F(ZERO) =	-.48766628E-09	.13125755E-09
11 ZERO AT	.25000000E+00	.17500000E+01	F(ZERO) =	-.25715938E-06	.28969053E-06
12 ZERO AT	.50000000E+00	.12500000E+01	F(ZERO) =	-.55518217E-06	.30493810E-06
13 ZERO AT	.75000000E+00	.15000000E+01	F(ZERO) =	-.48376999E-06	.23146903E-06
14 ZERO AT	.25000000E+00	.47500000E+01	F(ZERO) =	.30761459E-07	.49960265E-07
15 ZERO AT	.50000000E+00	.42500000E+01	F(ZERO) =	.31526461E-08	.25607578E-07
16 ZERO AT	.75000000E+00	.45000000E+01	F(ZERO) =	.22238213E-07	.14267090E-07

Table 1. The poles and zeros of $M_1(s)$ found by the present method

constant A in the exponent was varied and the results (pole-zero locations) did not vary significantly in regions where the singularities were not dense. This can be attributed to the efficient exponential normalization procedure, described earlier. An improved scheme of determining N_p will be necessary for highly oscillatory functions.

C. Input Impedance of a Biconical Antenna (Example 2)

The input impedance of a biconical antenna may be written as [6,7],

$$Z_{in}(k\ell) \simeq Z_c \left[\frac{e^{jk\ell} + T(k\ell)\, e^{-jk\ell}}{e^{jk\ell} - T(k\ell)\, e^{-jk\ell}} \right] \qquad (4.12)$$

where

$\ell \equiv$ slant height of the cone

$Z_c \equiv$ characteristic impedance of a symmetrical bicone

$$= \frac{Z_o}{\pi} \ell n \left(\cot \frac{\theta}{2}\right) \simeq \frac{Z_o}{\pi} \ell n \left(\frac{2}{\theta}\right) \text{ for small angles}$$

$Z_o \equiv$ characteristic impedance of free space

$\theta \equiv$ half angle of the bicone in radians

$k \equiv$ free space propagation constant

$T(k\ell) \equiv$ effective terminal reflection coefficient

Rewriting the input impedance in the normalized Laplace transform variable plane

$$\mathfrak{s} = \frac{s\ell}{\pi c} \qquad (4.13)$$

$$\tilde{Z}_{in}(s) \equiv \left(\frac{Z_{in}}{Z_c}\right) = \left\{\frac{e^{2\pi s} + \tilde{T}(s)}{e^{2\pi s} - \tilde{T}(s)}\right\} \tag{4.14}$$

where

$$k\ell = -j\pi s \tag{4.15a}$$

$$\tilde{T}(s) = \left\{\frac{1 - \tilde{y}(s)}{1 + \tilde{y}(s)}\right\} \tag{4.15b}$$

$$\tilde{y}(s) \equiv \text{normalized terminal admittance} = Z_c \tilde{Y}(s)$$

$$\simeq \frac{1}{4\pi}\left(\frac{Z_o}{Z_c}\right)\Big[2\,\mathrm{Ein}(2\pi s) + e^{2\pi s}\left\{\ln(2) + \mathrm{Ein}(2\pi s) - \mathrm{Ein}(4\pi s)\right\} + e^{-2\pi s}\left\{-\ln(2) + \mathrm{Ein}(-2\pi s)\right\}\Big] \tag{4.15c}$$

and Ein(z), following the notation of Ref. [8], is the exponential integral given by

$$\mathrm{Ein}(z) = \int_0^z \frac{1 - e^{-t}}{t}\,dt \tag{4.16}$$

Equation (4.14) was used as the input mermorphic function and its zeros and poles were accurately determined and they were found to be in excellent agreement (5 places or better) with earlier results [7] which employed a rather tedious 2-dimensional search method. The results for the case of $\theta = .001°$ is shown plotted in figure 4.3.

In concluding this section, we note that the two examples considered here have shown that an application of Cauchy's residue theorem is very useful in determining the pole and zero locations of a complex function of a complex variable which is meromorphic in a given region.

