



**Calhoun: The NPS Institutional Archive**  
**DSpace Repository**

---

NPS Scholarship

Theses

---

1992-03

Revising M-Layer: complex exponent representation.

Yin, Yuan Han

Monterey, California. Naval Postgraduate School

---

<https://hdl.handle.net/10945/28506>

---

*Downloaded from NPS Archive: Calhoun*



Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

**Dudley Knox Library / Naval Postgraduate School**  
**411 Dyer Road / 1 University Circle**  
**Monterey, California USA 93943**

<http://www.nps.edu/library>



DUDLEY KNOX LIBRARY  
NAVAL POSTGRADUATE SCHOOL  
MONTEREY CA 93943-5101





# NAVAL POSTGRADUATE SCHOOL Monterey, California



## THESIS

REVISING M-LAYER:  
COMPLEX EXPONENT REPRESENTATION

by

Yin Yuan Han

March 1992

Thesis Advisor:

Hung-Mou Lee

Approved for public release; distribution is unlimited.



## REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED		1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE			
4. PERFORMING ORGANIZATION REPORT NUMBER(S)		5. MONITORING ORGANIZATION REPORT NUMBER(S)	
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b. OFFICE SYMBOL (If applicable) EC	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School	
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000		7b. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	
8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c. ADDRESS (City, State, and ZIP Code)		10. SOURCE OF FUNDING NUMBERS	
		Program Element No	Project No
		Task No	Work Unit Accession Number
11. TITLE (Include Security Classification) REVISING M-LAYER: COMPLEX EXPONENT REPRESENTATION			
12. PERSONAL AUTHOR(S) YIN YUAN HAN			
13a. TYPE OF REPORT Master's Thesis	13b. TIME COVERED From To	14. DATE OF REPORT (year, month, day) 1992 March	15. PAGE COUNT 76
16. SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
17. COSATI CODES		18. SUBJECT TERMS (continue on reverse if necessary and identify by block number)	
FIELD	GROUP	Tropospheric Propagation	
	SUBGROUP		
19. ABSTRACT (continue on reverse if necessary and identify by block number)			
<p>The waveguide mode tropospheric propagation effect prediction program, M-Layer, originally written by Naval Command Control and Ocean Surveillance Center, Research, Development, Test and Engineering Division (NRaD), is revised for greater accuracy, speed and stability. The accuracy improvement is achieved first by converting the extended complex number representation into the representation by the complex exponent then by re-writing the group of Airy function computation subroutines. This accuracy improvement makes it possible to implement a self-checking procedure for determining the proper method to evaluate the height gaing function. Finally, a new mode locating algorithm is introduced which improves the efficiency of mode search and eliminates the looping problem observed. The revision has been documented and the new program source code has been delivered to NRaD. It is also recommended that the mode search protocol, not just the mode locating algorithm introduced in in this revision, be completely revised for better performance.</p>			
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS REPORT <input type="checkbox"/> DTIC USERS		21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Hung-Mou Lee		22b. TELEPHONE (Include Area code) (408)646-2846	22c. OFFICE SYMBOL EC/Lh

Approved for public release; distribution is unlimited.

REVISING M-LAYER:  
COMPLEX EXPONENT REPRESENTATION

by

Yin Yuan Han  
Lieutenant Commander, Taiwan Navy  
B.S., Chinese Naval Academy, 1982

Submitted in partial fulfillment  
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL

March 1992

---

## ABSTRACT

The waveguide mode tropospheric propagation effect prediction program, M-Layer, originally written by Naval Command Control and Ocean Surveillance Center, Research, Development, Test and Engineering Division (NRaD), is revised for greater accuracy, speed and stability. The accuracy improvement is achieved first by converting the extended complex number representation into the representation by the complex exponent then by re-writing the group of Airy function computation subroutines. This accuracy improvement makes it possible to implement a self-checking procedure for determining the proper method to evaluate the height gain function. Finally, a new mode locating algorithm is introduced which improves the efficiency of mode search and eliminates the looping problem observed. The revision has been documented and the new program source code has been delivered to NRaD. It is also recommended that the mode search protocol, not just the mode locating algorithm introduced in this revision, be completely revised for better performance.

1/20/15  
1/4/95  
C/

TABLE OF CONTENTS

I. INTRODUCTION ..... 1

    A. M-LAYER ..... 2

    B. EXTENDED COMPLEX NUMBER REPRESENTATION ..... 5

    C. OTHER REVISIONS ..... 7

II. PROGRAM REVISIONS ..... 9

    A. ADDITION SUBROUTINE ..... 13

    B. AIRY FUNCTION EVALUATION ..... 14

        1. XCDAIT ..... 14

        2. XCDAIG ..... 14

    C. MODE LOCATING ..... 15

        1. FNDMOD ..... 17

        2. FZEROX ..... 17

        3. FINDFX ..... 18

        4. ROOTS ..... 19

    D. EVALUATING  $A_i$  AND  $B_i$  ..... 20

III. CONCLUSIONS AND RECOMMENDATIONS ..... 32

A. Performance .....	32
B. Recommendations .....	38
APPENDIX A: SUBROUTINE XCADD .....	39
APPENDIX B: SUBROUTINE FZEROX .....	42
APPENDIX C: SUBROUTINE ROOTS .....	54
APPENDIX D: SUBROUTINE ABCOEF .....	58
LIST OF REFERENCES .....	68
INITIAL DISTRIBUTION LIST .....	69

## **ACKNOWLEDGEMENTS**

The author wishes to express his sincere gratitude to Professor Hung-Mou Lee for his professional guidance, invaluable advice and moral support throughout the thesis preparation period.

Special thanks are due to Professor Lawrence J. Ziomek for his careful review and constructive suggestions to the thesis work.

## I. INTRODUCTION

M-Layer is a FORTRAN program for computing the propagation factor of an electromagnetic (EM) wave in a stratified atmosphere. It is desirable to extend the capability of this program to include a layer of random medium representing the air-ocean interface. To achieve this goal, there are many basic theoretical problems which have to be answered. First of all, the effect of the earth curvature in this program is taken care of through the classical earth-flattening approximation [Ref. 1], but the result [Ref. 2] does not agree with the more recent diffraction theory of Fock [Ref. 3] near the surface of the earth. Then there is the question about the better method to model the atmospheric refractive index profile, either piecewise linear or quadratic, to be resolved by a new earth-flattening approximation under development at NPS. The new approximation will also determine the functions to be used for the representation of the EM fields in each layer through uniform asymptotic theories. Within some proper region, these new functions are expected to reduce to the Airy functions utilized by M-Layer. The evolutionary nature of this effort prompted this review to improve the inner workings of the M-Layer program. In particular, the subroutines to search for the modes and those for evaluating the Airy functions will remain as an important part of a program investigating questions about EM wave propagation by solving the related boundary value problem.

It can never be overemphasized that a boundary value problem which includes a layer of random medium or some range dependent inhomogeneity, set up according to the Maxwell equations, will include backscattering in its solution. This is in sharp contrast to those numerical procedures based on the parabolic approximation to the wave equation for which the backscattering is completely ignored.

In what follows, the M-Layer program and the reasons for replacing the extended complex numbers with their complex exponent representations are discussed, together with some other problems encountered and resolved during this investigation.

#### **A. M-LAYER**

In M-Layer, the index of refraction of the atmosphere is assumed to be height dependent and is approximated with a continuous piecewise linear profile. The classical earth-flattening approximation is utilized to allow the use of the cylindrical coordinate system while retaining the effect of the curvature of the earth. This is done simply by substituting the index of refraction with the modified index of refraction, which also has a piecewise linear profile [Ref. 1].

The source of the EM radiation is assumed to be either a vertical electric dipole or a vertical "magnetic dipole", with the latter providing an approximation to the radiation of a horizontal electric dipole. The dipole is located along the positive z-axis of the cylindrical coordinate system while the origin is sitting on the ground. The x-y plane is the 'flattened' earth surface. After carrying out the Hankel

transform along the radial direction, the resulting spectrum of the Hertzian dipole field within each layer of a linear segment of the modified refractive index profile is reduced to a linear combination of the Airy functions. Specifically, the layers are numbered to increase with height, with the first layer being the one above the ground. The spectrum of the Hertzian dipole field is proportional to the product of the values, at the transmitter height and at the receiver height respectively, of the height-gain function. At a height within the  $i$ -th layer, the height-gain function is given by [Ref. 4]:

$$f_i(\rho, z) = B_i(\rho) [A_i(\rho) k_1(q_i) + k_2(q_i)] , \quad (1)$$

where  $\rho$  is the radial component of the propagation vector and is also the spectral variable of the Hankel transform; hence it is the same throughout all layers. It is a complex variable whose imaginary part represents the radial attenuation rate of the spectral component of the Hertzian dipole field. Under the classical earth-flattening approximation, the spectrum of the Hertzian dipole field contains a discrete portion and a branch cut. The discrete spectrum gives rise to the creeping wave modes diffracted by the earth surface and the dielectric waveguide modes supported by the layered atmosphere. The contribution from the branch cut is usually negligible, especially for the field in the shadow of the earth. The M-Layer program locates the discrete spectrum for modes having a radial attenuation rate below a predetermined value. Contributions from these modes determine the propagation factor of the wave.

The variable  $q_i$  in the  $i$ -th layer is a dimensionless linear function of height  $z$  with the free space wavenumber  $k$ , the modified index of refraction  $m_i$  at the lower boundary  $z = z_i$ , the slope of the modified index of refraction  $\alpha_i/2$  and  $\rho$  as parameters:

$$q_i = \sqrt[3]{\left(\frac{k}{\alpha_i}\right)^2 \left(m_i^2 + \alpha_i(z - z_i) - \frac{\rho^2}{k^2}\right)}. \quad (2)$$

The height dependence of the field is given in terms of the functions  $k_1(q_i)$  and  $k_2(q_i)$ , which are proportional to the Airy functions  $Ai(-q e^{j2\pi/3})$  and  $Ai(-q_i)$  respectively. Of these two functions, at a height so large that  $q_i$  is large and positive,  $k_1(q_i)$  represents a downward going wave and  $e^{j4\pi/3}k_1(q_i) + k_2(q_i)$  represents an upward going wave. The coefficients  $A_i$  and  $B_i$  are determined by the conditions on the continuity of the Hertzian dipole field and its derivative across layer boundaries and by the normalization condition that the integral of the square of the height-gain function over all height equals unity.

To fulfill the radiation condition, the highest layer is given the same refractive index as the free space above it and only the outgoing wave is allowed within this layer. Below the 'flattened' earth surface, the field is assumed to be a plane wave propagating downward. Hence, only the normalization factors are required in the highest layer and in the ground. By assigning  $B_i$  to unity in the highest layer, all the coefficients  $A_i$  and  $B_i$  can be determined, according to the boundary conditions, to within a multiplicative factor for  $B_i$ . This multiplicative factor is then deduced from

the normalization condition. This procedure can also be carried out from the ground level up. That these coefficients can be computed either from the highest level down or from the lowest level up is a result of the fact that  $\rho$  belongs to the discrete spectrum of the Hertzian dipole field. Consequently, agreement between these two ways of evaluating the  $A_i$  and  $B_i$  coefficients confirms that a mode has been located accurately.

## B. EXTENDED COMPLEX NUMBER REPRESENTATION

The discrete spectrum of the Hertzian dipole field corresponds to the zeroes of the modal function which is a determinant whose elements consist of  $k_1(q_i)$  and  $k_2(q_i)$  at the layer boundaries. Numerically, the magnitude of this modal function causes overflow and underflow problems as  $k_1(q_i)$  or  $k_2(q_i)$  becomes exponentially large or small for complex  $q_i$  values. In the M-Layer program, to overcome this problem, a complex number is written as a scaled number, which is complex, multiplied by a scaling factor which is an integer power of  $e$ , the base of natural logarithm. This integer is chosen so that the greater of the absolute values of the real part and the imaginary part of the scaled number lies within  $e^{\pm 1}$ . A complex number written in this form is called an extended complex number. Multiplication of two extended complex numbers requires summing the two integer exponents in addition to carrying out the regular complex multiplication of the scaled numbers. Addition of two such numbers is achieved through the use of an addition subroutine: the larger scaling factor is factored out of both addends before they are combined.

The scaling factor is adjusted after each addition and after a sequence of multiplications to make sure that the resulting scaled number is still within the desired range. Addition is troublesome when the two numbers to be added nearly cancel each other. Under this circumstance, the scaling factors of the two numbers are identical and both the real parts and the imaginary parts of the scaled numbers are almost equal with opposite signs. It is clear that the real part and the imaginary part of the sum lose their accuracies to different degrees; hence the phase angle may incur substantial error. To remedy this situation, interpolation procedures have to be devised.

As two complex numbers come close to cancel each other, they must be out of phase by almost 180 degrees. By factoring out the square root of their product instead of the scale factor, the resulting addends become reciprocal to each other, both lying within an identical small angle to, and on the same side of, the imaginary axis. They are close to the unit circle, but one is on the inside and the other is on the outside. Taking out further a phase factor of  $\pi/2$  after writing the addends in their exponential forms, the exponents become small numbers for which a Taylor series expansion of the exponential function converges rapidly and can be used for interpolating the sum to achieve higher accuracy. Note that after the extra phase factor of  $\pi/2$  is removed from the addends, it is actually the difference of the resulting two reciprocals which is computed. This procedure effectively picks the direction on the complex plane along which the addends are almost opposing each

other to carry out their cancellation. The resulting sum has a phase angle nearly perpendicular to this chosen direction.

It is evident that the representation of a complex number by its complex exponent of base  $e$  provides better phase accuracy for addition. A one-to-one correspondence can be achieved by restricting the imaginary part of this exponent to within  $-\pi$  and  $\pi$ . This will be called the exponential representation or the complex exponent representation henceforth. It is convenient for multiplication: adding the complex exponents of the two factors will suffice. Conversion of the M-Layer program from the extended complex number to the complex exponent representation has been carried out.

