This manual and the associated NEC-2 manuals are the result of a concerted effort to produce a set of clean documents for the NEC-2 program. The original manuals were first released in January 1981 and are available from the US Government, but are of poor quality. Not the fault of any individual or group. It is just an illustration of the evolution of computer technology since the time of using typewriters for the production of manuals and documentation. The original manuals are also available online from several sources by doing a search for nec2part1.pdf, nec2part2.pdf, and nec2part3.pdf.

I have made an effort to correct as many errors as possible in the scanning of the documents and the OCR process itself. Any pointers to errors, either in the original documents or in my production of these documents is greatly appreciated. I will make the appropriate corrections and reproduce the new document as soon as possible and place them back on the web.

These documents were produced using LaTeX under the kubuntu 7.04 Linux operating system on a Compaq Presario AMD 64 system. I have redone all the graphics where possible to improve the quality of the documents. This work started in mid-June 2007 and most likely will continue for a very long time due to the immensity of the project.

I have chosen the fixed spacing typewriter font to reproduce the font used in the original documents.

The program listing is slightly different from the original as shown in Part II of the original manual, so it will take a long time to match the program line numbers with the correct lines. Just please patient as this work progresses.

Please note that all equations have been entered entirely by hand and are subject to extra scrutiny on the part of the reader.

Thanks.

Chuck Adams
Prescott, AZ
June, 2007
k7qo@commspeed.net

Preface

The Numerical Electromagnetics Code (NEC) has been developed at the Lawrence Livermore Laboratory, Livermore, California, under the sponsorship of the Naval Ocean Systems Center and the Air Force Weapons Laboratory. It is an advanced version of the Antenna Modeling Program (AMP) developed in the early 1970's by MBAssociates for the Naval Research Laboratory, Naval Ship Engineering Center, U.S. Army ECOM/Communications Systems, U.S. Army Strategic Communications Command, and Rome Air Development Center under Office of Naval Research Contract N00014-71-C-0187. The present version of NEC is the result of efforts by G. J. Berk and A. J. Poggio of Lawrence Livermore Laboratory.

The documentation for NEC consists of three volumes:

- Part I: NEC Program Description - Theory
- Part II: NEC Program Description - Code
- Part III: NEC User's Guide

The documentation has been prepared by using the AMP documents as foundations and by modifying those as needed. In some cases this led to minor changes in the original documents while in many cases major modifications were required.

Over the years many individuals have been contributors to AMP and NEC and are acknowledged here as follows:

R. W. Adams
J. N. Brittingham
G. J. Burke
F. J. Deadrick
K. K. Hazard
D. L. Knepp
D. L. Lager
R. J. Lytle
E. K. Miller
J. B. Morton
G. M. Pjerrou
A. J. Poggio
E. S. Selden

The support for the development of NEC-2 at the Lawrence Livermore Laboratory has been provided by the Naval Ocean Systems Center under MIPR-N0095376MP. Cognizant individuals under whom this project was carried out include: J. Rockway and J. Logan.

Previous development of NEC also included the support of the Air Force Weapons Laboratory (Project Order 76-090) and was monitored by J. Castillo and TSgt. H. Goodwin.

Work was performed under the auspices of the U. S. Department of Energy under contract No. W-7405-Eng-48. Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U. S. Department of Energy to the exclusion of others that may be suitable.

CONTENTS

LIST OF ILLUSTRATIONS

ABSTRACT

The Numerical Electromagnetics Code (NEC-2) is a computer code for analyzing the electromagnetic response of an arbitrary structure consisting of wires and surfaces in free space or over a ground plane. The analysis is accomplished by the numerical solution of integral equations for induced currents. The excitation may be an incident plane wave or a voltage source on a wire, while the output may include current and charge density, electric or magnetic field in the vicinity of the structure, and radiated fields. Hence, the code may be used for antenna analysis or scattering and EMP studies.

This document is Part II of a three-part report. It contains a detailed description of the Fortran coding, including the definitions of variables and constants, and a listing of the code. The other two documents cover the equations and numerical methods (Part I) and instructions for use of the code (Part III).

KEY WORDS FOR DD FORM 1473:
EM scattering
EMP
Wire Model
Method of Moments

Section I
INTRODUCTION

The Numerical Electromagnetics Code (NEC-2)[1] is a user-oriented computer code for the analysis of the electromagnetic response of antennas and other metal structures. It is built around the numerical solution of integral equations for the currents induced on the structure by sources or incident fields. This approach avoids many of the simplifying assumptions required by other solution methods and provides a highly accurate and versatile tool for electromagnetic analysis.

The code combines an integral equation for smooth surfaces with one specialized to wires to provide for convenient and accurate modeling of a wide range of structures. A model may include nonradiating networks and transmission lines connecting parts of the structure, perfect or imperfect conductors, and lumped-element loading. A structure may also be modeled over a ground plane that may be either a perfect or imperfect conductor.

The excitation may be either voltage sources on the structure or an incident plane wave of linear or elliptic polarization. The output may include induced currents and charges, near electric or magnetic fields, and radiated fields. Hence, the program is suited to either antenna analysis or scattering, and EMP studies.

This document is Vol. II of a three-part report on NEC. It contains a detailed description of the Fortran coding. Section II contains for each routine: (1) a statement of purpose, (2) a narrative description of the methodology, (3) definitions of variables and constants, and (4) a listing of the code. The remaining sections cover the common blocks, system library functions, array dimension limitations, and subroutine linkage.

The information in Vol. II will be of use mainly to persons attempting to modify the code or to use it on a computer system with which the delivered deck is not compatible.

Vol.I describes the equations and numerical methods used in NEC.

Vol. III contains instructions for using the code, including preparation of input data and interpretation of output.

Persons attempting to use NEC far the first time should start by reading Vol III. Vol. I will help the new user to understand the capabilities and limitations of NEC.

[1]NEC-2 will be abbreviated to NEC elsewhere in this volume.

SECTION II
CODE DESCRIPTION

In this section, each routine in NEC is described in detail. The main program is described first and is followed by the subroutines in alphabetical order. Far each routine, there is a brief statement of its purpose, a description of the code, an alphabetized listing and definition of important variables and constants, and a listing of the code. Variables that are in common blocks, and hence occur in several routines, are usually omitted from the lists for individual routines. They are defined in Section III under their common block labels.

Following line MA 495 in the main program, all quantities of length have been normalized to wavelength. Current is normalized to wavelength throughout the solution. This changes the appearance of many of the equations. In particular the wave number, $k = 2\pi/\lambda$, usually appears as 2π.

MAIN

PURPOSE

To handle input and output and to call the appropriate subroutines.

METHOD

The structure of MAIN is shown in the flow charts of Figures 1 and 2, where Figure 1 represents the first half of the code to about line MA 459.

Comment cards are read and printed after line MA 72 and subroutine DATACN is called at MA 90 to read and process structure data. If a Numerical Green's Function (NGF) file was read in DATAGN then subroutine FBNFG is called to determine whether file storage is needed for the matrix and to allocate core storage. When a NGF has not been read the mode of matrix storage cannot be determined until line MA 464 since it depends on whether a NFG file is to be written.

The box labeled 'Read data end' in Figure 1 refers to the READ statement at MA 139. Any of the types of data cards in Table 1 may be read at this point to set parameters or to request execution at the solution part of the code.

The integer variables IGO and IFLOW are keys to the operation of the code. IGO indicates the stage of completion of the solution as listed in Table 2. When a card requesting execution is read (NE, NH, RP, WG, or XQ) the solution part of the code (Figure 2) is entered at the point determined by IGO (see MA 385, MA 420, MA 429, and MA 457). After the current has been computed IGO is given the value five. If subsequent data cards change parameters, the value of IGO is reduced to the value in Table 1 to indicate the point beyond which the solution must be repeated. For example, when an EX card is read IGO is set equal to three if it was greater than three but is not changed if it was less than three. For cards that request execution "ex." is shown in Table 1.

IFLOW is used to indicate the type of the previous data card. When several cards of the same type can be used together (CP, LD, NT, '[‘L, and EX for voltage sources) a counter is incremented and data is added to arrays if the card is the same as the previous card as indicated by IFLOW. If the previous card was different the counter is initialized and previous data in the arrays is destroyed. IFLOW is also used to indicate what type of card requested the solution (NE, RP, etc.). Cards such as up may be stacked together but are not stored since they are acted upon as they are encountered.