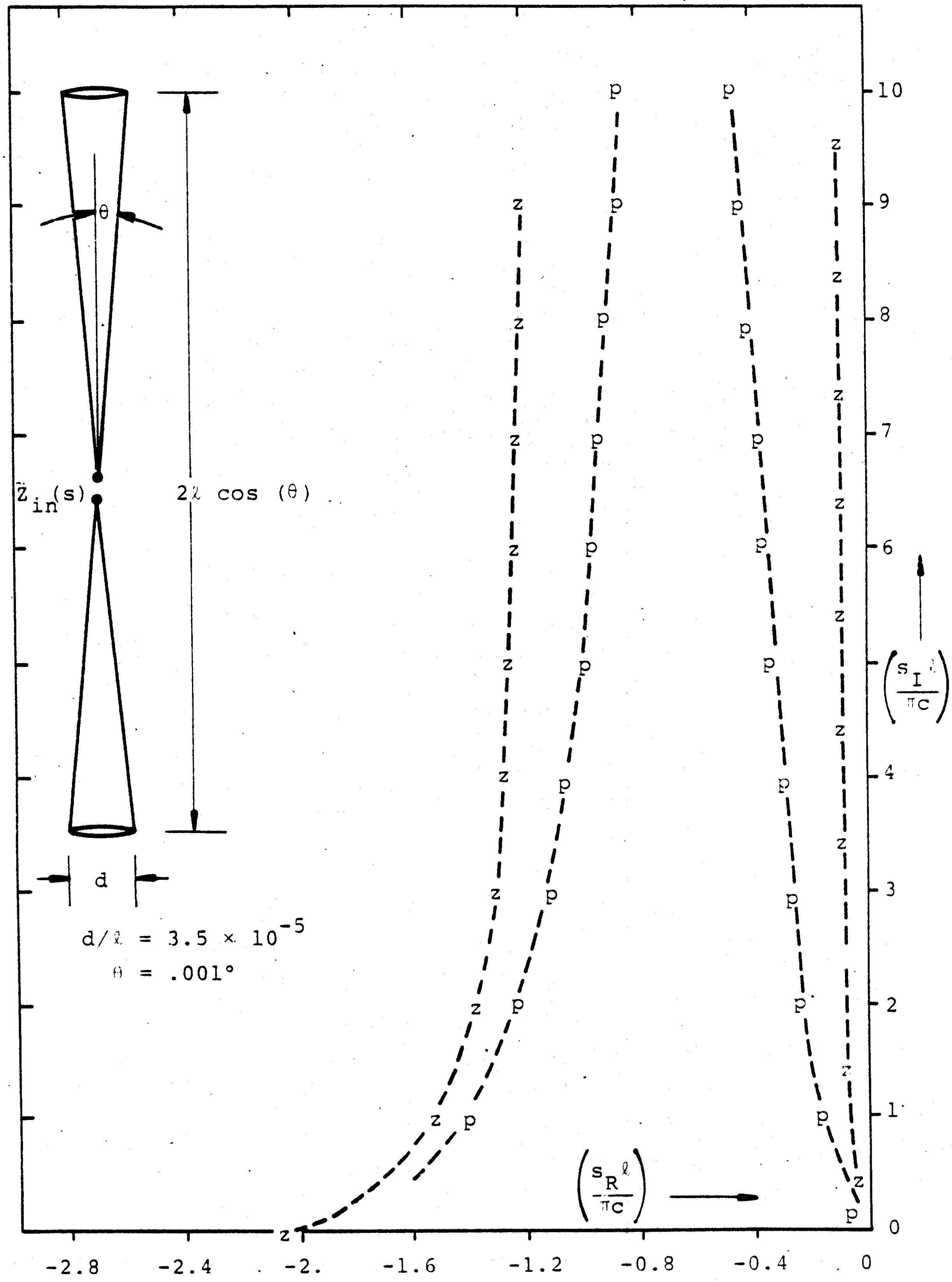

Figure 4.3 The pole-zero configuration of the input impedance $\tilde{Z}_{in}(s)$ of a biconical antenna (Example 2)

V. Summary

In this note, we have developed a pole finding procedure based on an application of Cauchy's residue theorem. The validity of the procedure in determining the poles and zeros of a meromorphic function has been demonstrated with two numerically illustrated examples of i) a ratio of polynomials, and ii) the input impedance of a biconical antenna. It is to be emphasized that a major problem in this procedure lies in unambigiously determining the number of poles N_p in a given contour. This is further discussed in Appendix B. In view of this, it is expected that this procedure may be successfully applied in physical problems where the user has some a priori knowledge of the behaviour of the meromorphic function. In other instances, modifications in terms of improving the accuracy with which the residue moments are determined, may be required to circumvent certain problems e.g., highly oscillatory functions or functions with dense population of poles and zeros.

This work is a sequel to an earlier work [2] which concerned itself with the numerical evaluation of the zeros of an analytic function.

References

1. C.E. Baum, "The Singularity Expansion Method", Chapter 3 of Transient Electromagnetic Fields, edited by L.B. Felsen, Springer-Verlag, 1975.

2. B.K. Singaraju, D.V. Giri and C.E. Baum, "Further Developments in the Application of Contour Integration to the Evaluation of the Zeros of Analytic Functions and Relevant Computer Programs", Mathematics Note 42, March 1976.

3. A.I. Markushevich, Theory of Functions of a Complex Variable, Volume I, Prentice-Hall, 1965.

4. E.C. Titchmarsh, The Theory of Functions, Chapters II and III, 2nd edition, The ELBS and Oxford University Press, 1939, reprinted in 1944,47,49,50,52,58,61 and 1962.

5. R. Mittra, "Message from the President", IEEE Newsletter: Antennas and Propagation Society, Vol. 18, No. 4, August 1976.

6. R.W.P. King, Theory of Linear Antennas, pp. 818-834, Harvard University Press, 1956.

7. D.V. Giri, C.E. Baum and C-T. Tai, "On the Use of Cauchy's Residue Theorem in Finding the Poles and Zeros of Meromorphic Functions and its Application to a Biconical Antenna", presented at the 1977 URSI Symposium on Electromagnetic Wave Theory, Stanford University, Palo Alto, CA, June 1977.