### C. OTHER REVISIONS

As better precision is achieved, problems with the mode search procedure and the evaluation of the  $A_i$  and  $B_i$  coefficients become severe. They are thoroughly investigated and resolved. For mode search, although the division of the region of interest into "contour rectangles" and further into square "meshes", and the search pattern to run along the sides of a "contour rectangle" to find and follow "phase lines" into the interior, the basic assumption of Shellman and Morfitt [Ref. 5] that both the real and imaginary parts of the modal function are linear along every edge of a mesh is completely abandoned. For the evaluation of the  $A_i$  and  $B_i$  coefficients, the "test for evanescence" conditions have been removed. A condition to determine whether to evaluate the coefficients from the ground level up or from

the top level down has been fomulated and incorporated into the program. This accomplishment leads to the relaxation of mode locating accuracy requirements which, combined with the improved precision of the revised program, makes the first order Newton-Raphson iteration unnecessary. The specific changes in the program and the resulting gains in speed, accuracy and execution stability are discussed in the following chapters. Suggestions to completely revise the mode search protocol to do without the "contour rectangles" and to look for the modes according to their range attenuation rates are also provided.

## II. PROGRAM REVISIONS

M-Layer is structured into three parts: setup, mode search and propagation factor evaluation. The main input is the modified refractive index values at specified heights so that a piecewise linear profile can be constructed. If the mode locations for the particular profile are available from a previous run of the program, they can also be included in the input and the mode search procedures will be bypassed. The various ranges and transmitter and receiver heights for which propagation factors are desired are also specified. The subroutine WVGSTDIN is called to input the information from an ASCII data file. The program then computes the constants to be used for mode search and propagation factor evaluation. The mode search is performed with the subroutine FNDMOD. The MODSUM subroutine is then invoked to first compute the  $A_i$  and  $B_i$  coefficients as explained in the Introduction, then compute the propagation factor and the propagation loss. The complete program structure is given in Figures 1 and 2. There are several other subroutines which are not included in these and other figures, such as DHORIZ for computing the horizon distance between a transmitter and a receiver for reference purpose; CHKMOD, a maintenance routine for removing zeroes from reported mode locations by older versions of the program; or AO2H2O, a routine to compute the atmospheric absorption coefficient due to oxygen and water vapor. They will not be

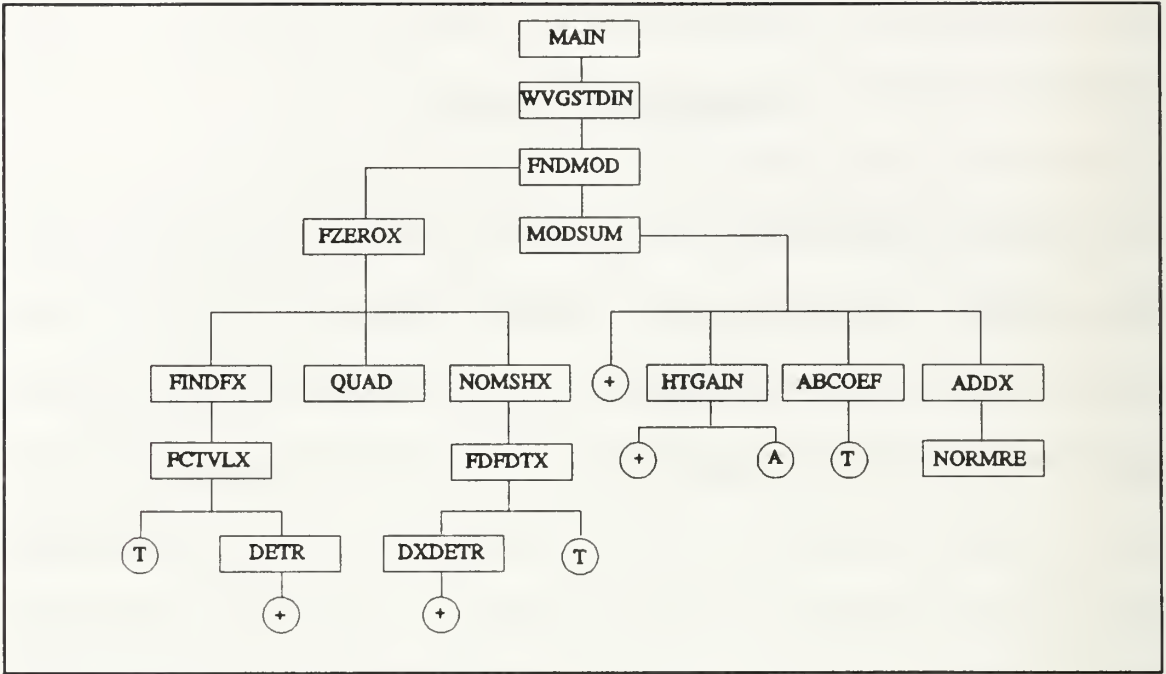


Figure 1 Original M-layer subroutines structure.

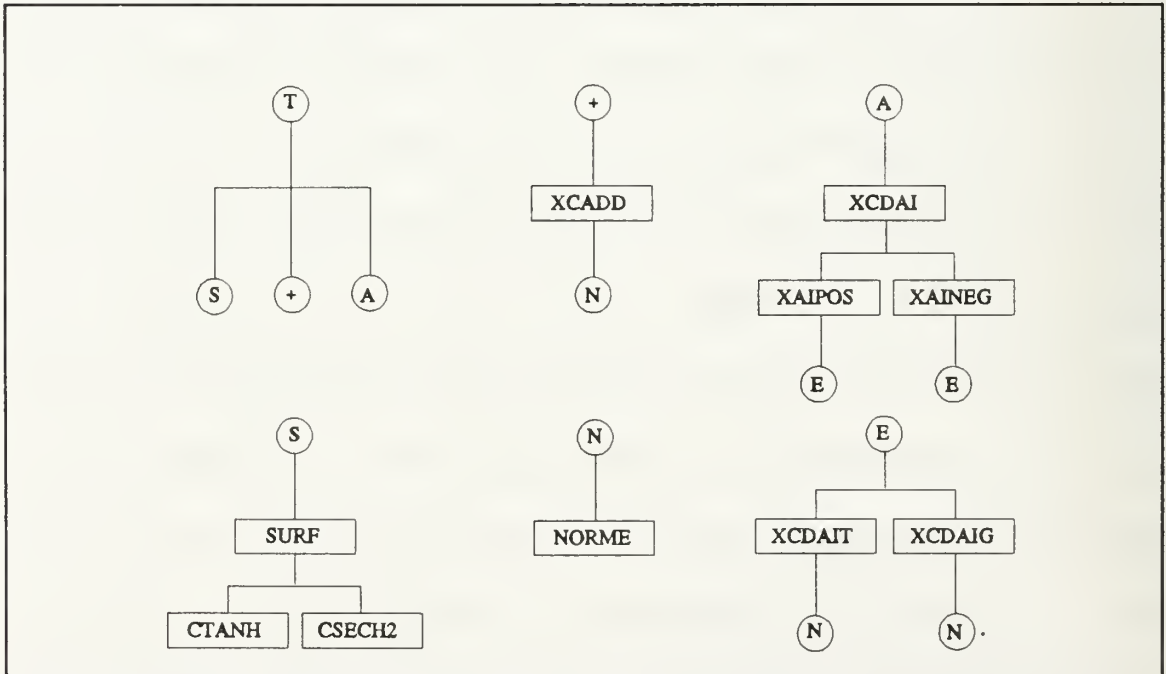


Figure 2 Original M-layer subroutines structure (continued).

discussed as they do not contribute directly to the main purpose of this program of locating the modes and computing the propagation factor.

The program structure has been altered as shown in Figures 3 and 4. Since the  $A_i$  and  $B_i$  coefficients have to be evaluated only once, they are now obtained through a call to the subroutine ABCOEF directly from the main program right after the modes are located. Several subroutines are dropped in this revision for various reasons: The subroutines NORME and NORMRE are eliminated because they are no longer needed due to the change in complex number representation; the subroutines NOMSHX, FDFDTX and DXDETR are not used because the modes are now located with adequate precision without further iteration; the subroutine ADDX is not listed separately because it is called only once and has been reduced to only a few lines which are placed where the subroutine is called in the original program. On the other hand, changes in the mode search algorithm require the addition of two new subroutines: SURF0 is a modified and simpler version of SURF; ROOTS replaces QUAD. Due to the change in complex number representation, all subroutines listed below FNDMOD and MODSUM have been revised, including their input/output lists. But except for SURF0 and ROOTS, the utilities of these subroutines are the same as those of the original ones. Descriptions of these subroutines can be found in the report by Yeoh [Ref. 4].

The most significant changes have been made in XCADD, XCDAIT and XCDAIG for adopting the complex exponent representation and improving

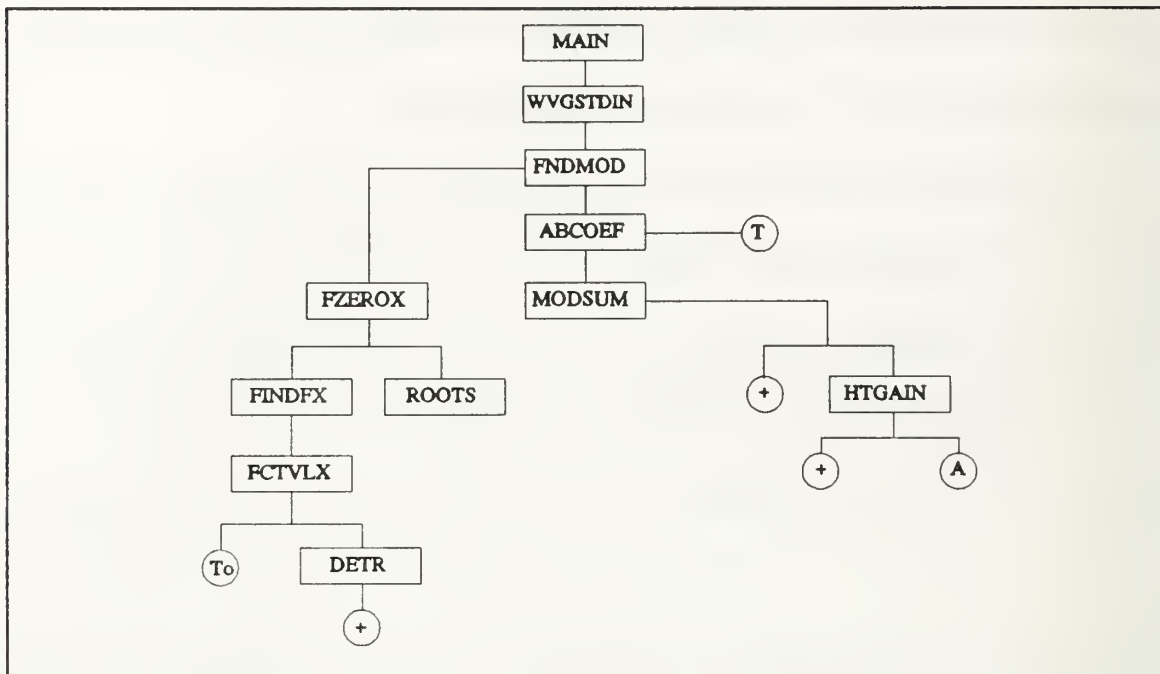


Figure 3 New M-layer subroutines structure.

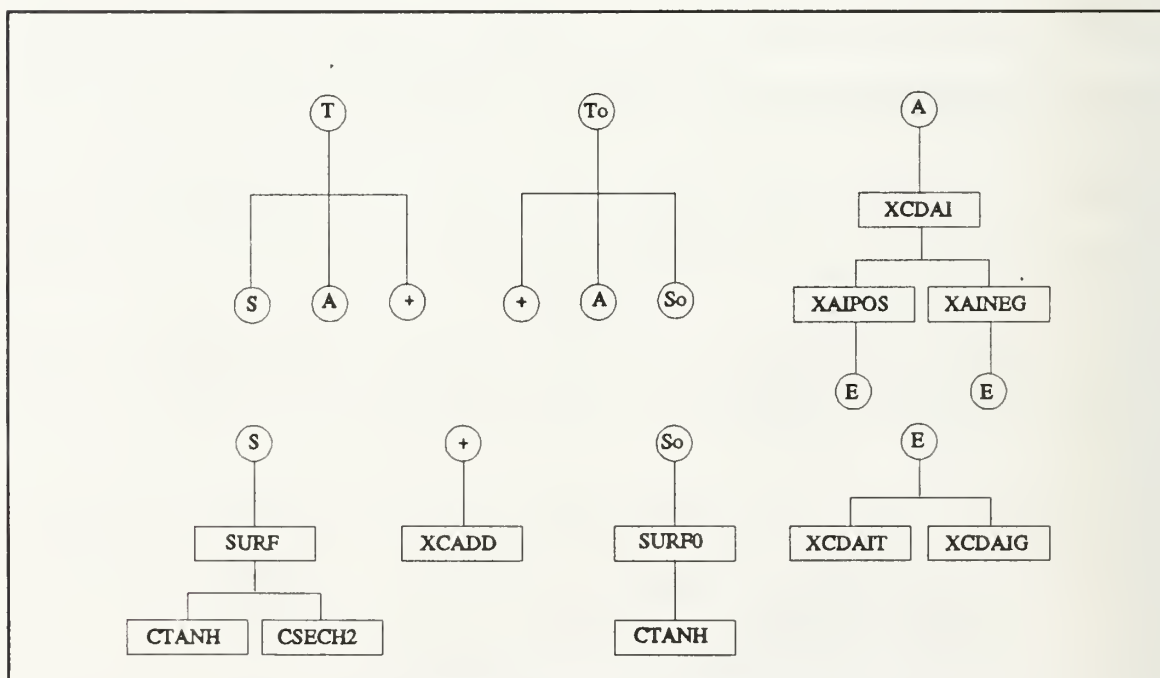


Figure 4 New M-layer subroutines structure (continued).

computation speed and accuracy; in FZEROX, FINDFX, ROOTS and SURF0 for stabilizing and simplifying the mode search algorithm; and in ABCOEF for implementing the criteria to determine the reliable manner for evaluating the  $A_i$  and  $B_i$  coefficients. These changes are discussed in the sections below. The source code listings of the completely new subroutines XCADD and ROOTS and the significantly revised subroutines FZEROX and ABCOEF, which are compiled with Microsoft FORTRAN version 5.00, are attached as Appendices A through D. Validation of the revised program has been carried out at 9.6 GHz for all the 21 profiles listed in Yeoh [Ref. 4].