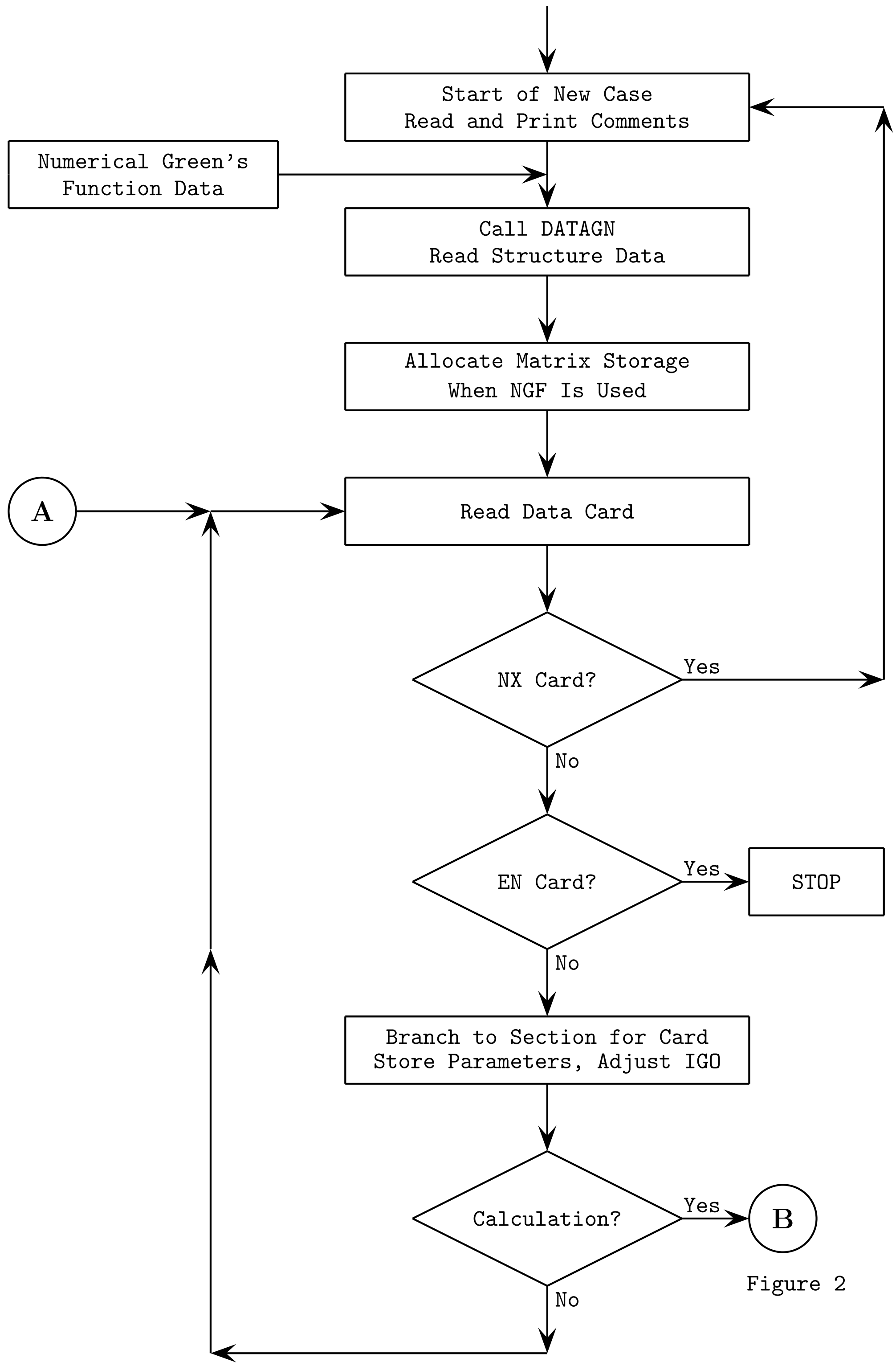


Figure 1. Flow Diagram af Main Program Input Section

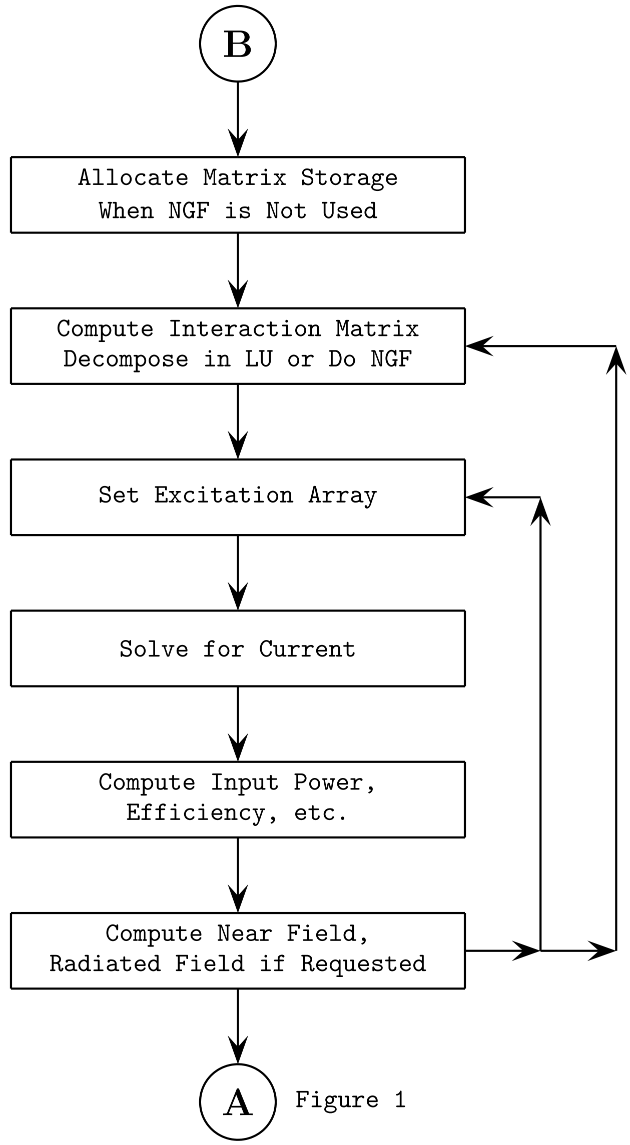


Figure 2. Flow Diagram of Main Program Computation Section

TABLE 1

	I	AIN(I)	GO TO	Line	IGO	IFLOW
1	21	CP	304	202	-	2
2	19	EK	320	194	2	1
3	13	EN	STOP	166	-	-
4	5	EX	24	275	3	5
5	2	FR	16	172	1	1
6	9	GD	34	369	-	9
7	4	GN	21	245	2	4
8	16	KH	305	187	2	1
9	3	LD	17	221	2	3
10	8	NE	32	370	ex.*	8
11	17	NH	208	368	ex.*	8
12	6	NT	28	321	3	6
13	12	NX	1	69	1	1
14	18	PQ	319	358	-	-
15	15	PT	31	348	-	-
16	10	RP	36	398	ex.	10
17	14	TL	28	321	3	6
18	20	WC	322	424	ex.	12
19	7	XQ	37	433	ex.	7 or 11

* NE and NH do not cause execution when multiple frequencies have been requested on the FR card. This allows computation of both near fields and radiated fie1ds in a frequency loop.

TABLE 2

IGO	Completion Point
1	Start
2	Frequency has been set and geometry scaled to wavelength
3	Interaction matrix filled and factored
4,5	Current computed and printed

The solution part of the code contains a loop over frequency starting at MA 463 and a loop over incident field direction starting at MA 562. FBLOCK is called at MA 465 to determine whether file storage is required for the matrix. From MA 466 to MA 493 the structure data are scaled from units of meters to wavelength or from one wavelength to the next when frequency is changed. Subroutine LOAD is called at MA 497 to fill array ZARRAY for the given frequency. At MA 520 the Sommerfeld interpolation tables are read from file TAPE21 if this option is used. NXA(l) is set to zero at MA 67 so the test ensures that the tape is read only once.