8. M. Abramowitz and I.A. Stegun (editors), Handbook of Mathematical Functions, pp. 228, Dover, 1965.

9. G.F. Carrier, M. Krook and C. E. Pearson, Functions of a Complex Variable, McGraw-Hill, 1966.

APPENDIX A

Computer Program Listings

The following TEST Program illustrates the way in which this package entitled CONTOUR may be called as a subroutine.

```
PROGRAM TEST (INPUT,OUTPUT)
IMPLICIT COMPLEX (C)
COMMON NOZ,CZ(30),CFZ(30),NOP,CP(30),CFP(30),IWARN
CSL=(0.,5.)
CSF=(5.,0.)
NR=5
NI=5
KDM=1
CALL CONTOUR(CSL,CSF,NR,NI,KDM)
END
```

```
      SUBROUTINE CONTOUR(CSL,CSF,NR,NI,KDM)
      IMPLICIT COMPLEX (C)
      COMMON NOZ,CZ(30),CFZ(30),NOP,CP(30),CFP(30),IWARN
      EXTERNAL CFCTS
      NOZ=0
      NOP=0
      IWARN=0
      RCSF=REAL(CSF)
      SCIF=AIMAG(CSF)
      RCSL=REAL(CSL)
      SCIL=AIMAG(CSL)
      RINC=(RCSL-RCSF)/FLOAT(NR)
      SCIINC=(SCIL-SCIF)/FLOAT(NI)
      ND4=(40*KDM) * 4
      PRINT 810, RCSF,SCIF,RCSL,SCIL,ND4,NR,NI
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***************,//////)
      DO 720 J=1,NR
      JJ=J-1
      SRMI=RCSF+RINC*FLOAT(J)
      SRM=RCSF+RINC*FLOAT(JJ)
      DO 710 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 RESIDUE(CFCTS,CSM,CSMI,KDM)
710   CONTINUE
720   CONTINUE
      PRINT 730,IWARN
730   FORMAT (1H1,50X,*SUMMARY OF RESULTS*,/,3X,*NO. OF WARNING MESSAGES
     2 = *,I3)
      PRINT 747
      IF (NOP.EQ.0) GO TO 745
      DO 740 M=1,NOP
      PRINT 760,M,CP(M),CFP(M)
740   CONTINUE
      IF (NOP.LE.30) GO TO 745
      PRINT 742,NOP
742   FORMAT (1H0,//,3X,*I FOUND*,2X,I3,2X,*POLES. INCREASE DIMENSION OF
     $ CP AND CFP ARRAYS ACCORDINGLY*)
745   CONTINUE
      IF (NOZ.EQ.0) GO TO 780
      PRINT 747
747   FORMAT (1H0,/,3X,**************************************************
     $****************************************************************,/)
      DO 750 M=1,NOZ
      PRINT 770,M,CZ(M),CFZ(M)
750   CONTINUE
      IF (NOZ.LE.30) GO TO 755
```

```
      PRINT 752,NOZ
752   FORMAT (1H0,//,3X,*I FOUND*,2X,I3,2X,*ZEROS. INCREASE DIMENSION OF
     $ CZ AND CFZ ARRAYS ACCORDINGLY*)
755   CONTINUE
      PRINT 747
760   FORMAT (1H0,3X,I3,2X,*POLE AT*,2E20.8,8X,*1/F(POLE)    =    *,2E17.8
     $)
770   FORMAT (1H0,3X,I3,2X,*ZERO AT*,2E20.8,8X,*F(ZERO)    =    *,2E17.8)
      RETURN
780   PRINT 790
790   FORMAT (1H0,10X,*SORRY ..RESIDUE.. COULD NOT FIND ANY ZEROS   OR
     $POLES IN THE GIVEN SCAN AREA*,//)
      RETURN
      END
```

```
      SUBROUTINE RESIDUE(CFCTS,CSM,CSMI,KDM)
      IMPLICIT COMPLEX (C)
      COMMON NOZ,CZ(30),CFZ(30),NGP,CP(30),CFP(30),IWARN
      DIMENSION CS(641),ARG(641),CF(641),CSTO(641)
      DIMENSION X(40),W(40),CD(8),CA(3),CQUAD(2),CUBE(3)
      DATA NP/40/, X/-.998237709710559,-.990726238699457,-.9772599499837
     174,-.957916819213792,-.932812808278677,-.902098806968874,-.8659595
     203212260,-.824612230833312,-.778305651426519,-.727318255189927,-.6
     371956684614180,-.612553889667980,-.549467125095128,-.483075801686l
     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  W/.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,.76110361900626
     72E-1,.747231690579683E-1,.728865823958041E-1,.706116473912868E-1,.
     867912045815233 9E-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
     $277098533 19E-2/
      IF (KDM.EQ.0) KDM=1
      CZERO=(0.,0.)
      CAUX1=CSM
      CAUX2=CSMI
      ND=KDM*NP
      ND1=ND+1
      ND2=2*ND
      ND3=3*ND
      ND4=ND*4
      NM1=ND4+1
      NMM1=ND4-1
      NDP=ND2+1