#### A. ADDITION SUBROUTINE

XCADD is the subroutine implementing the addition of complex numbers under the representation by their exponents. Given the double precision complex numbers  $z_1$  and  $z_2$  as the exponents of the addends, this subroutine returns the exponent of the sum. Since a double precision number has an accuracy of 53 bits, if the real parts of  $z_1$  and  $z_2$  differ by more than 53 bits, the exponent of their sum will simply be the one of the greater real part. When cancellation becomes serious, the square root of the addends is factored out first. Then the four-term Taylor series expansions of the resulting reciprocals are summed. Since the leading term of the sum of the Taylor series is a good estimate of the sum of the reciprocals and the relative error of the four-term Taylor series sum is proportional to the fourth order

of this leading term, the threshold for invoking this interpolation procedure is set at the highest possible value of  $2^{-14}$  allowed under double precision. Experimenting with this procedure shows that this interpolation improves accuracy as long as the threshold is set at a number between  $2^{-24}$  and  $2^{-14}$ .

## **B. AIRY FUNCTION EVALUATION**

Similar to the original program, the evaluation of the Airy function adopted the algorithm prescribed by Schulten, et. al. [Ref. 6]. In the new program, changes are made to follow the advice of Schulten, et. al. concerning the region within which a Taylor series expansion, instead of the faster Gaussian quadrature, has to be used to achieve double precision accuracy. Other changes in implementing the algorithm are described below.

### **1. XCDAIT**

Due to the similarity in their Taylor series coefficients, the Airy function and its derivative are evaluated within a single loop. The relative accuracy of the derivative of the Airy function is set at the double precision limit of  $2^{-54}$ .

### **2. XCDAIG**

Six term Gaussian quadrature is used for evaluating the Airy function and its derivative outside the circle of radius 4.97 centered at (0.90, 2.80) on the complex plane. The use of four-term quadrature outside a radius of 15 from the origin suggested by Schulten, et. al. is not adopted. The six-term quadrature in this range

retains a higher accuracy while overall speed improvement by using both the four-term and the six-term quadrature appears to be minimal.

### C. MODE LOCATING

As explained in the Introduction, the modes are located at the zeroes of the modal function. These zeroes are located on the upper complex  $q_{11}$  plane. Here  $q_{11}$  is the value of  $q_1$  on the earth's surface, which, according to Eq.(2) of Chapter I, is a linear function of  $\rho^2$ . For a horizontally propagating mode,  $\rho/k$  is close to unity. The maximum range attenuation rate specified for the desired modes, which corresponds to a limit on the imaginary part of  $\rho$ , determines approximately the upper bound for the imaginary part of the  $q_{11}$  complex plane to be searched for modes. The Shellman and Morffit mode search procedure first divides the search region horizontally into "contour rectangles" each of which spans 160 meshes along the real  $q_{11}$  direction. A mesh is a square whose size is an adjustable parameter of the order  $10^{-4}$  at 9.6 GHz for most of the cases considered herein. This parameter is determined by the frequency and the slope of the modified index of reflection in the lowest layer of the profile. The search commences at the top left corner of the "contour rectangle" whose left edge has a real coordinate value close to the difference of the real parts of the  $q_{11}$  values, with the minimum modified index of refraction and the index near the surface substituted into Eq.(2) of Chapter I. After the search over the initial rectangle is completed, the program moves to search the

next rectangle until a specified maximum number of modes are found or a specified number of "contour rectangles" have been searched.

The search for zeroes makes use of the fact that a real function changes sign when it crosses a simple zero. Since a zero of a complex valued function  $F(q)$  is where both its real part and imaginary part vanish, a necessary condition for a point  $q_m$  to be a zero is that it is on the intersection of two curves defined by  $\text{Im}\{F(q)\} = 0$  and  $\text{Re}\{F(q)\} = 0$ . The program searches around a "contour rectangle" for a sign change in  $\text{Im}\{F(q)\}$  across an edge of a mesh bordering the side of the "contour rectangle" to determine that a line of  $\text{Im}\{F(q)\} = 0$  has been encountered. The search then follows this line into the meshes within the "contour rectangle", checking each mesh to see if a curve  $\text{Re}\{F(q)\} = 0$  enters the mesh under investigation. All these steps make use only of the assumption that the zeroes of the modal function are simple. Once both the curve  $\text{Im}\{F(q)\} = 0$  and the curve  $\text{Re}\{F(q)\} = 0$  are determined to be present within a mesh, the location of their possible interception is estimated. An algorithm for this estimate is required.

Shellman and Morffit [Ref. 5] introduced a further assumption that the functions  $\text{Re}\{F(q)\}$  and  $\text{Im}\{F(q)\}$  are both linear along the edges of a mesh. Based on this assumption, they try to estimate the locations where the curve  $\text{Im}\{F(q)\} = 0$  enters and leaves a mesh square, and the location of  $q_m$  if a curve  $\text{Re}\{F(q)\} = 0$  also enters the same mesh. It is obvious that information about the locations where the curves enter and leave the mesh square is not essential. Furthermore, in the 18

m duct height case, the scheme causes the search path to loop around four contiguous meshes until the search is broken up by the limit on the number of meshes to be investigated. Replacing their technique requires major changes in the subroutines involved. A new subroutine **ROOTS** is provided to estimate the location of the intersection of the curves  $\text{Im}\{F(q)\} = 0$  and  $\text{Re}\{F(q)\} = 0$ . These changes eliminate the looping problem.

Another problem is encountered in the 40 m duct height case when a large number of zeroes are found in the lower half complex  $q_{11}$  plane. These zeroes appear to belong to the reflection coefficient on the wrong sheet of the branch cut and are not waveguide modes. This happens because the search region has been extended below the real  $q_{11}$  axis to avoid the singularity in **SURF**. The problem with this singularity should have been solved within **SURF**, especially because it occurs only when the derivative of the subroutine output variable **gamma** with respect to  $q_{11}$  is computed. Since this derivative is not needed during mode search, the extension of the search region to the negative  $q_{11}$  plane is unnecessary. A simplified routine, **SURF0**, is introduced which is exactly the same as **SURF** except that it does not evaluate the derivative of **gamma**. By using this subroutine instead of **SURF**, the search path in the revised program does not avoid the real and the imaginary axes.

## 1. **FNDMOD**

The search region is limited to the upper half  $q_{11}$  plane. All the modes found are ordered according to their range attenuation rates before those numbered beyond the maximum modes allowed are abandoned.

## 2. FZEROX

Since the curve  $\text{Im}\{F(q)\} = 0$  enters into a mesh square through an edge, the values of  $\text{Im}\{F(q)\}$  must change sign over the end points of either one or all three other edges. When there is only one other edge across which  $\text{Im}\{F(q)\}$  changes sign at its end points, it is the edge across which the curve  $\text{Im}\{F(q)\} = 0$  exits the mesh square. Ambiguity arises when all edges indicate a change of sign at their end points. When this occurs, a "right turn rule" is adopted which assumes that the curve exits the edge to the right of the one along which it enters the mesh square. Such a rule avoids the retracing of the search path when the mesh square is revisited as entering this same mesh square from the left side of an edge after exiting from its right side requires a crossing of the  $\text{Im}\{F(q)\} = 0$  curve, which is prohibited under the simple zero assumption. On the other hand, the actual curve may have turned left and then returns to this mesh square, i.e., following a "left turn rule." Under such a scenario, this wrong choice would have left a segment of the curve not searched. This difficulty has not been observed during testing. In fact, the ambiguous situation seldom occurs. Note also that, as remarked above, two lines of  $\text{Im}\{F(q)\} = 0$  do not cross each other unless a higher order zero is present. Hence, only a "right turn rule" or a "left turn rule" for the curve to exit the mesh is allowed.

Exiting the opposite edge demands a pair of crossing  $\text{Im}\{F(q)\} = 0$  curves within the mesh square. This violates the assumption that all zeroes are simple. Also note that, the possibility of vanishing  $\text{Re}\{F(q)\}$  or  $\text{Im}\{F(q)\}$  values at the corners of a mesh square is eliminated through a small adjustment in **FINDFX**.

### **3. FINDFX**

Both the vertical shift away from the real  $q_{11}$  axis and the horizontal offset away from the imaginary axis are unnecessary and have been removed from this routine. Furthermore, as a result of converting to the complex exponent representation, the sine and cosine of the argument of the modal functions are examined for sign changes in **FZEROX**. This is implemented in **FINDFX** by including the cosine and sine values of the argument of the modal function in the output list. To avoid the indeterminate case when either the real or the imaginary part of the modal function becomes zero at any corner of a mesh square, the argument for computing the cosine and sine values is increased by  $2^{-53}$  when this occurs. This is equivalent to a consistent small distortion of the particular corner of the mesh square. This will not cause any error in locating the zero because **FINDFX** still returns separately the unmodified exponent of the value of the modal function.

### **4. ROOTS**

Assuming that the modal function is analytic within the mesh, this subroutine utilizes the values of the modal function at the four corners of the mesh

square to determine the Taylor series expansion coefficients of the modal function to the third order. The roots of this cubic polynomial are then located using Cardan's solution by radicals. If the higher order coefficients fall below machine resolution for a root within the mesh square, these coefficients are regarded as zero and the order of the polynomial is reduced and can be solved more expediently. If the function is determined to be constant over the mesh square, the center of the square is taken as the root location.

#### D. EVALUATING $A_i$ AND $B_i$

As discussed in the Introduction, the  $A_i$  and  $B_i$  coefficients can be evaluated either from the top level down or from the lowest level up. These two procedures are simply called "integration down" and "integration up", respectively, in the original documentation [Ref. 4]. The location of a mode has been called an eigenvalue. That the results of "integration down" and "integration up" agree is a manifestation that the eigenvalue is located accurately.

The subroutine ABCOEF evaluates the coefficients  $A_i$  and  $B_i$  for each mode. If the range attenuation rate for a mode is greater than 0.1 dB/km, the coefficients are evaluated from the lowest layer up. Otherwise, it is evaluated from the top layer down. It is obvious that such a rule must be implemented because the results of "integration up" and "integration down" do not agree for many modes. Efforts are made to determine the cause of this discrepancy and to devise a means to resolve it.

Investigation reveals that inadequate precision in the location of the modes is one source of the problem. Since the  $B_i$  coefficients depend on the  $A_i$  coefficients, while the  $A_i$  coefficients are obtained directly, only the  $A_i$  coefficients need to be examined. The  $A_i$  coefficients of the six modes of lowest range attenuation rates for all 21 profiles except the one without evaporation duct are computed using eigenvalues of different accuracy controlled by the first order Newton-Raphson iteration method. Table 1 shows the  $A_i$  coefficient computed with the new program. They are arranged from the top layer down. In the  $i$ -th layer, the  $A_i$  coefficient computed by "integration downward" depends only on  $A_{i+1}$  in the layer above while that computed by "integration upward" depends only on  $A_{i-1}$  in the layer below. Hence in each layer, the coefficient obtained by "integration downward" is listed above that obtained by "integration upward". There are five sets of  $A_i$  values listed, with the magnitudes given in powers of 10, and the phase given as a multiple of  $\pi$ . They are obtained from eigenvalues of decreasing accuracy -the one used to compute the left most column being the most accurate. The first set is computed using an eigenvalue having a relative accuracy of  $2^{-40}$ . The second set uses an eigenvalue with a relative accuracy of  $2^{-36}$ . The relative accuracy of the eigenvalue for the third set is  $2^{-36}$ . For the fourth set, the first order Newton-Raphson iteration of the mode location is set at an absolute accuracy of 0.03 of the mesh size, same as that specified in the original program. The eigenvalue for the right most set is the mode location estimated by ROOTS without modification by the Newton-Raphson iteration. It is