When the NGF option is not in use the matrix is filled by subroutine CMSET at MA 537 and factored by subroutine FACTRS at MA 540. When the NCF is used the equivalent steps are performed by CMNGF and FACGF. If a NGF tile is to be written, subroutine GFOUT is called at MA 557 to write TAPE20.

Subroutine ETMNS, called at MA 582, fills the excitation array and the current is computed in subroutine NETWORK called at MA 611. If transmission lines or two port networks are used NETWK combines the network equations with driving-point interaction equations derived from the primary interaction matrix. Otherwise the current is computed directly from the primary matrix.

The remainder of MAIN prints the currents and calls subroutines for near fields, radiated fields or coupling.

SYMBOL DICTIONARY

AIN	=	mnemonic from data card
ATST	=	array of possible data card mnemonics
CMAG	=	magnitude of the current in amperes
COM	=	array to store text from comment cards
CURI	=	current on segment I in amperes
CVEL	=	(velocity of light) (10^{-6}) in meters/second
DELFRQ	=	frequency increment (additive or multiplicative)
DPH	=	far-field Φ angle increment in degrees (input quantity)
DTH	=	far-field θ angle increment in degrees (input quantity)
DXNR	=	near-field observation point increments (input
DYNR	=	quantities with multiple meanings -- see ME card)
DZNR	=	
EPH	=	current component in direction $\hat{t}_2$ on patch
EPHA	=	phase angle of EPH
EPHM	=	magnitude of EFH
EFSC	=	complex dielectric constant of ground $\epsilon_c = \epsilon_r - j\sigma/\omega\epsilon_0$.
EPSCF	=	ϵ_c read from file TAPE21
EPSK	=	ϵ_r
EPSR2	=	ϵ_r for outer ground region
ETH	=	current component in direction $\hat{t}_1$ on patch
ETHA	=	phase angle of ETH
ETHM	=	magnitude of ETH
EX	=	$\hat{x}$ component of current an a patch
EXTIM	=	time at start of run (seconds)
EY	=	$\hat{y}$ component of current on a patch
EZ	=	$\hat{z}$ component of current on a patch

```
FJ      =  √-1
FMHZ    =  frequency in MHz
FMHZS   =  frequency in MHz
FNORM   =  multiply used array; stores impedances for printing of
           the normalized impedance or stores currents in the
           receiving pattern case for printing normalized
           receiving pattern
FR      =  (next frequency)/(present frequency)
FR2     =  (FR)(FR)
GNOR    =  if non-zero, equals gain normalization factor (dB) from RP card
HPOL    =  array containing polarization types (Hollerith)
IAVP    =  input integer flag used in average gain logic (RP card)
IAX     =  input integer flag specifying gain type (RP card)
IB11    =  location in array CM for start of storage of submatrix
           B when NCF is used
IC11    =  location in array CM for start of storage of submatrix
           C when NCF is used
ID11    =  location in CM for submatrix D
IEXK    =  flag to select the extended thin-wire kernel
IFAR    =  input integer flag specifying type of field
           calculation and type of ground system in
           far field (RP card)
IFLOW   =  integer flag, used to distinguish various input sections
IFRQ    =  input integer flag specifying type of frequency
           stepping (FR card)
IGO     =  integer to indicate stage of completion of the solution
INC     =  incident field loop index
INOR    =  input integer flag used for normalized gain request (RP card)
IFD     =  input integer flag selects gain type for normalization (RP card)
IPED    =  input integer flag used for impedance normalization request (EX card)
IPTAG   =  input integer for print central equal to segment tag number (PT card)
IPTAGF  =  input integer for print control specifying segment
           placement in a set of equal tags (PT card)
IPTAGT  =  same function as IPTAGF (input, PT card)
IPTFLG  =  input integer flag specifying type of print control (PT card)
IPTAQ
IPTAQF  =  same as above four variables but for PQ card
IPTAQT
IPTFLQ
IRESRV  =  length of array CM in complex numbers
IRNGF   =  storage in array CM that is reserved for later use
           when a NGF file is written
ISANT   =  array of segment numbers for voltage sources
ISAVE   =  segment number for normalized receiving pattern
           calculation
```

ISEG1(I)	=	segment numbers of end I and end 2 of the ith
ISEG2(I)		network connection
ITMP1 to ITMP5	=	temporary storage
IX	=	array for matrix pivot element information
IX11	=	location in GM of the start of an array in the NGF solution
IXTYP	=	excitation type from EX card
KCOM	=	number of comment cards read
LDTAG	=	tag number of loaded segment
LDTAGF	=	number of first loaded segment in set of segments
	=	having given tag
LDTAGT	=	last loaded segment
LDTYP	=	loading type
LOADMX	=	maximum number of loading cards
MASYM	=	flag to request matrix asymmetry calculation
MHZ	=	frequency loop index
MPCNT	=	counter for data cards
NCOUP	=	number of excitation points for coupling calculation
NCSBG	=	excitation segment for coupling calculation
NCTAC	=	excitation segment for coupling calculation
NEAR	=	increment option for near field points
NEQ	=	order of the primary interaction matrix
NEQ2	=	number of new unknowns in NGF mode
NETMX	=	maximum number of network data cards
NFEH	=	0 for near E field, 1 for near H
NFRQ	=	number of frequency steps
NONET	=	number of network data cards
NORMF	=	dimension of FNORM
NPHI	=	number of phi steps in incident field
NPHIC	=	loop index for phi in incident field
NPRINT	=	print control flag for subroutine NETWK
NKX NKY NRZ	=	number of steps in near field evaluation loops
NSANT	=	number of voltage sources
NSMAX	=	maximum number of voltage sources
NTHI	=	number of theta steps in incident field
NTHIC	=	loop index for theta in incident field
PH	=	phase angle of current or charge (degrees)
PHISS	=	initial Φ value for incident field
PIN	=	P_{in} = total power supplied to a structure by all voltage sources ($\sum$ Re(VI*)/2). For a Hertzian dipole source $P_{in} = \eta(\pi/3)\|Il/\lambda\|^2$.
PLOSS	=	power lost in distributed and point structure loads in watts
PNET	=	array contains Hollerith transmission line type
RFLD	=	if non-zero, equal to input far-field observation distance in meters
RKH	=	minimum separation for use of approximate interaction equations