      C1=(1.,0.)
      RCSF=REAL(CSMI)
      RCSL=REAL(CSM)
      SCIF=AIMAG(CSMI)
      SCIL=AIMAG(CSM)
      PI=3.14159265
      PI3=1.5*PI
      TPI=2.*PI
      HPI=.5*PI
      NR=1
      NI=1
      ICKL=1
      ICKU=2
    5 CONTINUE
      DO 655 ICK=ICKL,ICKU
      RINT=(RCSL-RCSF)/FLOAT(NR)
```

```
      SCINT=(SCIL-SCIF)/FLOAT(NI)
      DO 650 JT=1,NR
      JJT=JT-1
      SRMI=RCSF+RINT*FLOAT(JT)
      SRM=RCSF+RINT*FLOAT(JJT)
      SAUX1=SRMI
      SAUX2=SRM
      DO 640 IT=1,NI
      IIT=IT-1
      SIMI=SCIF+SCINT*FLOAT(IIT)
      SIM=SCIF+SCINT*FLOAT(IT)
      IF (ICK.GT.1) GO TO 20
 10   CONTINUE
      PRINT 660,SRM,SIMI,SRMI,SIM
      PRINT 150
      GO TO 25
 20   PRINT 200
 25   CONTINUE
C
C     CHANGE OF VARIABLE
C
      SMR=(SRM+SRMI)/2.
      SMI=(SIM+SIMI)/2.
      CSC=CMPLX(SMR,SMI)
      CSC=CSC-(1.,0.)
      CSM=CMPLX(SRM,SIMI)-CSC
      CSMI=CMPLX(SRMI,SIM)-CSC
      SRM=REAL(CSMI)
      SRMI=REAL(CSM)
      SIMI=AIMAG(CSMI)
      SIM=AIMAG(CSM)
      DELX=(SRM-SRMI)/(2.*FLOAT(KDM))
      DELY=(SIM-SIMI)/(2.*FLOAT(KDM))
      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
  150 FORMAT(1H0,2X,*WORKING WITH THE FUNCTION F(S) TO EXTRACT POLES....
     2....*,/)
  200 FORMAT (1H0,2X,*WORKING WITH THE RECIPROCAL OF F(S) TO EXTRACT ZER
     2OS.........*,/)
  220 CONTINUE
      DO 250 K=1,ND4
      IF (ICK.EQ.1) CF(K)=CFCTS(CS(K),CSC)
      IF(ICK.EQ.2) GO TO 225
      CSTO(K)=CF(K)
      GO TO 250
  225 CF(K)=1./CSTO(K)
  250 CONTINUE
      CS(NM1)=CS(1)
      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)
  325 CONTINUE
      DO 330 L=1,ND4
      RF=REAL(CF(L))
```

```
      Z=AIMAG(CF(L))
      ARG(L)=ANGLER(RF,Z)
  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 (ICK.EQ.1) IAN1=IO
      IF (ICK.LT.2) GO TO 444
      IAN2=IO
      ICHECK=IAN1+IAN2
      IF (ICHECK.EQ.0) GO TO 444
      IWARN=IWARN+1
      PRINT 442, IWARN,IAN1,IAN2
  442 FORMAT (1HO,//,3X,*WARNING NUMBER = *,I3,//,3X,
     2*ARGUMENT NUMBER OF F(S)     = *,I3,/,3X,
     3*ARGUMENT NUMBER OF 1/F(S)   = *,I3,/)
  444 CONTINUE
C     FINDING THE RESIDUE MOMENTS DO THRU D8.
      DO 520 L=1,9
      CON1=(0.,0.)
      CON2=(0.,0.)
      CON3=(0.,0.)
      CON4=(0.,0.)
      LM1=L-1
      DO 500 K=1,4
      KK=(K-1)*KDM*NP
      IF (K-2) 450,465,445
  445 IF (K-3) 465,480,495
  450 DO 460 M=1,KDM
      MM=(M-1)*NP
      DO 455 N=1,NP
```

```
      NN=KK+MM+N
      CON1=CON1+((CS(NN)**LM1))*W(N)*CF(NN)
455   CONTINUE
460   CONTINUE
      GO TO 498
465   DO 475 M=1,KDM
      MM=(M-1)*NP
      DO 470 N=1,NP
      NN=KK+MM+N
      CON2=CON2+(CS(NN)**LM1) *W(N)*CF(NN)
470   CONTINUE
475   CONTINUE
      GO TO 498
480   DO 490 M=1,KDM
      MM=(M-1)*NP
      DO 485 N=1,NP
      NN=KK+MM+N
      CON3=CON3+((CS(NN)**LM1))*W(N)*CF(NN)
485   CONTINUE
490   CONTINUE
      GO TO 498
495   DO 497 M=1,KDM
      MM=(M-1)*NP
      DO 496 N=1,NP
      NN=KK+MM+N
      CON4=CON4+((CS(NN)**LM1))*W(N)*CF(NN)
496   CONTINUE
497   CONTINUE
498   CONTINUE
500   CONTINUE
      CON1=CON1*DELY*(0.,1.)
      CON2=-CON2*DELX
      CON3=CON3*DELY*(0.,-1.)
      CON4=CON4*DELX
      IF (L.GT.1) GO TO 510
      CD0=(CON1+CON2+CON3+CON4)/(TPI*(0.,1.))
      GO TO 520
510   CD(LM1)=(CON1+CON2+CON3+CON4)/(TPI*(0.,1.))
520   CONTINUE
C     ALL THE 9 RESIDUE MOMENTS ARE NOW COMPUTED.
      CALL POLECHK(CD0,CD,CA0,CA,NQ)
      NP1=NQ+1
      IF (ICK.EQ.1) NQ1=NQ
      IF (ICK.EQ.2) NQ2=NQ
      IF (NQ-4) 521,570,570
521   IF (ICK.LT.2) GO TO 525
      NCK=NQ2-NQ1-IAN1