**TABLE 1. IMPROVING  $A_i$  ACCURACY WITH EIGENVALUE 18 M DUCT**

mode 4		q-eigenvalue: .1888574325176803D+00		.1080678744810598D-01		eigenvalue difference: .00D+00 .00D+00		-.49D-13 -.66D-15		.12D-11 -.12D-10		.15D-06 .60D-07		
layer #	Ai/down	Ai/down		Ai/down		Ai/down		Ai/down		Ai/down		Ai/down		
layer #	Ai/up	Ai/up		Ai/up		Ai/up		Ai/up		Ai/up		Ai/up		
18	.0261	.6719	.0261	.6719	.0261	.6719	.0261	.6719	.0261	.6719	.0261	.6719	.0261	.6719
18	.0261	.6719	.0261	.6719	.0261	.6719	.0261	.6719	.0261	.6719	.0261	.6719	.0261	.6719
17	-.0625	.6368	-.0625	.6368	-.0625	.6368	-.0625	.6368	-.0625	.6368	-.0625	.6368	-.0625	.6368
17	-.0625	.6368	-.0625	.6368	-.0625	.6368	-.0625	.6368	-.0625	.6368	-.0625	.6368	-.0625	.6368
16	.0139	.7440	.0139	.7440	.0139	.7440	.0139	.7440	.0139	.7440	.0139	.7440	.0139	.7440
16	.0139	.7440	.0139	.7440	.0139	.7440	.0139	.7440	.0139	.7440	.0139	.7440	.0139	.7440
15	.1216	.6353	.1216	.6353	.1216	.6353	.1216	.6353	.1216	.6353	.1216	.6353	.1216	.6353
15	.1216	.6353	.1216	.6353	.1216	.6353	.1216	.6353	.1216	.6353	.1216	.6353	.1216	.6353
14	.0166	.5471	.0166	.5471	.0166	.5471	.0166	.5471	.0166	.5471	.0166	.5471	.0166	.5471
14	.0166	.5471	.0166	.5471	.0166	.5471	.0166	.5471	.0166	.5471	.0166	.5471	.0166	.5471
13	-.1565	.5310	-.1565	.5310	-.1565	.5310	-.1565	.5310	-.1565	.5310	-.1565	.5310	-.1565	.5310
13	-.1565	.5310	-.1565	.5310	-.1565	.5310	-.1565	.5310	-.1565	.5310	-.1565	.5310	-.1565	.5310
12	-.3842	.5659	-.3842	.5659	-.3842	.5659	-.3842	.5659	-.3842	.5659	-.3842	.5659	-.3843	.5659
12	-.3842	.5659	-.3842	.5659	-.3842	.5659	-.3842	.5659	-.3842	.5659	-.3842	.5659	-.3842	.5659
11	-2.2002	-.8081	-2.2002	-.8081	-2.2002	-.8081	-2.2002	-.8081	-2.2002	-.8081	-2.2002	-.8081	-2.1909	-.8068
11	-2.2002	-.8081	-2.2002	-.8081	-2.2002	-.8081	-2.2002	-.8081	-2.2002	-.8081	-2.2002	-.8081	-2.2002	-.8081
10	-5.4648	.2423	-5.4648	.2423	-5.4648	.2423	-5.4648	.2423	-5.4654	.2423	-4.1810	-.2161	-5.4647	.2423
10	-5.4648	.2423	-5.4648	.2423	-5.4648	.2423	-5.4648	.2423	-5.4648	.2423	-5.4647	.2423	-5.4647	.2423
9	-3.6974	-.6979	-3.6974	-.6979	-3.6974	-.6980	-3.6974	-.6980	-3.6783	-.7012	-6.4611	-.2121	-3.6977	-.6978
9	-3.6978	-.6978	-3.6978	-.6978	-3.6978	-.6978	-3.6978	-.6978	-3.6978	-.6978	-3.6977	-.6978	-3.6977	-.6978
8	.3459	-.7982	.3459	-.7982	.3460	-.7982	.3459	-.7982	.3482	-.7926	-1.9078	-.9148	.3459	-.7983
8	.3459	-.7983	.3459	-.7983	.3459	-.7983	.3459	-.7983	.3459	-.7983	.3459	-.7983	.3459	-.7983
7	.4098	.8794	.4098	.8794	.4098	.8794	.4098	.8794	.4136	.8836	-1.0899	.5364	.4097	.8793
7	.4097	.8793	.4097	.8793	.4097	.8793	.4097	.8793	.4097	.8793	.4097	.8793	.4097	.8793
6	.3480	.8161	.3480	.8161	.3480	.8161	.3480	.8161	.3526	.8205	-.5879	.4005	.3479	.8160
6	.3479	.8160	.3479	.8160	.3479	.8160	.3479	.8160	.3479	.8160	.3479	.8160	.3479	.8160
5	.2923	.8304	.2923	.8304	.2923	.8304	.2923	.8304	.2972	.8358	-.3490	.3749	.2922	.8303
5	.2922	.8303	.2922	.8303	.2922	.8303	.2922	.8303	.2922	.8303	.2922	.8303	.2922	.8303
4	.2359	.8619	.2359	.8619	.2360	.8619	.2359	.8619	.2408	.8690	-.2058	.3731	.2358	.8618
4	.2358	.8618	.2358	.8618	.2358	.8618	.2358	.8618	.2358	.8618	.2358	.8618	.2358	.8618
3	.1831	.8910	.1831	.8910	.1832	.8910	.1831	.8910	.1878	.9003	-.1250	.3753	.1831	.8908
3	.1831	.8908	.1831	.8908	.1831	.8908	.1831	.8908	.1831	.8908	.1831	.8908	.1831	.8908
2	.1300	.9149	.1300	.9149	.1301	.9149	.1300	.9149	.1342	.9275	-.0734	.3750	.1300	.9146
2	.1300	.9146	.1300	.9146	.1300	.9146	.1300	.9146	.1300	.9146	.1300	.9146	.1300	.9146
1	.0586	.9335	.0586	.9335	.0588	.9335	.0586	.9335	.0618	.9545	-.0318	.3670	.0586	.9331
1	.0586	.9331	.0586	.9331	.0586	.9331	.0586	.9331	.0586	.9331	.0586	.9331	.0586	.9331

clear that, for this mode, the difference between these two methods of computing the coefficients becomes negligible as the accuracy in mode location increases. For example, in the 8-th layer, the magnitude of  $A_i$  computed by integrating downward changes from  $-1.9078$  to  $0.3482$ , to  $0.3460$  to  $0.3459$ , which agrees with the result computed by integrating upward. The phase follows the same trend to an agreement within  $0.001\pi$ . Table 2 shows a similar set of output, but the coefficients fail to agree even when the relative accuracy is increased to  $2^{-40}$ . Note that the actual difference in both the real part and the imaginary part of the two most accurate eigenvalues is about  $2^{-48}$ . Double precision accuracy appears to be insufficient for the coefficients computed with these two methods to agree for all modes. Some interesting features can be observed in both tables, which are present in all 120 sets of values computed. When disagreement is present in one set of  $A_i$  coefficients, such as those in either Table 1 or Table 2, the change toward smaller differences with improving eigenvalue accuracy occurs mainly in one way of computation, but not both. For example, in Table 1, the values of "integration downward" improve with better eigenvalue accuracy, while those computed by "integrating upward" change little. In Table 2, the results of "integration downward" are the ones that are holding steady as the accuracy in eigenvalue improves. Furthermore, when disagreement occurs, the layer in which the  $A_i$  coefficient has the smallest magnitude, i.e., the one having the most negative power of 10, divides the table into two parts. The results of two different ways of computation agree in the layers above this one if they disagree in those below it, and vice versa. No explanation will be attempted. Instead, practical rules are drawn up



to take advantage of these facts. In Table 1, the process of "integration upward" goes through the troublesome 10-th layer and produces results which agree with the results of downward integration before the downward process goes through the 10-th layer. On the other hand, the downward integration is tripped up going across the 10-th layer and produces results which fail to agree with the results from upward integration. It is clear that the results from upward integration are the correct ones. This conclusion is further supported by the fact that improving the accuracy of the eigenvalue does not change significantly the results of upward integration. Similar argument leads to the conclusion that in Table 2, the results of downward integration are the correct values.

It can be concluded from the above observations that one of the methods of computing the  $A_i$  coefficients converges to the correct value much faster than the other. It is also found that this method of faster convergence is always able to arrive at the correct values for  $A_i$  for all the cases under investigation.

Table 3 lists the statistics of the method of integration which yields the correct  $A_i$  coefficients for each of the 120 modes investigated. The differences in magnitudes and phases in the lowest layer and in the layer below the highest are also listed. Since for most of the cases, when disagreement in  $A_i$  values occurs, the correct integration is upward -this is used as the default. To decide that downward integration should be utilized, the following steps are taken: The first  $A_i$  value of downward integration is computed and compared to the value from upward integration. If the magnitudes in dB disagree by less than 0.02 dB, their phases will

be checked. If the phases differ by less than  $10^{-3}\pi$ , the agreement is deemed acceptable and the  $A_i$  and  $B_i$  coefficients computed from the lowest layer up are used. Otherwise, the coefficients are re-evaluated again from the highest layer down.

Once the correct method of evaluating the  $A_i$  and  $B_i$  coefficients is used, the accuracy of the mode location becomes less critical. For all the cases investigated, the  $A_i$  coefficients obtained from mode locations estimated with or without the Newton-Raphson first order iteration differ only by 0.06 dB in magnitude and  $0.0013\pi$  in phase at most. In fact, few cases show differences more than 0.002 dB and  $0.0001\pi$ . The Newton-Raphson iteration is not needed. Hence the subroutines NOMSHX, FDFDTX and DXDETR are removed.

**TABLE 3. STATISTICS FOR EVALUATING  $A_i$  COEFFICIENT**

Duct height	Mode #	Evaluating Method		$\Delta  A_i $ (dB)		$\Delta \arg(A_i)/\pi$	
				Layer		Layer	
		up	down	bottom	top-1	bottom	top-1
02	1	x					
	2	x					
	3	x					
	3	x		0.172		0.093	
	5	x					
	6	x		8.362		1.3234	
04	1	x					
	2	x					
	3	x		0.008		0.0002	
	3	x		1.030		1.8717	
	5	x		7.814		1.2948	
	6	x		0.002		0.0001	
06	1	x					
	2	x		0.002		0.0004	
	3	x		0.522		0.0158	
	4	x					
	5	x		13.278		0.4377	
	6	x		0.002		0.0001	
08	1	x					
	2	x		0.002			
	3	x		0.002		0.0001	0.0001
	4	x		0.016		0.0026	
	5	x		4.066		0.6355	
	6	x		3.978		0.6186	

TABLE 3. CONTINUED 1.

Duct height	Mode #	Evaluating Method		$\Delta  A_i $ (dB)		$\Delta \arg(A_i)/\pi$	
				Layer		Layer	
		up	down	bottom	top-1	bottom	top-1
10	1	x					
	2	x				0.0002	
	3	x				0.0001	
	4	x		0.04		0.0008	
	5	x		0.206		0.0402	0.0001
	6	x		0.002		0.0008	
12	1	x					
	2	x		0.006		0.0008	
	3	x		0.004			
	4	x		1.808		0.5661	
	5	x		1.732		0.5429	
	6	x		1.472		0.0414	
14	1	x					
	2	x		0.002		0.0008	
	3	x		0.178		0.0052	
	4	x		0.024		0.0005	
	5	x		0.004		0.0001	
	6	x		0.85		0.4711	
16	1	4					
	2	4		0.006		0.0002	
	3	x		0.004			
	4	x		0.006		0.0001	
	5	x		0.002		0.0001	
	6	x		0.004		0.0077	

TABLE 3. CONTINUED 2.

Duct height	Mode #	Evaluating Method		$\Delta  A_i $ (dB)		$\Delta \arg(A_i)/\pi$	
				Layer		Layer	
		up	down	bottom	top-1	bottom	top-1
18	1		x		0.008		0.0001
	2	4		0.002		0.0001	
	3	x				0.0001	
	4	x					
	6	x		0.016		0.0003	
	6	4		0.002			
20	1		4		0.078		0.0164
	2	x					
	3	x		0.002		0.0001	
	4	x				0.0008	
	5	4		0.16		0.0195	
	6	4		0.002		0.0001	
22	1		x		8.708		0.239
	2	x					
	3	x		0.004			
	4	4		0.016			
	6	x		0.002		0.0001	
	6	x		0.31		0.0117	
24	1	x					
	2		x		0.868		0.2842
	3	x		0.006		0.0009	
	4	x		0.002		0.0001	
	5	x		0.026		0.0009	
	6	x		0.008		0.0001	

TABLE 3. CONTINUED 3.

Duct height	Mode #	Evaluating Method		$\Delta  A_i $ (dB)		$\Delta \arg(A_i)/\pi$	
				Layer		Layer	
		up	down	bottom	top-1	bottom	top-1
26	1	x		0.002	0.002	0.0002	0.0001
	2		x		4.308		0.121
	3	x		0.006			
	4	x		0.002		0.0001	
	5	x				0.0001	
	6	6			0.034		0.0039
28	1		x		0.028		0.0014
	2		x		4.806		0.0728
	3	x					
	3	x					
	5	x		0.004	0.002	0.0002	
	6	x		0.004		0.0019	
30	1		x		1.562		0.0165
	2	6					
	3		x		0.718		0.2455
	x	6					
	5	x		0.004			
	6	x		0.724		0.0522	
32	1		x		3.194		0.1648
	2	x		0.002			
	3		x		13.12		0.1026
	4	x		0.002			
	5	x		0.382		0.0099	
	6	x		0.002		0.0001	

TABLE 3. CONTINUED 4.

Duct height	Mode #	Evaluating Method		$\Delta  A_i $ (dB)		$\Delta \arg(A_i)/\pi$	
				Layer		Layer	
		up	down	bottom	top-1	bottom	top-1
34	1	x		0.002	0.002		
	2		x		13.456		0.0311
	3		x		1.014		0.2347
	4	x					
	5	x		0.03		0.0006	
	6	x		0.014		0.0006	
36	1		x			0.0001	0.0014
	2		x		1.686		0.2224
	4		x		4.724		0.0919
	4	x					
	6	x		0.006		0.0001	
	6	x		0.02		0.0001	
38	1		x		0.996		0.0115
	2		x		4.974		0.0152
	3	x					
	4		x		5.052		0.0417
	5	x				0.0001	
	6	x		0.002			
40	1	x		0.002	0.002		
	2		x		3.85		0.1226
	3		x		3.568		0.1555
	4		x		3.448		0.1678
	5	x				0.0001	
	6	x					

### III. CONCLUSIONS AND RECOMMENDATIONS

#### A. Performance

This revision of M-Layer converts the extended complex number representation of an exponentially large or small number into the direct representation by its complex exponent. The accuracy of the computation has been improved in two ways: First, an interpolation algorithm has been devised when severe cancellation of the addends is detected. Secondly, accuracy for the evaluation of the Airy function has been improved, not just by summing the Taylor series to double precision resolution and by adopting six-term Gaussian quadrature, but also by expanding the region within which the more expedient Gaussian quadrature is excluded in favor of the more accurate, but time-consuming, Taylor series summation. The improvement in accuracy is most easily seen from Table 1.