SCRWLT	=	input length of radials in radial wire screen (GN Card) in meters
SCRWRT	=	radius at wires in radial wire ground screen in meters
SIG	=	conductivity of ground (σ in mhos/meter on GN card)
SIG2	=	conductivity of second medium in mhos/meter (GN and GD card)
TA	=	$\pi/180$
THETIS	=	initial θ for incident field
THETS	=	initial θ for radiated field
TIM	=	matrix computation time (seconds)
TMPl to TMP6	=	temporary input variables
XPR1 to XPR6	=	input quantities for incident field or Hertzian dipole illumination
ZLC		
ZLI	=	input quantities for loading
ZLR		
ZPNORM	=	impedance normalization quantity
l.E-20	=	used as small value test
1.745329252	=	π/180
2367.067	=	$2\pi\eta_0$
59.96	=	1/$(2\pi c\epsilon_0)$
299.8	=	c/10^6

```
C     PROGRAM NEC(INPUT,TAPE5=INPUT,OUTPUT,TAPE11,TAPE12,TAPE13,TAPE14,
C    1TAPE15,TAPE16,TAPE20,TAPE21)
C
C     NUMERICAL ELECTROMAGNETICS CODE (NEC2)  DEVELOPED AT LAWRENCE
C     LIVERMORE LAB., LIVERMORE, CA.  (CONTACT G. BURKE AT 415-422-8414
C     FOR PROBLEMS WITH THE NEC CODE.  FOR PROBLEMS WITH THE VAX IMPLEM-
C     ENTATION, CONTACT J. BREAKALL AT 415-422-8196 OR E. DOMNING AT 415
C     422-5936)
C     FILE CREATED 4/11/80.
C
C                ***********NOTICE**********
C     THIS COMPUTER CODE MATERIAL WAS PREPARED AS AN ACCOUNT OF WORK
C     SPONSORED BY THE UNITED STATES GOVERNMENT.  NEITHER THE UNITED
C     STATES NOR THE UNITED STATES DEPARTMENT OF ENERGY, NOR ANY OF
C     THEIR EMPLOYEES, NOR ANY OF THEIR CONTRACTORS, SUBCONTRACTORS, OR
C     THEIR EMPLOYEES, MAKES ANY WARRANTY, EXPRESS OR IMPLIED, OR
C     ASSUMES ANY LEGAL LIABILITY OR RESPONSIBILITY FOR THE ACCURACY,
C     COMPLETENESS OR USEFULNESS OF ANY INFORMATION, APPARATUS, PRODUCT
C     OR PROCESS DISCLOSED, OR REPRESENTS THAT ITS USE WOULD NOT
C     INFRINGE PRIVATELY-OWNED RIGHTS.
C
      CHARACTER  AIN*2, ATST*2, INFILE*80, OTFILE*80
      INTEGER*4 COM
      CHARACTER*6 HPOL,PNET
      COMPLEX CM,FJ,VSANT,ETH,EPH,ZRATI,CUR,CURI,ZARRAY,ZRATI2
      COMPLEX EX,EY,EZ,ZPED,VQD,VQDS,T1,Y11A,Y12A,EPSC,U,U2,XX1,XX2
      COMPLEX  AR1, AR2, AR3, EPSCF, FRATI
      COMMON/DATA/ LD,N1,N2,N,NP,M1,M2,M,MP,X(NM),Y(NM),
     * Z(NM),SI(NM),BI(NM),ALP(NM),BET(NM),ICON1(N2M),ICON2(
     * N2M), ITAG( N2M), ICONX( NM), WLAM, IPSYM
      COMMON/CMB/ CM(90000)
      COMMON/MATPAR/ ICASE, NBLOKS, NPBLK, NLAST, NBLSYM, NPSYM,
     *NLSYM, IMAT, ICASX, NBBX, NPBX, NLBX, NBBL, NPBL, NLBL
      COMMON/SAVE/ IP( N2M), KCOM, COM(20,5), EPSR, SIG, SCRWLT,
     *SCRWRT, FMHZ
      COMMON/CRNT/ AIR( NM), AII( NM), BIR( NM), BII( NM), CIR( NM),
     *CII( NM), CUR( N3M)
      COMMON/GND/ ZRATI, ZRATI2, FRATI, CL, CH, SCRWL, SCRWR, NRADL,
     *KSYMP, IFAR, IPERF, T1, T2
      COMMON/ZLOAD/ ZARRAY( NM), NLOAD, NLODF
      COMMON/YPARM/ NCOUP,ICOUP,NCTAG(5),NCSEG(5),Y11A(5),Y12A(20)
      COMMON/SEGJ/ AX(30), BX(30), CX(30), JCO(30), JSNO, ISCON(50),
     *NSCON, IPCON(10), NPCON
      COMMON/VSORC/ VQD(30), VSANT(30), VQDS(30), IVQD(30), ISANT(30)
     *, IQDS(30), NVQD, NSANT, NQDS
      COMMON/NETCX/ ZPED, PIN, PNLS, NEQ, NPEQ, NEQ2, NONET, NTSOL,
     *NPRINT, MASYM, ISEG1(150), ISEG2(150), X11R(150), X11I(150),
     *X12R(150), X12I(150), X22R(150), X22I(150), NTYP(150)
      COMMON/FPAT/ NTH, NPH, IPD, IAVP, INOR, IAX, THETS, PHIS, DTH,
```

```
     *DPH, RFLD, GNOR, CLT, CHT, EPSR2, SIG2, IXTYP, XPR6, PINR, PNLR,
     *PLOSS, NEAR, NFEH, NRX, NRY, NRZ, XNR, YNR, ZNR, DXNR, DYNR, DZNR
     *
      COMMON/GGRID/ AR1(11,10,4), AR2(17,5,4), AR3(9,8,4), EPSCF, DXA
     *(3), DYA(3), XSA(3), YSA(3), NXA(3), NYA(3)

      COMMON/GWAV/ U, U2, XX1, XX2, R1, R2, ZMH, ZPH

      COMMON /PLOT/ IPLP1, IPLP2, IPLP3, IPLP4
      DIMENSION  CAB(1), SAB(1), X2(1), Y2(1), Z2(1)
      DIMENSION  LDTYP(200), LDTAG(200), LDTAGF(200), LDTAGT(200),
     * ZLR(200), ZLI(200), ZLC(200)
      DIMENSION  ATST(22), PNET(6), HPOL(3), IX( N2M)
      DIMENSION  FNORM(200)

      DIMENSION  T1X(1), T1Y(1), T1Z(1), T2X(1), T2Y(1), T2Z(1)
      DIMENSION  XTEMP( NM), YTEMP( NM), ZTEMP( NM), SITEMP( NM),
     *BITEMP( NM)
      EQUIVALENCE(CAB,ALP),(SAB,BET),(X2,SI),(Y2,ALP),(Z2,BET)
      EQUIVALENCE(T1X,SI),(T1Y,ALP),(T1Z,BET),(T2X,ICON1),(T2Y,ICON2),(
     *T2Z,ITAG)

      DATA   ATST/'CE','FR','LD','GN','EX','NT','XQ','NE','GD','RP',
     * 'CM','NX','EN','TL','PT','KH','NH','PQ','EK','WG','CP','PL'/
      DATA HPOL/6HLINEAR,5HRIGHT,4HLEFT/
      DATA PNET/6H      ,2H  ,6HSTRAIG,2HHT,6HCROSSE,1HD/
      DATA TA/1.745329252D-02/, CVEL/299.8/
      DATA LOADMX, NSMAX, NETMX/200,150,150/, NORMF/200/

  706 CONTINUE
      PRINT 700
  700 FORMAT(' ENTER DATA INPUT FILENAME [HIT RETURN FOR TERMINAL',
     *' INPUT] : ',/,'      >')
  701 FORMAT(A)
      READ(*,701,ERR=702)  INFILE
      CALL STROPC( INFILE, INFILE)

      IF(INFILE.NE.' ') THEN
      OPEN ( UNIT=1,FILE=INFILE,STATUS='OLD',ERR=702)
      ENDIF
  707 CONTINUE
      PRINT 703
  703 FORMAT(' ENTER DATA OUTPUT FILENAME [HIT RETURN FOR TERMINAL',
     *' OUTPUT] : ',/,'      >')
      READ(*,701,ERR=704)  OTFILE
      CALL STROPC( OTFILE, OTFILE)

      IF(OTFILE.NE.' ') THEN
      OPEN(UNIT=2,FILE=OTFILE,STATUS='NEW',ERR=704)
```

```
      ENDIF
      GOTO 705
  702 PRINT *, 'ERROR ON TERMINAL INPUT'
      CALL ERROR
      GOTO 706
  704 CALL ERROR
      GOTO 707

  705 CONTINUE
      CALL SECONDS(EXTIM)
      FJ=(0.,1.)
      LD=600
      NXA(1)=0
      IRESRV=90000

    1 KCOM=0
      IFRTMW=0

      IFRTMP=0
    2 KCOM=KCOM+1
      IF(KCOM.GT.5) KCOM=5


      READ(1,125)  AIN,( COM( I, KCOM), I=1,19)

      CALL STROPC( AIN, AIN)

      IF(KCOM .LE. 0) THEN
          WRITE(2,126)
          WRITE(2,127)
          WRITE(2,128)
      ENDIF

      WRITE(2,129) ( COM( I, KCOM), I=1,19)

      IF(AIN.EQ. ATST(11)) GOTO 2

      IF(AIN .NE. ATST(1)) THEN
          WRITE(2,130)
          STOP
      ENDIF

      DO 5  I=1, LD
    5 ZARRAY( I)=(0.,0.)
      MPCNT=0

C     SET UP GEOMETRY DATA IN SUBROUTINE DATAGN

      IMAT=0
```

```
      CALL DATAGN
      IFLOW=1

C     CORE ALLOCATION FOR ARRAYS B, C, AND D FOR N.G.F. SOLUTION

      IF(IMAT.EQ.0) GOTO 326
      NEQ=N1+2* M1
      NEQ2=N- N1+2*( M- M1)+ NSCON+2* NPCON
      CALL FBNGF( NEQ, NEQ2, IRESRV, IB11, IC11, ID11, IX11)
      GOTO 6
  326 NEQ=N+2* M
      NEQ2=0
      IB11=1
      IC11=1
      ID11=1
      IX11=1
      ICASX=0
    6 NPEQ=NP+2* MP

C     DEFAULT VALUES FOR INPUT PARAMETERS AND FLAGS

      WRITE(2,135)
      IPLP1=0
      IPLP2=0
      IPLP3=0
      IPLP4=0
      IGO=1
      FMHZS=CVEL
      NFRQ=1
      RKH=1.
      IEXK=0
      IXTYP=0
      NLOAD=0
      NONET=0
      NEAR=-1
      IPTFLG=-2
      IPTFLQ=-1
      IFAR=-1
      ZRATI=(1.,0.)
      IPED=0
      IRNGF=0
      NCOUP=0
      ICOUP=0
      IF(ICASX.GT.0) GOTO 14
      FMHZ=CVEL
      NLODF=0
      KSYMP=1
      NRADL=0