      IF (NCK.EQ.0) GO TO 525
      IWARN=IWARN+1
      PRINT 522,IWARN,NQ2,NQ1,IAN1
522   FORMAT (1H0,//,3X,*WARNING NUMBER = *,I3,//,3X,*NUMBER OF ZEROS NQ
     22 = *,I3,5X,*NUMBER OF POLES NQ1 = *,I3,/,3X,
     3*(NQ2-NQ1) DOES NOT EQUAL THE ARGUMENT NUMBER IAN1 = *,I3,/)
525   CONTINUE
      GO TO 528
526   IF (NQ-4) 528,570,570
528   CONTINUE
      GO TO (530,540,550,560),NP1
```

```
530   PRINT 590,(ICK,NQ,CAO,(CA(K),K=1,3))
      IF (ICK.EQ.2) PRINT 620
      GO TO 580
540   PRINT 590,(ICK,NQ,CAO,(CA(K),K=1,3))
      CALL POLY 1(CAO,C1,CLIN)
      IF (ICK.EQ.2) CLIN=CLIN+CSC
      IF (ICK.EQ.2) GO TO 545
      PRINT 600,CLIN
      NOP=NOP+1
      CP(NOP)=CLIN
      CFP(NOP)=1./CFCTS(CLIN,CZERO)
      GO TO 580
545   PRINT 610,CLIN
      NOZ=NOZ+1
      CZ(NOZ)=CLIN
      CFZ(NOZ)=CFCTS(CLIN,CZERO)
      PRINT 620
      GO TO 580
550   PRINT 590,(ICK,NQ,CAO,(CA(K),K=1,3))
      CALL POLY 2(CAO,CA(1),C1,CQUAD)
      IF (ICK.EQ.1) GO TO 551
      CQUAD(1)=CQUAD(1)+CSC
      CQUAD(2)=CQUAD(2)+CSC
      GO TO 555
551   CONTINUE
      PRINT 600,(CQUAD(K),K=1,2)
      DO 552 L=1,2
      NOP=NOP+1
      CP(NOP)=CQUAD(L)
      CFP(NOP)=1./CFCTS(CQUAD(L),CZERO)
552   CONTINUE
      GO TO 580
555   PRINT 610,(CQUAD(K),K=1,2)
      DO 557 L=1,2
      NOZ=NOZ+1
      CZ(NOZ)=CQUAD(L)
      CFZ(NOZ)=CFCTS(CQUAD(L),CZERO)
557   CONTINUE
      PRINT 620
      GO TO 580
560   PRINT 590,(ICK,NQ,CAO,(CA(K),K=1,3))
      CALL POLY 3(CAO,CA(1),CA(2),C1,CUBE)
      IF (ICK.EQ.1) GO TO 561
      CUBE(1)=CUBE(1)+CSC
      CUBE(2)=CUBE(2)+CSC
      CUBE(3)=CUBE(3)+CSC
      GO TO 565
561   CONTINUE
      PRINT 600,(CUBE(K),K=1,3)
      DO 562 L=1,3
      NOP=NOP+1
      CP(NOP)=CUBE(L)
      CFP(NOP)=1./CFCTS(CUBE(L),CZERO)
562   CONTINUE
      GO TO 580
565   PRINT 610,(CUBE(K),K=1,3)
      DO 567 L=1,3
      NOZ=NOZ+1
```

```
      CZ(NOZ)=CUBE(L)
      CFZ(NOZ)=CFCTS(CUBE(L),CZERO)
567   CONTINUE
      PRINT 620
      GO TC 580
570   IF (ICK.EQ.2) GO TO 575
      PRINT 572
572   FORMAT (1HC,2X,*CAUTION ......*,/,3X,*THIS CONTOUR HAS MORE THAN 3
     $ POLES.*,/,3X,*THE USER IS URGEC TO REWORK THIS CONTOUR*)
      GO TO 630
575   CONTINUE
576   PRINT 577
577   FORMAT (1HC,3X,*CAUTION ......*,/,3X,*THIS CONTOUR HAS MORE THAN 3
     $ ZEROS.*,/,3X,*THE USER IS URGED TO REWORK THIS CONTOUR*)
      GO TO 630
580   CONTINUE
590   FORMAT (1H0,3X,*ICK = *,I1,3X,*NP = *,I1,3X,8E14.5,/)
600   FORMAT (1H0,3X,*POLE AT*,2X,2E20.8,/)
610   FORMAT (1H0,3X,*ZERO AT*,2X,2E2C.8,/)
620   FORMAT (1H0,1X,//////)
630   CONTINUE
      SRMI=SAUX1
      SRM=SAUX2
640   CONTINUE
650   CONTINUE
655   CONTINUE
660   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))
      CSM=CAUX1
      CSMI=CAUX2
      RETURN
      END
```

```
      SUBROUTINE POLECHK(D0,D,DA0,DA,NP)
      IMPLICIT COMPLEX (D)
      DIMENSION D(8),DA(3),AD(8),D3(3,3)
      DIMENSION D3A(3,3),DW(3),DV(3)
C
C     THIS SUBROUTINE FINDS A POTENTIAL VALUE FOR NP = NUMBER
C     OF POLES IN THE GIVEN CONTOUR C.
C     IT ALSO DETERMINES THE COEFFICIENTS DA(4) OF THE
C     DENOMINATOR POLYNOMIAL.
C
      NP=0
      DA0=(0.,0.)
      DO 10 IC=1,3
      DA(IC)=(0.,0.)
   10 CONTINUE
      EPS=1.E-02
      AD0=CABS(D0)
      DO 20 I=1,8
      AD(I)=CABS(D(I))
   20 CONTINUE
      PRINT 25,AD0,(AD(I),I=1,8)
   25 FORMAT (1H0,2X,*MAGNITUDES OF THE RESIDUE MOMENTS D0 THRU D8*,//,3
     $X,9E14.5,/)
      IF (AD0.LE.EPS) GO TO 40
C     CHECKING TO SEE IF NP = 1
      DA0=-D(1)/D0
      DO 30 I=1,7