As discussed in the Introduction, evaluating the  $A_i$  and  $B_i$  coefficients either from the lowest layer up (integration up), or from the top layer down (integration down), must result in the same values. This property provides a consistency check for the accuracy of the computation. For the six modes of lowest range attenuation rates of the 20 profiles of different duct heights, Table 1 lists the maximum difference for each mode which shows a discrepancy between these two methods of evaluating the  $A_i$  coefficients. For each profile, the maximum value in magnitude

**TABLE 1. MAXIMUM DIFFERENCE IN  $A_1$  COEFFICIENT BETWEEN INTEGRATION UP AND DOWN**

Duct height	Mode #	Difference in A coefficient			
		Magnitude difference in (dB)		Phase difference over $0.1 \pi$	
		old	new	old	new
02	6	5.22		Yes	
	6	61.16		Yes	
04	6	22.46	2.3		
	5	106.9		Yes	
06	4	8.62		Yes	
	5	32.36			
08	5	77.84		Yes	
	6	44.9		Yes	
14	6			Yes	
12	4	69.38		Yes	
	5	46.32		Yes	
	6	7.46		Yes	
14	6	30.6		Yes	
22	6	8.64		Yes	
24	2	80.48		Yes	
26	2	110.68		Yes	
28	2	106.9	67.68	Yes	Yes
30	3	173.28	143.42	Yes	Yes
32	1	11.38		Yes	
	3	525.04	188.04	Yes	Yes

**TABLE 1. CONTINUED**

Duct height	Mode #	Difference in A coefficient			
		Magnitude difference in (dB)		Phase difference over $0.1 \pi$	
		old	new	old	new
34	2	37.98		Yes	
	4	715.7	209.94	Yes	Yes
34	2	112.74		Yes	
	4	957.92	231.68	Yes	Yes
38	2	107.44	52.26	Yes	Yes
	4	1249	255.8	Yes	Yes
34	3	167	112.72	Yes	Yes
	4	823.56	258.18	Yes	Yes
	Magnitude difference within 2dB are not listed.				

difference in dB among all the layers is listed if it is greater than 2. If the phases of the coefficients deviate more than  $0.1\pi$  in any layer, that particular mode is also singled out. The location of the mode of the revised program is within a relative accuracy of  $2^{-40}$  achieved through first order Newton-Raphson iteration. Even though discrepancies still exist when the duct is 28 meters or higher, it is clear that the revised program computes more accurately than the original one.

For the cases where the two methods of evaluating the  $A_i$  and  $B_i$  coefficients disagree, it has been observed that one of the methods always leads to  $A_i$  values which are little changed when the accuracy in mode location is varied, while the other method produces  $A_i$  values which shift toward the results of the other method

as the accuracy of mode location improves. Based on this observation, a consistency check is implemented into the program to identify the method which converges better. For the 120 cases investigated, when this method of faster convergence is used, the  $A_i$  coefficients obtained from mode locations estimated with or without the Newton-Raphson first order iteration differ only by 0.06 dB in magnitude and  $0.0013\pi$  in phase at most. In fact, few cases show differences more than 0.002 dB and  $0.0001\pi$ . This allowed the Newton-Raphson iteration to be removed in this revision.

Table 2 compares the performance between the original and the revised programs. The time spent to find the modes has been reduced by an average of 22.58%. The revised program can always produce the modes found by the original program. Moreover, the mode search is stable for the new program: the time it requires to search for the modes is about the same for similar profiles. The sudden jumps in mode search time for the 24 m and the 40 m cases, which indicate troubles during the search, no longer happen.

With the proper method of evaluating the  $A_i$  and  $B_i$  coefficients determined by the consistency check, the output of the revised program differs from the original program in some cases. The most serious deviation has been observed for the 38 m duct height case as shown in Tables 3 and 4. For example, at a range of 36.5 km with the transmitter at a height of 25 m and the receiver at 10 m, the coherent path loss is 175.93 dB from the original program, and is 167.90 dB from the revised program.

**TABLE 2 OVERALL MODE SEARCH PERFORMANCE COMPARISON**

DUCT HEIGHT (meter)	ORIGINAL		REVISED		Time Improvement
	Time	Modes	Time	Modes	
00	0:00:37	3	0:00:35	3	5.40%
02	0:32:14	9	0:31:55	9	0.98%
04	1:14:12	25	1:05:04	25	12.31%
08	2:10:18	53	1:56:50	53	10.33%
08	0:35:58	39	0:29:25	39	18.21%
10	0:53:24	59	0:48:32	61	9.11%
12	1:09:40	86	1:01:44	89	11.39%
14	1:20:42	94	1:11:13	97	11.75%
16	1:54:35	95	1:46:07	97	31.82%
40	1:45:09	109	1:27:15	104	17.02%
20	1:46:19	108	1:34:20	105	11.27%
22	1:52:54	105	1:35:18	106	15.59%
24	3:42:59	109	1:46:47	107	52.11%
28	2:07:42	109	1:43:55	106	18.62%
28	2:00:05	107	1:44:59	109	12.57%
30	1:59:59	107	1:46:49	106	11.39%
32	1:55:29	108	1:42:58	110	10.84%
34	2:29:57	109	2:15:58	111	9.32%
36	2:31:40	109	2:17:20	112	9.45%
38	2:38:44	110	2:18:09	111	12.97%
40	5:41:17	95	2:39:39	111	53.22%
Total	40:23:54		31:16:22		22.58%

**TABLE 3. ORIGINAL PROGRAM 38 M DUCT OUTPUT**

frequency = 9600.0000 mhz							
range (km)	zt (m)	zr (m)	coherent mode sum (dB)	incoherent mode sum (dB)	coherent path loss (dB)	incoherent path loss (dB)	horizon (km)
27.3	25.0	4.0	-15.30	-15.62	156.10	156.43	28.9
27.3	25.0	6.0	.62	-2.35	140.18	143.16	30.7
27.3	25.0	8.0	-1.11	-4.21	141.92	145.01	32.3
27.3	25.0	10.0	-27.26	-12.66	168.06	153.46	33.6
36.5	25.0	4.0	-16.94	-16.62	160.28	159.96	28.9
36.5	25.0	6.0	-.73	-2.05	144.07	145.39	30.7
36.5	25.0	8.0	-2.21	-3.72	145.55	147.06	32.3
36.5	25.0	10.0	-32.59	-14.29	175.93	157.64	33.6
45.8	25.0	4.0	-19.89	-16.96	165.20	162.26	28.9
45.8	25.0	6.0	-2.81	-1.89	148.11	147.19	30.7
45.8	25.0	8.0	-4.11	-3.43	149.41	148.74	32.3
45.8	25.0	10.0	-28.57	-15.22	173.88	160.52	33.6

**TABLE 4. REVISED PROGRAM 38 M DUCT OUTPUT**

frequency = 9600.0000 mhz							
range (km)	zt (m)	zr (m)	coherent mode sum (dB)	incoherent mode sum (dB)	coherent path loss (dB)	incoherent path loss (dB)	horizon (km)
27.3	25.0	4.0	-14.38	-15.66	155.18	156.47	28.9
27.3	25.0	6.0	.42	-2.37	140.39	143.18	30.7
27.3	25.0	8.0	-1.52	-4.21	142.33	145.02	32.3
27.3	25.0	10.0	-21.20	-12.51	162.01	153.31	33.6
36.5	25.0	4.0	-17.32	-16.60	160.66	159.94	28.9
36.5	25.0	6.0	-.48	-2.08	143.82	145.42	30.7
36.5	25.0	8.0	-1.62	-3.73	144.96	147.07	32.3
36.5	25.0	10.0	-24.56	-14.04	167.90	157.38	33.6
45.8	25.0	4.0	-20.26	-16.93	165.57	162.23	28.9
45.8	25.0	6.0	-3.14	-1.93	148.44	147.23	30.7
45.8	25.0	8.0	-4.62	-3.46	149.92	148.76	32.3
45.8	25.0	10.0	-25.40	-14.90	170.71	160.21	33.6

## **B. Recommendations**

The mode search protocol of this program needs to be revised. Since the search is limited by the number of modes to be found and the maximum range attenuation rate accepted, it is more logical to begin with locating the mode of the lowest attenuation, and then proceed to look for the next one in the order of increasing attenuation rate. Furthermore, there appears to be only a single 'phase line' of vanishing real part of the modal function on which all the modes are located. This line extends from lower to higher range attenuation rates. The partition of the search region into rectangles, as has been done in this program, tends to cut the "phase line" into segments before the program starts to search for the end points of these segments and then follow the segments in different directions. It is clear that a better way for mode search is to find the lower end of the single "phase line" then follow it to the other end.

## APPENDIX A: SUBROUTINE XCADD

This Appendix lists the addition subroutine `XCADD` which returns the complex exponent of the sum when the complex exponents of the addends are given. This is a complete re-write of the original subroutine of the same name.

```

1      subroutine xcadd(zx,z1x,z2x)
2  c
3  c      Given z1x and z2x, this subroutine adds the two complex numbers
4  c      z1=exp(z1x) and z2=exp(z2x) for z=exp(zx) and returns zx.
5  c
6  c      inputs...
7  c          z1x=complex exponent of the complex number z1
8  c          z2x=complex exponent of the complex number z2
9  c
10 c      outputs...
11 c          zx=complex exponent of the complex number z
12 c
13 c      subroutines called...
14 c
15 c*****
16      implicit real*8 (a-h,o-z)
17      complex*16 zx,z1x,z2x,zt1x,zt2x,clogzh,dsum,czero,cerrx,cone,chpi
18      parameter(pi=3.141592653589793238462643d0,twopi=2.d0*pi,
19 + hpi=0.5d0*pi,zero=0.d0,c16=1.d0/6.d0,
20 + bit14=1.d0/16384.d0,bit24=bit14/1024.d0,ctol=bit14,
21 + dpi=2259.d0/4294967296.d0/4294967296.d0,hdpi=dpi/2.d0,
22 + e2m54=-3.742994775023704819d1,e2p27=-0.5d0*e2m54,
23 + chpi=(0.d0,1.57079632679489661923132d0),cone=(1.d0,0.d0),
24 + czero=(0.d0,0.d0),cerrx=(-3.742994775023704819d1,0.d0))
25 c      cerrx=e2m54=-54*log(2)=exponent below machine accuracy
26      dimension ztmp(2),stmp(2)
27      equivalence (ztmp,clogzh),(stmp,dsum)
28 c*****
29 c      Replace the input variables with a local variable so that
30 c      equations in the form of y=x+y will not lead to confusion.
31 c
32      zt1x=z1x
33      zt2x=z2x
34 c
35      clogzh=0.5d0*(zt1x-zt2x)
36      dxh=ztmp(1)
37      if(dxh .lt. zero) then
38          zx=zt2x
39          dxh=-dxh
40      else
41          zx=zt1x
42      end if
43 c*****
44 c      machine accuracy = 2**(-53)
45 c      2**(27)=e**e2p27

```

```

46 c
47     if (dxh .ge. e2p27) then
48         return
49     else
50         zx=0.5d0*(zt1x+zt2x)
51         dsum=cdexp(clogzh)
52         dsum=1.d0/dsum+dsum
53         if (cdabs(dsum) .gt. ctol) then
54             zx=cdlog(dsum)+zx
55         else
56 c     Cancellation is serious. Im[clogzh] is close to pi/2 or -pi/2.
57             yi=dnint(ztmp(2)/twopi)*2.d0
58             ztmp(2)=ztmp(2)-pi*yi
59             dyi=dpi*yi
60             if (ztmp(2) .lt. zero) then
61                 clogzh=-clogzh
62                 dyi=-dyi
63             end if
64             ztmp(2)=(ztmp(2)-hpi)-hdpi-dyi
65             dsum=2.d0*clogzh*(cone+c16*clogzh*clogzh)
66             if (dsum .eq. czero) then
67 c     Note that a complete cancellation of two nonzero numbers of
68 c     order one is considered to be as accurate as what is allowed
69 c     by the machine and the algorithm.
70                 zx=cerrx+chpi+zx
71             else
72                 dsum=cdlog(dsum)
73                 if (stmp(1) .lt. e2m54) stmp(1)=e2m54
74                 zx=dsum+chpi+zx
75             end if
76         end if
77         return
78     end if
79 c
80     end

```

## **APPENDIX B: SUBROUTINE FZEROX**

This Appendix includes the listing of the subroutine FZEROX which searches identifies the meshes which may contain modes within a contour rectangle. The Shellman-Morffit mode locating algorithm has been completely replaced.

```

1      subroutine fzerox(tleft,tright,tbot,ttop,tmesh,zeros,ni,nf)
2  c*****
3  c  fzerox is a routine for finding the zeroes of a complex function, f,
4  c  which lie within a specified rectangular region of the
5  c  complex q11 plane, assuming that the function has only
6  c  simple zeroes over this rectangle.
7  c
8  c  parameters specifying the search rectangle:
9  c  tleft - value of the real part of q11 at the left edge.
10 c  tright- value of the real part of q11 at the right edge.
11 c  tbot  - value of the imaginary part of q11 at the bottom edge.
12 c      (this is set to 0.)
13 c  ttop  - value of the imaginary part of q11 at the top edge.
14 c  tmesh - set equal to about half the average spacing between
15 c        zeroes within the rectangle. A smaller value may be used
16 c        as a safety measure, but too small a value will result
17 c        in excessively long run time.
18 c  zeros - output list of (complex) values of q11 at which
19 c        zeroes are found.
20 c  nf-ni - the number of zeroes found
21 c
22 c  subroutines called--
23 c      findfx
24 c      roots
25 c      nomshx
26 c*****
27      implicit double precision (a-h,o-z)
28      complex*16 f10,f01,f11,fxnew,fxold,fx00,fx10,fx01,fx11,
29      +          czero,one,ci,sol,zeros
30      parameter(czero=(0.d0,0.d0),one=(1.d0,0.d0),ci=(0.d0,1.d0))
31  $include: 'mlaparm.inc'
      ***** Begin listing of: mlaparm.inc
1  c
2  c      include file to define the
3  c          maximum # of layers (mxlayr)
4  c          maximum # of modes (mxmode)
5  c
6  c      parameter (mxlayr=35 )
7  c      parameter (mxmode=127)
      ***** End listing of: mlaparm.inc
32      dimension kedge1(100),kedge2(100),kedge3(100),kedge4(100),
33 c  +   loc12r(mxmode),loc12i(mxmode),loc23r(mxmode),loc23i(mxmode),
34 c  +   loc34r(mxmode),loc34i(mxmode),loc41r(mxmode),loc41i(mxmode),
35 c  +   sol(3),theta(2),zeros(2*mxmode+1)
36 c