```

```
C     MAIN INPUT SECTION - STANDARD READ STATEMENT - JUMPS TO APPRO-
C     PRIATE SECTION FOR SPECIFIC PARAMETER SET UP


      IPERF=0

   14 CALL READMN( AIN, ITMP1, ITMP2, ITMP3, ITMP4, TMP1, TMP2, TMP3,
     *TMP4, TMP5, TMP6)

      MPCNT=MPCNT+1

      WRITE(2,137)  MPCNT, AIN, ITMP1, ITMP2, ITMP3, ITMP4, TMP1, TMP2
     *, TMP3, TMP4, TMP5, TMP6

      IF(AIN.EQ. ATST(2)) GOTO 16
      IF(AIN.EQ. ATST(3)) GOTO 17
      IF(AIN.EQ. ATST(4)) GOTO 21
      IF(AIN.EQ. ATST(5)) GOTO 24
      IF(AIN.EQ. ATST(6)) GOTO 28
      IF(AIN.EQ. ATST(14)) GOTO 28
      IF(AIN.EQ. ATST(15)) GOTO 31
      IF(AIN.EQ. ATST(18)) GOTO 319
      IF(AIN.EQ. ATST(7)) GOTO 37
      IF(AIN.EQ. ATST(8)) GOTO 32
      IF(AIN.EQ. ATST(17)) GOTO 208
      IF(AIN.EQ. ATST(9)) GOTO 34
      IF(AIN.EQ. ATST(10)) GOTO 36
      IF(AIN.EQ. ATST(16)) GOTO 305
      IF(AIN.EQ. ATST(19)) GOTO 320
      IF(AIN.EQ. ATST(12)) GOTO 1
      IF(AIN.EQ. ATST(20)) GOTO 322

      IF(AIN.EQ. ATST(21)) GOTO 304

      IF(AIN.EQ. ATST(22)) GOTO 330
      IF(AIN.NE. ATST(13)) GOTO 15
      CALL SECONDS( TMP1)
      TMP1=TMP1- EXTIM
      WRITE(2,201)  TMP1
      STOP
   15 WRITE(2,138)

C     FREQUENCY PARAMETERS

      STOP
   16 IFRQ=ITMP1
      IF(ICASX.EQ.0) GOTO 8
      WRITE(2,303)  AIN
      STOP
```

```
    8 NFRQ=ITMP2                                                        MA 246
      IF(NFRQ.EQ.0) NFRQ=1                                              MA 247
      FMHZ=TMP1                                                         MA 248
      DELFRQ=TMP2                                                       MA 249
      IF(IPED.EQ.1) ZPNORM=0.                                           MA 250
      IGO=1                                                             MA 251
      IFLOW=1                                                           MA 252
                                                                        MA 253
C     MATRIX INTEGRATION LIMIT                                          MA 254
                                                                        MA 255
      GOTO 14                                                           MA 256
  305 RKH=TMP1                                                          MA 257
      IF(IGO.GT.2) IGO=2                                                MA 258
      IFLOW=1                                                           MA 259
                                                                        MA 260
C     EXTENDED THIN WIRE KERNEL OPTION                                  MA 261
                                                                        MA 262
      GOTO 14                                                           MA 263
  320 IEXK=1                                                            MA 264
      IF(ITMP1.EQ.-1) IEXK=0                                            MA 265
      IF(IGO.GT.2) IGO=2                                                MA 266
      IFLOW=1                                                           MA 267
                                                                        MA 268
C     MAXIMUM COUPLING BETWEEN ANTENNAS                                 MA 269
                                                                        MA 270
      GOTO 14                                                           MA 271
  304 IF(IFLOW.NE.2) NCOUP=0                                            MA 272
      ICOUP=0                                                           MA 273
      IFLOW=2                                                           MA 274
      IF(ITMP2.EQ.0) GOTO 14                                            MA 275
      NCOUP=NCOUP+1                                                     MA 276
      IF(NCOUP.GT.5) GOTO 312                                           MA 277
      NCTAG( NCOUP)=ITMP1                                               MA 278
      NCSEG( NCOUP)=ITMP2                                               MA 279
      IF(ITMP4.EQ.0) GOTO 14                                            MA 280
      NCOUP=NCOUP+1                                                     MA 281
      IF(NCOUP.GT.5) GOTO 312                                           MA 282
      NCTAG( NCOUP)=ITMP3                                               MA 283
      NCSEG( NCOUP)=ITMP4                                               MA 284
      GOTO 14                                                           MA 285
  312 WRITE(2,313)                                                      MA 286
C                                                                       MA 287
C     LOADING PARAMETERS                                                MA 288
C                                                                       MA 289
      STOP                                                              MA 290
   17 IF(IFLOW.EQ.3) GOTO 18                                            MA 291
      NLOAD=0                                                           MA 292
      IFLOW=3                                                           MA 293
      IF(IGO.GT.2) IGO=2                                                MA 294
```

```
      IF(ITMP1.EQ.(-1)) GOTO 14                                         MA 295
   18 NLOAD=NLOAD+1                                                     MA 296
      IF(NLOAD.LE. LOADMX) GOTO 19                                      MA 297
      WRITE(2,139)                                                      MA 298
      STOP                                                              MA 299
   19 LDTYP( NLOAD)=ITMP1                                               MA 300
      LDTAG( NLOAD)=ITMP2                                               MA 301
      IF(ITMP4.EQ.0) ITMP4=ITMP3                                        MA 302
      LDTAGF( NLOAD)=ITMP3                                              MA 303
      LDTAGT( NLOAD)=ITMP4                                              MA 304
      IF(ITMP4.GE. ITMP3) GOTO 20                                       MA 305
      WRITE(2,140)  NLOAD, ITMP3, ITMP4                                 MA 306
      STOP                                                              MA 307
   20 ZLR( NLOAD)=TMP1                                                  MA 308
      ZLI( NLOAD)=TMP2                                                  MA 309
      ZLC( NLOAD)=TMP3                                                  MA 310
C                                                                       MA 311
C     GROUND PARAMETERS UNDER THE ANTENNA                               MA 312
C                                                                       MA 313
      GOTO 14                                                           MA 314
   21 IFLOW=4                                                           MA 315
      IF(ICASX.EQ.0) GOTO 10                                            MA 316
      WRITE(2,303)  AIN                                                 MA 317
      STOP                                                              MA 318
   10 IF(IGO.GT.2) IGO=2                                                MA 319
      IF(ITMP1.NE.(-1)) GOTO 22                                         MA 320
      KSYMP=1                                                           MA 321
      NRADL=0                                                           MA 322