      IP1=I+1
      DQTY=-D(IP1)/D(I)
      TR=ABS(REAL(DA0-DQTY))
      TI=ABS(AIMAG(DA0-DQTY))
      IF (TR.GT.EPS.AND.TI.GT.EPS) GO TO 40
   30 CONTINUE
      NP=1
      RETURN
   40 CONTINUE
      IF (AD0.LE.EPS.AND.AD(1).LE.EPS) GO TO 60
C     CHECKING T SEE IF NP = 2 ......
      DET=(D0*D(2))-(D(1)**2)
      DA0=((D(1)*D(3))-(D(2)**2))/DET
      DA1=((D(1)*D(2))-(D0*D(3)))/DET
      DA(1)=DA1
      DO 50 J=2,6
      JP1=J+1
      JP2=J+2
      DQTY=D(JP2)+(D(JP1)*DA1)+(D(J)*DA0)
      TR=ABS(REAL(DQTY))
      TI=ABS(AIMAG(DQTY))
      IF (TR.GT.EPS.AND.TI.GT.EPS) GO TO 60
   50 CONTINUE
      NP=2
      RETURN
   60 CONTINUE
      IF (AD0.LE.EPS.AND.AD(1).LE.EPS.AND.AD(2).LE.EPS) GO TO 180
C     CHECKING TO SEE IF NP = 3 ......
      DO 90 IROW=1,3
      DO 80 JCOL=1,3
```

```
      IND=IROW-2+JCOL
      IF (IROW.EQ.1.AND.JCOL.EQ.1) GO TO 70
      D3(IROW,JCOL)=D(IND)
      GO TO 80
70    D3(1,1)=D0
80    CONTINUE
90    CONTINUE
      CALL DETER(D3,D3A,DW,DV,3,DENOM)
      DO 100 IROW=1,3
      IND=IROW+2
      D3(IROW,1)=-D(IND)
100   CONTINUE
      CALL DETER(D3,D3A,DW,DV,3,DN0)
      DO 130 IROW=1,3
      DO 120 JCOL=1,2
      IF (JCOL.EQ.2) GO TO 110
      IND1=IROW-2+JCOL
      IF (IND1.EQ.0) GO TO 105
      D3(IROW,JCOL)=D(IND1)
      GO TO 120
105   D3(1,1)=D0
      GO TO 120
110   IND2=IROW+2
      D3(IROW,JCOL)=-D(IND2)
120   CONTINUE
130   CONTINUE
      CALL DETER(D3,D3A,DW,DV,3,DN1)
      DO 160 IROW=1,3
      DO 150 JCOL=2,3
      IF (JCOL.EQ.3) GO TO 140
      IND1=IROW-2+JCOL
      D3(IROW,JCOL)=D(IND1)
      GO TO 150
140   IND2=IROW+2
      D3(IROW,JCOL)=-D(IND2)
150   CONTINUE
160   CONTINUE
      CALL DETER(D3,D3A,DW,DV,3,DN2)
      DA0=DN0/DENOM
      DA(1)=DN1/DENOM
      DA(2)=DN2/DENOM
      DO 170 J=3,5
      JP1=J+1
      JP2=J+2
      JP3=J+3
      DQTY=D(JP3)+(DA(2)*D(JP2))+(DA(1)*D(JP1))+(DA0*D(J))
      TR=ABS(REAL(DQTY))
      TI=ABS(AIMAG(DQTY))
      IF (TR.GT.EPS.AND.TI.GT.EPS) GO TO 180
170   CONTINUE
      NP=3
      RETURN
180   CONTINUE
      IF (AD0.GT.EPS) GO TO 200
      DO 190 ID=1,3
      IF (AD(ID).GT.EPS) GO TO 200
190   CONTINUE
      GO TO 340
```

```
  200   CONTINUE
  340   CONTINUE
        DO 350 ID=4,8
        IF (AD(ID).GT.EPS) GO TO 370
  350   CONTINUE
C       ALL MOMENTS ARE LESS THAN EPSILON.
C       SO NP=0 FOR THIS CONTOUR.
        NP=0
        DA0=(0.,0.)
        DO 360 I=1,3
        DA(I)=(0.,0.)
  360   CONTINUE
        RETURN
  370   CONTINUE
        NP=4
        RETURN
        END
```

```
      SUBROUTINE DETER(CM,CB,CW,CV,MS,CD)
C     THIS SUBROUTINE COMPUTES THE VALUE OF THE DETERMINANT
C     CD OF A SQUARE MATRIX CM OF SIZE MS
C
C     INPUTS. 1)COMPLEX MATRIX ELEMENTS CM
C             2)SIZE OF THE SQUARE MATRIX MS
C     OUTPUT. 1)COMPLEX VALUE OF DETERMINANT CD
C
      IMPLICIT COMPLEX (C)
      DIMENSION CM(MS,MS),CB(MS,MS),CW(MS),CV(MS)
C
C     BEGIN TRIANGULARIZATION
C     MOVE MATRIX CM TO CB SO THAT INPUT MATRIX WILL NOT
C     BE DESTROYED
 1    DO 15 I=1,MS
      DO 10 J=1,MS
 10   CB(I,J) =CM(I,J)
 15   CONTINUE
      DO 60 J=1,MS
      Q=0.
C     CALCULATE NORM SQUARED OF COLUMN J
      DO 20 K=J,MS
 20   Q=Q+CABS(CB(K,J))**2
      IF (Q.GT.0.) GO TO 21
C     IF THE NORM IS ZERO, THE MATRIX IS SINGULAR
      ISW=3
      PRINT 9
 9    FORMAT (1H0,*THE MATRIX IS SINGULAR*,//)
      RETURN
C     CALCULATE THE DIAGONAL ELEMENT OF THE MATRIX T. (CW(J))
C     CALCULATE THE DIAGONAL ELEMENT OF MATRIX U. (CB(J,J))
C     CALCULATE THE ELEMENT OF VECTOR V