```

```

37 c
38     common /tmccom/tmesh
39 c*****
40 c     maxnsq - maximum number of mesh squares allowed on any one
41 c         phase line
42 c     maxnt - maximum number of times fzero will reduce tmesh
43 c
44     maxnsq=3*max0(int((ttop-tbot)/tmsh0),int((tright-tleft)/tmsh0))
45     maxnt=2
46 c*****
47     tmesh = tmsh0
48     ntime = 0
49     go to 7
50 c
51 5     tmesh=tmesh/2.0d0
52     ntime = ntime+1
53     if(ntime .gt. maxnt) go to 97
54 c
55 7     continue
56
57 c*****
58 c     calculate coordinates of rectangle edges in tmesh units
59 c
60     jlt = idnint(tleft/tmesh-0.5d0)
61     jrt = idnint(tright/tmesh+0.5d0)
62     jtop = idnint(ttop/tmesh+1.5d0)
63     jbot = 0
64 c
65 c     initialize parameters for starting search at upper left
66 c     corner of search rectangle
67 c
68     ki = jtop
69     kr = jlt
70     kedge = 1
71     call findfx(kr,ki,fxnew,xnew,ynew)
72     nre1=0
73     nre2=0
74     nre3=0
75     nre4=0
76     knot12=0
77     knot23=0
78     knot34=0
79     knot41=0
80     nf=ni
81     ni1=ni+1

```

```

82      go to 15
83 c*****
84 10   continue
85     if(nrzl .lt. 2) go to 15
86 c   write(16,2000) nrzl
87     go to 5
88 15   nrzl=0
89     nrsqu = 0
90 20   fxold=fxnew
91     xold=xnew
92     yold=ynew
93     go to (21,26,31,36),kedge
94 c*****
95 c   search along left edge of rectangle for changes in the
96 c   sign of imag(f)
97 c
98 21   continue
99     if(ki.eq.jbot) then
100       kedge=2
101       go to 26
102     end if
103     ki = ki-1
104     call findfx(kr,ki,fxnew,xnew,ynew)
105     if (yold*ynew .gt. 0.d0) go to 20
106     if(nre1.eq.0) go to 23
107 c
108 c   check if crossing point has been previously found
109 c
110     do 22 k=1,nre1
111       if(ki.eq.kedge1(k)) go to 20
112 22   continue
113 c
114 c   follow phase line through rectangular region
115 c
116 23   fx01=fxold
117       fx01r=xold
118       fx01i=yold
119       fx00=fxnew
120       fx00r=xnew
121       fx00i=ynew
122       li = ki
123       lr = jlt
124       go to 43
125 c*****
126 c   search along bottom edge of rectangle for changes in the

```

```

127 c    sign of imag(f)
128 c
129 26   continue
130     if(kr.eq.jrt) then
131         kedge=3
132         go to 31
133     end if
134     kr = kr+1
135     call findfx(kr,ki,fxnew,xnew,ynew)
136     if (yold*ynew .gt. 0.d0) go to 20
137     if(nre2.eq.0) go to 28
138 c
139 c    check if crossing point has been previously found
140 c
141     do 27 k=1,nre2
142     if(kr.eq.kedge2(k)) go to 20
143 27   continue
144 c
145 c    follow phase line through rectangular region
146 c
147 28   fx00=fxold
148     fx00r=xold
149     fx00i=yold
150     fx10=fxnew
151     fx10r=xnew
152     fx10i=ynew
153     li = jbot
154     lr = kr-1
155     go to 48
156 c*****
157 c    search along right edge of rectangle for sign changes in imag(f).
158 c
159 31   continue
160     if(ki.eq.jtop) then
161         kedge=4
162         go to 36
163     end if
164     ki = ki+1
165     call findfx(kr,ki,fxnew,xnew,ynew)
166     if (yold*ynew .gt. 0.d0) go to 20
167     if(nre3.eq.0) go to 33
168 c
169 c    check if crossing point has been previously found
170 c
171     do 32 k=1,nre3

```

```

172      if(ki.eq.kedge3(k)) go to 20
173 32   continue
174  c
175  c   follow phase line through rectangular region
176  c
177 33   fx10=fxold
178      fx10r=xold
179      fx10i=yold
180      fx11=fxnew
181      fx11r=xnew
182      fx11i=ynew
183      li = ki-1
184      lr = jrt-1
185      go to 53
186  c*****
187  c   search along top edge of rectangle for sign changes in imag(f).
188  c
189 36   continue
190      if(kr.eq.jlt) go to 80
191      kr = kr-1
192      call findfx(kr,ki,fxnew,xnew,ynew)
193      if (yold*ynew .gt. 0.d0) go to 20
194      if(nre4.eq.0) go to 38
195  c
196  c   check if crossing point has been previously found
197  c
198      do 37 k=1,nre4
199      if(kr.eq.kedge4(k)) go to 20
200 37   continue
201  c
202  c   follow phase line through rectangular region
203  c
204 38   fx11=fxold
205      fx11r=xold
206      fx11i=yold
207      fx01=fxnew
208      fx01r=xnew
209      fx01i=ynew
210      li = jtop-1
211      lr = kr
212      go to 58
213  c*****
214  c   enter mesh square from left side or exit rectangle at right edge.
215
216 41   lr=lr+1

```

```

217     if (lr .le. jrt-1) go to 42
218     nre3=nre3+1
219     kedge3(nre3)=li+1
220     go to 10
221 42   fx01=fx11
222     fx01r=fx11r
223     fx01i=fx11i
224     fx00=fx10
225     fx00r=fx10r
226     fx00i=fx10i
227 43   continue
228     call findfx(lr+1,li+1,fx11,fx11r,fx11i)
229     call findfx(lr+1,li,fx10,fx10r,fx10i)
230 c*****
231 c   Determine the edge of exit of im(f)=0 from current mesh.
232     edgeit=fx01i*fx11i
233     edgeib=fx00i*fx10i
234     if (edgeib .gt. 0.d0) then
235 c       Im(f)=0 goes through the 01 to 10 line.
236         if (edgeit .gt. 0.d0) then
237 c           Im(f)=0 goes through the 10 to 11 edge (edge 1).
238             lout=1
239         else
240 c           Im(f)=0 goes through the 01 to 11 edge (edge 2)
241             lout=2
242         end if
243     else
244 c       Im(f)=0 goes through the 00 to 10 edge (edge 4)
245         lout=4
246         if (edgeit .lt. 0.d0) then
247 c           Im(f)=0 also runs through 01 to 11 and 10 to 11 edges.
248 c           Store crossing location and in/out information.
249             knot34=knot34+1
250 c           loc34r(knot34)=lr
251 c           loc34i(knot34)=li
252         end if
253     end if
254 c*****
255     go to 60
256 c*****
257 c   enter mesh square from bottom side or exit rectangle at top edge.
258 46   li=li+1
259     if (li .le. jtop-1) go to 47
260     nre4=nre4+1
261     kedge4(nre4)=lr

```

```

262      go to 10
263 47   fx00=fx01
264     fx00r=fx01r
265     fx00i=fx01i
266     fx10=fx11
267     fx10r=fx11r
268     fx10i=fx11i
269 48   continue
270     call findfx(lr,li+1,fx01,fx01r,fx01i)
271     call findfx(lr+1,li+1,fx11,fx11r,fx11i)
272 c*****
273 c     Determine the edge of exit of im(f)=0 from current mesh.
274     edgeil=fx00i*fx01i
275     edgeir=fx10i*fx11i
276     if (edgeir .gt. 0.d0) then
277 c       Im(f)=0 goes through the 00 to 11 line.
278       if (edgeil .gt. 0.d0) then
279 c         Im(f)=0 goes through the 01 to 11 edge (edge 2)
280         lout=2
281       else
282 c         Im(f)=0 goes through the 00 to 01 edge (edge 3).
283         lout=3
284       end if
285     else
286 c       Im(f)=0 goes through the 10 to 11 edge (edge 1)
287       lout=1
288       if (edgeil .lt. 0.d0) then
289 c         Im(f)=0 also runs through 00 to 01 and 01 to 11 edges.
290 c         Store crossing location and in/out information.
291         knot41=knot41+1
292 c         loc41r(knot41)=lr
293 c         loc41i(knot41)=li
294       end if
295     end if
296 c*****
297     go to 60
298 c*****
299 c     enter mesh square from right side or exit rectangle at left edge.
300
301 51   lr=lr-1
302     if (lr .ge. jlt) go to 52
303     nre1=nre1+1
304     kedge1(nre1)=li
305     go to 10
306 52   fx11=fx01

```

```

307     fx11r=fx01r
308     fx11i=fx01i
309     fx10=fx00
310     fx10r=fx00r
311     fx10i=fx00i
312 53  continue
313     call findfx(lr,li+1,fx01,fx01r,fx01i)
314     call findfx(lr,li,fx00,fx00r,fx00i)
315 c*****
316 c   Determine the edge of exit of im(f)=0 from current mesh.
317     edgeit=fx01i*fx11i
318     edgeib=fx00i*fx10i
319     if (edgeit .gt. 0.d0) then
320 c       Im(f)=0 goes through the 01 to 10 line.
321       if (edgeib .gt. 0.d0) then
322 c       Im(f)=0 goes through the 00 to 01 edge (edge 3).
323         lout=3
324       else
325 c       Im(f)=0 goes through the 00 to 10 edge (edge 4)
326         lout=4
327       end if
328     else
329 c       Im(f)=0 goes through the 01 to 11 edge (edge 2)
330       lout=2
331       if (edgeib .lt. 0.d0) then
332 c       Im(f)=0 also runs through 00 to 10 and 00 to 01 edges.
333 c       Store crossing location and in/out information.
334         knot12=knot12+1
335 c       loc12r(knot12)=lr
336 c       loc12i(knot12)=li
337       end if
338     end if
339 c*****
340     go to 60
341 c*****
342 c   enter mesh square from top side or exit rectangle at bottom edge.
343 56  li=li-1
344     if (li .ge. jbot) go to 57
345     nre2=nre2+1
346     kedge2(nre2)=lr+1
347     go to 10
348 57  fx01=fx00
349     fx01r=fx00r
350     fx01i=fx00i
351     fx11=fx10

```

```

352     fx11r=fx10r
353     fx11i=fx10i
354 58  continue
355     call findfx(lr,li,fx00,fx00r,fx00i)
356     call findfx(lr+1,li,fx10,fx10r,fx10i)
357 c*****
358 c   Determine the edge of exit of  $im(f)=0$  from current mesh.
359     edgeil=fx00i*fx01i
360     edgeir=fx10i*fx11i
361     if (edgeil .gt. 0.d0) then
362 c        $Im(f)=0$  goes through the 00 to 11 line.
363         if (edgeir .gt. 0.d0) then
364 c            $Im(f)=0$  goes through the 00 to 10 edge (edge 4)
365             lout=4
366         else
367 c            $Im(f)=0$  goes through the 10 to 11 edge (edge 1).
368             lout=1
369         end if
370     else
371 c        $Im(f)=0$  goes through the 00 to 01 edge (edge 3)
372         lout=3
373         if (edgeir .lt. 0.d0) then
374 c            $Im(f)=0$  also runs through 00 to 10 and 10 to 11 edges.
375 c           Store crossing location and in/out information.
376             knot23=knot23+1
377 c             loc23r(knot23)=lr
378 c             loc23i(knot23)=li
379         end if
380     end if
381 c
382 c*****
383 60  continue
384     nrsqu=nrsqu+1
385     if(nrsqu .gt. maxnsq) go to 95
386 c*****
387 c   Test for there being at least one  $re(f)=0$  line entering and
388 c       leaving the mesh square.
389 c
390     if ((fx00r*fx10r .gt. 0.d0) .and. (fx01r*fx11r .gt. 0.d0)
391 + .and. (fx00r*fx01r .gt. 0.d0)) go to (41,46,51,56) lout
392 c
393 c   Compute the values of the modal function at the corners of a
394 c     mesh square to determine its Taylor series to the 3rd order
395 c     for estimating its root locations.
396 c