      IPERF=0                                                           MA 323
      GOTO 14                                                           MA 324
   22 IPERF=ITMP1                                                       MA 325
      NRADL=ITMP2                                                       MA 326
      KSYMP=2                                                           MA 327
      EPSR=TMP1                                                         MA 328
      SIG=TMP2                                                          MA 329
      IF(NRADL.EQ.0) GOTO 23                                            MA 330
      IF(IPERF.NE.2) GOTO 314                                           MA 331
      WRITE(2,390)                                                      MA 332
      STOP                                                              MA 333
  314 SCRWLT=TMP3                                                       MA 334
      SCRWRT=TMP4                                                       MA 335
      GOTO 14                                                           MA 336
   23 EPSR2=TMP3                                                        MA 337
      SIG2=TMP4                                                         MA 338
      CLT=TMP5                                                          MA 339
      CHT=TMP6                                                          MA 340
C                                                                       MA 341
C     EXCITATION PARAMETERS                                             MA 342
C                                                                       MA 343
```

```
      GOTO 14                                                           MA 344
   24 IF(IFLOW.EQ.5) GOTO 25                                            MA 345
      NSANT=0                                                           MA 346
      NVQD=0                                                            MA 347
      IPED=0                                                            MA 348
      IFLOW=5                                                           MA 349
      IF(IGO.GT.3) IGO=3                                                MA 350
   25 MASYM=ITMP4/10                                                    MA 351
      IF(ITMP1.GT.0.AND. ITMP1.NE.5) GOTO 27                            MA 352
      IXTYP=ITMP1                                                       MA 353
      NTSOL=0                                                           MA 354
      IF(IXTYP.EQ.0) GOTO 205                                           MA 355
      NVQD=NVQD+1                                                       MA 356
      IF(NVQD.GT. NSMAX) GOTO 206                                       MA 357
      IVQD( NVQD)=ISEGNO( ITMP2, ITMP3)                                 MA 358
      VQD( NVQD)=CMPLX( TMP1, TMP2)                                     MA 359
      IF(ABS( VQD( NVQD)).LT.1.D-20) VQD( NVQD)=(1.,0.)                 MA 360
      GOTO 207                                                          MA 361
  205 NSANT=NSANT+1                                                     MA 362
      IF(NSANT.LE. NSMAX) GOTO 26                                       MA 363
  206 WRITE(2,141)                                                      MA 364
      STOP                                                              MA 365
   26 ISANT( NSANT)=ISEGNO( ITMP2, ITMP3)                               MA 366
      VSANT( NSANT)=CMPLX( TMP1, TMP2)                                  MA 367
      IF(ABS( VSANT( NSANT)).LT.1.D-20) VSANT( NSANT)=(1.,0.)           MA 368
  207 IPED=ITMP4- MASYM*10                                              MA 369
      ZPNORM=TMP3                                                       MA 370
      IF(IPED.EQ.1.AND. ZPNORM.GT.0) IPED=2                             MA 371
      GOTO 14                                                           MA 372
   27 IF(IXTYP.EQ.0.OR. IXTYP.EQ.5) NTSOL=0                             MA 373
      IXTYP=ITMP1                                                       MA 374
      NTHI=ITMP2                                                        MA 375
      NPHI=ITMP3                                                        MA 376
      XPR1=TMP1                                                         MA 377
      XPR2=TMP2                                                         MA 378
      XPR3=TMP3                                                         MA 379
      XPR4=TMP4                                                         MA 380
      XPR5=TMP5                                                         MA 381
      XPR6=TMP6                                                         MA 382
      NSANT=0                                                           MA 383
      NVQD=0                                                            MA 384
      THETIS=XPR1                                                       MA 385
      PHISS=XPR2                                                        MA 386
C                                                                       MA 387
C     NETWORK PARAMETERS                                                MA 388
C                                                                       MA 389
      GOTO 14                                                           MA 390
   28 IF(IFLOW.EQ.6) GOTO 29                                            MA 391
      NONET=0                                                           MA 392
```

```
      NTSOL=0
      IFLOW=6
      IF(IGO.GT.3) IGO=3
      IF(ITMP2.EQ.(-1)) GOTO 14
   29 NONET=NONET+1
      IF(NONET.LE. NETMX) GOTO 30
      WRITE(2,142)
      STOP
   30 NTYP( NONET)=2
      IF(AIN.EQ. ATST(6)) NTYP( NONET)=1
      ISEG1( NONET)=ISEGNO( ITMP1, ITMP2)
      ISEG2( NONET)=ISEGNO( ITMP3, ITMP4)
      X11R( NONET)=TMP1
      X11I( NONET)=TMP2
      X12R( NONET)=TMP3
      X12I( NONET)=TMP4
      X22R( NONET)=TMP5
      X22I( NONET)=TMP6
      IF(NTYP( NONET).EQ.1.OR. TMP1.GT.0.) GOTO 14
      NTYP( NONET)=3


C     PLOT FLAGS

      X11R( NONET)=- TMP1
  330 IPLP1=ITMP1
      IPLP2=ITMP2
      IPLP3=ITMP3

      IPLP4=ITMP4
C
C     PRINT CONTROL FOR CURRENT
C
      GOTO 14
   31 IPTFLG=ITMP1
      IPTAG=ITMP2
      IPTAGF=ITMP3
      IPTAGT=ITMP4
      IF(ITMP3.EQ.0.AND. IPTFLG.NE.-1) IPTFLG=-2
      IF(ITMP4.EQ.0) IPTAGT=IPTAGF
C
C     WRITECONTROL FOR CHARGE
C
      GOTO 14
  319 IPTFLQ=ITMP1
      IPTAQ=ITMP2
      IPTAQF=ITMP3
      IPTAQT=ITMP4
      IF(ITMP3.EQ.0.AND. IPTFLQ.NE.-1) IPTFLQ=-2
```

```
      IF(ITMP4.EQ.0) IPTAQT=IPTAQF
C
C     NEAR FIELD CALCULATION PARAMETERS
C
      GOTO 14
  208 NFEH=1
      GOTO 209
   32 NFEH=0
  209 IF(.NOT.( IFLOW.EQ.8.AND. NFRQ.NE.1)) GOTO 33
      WRITE(2,143)
   33 NEAR=ITMP1
      NRX=ITMP2
      NRY=ITMP3
      NRZ=ITMP4
      XNR=TMP1
      YNR=TMP2
      ZNR=TMP3
      DXNR=TMP4
      DYNR=TMP5
      DZNR=TMP6
      IFLOW=8
      IF(NFRQ.NE.1) GOTO 14
C
C     GROUND REPRESENTATION
C
      GOTO (41,46,53,71,72), IGO
   34 EPSR2=TMP1
      SIG2=TMP2
      CLT=TMP3
      CHT=TMP4
      IFLOW=9
C
C     STANDARD OBSERVATION ANGLE PARAMETERS
C
      GOTO 14
   36 IFAR=ITMP1
      NTH=ITMP2
      NPH=ITMP3
      IF(NTH.EQ.0) NTH=1
      IF(NPH.EQ.0) NPH=1
      IPD=ITMP4/10
      IAVP=ITMP4- IPD*10
      INOR=IPD/10
      IPD=IPD- INOR*10
      IAX=INOR/10
      INOR=INOR- IAX*10
      IF(IAX.NE.0) IAX=1
      IF(IPD.NE.0) IPD=1