C     BEGIN ITERATION
 21   BSQ=CABS(CB(J,J))**2
      IF (BSQ.EQ.0.) CW(J)=SQRT(Q)
      IF (BSQ.GT.0.) CW(J)=SQRT(Q/BSQ)*CB(J,J)
      CB(J,J)=CB(J,J) + CW(J)
      CV(J)=-CB(J,J)*CONJG(CW(J))
      IF (J.EQ.MS) GO TO 60
      IB=J+1
      DO 50 I=IB,MS
      CS=(0.,0.)
      DO 30 K=J,MS
 30   CS=CS+CB(K,I)*CONJG(CB(K,J))
      CS=CS/CV(J)
      DO 40 K=J,MS
 40   CB(K,I)=CB(K,I)+CS*CB(K,J)
 50   CONTINUE
 60   CONTINUE
      CD=(1.,0.)
      DO 61 I=1,MS
 61   CD=CD*CW(I)
      RETURN
      END
```

```
      FUNCTION ANGLER (X,Y)
      PI=3.14159265
      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
```

```
      SUBROUTINE POLY 1(C0,C1,CLIN)
      IMPLICIT COMPLEX (C)
C
C     THIS SUBROUTINE SOLVES A LINEAR EQUATION OF THE
C     FORM C1*S+C0 = 0
C
      CLIN=-C0/C1
      RETURN
      END
```

```
      SUBRCUTINE POLY 2(C0,C1,C2,CQUAD)
      IMPLICIT COMPLEX (C)
      DIMENSION CQUAD(2)
C     THIS SUBRCUTINE SOLVES A QUACRATIC EQUATION OF THE
C     FCRM C2*(S**2)+C1*S+C0 = 0
C
      CQ=(C1**2)-(4.*C2*C0)
      CQ=CSQRT(CQ)
      CDR=(2.*C2)
      CQUAD(1)=(-C1+CQ)/CDR
      CQUAD(2)=(-C1-CQ)/CDR
      RETURN
      END

      SUBRCUTINE POLY 3(C0,C1,C2,C3,CUBE)
      IMPLICIT COMPLEX (C)
      DIMENSION CUBE(3),CAUX1(3),CAUX2(3)
C
C     THIS SUBRCUTINE SOLVES A CUBIC EQUATION OF THE
C     FORM C3*(S**3)+C2*(S**2)+C1*S+C0 = 0
C     REFERENCE........EQN. 3.8.2. OF AMS 55. PAGE 17.
C
      CA2=C2/C3
      CA1=C1/C3
      CA0=C0/C3
      CQ=(CA1/3.)-((CA2**2)/9.)
      CR=((CA1*CA2-3.*CA0)/6.)-((CA2**3)/27.)
      CRQ=CSQRT((CR**2) + (CQ**3))
      TPI=2.*3.14159265
      CRP=CR+CRQ
      CRM=CR-CRQ
      CS1=CEXP(CLOG(CRP)/3.)
      CS2=CEXP(CLOG(CRM)/3.)
      CQTY=-CQ
      CI=(0.,1.)
      EPS=1.E-5
      DO 20 IC=1,3
      TIC=FLOAT(IC)
      CAUX1(IC)=CEXP(CI*TPI*TIC/3.)*CS1
      DO 10 JC=1,3
      TJC=FLOAT(JC)
      CAUX2(JC)=CEXP(CI*TPI*TJC/3.)*CS2
      CP=CAUX1(IC)*CAUX2(JC)-CQTY
      DELR=ABS(REAL(CP))
      DELI=ABS(AIMAG(CP))
      IF (DELR.LE.EPS.AND.DELI.LE.EPS) GO TO 30
   10 CONTINUE
   20 CONTINUE
   30 CS1=CAUX1(IC)
      CS2=CAUX2(JC)
      CSP=(CS1+CS2)/2.
      CSM=(CS1-CS2)/2.
      CR3=CI*SQRT(3.)
      CA23=CA2/3.
      CUBE(1)=CS1+CS2-CA23
      CUBE(2)=-CSP-CA23+(CR3*CSM)
      CUBE(3)=-CSP-CA23-(CR3*CSM)
      RETURN
      END
```

```
      COMPLEX FUNCTION CFCTS(CS,CSHIFT)
      IMPLICIT COMPLEX (C)
      DIMENSION CDR(10),CFD(10),CNR(16),CFN(16)
      CS=CS+CSHIFT
      NP=10
      CDR(1)=(3.5,1.5)
      CDR(2)=(1.75,2.25)
      CDR(3)=(1.5,2.75)
      CDR(4)=(2.25,3.25)
      CDR(5)=(2.75,3.5)
      CDR(6)=(2.25,3.75)
      CDR(7)=(1.25,0.75)
      CDR(8)=(2.75,1.75)
      CDR(9)=(1.5,0.5)
      CDR(10)=(2.25,1.25)
      DO 10 I=1,NP
      CFD(I)=CS-CDR(I)
   10 CONTINUE
      CDENOM=(1.,0.)
      DO 20 I=1,NP
      CDENOM=CDENOM*CFD(I)
   20 CONTINUE
      NO=16
      CNR(1)=(4.5,.5)
      CNR(2)=(4.25,4.75)
      CNR(3)=(3.75,1.25)
      CNR(4)=(3.75,1.75)
      CNR(5)=(4.5,2.5)
      CNR(6)=(4.25,2.75)
      CNR(7)=(2.75,3.75)
      CNR(8)=(1.25,2.25)
      CNR(9)=(1.75,2.75)
      CNR(10)=(4.5,4.5)
      CNR(11)=(.75,1.5)
      CNR(12)=(.5,1.25)
      CNR(13)=(.25,1.75)
      CNR(14)=(.5,4.25)
      CNR(15)=(.75,4.5)
      CNR(16)=(.25,4.75)
      DO 30 I=1,NO
      CFN(I)=CS-CNR(I)
   30 CONTINUE
      CNUM=(1.,0.)
      DO 40 I=1,NO
      CNUM=CNUM*CFN(I)
   40 CONTINUE
      CFCTS=CNUM/CDENOM
      A=0.
      CFCTS=CFCTS*CEXP(A*CS)
      RETURN
      END
```