```

```

397 c      f00=one
398      f10=cdexp(fx10-fx00)-one
399      f01=cdexp(fx01-fx00)-one
400      f11=cdexp(fx11-fx00)-one
401 c
402 c*****
403 c      write (16,3001) ni,nf,lr,li,knot12,knot23,knot34,knot41
404 c 3001 format('/ ni, nf, lr, li and knot12, 23, 34 and 43 before ROOTS
405 c + :/, 2i6,2x,2i6,2x,4i6)
406 c
407 c***** estimate locations of zeroes by radicals *****
408 c
409      call roots(f10,f01,f11,sol,nrsol)
410 c
411      do 63 n=1,nrsol
412          ureal = dreal(sol(n))
413          uimag = dimag(sol(n))
414          if (ureal .lt. 0.d0 .or. ureal .gt. 1.0d0) go to 63
415          if (uimag .lt. 0.d0 .or. uimag .gt. 1.0d0) go to 63
416 62      theta(1)=(lr+ureal)*tmesh
417          theta(2)=(li+uimag)*tmesh
418          nf = nf+1
419          zeros(nf)=dcmplx(theta(1),theta(2))
420          nrzl=nrzl+1
421 63      continue
422 c*****
423 c      write (16,3002) ni,nf,nrsol
424 c 3002 format('/ out of ROOTS at 63, ni, nf and # of roots ',3i4)
425 c*****
426 c      continue following the phase line
427      go to (41,46,51,56) lout
428 c*****
429 cc
430 80      continue
431 c
432      return
433 c*****
434 95      continue
435          write(16,9500)
436          write(16,4001)lr,li,ni,nf,tmesh
437          write(* ,9500)
438 4001 format('go to 5 from 95 at lr, li =',i6,',',i6,', ni, nf =',i6,
439          +',',i6,', mesh size =',d14.6)
440      go to 5
441 c*****

```

```

442 97 continue
443 write(16,9700)
444 write(16,4002)lr,li,ni,nf,tmesh
445 write(* ,9700)
446 4002 format('go to 5 from 97 at lr, li =',i6,',',i6,' ni, nf =',i6,
447 +',',i6,', mesh size =',d14.6,/'zeroes found are kept.')
```

```

448 c nf=ni
449 c
450 return
451 c
452 c**** format statements
453 9500 format(/5x,'too many squares on same phase line -- ',
454 $ 'reduce tmesh and start over')
455 9700 format(/5x,'tmesh has been reduced but problems remain in',
456 $ ' executing fzerox')
```

```

457 c
458 end
```

## APPENDIX C: SUBROUTINE ROOTS

This Appendix contains the listing of the subroutine ROOTS. This subroutine replaces the portion of the subroutine FZEROX where the coefficients of a quadratic equation are determined, and the subroutine QUAD for locating the zeroes of a quadratic polynomial. In the revised subroutine FZEROX, the roots of a cubic polynomial has to be found. This subroutine determines these zeroes by radicals.

```

1      subroutine roots (f1,f2,f3,sol,nrsol)
2  c*****
3  c This subroutine finds the roots of a third order polynomial by
4  c radicals when the values of this polynomial at z=0, z=1, z=i and
5  c z=1+i are given as f0=1, f1+f0, f2+f0 and f3+f0 respectively.
6  c Note that this algorithm takes cubic roots of two complex numbers
7  c (hence the name 'solution by radicals') and use their linear
8  c combinations as the roots of a third order polynomial.
9  c*****
10     implicit real*8 (a-h, o-z)
11     complex*16 f1,f2,f3,zero,one,ci,ep14,em14,ep23,em23,
12     +         fa,fb,fc,fd,fa1,fa2,fa3,fa1s,p,q,delt,z,zm,u,v,sol
13     parameter (xbit52=52.d0*0.69314718055994531d0,thrd=1.d0/3.d0,
14     +         bit50=1.d0/33554432.d0/33554432.d0,bit51=bit50/2.d0,
15     +         bit52=bit51/2.d0,tol=0.001d0,
16     +         zero=(0.d0,0.d0),one=(1.d0,0.d0),ci=(0.d0,1.d0),
17     +         ep14=(0.5d0,0.5d0),em14=(0.5d0,-0.5d0),
18     +         ep23=(-0.5d0,0.86602540378443864675d0),
19     +         em23=(-0.5d0,-0.86602540378443864675d0))
20     dimension sol(*)
21     fa=one
22     fb=(f2-ci*f1+em14*f3)
23     fc=((ep14+one)*f1-(em14+one)*f2+ci*f3)
24     fd=(em14*(f2-f1)-ep14*f3)
25     if (cdabs(fb) .le. bit50) fb=zero
26     if (cdabs(fc) .le. bit51) fc=zero
27     if (cdabs(fd) .le. bit52) fd=zero
28     if (fd .ne. zero) then
29         fa1=(-thrd)*fc/fd
30         fa2=fb/fd
31         fa3=fa/fd
32         fa1s=fa1*fa1
33         p=thrd*fa2-fa1s
34         q=0.5d0*(fa3+fa1*fa2)-fa1*fa1s
35         if (p .eq. zero) then
36             if (q .eq. zero) then
37                 nrsol=1
38                 sol(1)=fa1
39                 return
40             else
41                 nrsol=3
42                 u=((-2.d0)*q)**thrd
43                 sol(1)=u+fa1
44                 sol(2)=ep23*u+fa1
45                 sol(3)=em23*u+fa1
46                 return
47             end if
48     else

```

```

49         if (q. eq. zero) then
50             nrsol=3
51             sol(1)=fa1
52             u=cdsqrt((-3.d0)*p)
53             sol(2)=fa1+u
54             sol(3)=fa1-u
55             return
56         else
57             v=p/q
58             z=p*v*v
59             absz=cdabs(z)
60             if (absz .lt. tol) then
61                 zm=-z
62                 fn=dint(1.d0-xbit52/dlog(absz))
63                 lastn=idint(fn)-1
64                 dnn=fn-0.5d0
65                 dnd=fn+1.0d0
66                 delt=one
67                 do 100 nt=1,lastn
68                     dnn=dnn-1.d0
69                     dnd=dnd-1.d0
70                     delt=(dnn/dnd)*delt*zm+one
71 100             continue
72                 delt=(0.5d0*delt/q)**thrd
73                 u=p*delt
74                 v=-1.d0/delt
75             else
76                 delt=cdsqrt(one+z)-one
77                 u=(q*delt)**thrd
78                 v=-p/u
79             end if
80             nrsol=3
81             sol(1)=u+v+fa1
82             sol(2)=ep23*u+em23*v+fa1
83             sol(3)=em23*u+ep23*v+fa1
84             return
85         end if
86     end if
87     else if (fc .ne. zero) then
88         if (fb .eq. zero) then
89             if (fa .eq. zero) then
90                 nrsol=1
91                 sol(1)=zero
92                 return
93             else
94                 nrsol=2
95                 z=cdsqrt(-fa/fc)
96                 sol(1)=z

```

```

97         sol(2)=-z
98         return
99     end if
100    else
101        fa1=0.5d0*fb/fc
102        fa2=fa/fc
103        z=fa2/fa1/fa1
104        absz=cdabs(z)
105        if (absz .lt. tol) then
106            fn=dint(1.d0-xbit52/dlog(absz))
107            lastn=idint(fn)-1
108            dnn=fn-0.5d0
109            dnd=fn+1.0d0
110            delt=one
111            do 200 nt=1,lastn
112                dnn=dnn-1.d0
113                dnd=dnd-1.d0
114                delt=(dnn/dnd)*delt*z+one
115 200        continue
116            delt=-0.5d0*delt/fa1
117            nrsol=2
118            sol(1)=fa2*delt
119            sol(2)=1.d0/delt
120            return
121        else
122            delt=cdsqrt(one-z)
123            nrsol=2
124            sol(1)=-fa1*(one-delt)
125            sol(2)=-fa1*(one+delt)
126            return
127        end if
128    end if
129    else if (fb .ne. zero) then
130        nrsol=1
131        sol(1)=-fa/fb
132        return
133    else
134        nrsol=1
135        sol(1)=ep14
136        return
137    end if
138    end

```

## APPENDIX D: SUBROUTINE ABCOEF

This Appendix contains the listing of the subroutine ABCOEF. The consistency self-checking procedure has been implemented to determine the correct method to evaluate the  $A_i$  and  $B_i$  coefficients.

```

1      subroutine abcoef(zero,m)
2  c*****
3  c      For each mode m, this subroutine calculates A-B coefficients in
4  c      all layers for combining two linearly independent solutions of
5  c      Stokes' equation to form the height gain function:
6  c
7  c          height gain=exp(bcoefx(l,m))*(k1*exp(acoefx(l,m))+k2)
8  c
9  c      where k1 and k2 are two independent solutions to Stokes'
10 c      equation. In the top layer (i.e. nzlayr) the height gain is:
11 c
12 c          height gain=exp(bcoefx(l,m))*h2
13
14 c      where h2 is a solution to the Stokes' equation associated
15 c      with outgoing energy flow. Here k1 and k2 are proportional
16 c      to the k1 and k2 used by Marcus and the h2 is proportional
17 c      to a modified Hankle function of order 1/3.
18
19 c      inputs...
20 c          zero-an eigenvalue in q11 space
21
22 c      outputs...
23 c          acoefx-two dimensional array of complex exponents
24 c              coefficients used to combine two linearly
25 c              independent solutions of stokes' equation
26 c          bcoefx-two dimensional array of complex exponents
27 c              coefficients used for normalizing the height gains
28
29 c          note: acoefx and bcoefx are passed by the
30 c              common block /pap2/
31
32 c      subroutines called...
33 c          xcdai
34 c          xcadd
35
36 c      common block areas...
37 c          com1
38 c          com2
39 c          pap1
40 c          pap2
41 c*****
42
43      implicit real*8(a-h,o-z)
44      complex*16 acoefx,bcoefx,cqij,h2xq1,dh2xq1,h2xq2,dh2xq2,k1xq1,
45      $ dk1xq1,k1xq2,dk1xq2,k2xq1,dk2xq1,k2xq2,dk2xq2,h2dk1x,

```

```

46 $ dh2k1x,h2dk2x,dh2k2x,numax,denax,numbx,denbx,int1x,int2x,
47 $ hyx,dhyx,k1dhyx,dk1hyx,dk2hyx,k2dhyx,gamma,dgamdq,i,
48 $ koa123,rtsumx,zero,q1,q2,sumx,surfno,dqij,dqijdz,sqng,
49 $ dnumbx,dhux,dhlx,e13x,cneg,cldqzl,cldqzm,cigama,koawav,tthd,
50 + tacoef,dacoef
51
52 parameter(downi=1.d-3,downr=1.d-3/0.4342944819032518d0,
53 + pi=3.141592653589793238462643d0,
54 + i=(0.0d0,1.0d0),tthd=(2.d0/3.d0)*i,
55 + cneg=(0.0d0,3.141592653589793238462643d0),e13x=cneg/3.d0)
56 c*****
57 c mxlayr=maximum number of layers allowed
58 c mxmode=maximum number of modes allowed
59
60 c
61 c use include file for parameters of
62 c use include file for parameters of
63 c mxlayr max # layers
64 c mxmode max # modes
65 c
66 $include: 'mlaparm.inc'
    ***** Begin listing of: mlaparm.inc
    1 c
    2 c include file to define the
    3 c maximum # of layers (mxlayr)
    4 c maximum # of modes (mxmode)
    5 c
    6 parameter (mxlayr=35 )
    7 parameter (mxmode=127)
    ***** End listing of: mlaparm.inc
67 c
68 c
69 c*****
70 c acoefx-two dimensional complex array used for combining two
71 c independent solutions to stokes' equation
72 c bcoefx-two dimensional complex array used for normalizing height
73 c gain
74 c cqij-two dimensional array containing coefficients for evaluating
75 c qij in terms of q11
76 c dqij-array containing coefficients for evaluating qij in terms of
77 c q11
78 c dqijdz-array containing derivatives of qi(z) in the different
79 c layers
80 c zi-array containing input hesights for the modified refractivity
81

```

```

82     dimension acoefx(mxlayr,mxmode),
83     $         bcoefx(mxlayr,mxmode),
84     $         dqij(mxlayr),cqij(mxlayr,2),dqijdz(mxlayr),zi(mxlayr+1)
85 c*****
86
87     common /com1/freq,waveno,sqng
88     common /com2/cqij,dqij,dqijdz,nzlayr
89     common /pap1/nrmode,koa123,surfno,zi
90     common /pap2/acoefx,bcoefx
91
92 c*****
93 c     check for single layer
94 c
95 c     set a complex variable koawav=-i*koa123/(waveno*waveno) to
96 c     avoid repeating computations
97
98     koawav=-i*koa123/(waveno*waveno)
99
100    if(nzlayr .eq. 1)then
101        q1=cqij(1,1)+zero*dqij(1)
102        call surf(q1,gamma,dgamdq)
103        call xcdei(-q1,k2xq1,dk2xq1,k1xq1,dk1xq1,h2xq1,dh2xq1)
104        dh2xq1=dh2xq1+e13x
105        int1x=cdlog(koawav*dgamdq-q1/dqijdz(1))+2.0d0*h2xq1
106        int2x=2.0d0*dh2xq1-cdlog(-dqijdz(1))
107        call xcadd(sumx,int1x,int2x)
108        rtsumx=0.5d0*sumx
109        bcoefx(1,m)=-rtsumx
110        return
111    end if
112
113        cldqzl=cdlog(-dqijdz(1))
114
115 c     if l equals one then initialize cumulants and caculate a's and
116 c     b's in bottom layer using ground boundary conditions.
117
118        q1=cqij(1,1)+zero*dqij(1)
119        call xcdei(-q1,k2xq1,dk2xq1,k1xq1,dk1xq1,h2xq1,dh2xq1)
120        dk2xq1=dk2xq1+cneg
121        dk1xq1=dk1xq1-e13x
122        call surf(q1,gamma,dgamdq)
123        cigama=cdlog(i*gamma)
124        call xcadd(numax,cldqzl-cneg+dk2xq1,cigama+cneg+k2xq1)
125        call xcadd(denax,cigama+k1xq1,cldqzl+dk1xq1)
126        acoefx(1,m)=numax-denax