      IF(NTH.LT.2.OR. NPH.LT.2) IAVP=0
```

```
      IF(IFAR.EQ.1) IAVP=0                                              MA 491
      THETS=TMP1                                                        MA 492
      PHIS=TMP2                                                         MA 493
      DTH=TMP3                                                          MA 494
      DPH=TMP4                                                          MA 495
      RFLD=TMP5                                                         MA 496
      GNOR=TMP6                                                         MA 497
      IFLOW=10                                                          MA 498
C                                                                       MA 499
C     WRITENUMERICAL GREEN'S FUNCTION TAPE                              MA 500
C                                                                       MA 501
      GOTO (41,46,53,71,78), IGO                                        MA 502
  322 IFLOW=12                                                          MA 503
      IF(ICASX.EQ.0) GOTO 301                                           MA 504
      WRITE(2,302)                                                      MA 505
      STOP                                                              MA 506
  301 IRNGF=IRESRV/2                                                    MA 507
C                                                                       MA 508
C     EXECUTE CARD  -  CALC. INCLUDING RADIATED FIELDS                  MA 509
C                                                                       MA 510
      GOTO (41,46,52,52,52), IGO                                        MA 511
   37 IF(IFLOW.EQ.10.AND. ITMP1.EQ.0) GOTO 14                           MA 512
      IF(NFRQ.EQ.1.AND. ITMP1.EQ.0.AND. IFLOW.GT.7) GOTO 14             MA 513
      IF(ITMP1.NE.0) GOTO 39                                            MA 514
      IF(IFLOW.GT.7) GOTO 38                                            MA 515
      IFLOW=7                                                           MA 516
      GOTO 40                                                           MA 517
   38 IFLOW=11                                                          MA 518
      GOTO 40                                                           MA 519
   39 IFAR=0                                                            MA 520
      RFLD=0.                                                           MA 521
      IPD=0                                                             MA 522
      IAVP=0                                                            MA 523
      INOR=0                                                            MA 524
      IAX=0                                                             MA 525
      NTH=91                                                            MA 526
      NPH=1                                                             MA 527
      THETS=0.                                                          MA 528
      PHIS=0.                                                           MA 529
      DTH=1.0                                                           MA 530
      DPH=0.                                                            MA 531
      IF(ITMP1.EQ.2) PHIS=90.                                           MA 532
      IF(ITMP1.NE.3) GOTO 40                                            MA 533
      NPH=2                                                             MA 534
      DPH=90.                                                           MA 535
C                                                                       MA 536
C     END OF THE MAIN INPUT SECTION                                     MA 537
C                                                                       MA 538
C     BEGINNING OF THE FREQUENCY DO LOOP                                MA 539
```

```
C                                                                       MA 540
   40 GOTO (41,46,53,71,78), IGO                                        MA 541
                                                                        MA 542
   41 MHZ=1                                                             MA 543
      IF(N.EQ.0.OR. IFRTMW.EQ.1) GOTO 406                               MA 544
      IFRTMW=1                                                          MA 545
      DO 445  I=1, N                                                    MA 546
      XTEMP( I)=X( I)                                                   MA 547
      YTEMP( I)=Y( I)                                                   MA 548
      ZTEMP( I)=Z( I)                                                   MA 549
      SITEMP( I)=SI( I)                                                 MA 550
      BITEMP( I)=BI( I)                                                 MA 551
  445 CONTINUE                                                          MA 552
  406 IF(M.EQ.0.OR. IFRTMP.EQ.1) GOTO 407                               MA 553
      IFRTMP=1                                                          MA 554
      J=LD+1                                                            MA 555
      DO 545  I=1, M                                                    MA 556
      J=J-1                                                             MA 557
      XTEMP( J)=X( J)                                                   MA 558
      YTEMP( J)=Y( J)                                                   MA 559
      ZTEMP( J)=Z( J)                                                   MA 560
      BITEMP( J)=BI( J)                                                 MA 561
  545 CONTINUE                                                          MA 562
  407 CONTINUE                                                          MA 563
                                                                        MA 564
C     CORE ALLOCATION FOR PRIMARY INTERACTON MATRIX.  (A)               MA 565
      FMHZ1=FMHZ                                                        MA 566
      IF(IMAT.EQ.0) CALL FBLOCK( NPEQ, NEQ, IRESRV, IRNGF, IPSYM)       MA 567
   42 IF(MHZ.EQ.1) GOTO 44                                              MA 568
C      FMHZ=FMHZ+DELFRQ                                                 MA 569
                                                                        MA 570
      IF(IFRQ.EQ.1) GOTO 43                                             MA 571
      FMHZ=FMHZ1+( MHZ-1)* DELFRQ                                       MA 572
      GOTO 44                                                           MA 573
   43 FMHZ=FMHZ* DELFRQ                                                 MA 574
                                                                        MA 575
   44 FR=FMHZ/ CVEL                                                     MA 576
      WLAM=CVEL/ FMHZ                                                   MA 577
      WRITE(2,145)  FMHZ, WLAM                                          MA 578
      WRITE(2,196)  RKH                                                 MA 579
C     FREQUENCY SCALING OF GEOMETRIC PARAMETERS                         MA 580
C***      FMHZS=FMHZ                                                    MA 581
      IF(IEXK.EQ.1) WRITE(2,321)                                        MA 582
      IF(N.EQ.0) GOTO 306                                               MA 583