APPENDIX B

A List of Relevant Definitions and Some Interesting but not Very Useful Results

Analytic function:

If $A(s)$ has a derivative at a point s_0 and also at each point in some neighborhood of s_0, then $A(s)$ is said to be analytic at s_0. The terms *holomorphic*, *monogenic* and *regular* are also sometimes used [9].

Entire function:

An entire function is one which is analytic everywhere in the plane and may have its singularities only at infinity [9].

Meromorphic function:

A meromorphic function is one whose only singularities, except at infinity, are poles [4].

Argument number:

The argument number N_a is the number of excess zeros over poles $(N_o - N_p)$ of a meromorphic function, inside a simple closed contour. Note that N_a is the order of the pole at infinity when $N_a > 0$ and it is the order of the zero at infinity when $N_a < 0$.

Exceptional point:

For a given function, certain values may be exceptional, in the sense that the function can not take these values [4]. For instance, the points $(0 \pm j1)$ are exceptional points of the meromorphic function $\tan(s)$.

A major problem in the procedure developed in this note, of finding the poles and zeros of a given complex meromorphic function lies in obtaining the number of poles N_p, deterministically in a given contour. If N_p is known unambigiously, there is no difficulty in accurately determining the pole locations. Since the excess number of zeros over poles, i.e., $(N_o - N_p)$ can be easily determined from the principle of the argument, attempt was made to obtain another expression involving N_o and N_p so that one can solve a set of simultaneous equations for N_o and N_p. In this unsuccessful attempt, the following results were obtained. The following results are not useful in determining N_o and N_p, when used along with the known argument number. When $M(s)$ has all its poles simple,

$$\frac{1}{2\pi j}\oint_C s\left\{\frac{M'(s)}{M(s)}\right\}^2 ds = (N_o - N_p)^2 \qquad (B.1)$$

$$\frac{1}{2\pi j}\oint_C s\left\{\frac{M''(s)}{M(s)}\right\} ds = (N_o - N_p)^2 - (N_o - N_p) \qquad (B.2)$$

$$\frac{1}{2\pi j}\oint_C \frac{M''(s)}{M'(s)}\, ds = (N_o' - N_p') = (N_o' - 2N_p) \qquad (B.3)$$

with N_o' = number of zeros of $M'(s)$ in C,

N_p' = number of poles of $M'(s)$ in C

= twice the number of poles of $M(s)$ in $C = 2N_p$.

$$\frac{1}{2\pi j}\oint_C \frac{M'''(s)}{M''(s)}\, ds = (N_1' - 3N_p) \qquad (B.4)$$

with N_1' = number of zeros of $(M'(s) - 1)$ in C.

In this context, it may be noted that N_p can be obtained deterministically, if

i) we develop a method of determining the number of essential singularities of $M(s)$ in C.

or ii) we know an exceptional value ε of $M(s)$ in C as illustrated below.

i) Essential singularities

Observe that the function $\exp(M(s))$ has no zeros in C, but all the poles of $M(s)$ become essential singularities of $\exp(M(s))$ within C. So, if one can determine the number of essential singularities of a given function in a given contour, this concept is useful in determining the number of poles of a function inside the contour.

ii) Exceptional value

Let ε be a known exceptional value of $M(s)$ inside the contour C. The existence of ε is improbable, if not impossible, since the affinity of a function for every value is the same. If ε does exist, one may define a new function $a(s)$

$$a(s) = \frac{1}{M(s) - \varepsilon} \tag{B.5}$$

then the argument number of $a(s)$ is given by

$$\frac{1}{2\pi j} \oint_C \frac{a'(s)}{a(s)} ds = N_p \tag{b.6}$$

and thus N_p, the number of poles of $M(s)$ in C is easily determined. Note that $a(s)$ is an analytic function in C since $M(s) \neq \varepsilon$ and the poles of $M(s)$ in C become the zeros of $a(s)$.