```

```

127      call xcadd(denbx,k2xq1,acoefx(1,m)+k1xq1)
128      bcoefx(1,m)=-denbx
129
130 c    calculate contributions to normalizing integrals.
131
132      call xcadd(hyx,k2xq1,acoefx(1,m)+k1xq1)
133      hyx=bcoefx(1,m)+hyx
134      call xcadd(dhyx,dk2xq1,acoefx(1,m)+dk1xq1)
135      dhyx=bcoefx(1,m)+dhyx
136      int1x=cdlog(koawav*dgamdq-q1/dqijdz(1))+2.0d0*hyx
137      int2x=2.0d0*dhyx-cldqzl
138      call xcadd(sumx,int1x,int2x)
139
140      do 9010 l=2,nzlayer-1
141          lm1=l-1
142          cldqzl=cdlog(-dqijdz(l))
143          cldqzm=cdlog(dqijdz(lm1))
144          q1=cqij(l,1)+zero*dqij(l)
145          call xcdei(-q1,k2xq1,dk2xq1,k1xq1,dk1xq1,h2xq1,dh2xq1)
146          dk2xq1=dk2xq1+cneg
147          dk1xq1=dk1xq1-e13x
148          q2=cqij(lm1,2)+zero*dqij(lm1)
149          call xcdei(-q2,k2xq2,dk2xq2,k1xq2,dk1xq2,h2xq2,dh2xq2)
150          dk2xq2=dk2xq2+cneg
151          dk1xq2=dk1xq2-e13x
152          call xcadd(hyx,k2xq2,acoefx(lm1,m)+k1xq2)
153          call xcadd(dhyx,dk2xq2,acoefx(lm1,m)+dk1xq2)
154          k1dhyx=k1xq1+dhyx
155          dk1hyx=dk1xq1+hyx
156          dk2hyx=dk2xq1+hyx
157          k2dhyx=k2xq1+dhyx
158          call xcadd(denax,cldqzm+k1dhyx,cldqzl+dk1hyx)
159          call xcadd(numax,cldqzl-cneg+dk2hyx,cldqzm+cneg+k2dhyx)
160          acoefx(l,m)=numax-denax
161          call xcadd(denbx,k2xq1,acoefx(l,m)+k1xq1)
162          numbx=bcoefx(lm1,m)+hyx
163          dnumbx=bcoefx(lm1,m)+dhyx
164          bcoefx(l,m)=numbx-denbx
165
166 c    calculate contribution to normalizing integrals.
167
168      int1x=cdlog(-q1/dqijdz(l)+q2/dqijdz(lm1))+2.0d0*numbx
169      call xcadd(sumx,sumx,int1x)
170      call xcadd(dhux,dk2xq1,acoefx(l,m)+dk1xq1)
171      dhux=bcoefx(l,m)+dhux

```

```

172         int1x=2.0d0*dnumbx-cldqzm
173         int2x=2.0d0*dhux-cldqzl
174         call xcadd(sumx,sumx,int1x)
175         call xcadd(sumx,sumx,int2x)
176 9010    continue
177
178
179 c      if l equals nzlayer, calculate a's and b's using outgoing
180 c      wave in top layer.
181
182         nzm1=nzlayr-1
183         q1=cqij(nzlayr,1)+zero*dqij(nzlayr)
184         call xcdai(-q1,k2xq1,dk2xq1,k1xq1,dk1xq1,h2xq1,dh2xq1)
185         dh2xq1=dh2xq1+e13x
186         q2=cqij(nzm1,2)+zero*dqij(nzm1)
187         call xcdai(-q2,k2xq2,dk2xq2,k1xq2,dk1xq2,h2xq2,dh2xq2)
188         dk2xq2=dk2xq2+cneg
189         dk1xq2=dk1xq2-e13x
190         call xcadd(hyx,k2xq2,acoefx(nzm1,m)+k1xq2)
191         numbx=bcoefx(nzlayr-1,m)+hyx
192         bcoefx(nzlayr,m)=numbx-h2xq1
193
194 c      calculate contribution to cumulants.
195
196         int1x=cdlog(-q1/dqijdz(nzlayr)+q2/dqijdz(nzm1))+
197 $      2.0d0*numbx
198         call xcadd(sumx,sumx,int1x)
199         call xcadd(dhyx,dk2xq2,acoefx(nzm1,m)+dk1xq2)
200         dnumbx=bcoefx(nzm1,m)+dhyx
201         int1x=2.0d0*dnumbx-cdlog(dqijdz(nzm1))
202         call xcadd(sumx,sumx,int1x)
203         dhux=bcoefx(nzlayr,m)+dh2xq1
204         int2x=2.0d0*dhux-cdlog(-dqijdz(nzlayr))
205         call xcadd(sumx,sumx,int2x)
206
207 c      renormalize b's so that height gain integral equals unity.
208
209         rtsumx=.5d0*sumx
210         do 9000 ll=1,nzlayr
211             bcoefx(ll,m)=bcoefx(ll,m)-rtsumx
212 9000    continue
213
214 c*****
215
216

```

```

217     l=nzlayr
218     lm1=l-1
219     cldqzm=cdlog(dqijdz(lm1))
220     cldqzl=cdlog(-dqijdz(l))
221
222 c     calculate q and associated quantities at bottom of layer l
223
224     q1=cqij(l,1)+zero*dqij(l)
225     call xcdai(-q1,k2xq1,dk2xq1,k1xq1,dk1xq1,h2xq1,dh2xq1)
226     dh2xq1=dh2xq1+e13x
227
228     q2=cqij(lm1,2)+zero*dqij(lm1)
229     call xcdai(-q2,k2xq2,dk2xq2,k1xq2,dk1xq2,h2xq2,dh2xq2)
230     dk2xq2=dk2xq2+cneg
231     dk1xq2=dk1xq2-e13x
232
233 c*****
234 c     Caculate acoefx(lm1,m),bcoefx(lm1,m)
235 c     and cumulants using outgoing wave in nzlayr
236 c*****
237     dh2k1x=dh2xq1+k1xq2
238     h2dk1x=h2xq1+dk1xq2
239     h2dk2x=h2xq1+dk2xq2
240     dh2k2x=dh2xq1+k2xq2
241
242     call xcadd(denax,cldqzl+cneg+dh2k1x,cldqzm+cneg+h2dk1x)
243     call xcadd(numax,cldqzm+h2dk2x,cldqzl+dh2k2x)
244
245 c     If in the nzlayr-1 layer the magnitudes of A coefficients from
246 c     integration up and down differ by less than 0.02 dB and their
247 c     phases differ by less than 0.001pi, the A and B coefficients
248 c     obtained from integration up will be accepted.
249
250     tcoef=numax-denax
251     dcoef=tcoef-acoefx(lm1,m)
252     difr=dabs(dreal(dcoef))
253     if (difr .lt. downr) then
254         difi=dimag(dcoef)/pi
255         difi=dabs(difi-dnint(difi/2.d0)*2.d0)
256         if (difi .lt. downi) return
257     end if
258
259     acoefx(lm1,m)=tcoef
260     call xcadd(denbx,k2xq2,acoefx(lm1,m)+k1xq2)
261     bcoefx(lm1,m)=h2xq1-denbx

```

```

262
263 c      calculate contributions to cumulants
264
265         sumx=cdlog(-q1/dqijdz(l)+q2/dqijdz(lm1))+2.0d0*h2xq1
266         call xcadd(dhlx,dk2xq2,acoeffx(lm1,m)+dk1xq2)
267         dhlx=bcoefx(lm1,m)+dhlx
268         int1x=2.0d0*dh2xq1-cldqzl
269         call xcadd(int1x,sumx,int1x)
270         int2x=2.0d0*dhlx-cldqzm
271         call xcadd(sumx,int1x,int2x)
272
273         do 9030 l=nzlayr-1,2,-1
274             lm1=l-1
275             cldqzl=cdlog(-dqijdz(l))
276             cldqzm=cdlog(dqijdz(lm1))
277
278 c      calculate q and associated quantities at bottom of layer l
279
280         q1=cqij(l,1)+zero*dqij(l)
281         call xcdei(-q1,k2xq1,dk2xq1,k1xq1,dk1xq1,h2xq1,dh2xq1)
282         dk2xq1=dk2xq1+cneg
283         dk1xq1=dk1xq1-e13x
284
285         q2=cqij(lm1,2)+zero*dqij(lm1)
286         call xcdei(-q2,k2xq2,dk2xq2,k1xq2,dk1xq2,h2xq2,dh2xq2)
287         dk2xq2=dk2xq2+cneg
288         dk1xq2=dk1xq2-e13x
289         dh2xq2=dh2xq2+e13x
290
291 c*****
292 c      Calculate acoefx(lm1,m),bcoefx(lm1,m) and cumulants
293 c      using continuity relations in terms of the linearly
294 c      independent functions k1 and k2
295
296         call xcadd(hyx,k2xq1,acoeffx(l,m)+k1xq1)
297         call xcadd(dhyx,dk2xq1,acoeffx(l,m)+dk1xq1)
298         k1dhyx=k1xq2+dhyx
299         dk1dhyx=dk1xq2+hyx
300         dk2dhyx=dk2xq2+hyx
301         k2dhyx=k2xq2+dhyx
302
303         call xcadd(denax,cldqzl-cneg+k1dhyx,cldqzm+cneg+dk1dhyx)
304         call xcadd(numax,cldqzm+dk2dhyx,cldqzl+k2dhyx)
305         acoefx(lm1,m)=numax-denax
306         call xcadd(denbx,k2xq2,acoeffx(lm1,m)+k1xq2)

```

```

307      numbx=bcoefx(l,m)+hyx
308      dnumbx=bcoefx(l,m)+dhyx
309      bcoefx(lm1,m)=numbx-denbx
310
311 c    calculate contributions to cumulants.
312
313      int1x=cdlog(-q1/dqijdz(l)+q2/dqijdz(lm1))+2.0d0*numbx
314      call xcadd(sumx,sumx,int1x)
315      call xcadd(dhlx,dk2xq2,acoefx(lm1,m)+dk1xq2)
316      dhlx=bcoefx(lm1,m)+dhlx
317      int1x=2.0d0*dnumbx-cldqzl
318      int2x=2.0d0*dhlx-cldqzm
319      call xcadd(sumx,sumx,int1x)
320      call xcadd(sumx,sumx,int2x)
321
322 9030    continue
323
324 c*****
325 c    if l equal to one calculate ground
326 c    contribution to cumulants and renormalize bcoefx's
327
328      l=1
329      q1=cqij(l,1)+zero*dqij(l)
330      call xcdei(-q1,k2xq1,dk2xq1,k1xq1,dk1xq1,h2xq1,dh2xq1)
331      dk2xq1=dk2xq1+cneg
332      dk1xq1=dk1xq1-e13x
333
334      call xcadd(hyx,k2xq1,acoefx(l,m)+k1xq1)
335      call xcadd(dhyx,dk2xq1,acoefx(l,m)+dk1xq1)
336      call surf(q1,gamma,dgamdq)
337      numbx=bcoefx(l,m)+hyx
338      dnumbx=bcoefx(l,m)+dhyx
339      int1x=cdlog(koawav*dgamdq-q1/dqijdz(l))+2.0d0*numbx
340      int2x=2.0d0*dnumbx-cdlog(-dqijdz(1))
341      call xcadd(sumx,sumx,int1x)
342      call xcadd(sumx,sumx,int2x)
343
344 c    renormalize b's so that height gain integrals equal unity.
345
346      rtsumx=.5d0*sumx
347
348      do 9020 ll=1,nzlayer-1
349          bcoefx(ll,m)=bcoefx(ll,m)-rtsumx
350 9020    continue
351

```

```
352         bcoefx(nzlayr,m)=-rtsumx
353
354
355     return
356 end
```

## LIST OF REFERENCES

1. D.E. Kerr, *Propagation of Short Radio Waves*, Peregrinus Ltd, London, United Kingdom, 1987.
2. S.W. Marcus (1982), "A model to calculate EM fields in tropospheric duct environments at frequencies through SHF," *Radio Science* 17(5), 1108-1124.
3. V.I. Fock (1965), *Electromagnetic Diffraction and Propagation Problems*, 414 + ix pp., Pergamon Press, New York.
4. L.W. Yeoh (1990), "An analysis of M-Layer: a multilayer tropospheric propagation program," *Technical Report NPS-62-90-009*, Naval Postgraduate School, Monterey, California 93943.
5. D.G. Morfitt and C.H. Shellman, "MODESRCH, An improved computer program for obtaining ELF/VLF/LF mode constants in an earth-ionosphere waveguide," Interim Report 77T, Naval Ocean Systems Center, San Diego, CA 92152, October 1976.
6. Z. Schulten and D.G.M. Anderson, "An algorithm for the evaluation of the complex Airy functions", *Journal of Computational Physics*, Vol. 31, No. 60-75, 1979.

## INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center  
Cameron Station  
Alexandria, VA 22304-6145 1
2. Library, Code 52  
Naval Postgraduate School  
Monterey, CA 93943-5002 1
3. Department Chairman, Code EC  
Department of Electrical and Computer Engineering  
Naval Postgraduate School  
Monterey, CA 93943-5100 1
4. Professor Hung-Mou Lee, Code EC/Lh  
Department of Electrical and Computer Engineering  
Naval Postgraduate School  
Monterey, CA 93943-5100 1
5. Professor Lawrence J. Ziomek, Code EC/Zm  
Department of Electrical and Computer Engineering  
Naval Postgraduate School  
Monterey, CA 93943-5100 1
6. Han, Yin Yuan  
5Fl, No. 10, Lane 165, HSIN-LONG RD. SEC. 4,  
Taipei Taiwan, R.O.C. 2
7. Naval Academy Library  
Kao-Hsiung Tso-Ying P.O. 90175  
Taiwan, R.O.C. 1
8. Ting, Chueh  
Department of Electrical Engineering  
Chung Cheng Institute of Technology  
Tao-Yuan, Tai-Hsi  
Taiwan, R.O.C. 1







DUDLEY KNOX LIBRARY  
NAVAL POSTGRADUATE SCHOOL  
MONTEREY CA 93943-5101



GAYLORD S





3 2768 00019081 3