                                                                        MA 584
      DO 45  I=1, N                                                     MA 585
      X( I)=XTEMP( I)* FR                                               MA 586
      Y( I)=YTEMP( I)* FR                                               MA 587
      Z( I)=ZTEMP( I)* FR                                               MA 588
```

```
      SI( I)=SITEMP( I)* FR
   45 BI( I)=BITEMP( I)* FR
  306 IF(M.EQ.0) GOTO 307
      FR2=FR* FR
      J=LD+1
      DO 245  I=1, M
      J=J-1
      X( J)=XTEMP( J)* FR
      Y( J)=YTEMP( J)* FR
      Z( J)=ZTEMP( J)* FR
  245 BI( J)=BITEMP( J)* FR2
C     STRUCTURE SEGMENT LOADING
  307 IGO=2
   46 WRITE(2,146)
      IF(NLOAD.NE.0) CALL LOAD( LDTYP, LDTAG, LDTAGF, LDTAGT, ZLR, ZLI
     *, ZLC)
      IF(NLOAD.EQ.0.AND. NLODF.EQ.0) WRITE(2,147)
C     GROUND PARAMETER
      IF(NLOAD.EQ.0.AND. NLODF.NE.0) WRITE(2,327)
      WRITE(2,148)
      IF(KSYMP.EQ.1) GOTO 49
      FRATI=(1.,0.)
      IF(IPERF.EQ.1) GOTO 48
      IF(SIG.LT.0.) SIG=- SIG/(59.96* WLAM)
      EPSC=CMPLX( EPSR,- SIG* WLAM*59.96)
      ZRATI=1./ SQRT( EPSC)
      U=ZRATI
      U2=U* U
      IF(NRADL.EQ.0) GOTO 47
      SCRWL=SCRWLT/ WLAM
      SCRWR=SCRWRT/ WLAM
      T1=FJ*2367.067D+0/ DFLOAT( NRADL)
      T2=SCRWR* DFLOAT( NRADL)
      WRITE(2,170)  NRADL, SCRWLT, SCRWRT
      WRITE(2,149)
   47 IF(IPERF.EQ.2) GOTO 328
      WRITE(2,391)
      GOTO 329
  328 IF(NXA(1).EQ.0) READ(21)  AR1, AR2, AR3, EPSCF, DXA, DYA, XSA,
     *YSA, NXA, NYA
      FRATI=( EPSC-1.)/( EPSC+1.)
      IF(ABS(( EPSCF- EPSC)/ EPSC).LT.1.D-3) GOTO 400
      WRITE(2,393)  EPSCF, EPSC
      STOP
  400 WRITE(2,392)
  329 WRITE(2,150)  EPSR, SIG, EPSC
```

```
      GOTO 50
   48 WRITE(2,151)
      GOTO 50
   49 WRITE(2,152)
C * * *
C     FILL AND FACTOR PRIMARY INTERACTION MATRIX
C
   50 CONTINUE
      CALL SECONDS( TIM1)
      IF(ICASX.NE.0) GOTO 324
      CALL CMSET( NEQ, CM, RKH, IEXK)
      CALL SECONDS( TIM2)
      TIM=TIM2- TIM1
      CALL FACTRS( NPEQ, NEQ, CM, IP, IX,11,12,13,14)
C
C     N.G.F. - FILL B, C, AND D AND FACTOR D-C(INV(A)B)
C
C ****
      GOTO 323
C ****
  324 IF(NEQ2.EQ.0) GOTO 333
      CALL CMNGF( CM( IB11), CM( IC11), CM( ID11), NPBX, NEQ, NEQ2, RKH
     *, IEXK)
      CALL SECONDS( TIM2)
      TIM=TIM2- TIM1
      CALL FACGF( CM, CM( IB11), CM( IC11), CM( ID11), CM( IX11), IP,
     *IX, NP, N1, MP, M1, NEQ, NEQ2)
  323 CALL SECONDS( TIM1)
      TIM2=TIM1- TIM2
      WRITE(2,153)  TIM, TIM2
  333 IGO=3
      NTSOL=0
C     WRITEN.G.F. FILE
      IF(IFLOW.NE.12) GOTO 53
   52 CALL GFOUT
C
C     EXCITATION SET UP (RIGHT HAND SIDE, -E INC.)
C
      GOTO 14
   53 NTHIC=1
      NPHIC=1
      INC=1
      NPRINT=0
   54 IF(IXTYP.EQ.0.OR. IXTYP.EQ.5) GOTO 56
      IF(IPTFLG.LE.0.OR. IXTYP.EQ.4) WRITE(2,154)
      TMP5=TA* XPR5
      TMP4=TA* XPR4
      IF(IXTYP.NE.4) GOTO 55
      TMP1=XPR1/ WLAM
```

```
      TMP2=XPR2/ WLAM
      TMP3=XPR3/ WLAM
      TMP6=XPR6/( WLAM* WLAM)
      WRITE(2,156)  XPR1, XPR2, XPR3, XPR4, XPR5, XPR6
      GOTO 56
   55 TMP1=TA* XPR1
      TMP2=TA* XPR2
      TMP3=TA* XPR3
      TMP6=XPR6
      IF(IPTFLG.LE.0) WRITE(2,155)  XPR1, XPR2, XPR3, HPOL( IXTYP),
     *XPR6
C
C     MATRIX SOLVING  (NETWK CALLS SOLVES)
C
   56 CALL ETMNS( TMP1, TMP2, TMP3, TMP4, TMP5, TMP6, IXTYP, CUR)
      IF(NONET.EQ.0.OR. INC.GT.1) GOTO 60
      WRITE(2,158)
      ITMP3=0
      ITMP1=NTYP(1)
      DO 59  I=1,2
      IF(ITMP1.EQ.3) ITMP1=2
      IF(ITMP1.EQ.2) WRITE(2,159)
      IF(ITMP1.EQ.1) WRITE(2,160)
      DO 58  J=1, NONET
      ITMP2=NTYP( J)
      IF(( ITMP2/ ITMP1).EQ.1) GOTO 57
      ITMP3=ITMP2
      GOTO 58
   57 ITMP4=ISEG1( J)
      ITMP5=ISEG2( J)
      IF(ITMP2.GE.2.AND. X11I( J).LE.0.) X11I( J)=WLAM* SQRT(( X(
     *ITMP5)- X( ITMP4))**2+( Y( ITMP5)- Y( ITMP4))**2+( Z( ITMP5)- Z(
     *ITMP4))**2)
      WRITE(2,157)  ITAG( ITMP4), ITMP4, ITAG( ITMP5), ITMP5, X11R( J)
     *, X11I( J), X12R( J), X12I( J), X22R( J), X22I( J), PNET(2* ITMP2
     *-1), PNET(2* ITMP2)
   58 CONTINUE
      IF(ITMP3.EQ.0) GOTO 60
      ITMP1=ITMP3
   59 CONTINUE
   60 CONTINUE
      IF(INC.GT.1.AND. IPTFLG.GT.0) NPRINT=1
      CALL NETWK( CM, CM( IB11), CM( IC11), CM( ID11), IP, CUR)
      NTSOL=1
      IF(IPED.EQ.0) GOTO 61
      ITMP1=MHZ+4*( MHZ-1)
      IF(ITMP1.GT.( NORMF-3)) GOTO 61
      FNORM( ITMP1)=REAL( ZPED)
      FNORM( ITMP1+1)=AIMAG( ZPED)
```

```
      FNORM( ITMP1+2)=ABS( ZPED)
      FNORM( ITMP1+3)=CANG( ZPED)
      IF(IPED.EQ.2) GOTO 61
      IF(FNORM( ITMP1+2).GT. ZPNORM) ZPNORM=FNORM( ITMP1+2)
C
C     PRINTING STRUCTURE CURRENTS
C
   61 CONTINUE
      IF(N.EQ.0) GOTO 308
      IF(IPTFLG.EQ.(-1)) GOTO 63
      IF(IPTFLG.GT.0) GOTO 62
      WRITE(2,161)
      WRITE(2,162)
      GOTO 63
   62 IF(IPTFLG.EQ.3.OR. INC.GT.1) GOTO 63
      WRITE(2,163)  XPR3, HPOL( IXTYP), XPR6
   63 PLOSS=0.
      ITMP1=0
      JUMP=IPTFLG+1
      DO 69  I=1, N
      CURI=CUR( I)* WLAM
      CMAG=ABS( CURI)
      PH=CANG( CURI)
      IF(NLOAD.EQ.0.AND. NLODF.EQ.0) GOTO 64
      IF(ABS( REAL( ZARRAY( I))).LT.1.D-20) GOTO 64
      PLOSS=PLOSS+.5* CMAG* CMAG* REAL( ZARRAY( I))* SI( I)
   64 IF(JUMP) 68,69,65
   65 IF(IPTAG.EQ.0) GOTO 66
      IF(ITAG( I).NE. IPTAG) GOTO 69
   66 ITMP1=ITMP1+1
      IF(ITMP1.LT. IPTAGF.OR. ITMP1.GT. IPTAGT) GOTO 69
      IF(IPTFLG.EQ.0) GOTO 68
      IF(IPTFLG.LT.2.OR. INC.GT. NORMF) GOTO 67
      FNORM( INC)=CMAG
      ISAVE=I
   67 IF(IPTFLG.NE.3) WRITE(2,164)  XPR1, XPR2, CMAG, PH, I
      GOTO 69

   68 WRITE(2,165)  I, ITAG( I), X( I), Y( I), Z( I), SI( I), CURI,
     *CMAG, PH
      IF(IPLP1.NE.1) GOTO 69
      IF(IPLP2.EQ.1) WRITE( 8,*)  CURI

      IF(IPLP2.EQ.2) WRITE( 8,*)  CMAG, PH
   69 CONTINUE
      IF(IPTFLQ.EQ.(-1)) GOTO 308
      WRITE(2,315)
      ITMP1=0
      FR=1.D-6/ FMHZ
```

```
      DO 316  I=1, N
      IF(IPTFLQ.EQ.(-2)) GOTO 318
      IF(IPTAQ.EQ.0) GOTO 317
      IF(ITAG( I).NE. IPTAQ) GOTO 316
  317 ITMP1=ITMP1+1
      IF(ITMP1.LT. IPTAQF.OR. ITMP1.GT. IPTAQT) GOTO 316
  318 CURI=FR* CMPLX(- BII( I), BIR( I))
      CMAG=ABS( CURI)
      PH=CANG( CURI)
      WRITE(2,165)  I, ITAG( I), X( I), Y( I), Z( I), SI( I), CURI,
     *CMAG, PH
  316 CONTINUE
  308 IF(M.EQ.0) GOTO 310
      WRITE(2,197)
      J=N-2
      ITMP1=LD+1
      DO 309  I=1, M
      J=J+3
      ITMP1=ITMP1-1
      EX=CUR( J)
      EY=CUR( J+1)
      EZ=CUR( J+2)
      ETH=EX* T1X( ITMP1)+ EY* T1Y( ITMP1)+ EZ* T1Z( ITMP1)
      EPH=EX* T2X( ITMP1)+ EY* T2Y( ITMP1)+ EZ* T2Z( ITMP1)
      ETHM=ABS( ETH)
      ETHA=CANG( ETH)
      EPHM=ABS( EPH)
C309   WRITE(6,198) I,X(ITMP1),Y(ITMP1),Z(ITMP1),ETHM,ETHA,EPHM,EPHA,E
C     1X,EY, EZ

      EPHA=CANG( EPH)
      WRITE(2,198)  I, X( ITMP1), Y( ITMP1), Z( ITMP1), ETHM, ETHA,
     *EPHM, EPHA, EX, EY, EZ
      IF(IPLP1.NE.1) GOTO 309
      IF(IPLP3.EQ.1) WRITE( 8,*)  EX
      IF(IPLP3.EQ.2) WRITE( 8,*)  EY
      IF(IPLP3.EQ.3) WRITE( 8,*)  EZ
      IF(IPLP3.EQ.4) WRITE( 8,*)  EX, EY, EZ

  309 CONTINUE
  310 IF(IXTYP.NE.0.AND. IXTYP.NE.5) GOTO 70
      TMP1=PIN- PNLS- PLOSS
      TMP2=100.* TMP1/ PIN
      WRITE(2,166)  PIN, TMP1, PLOSS, PNLS, TMP2
   70 CONTINUE
      IGO=4
      IF(NCOUP.GT.0) CALL COUPLE( CUR, WLAM)
      IF(IFLOW.NE.7) GOTO 71
      IF(IXTYP.GT.0.AND. IXTYP.LT.4) GOTO 113
```

```
      IF(NFRQ.NE.1) GOTO 120
      WRITE(2,135)
      GOTO 14
C
C     NEAR FIELD CALCULATION
C
   71 IGO=5
   72 IF(NEAR.EQ.(-1)) GOTO 78
      CALL NFPAT
      IF(MHZ.EQ. NFRQ) NEAR=-1
      IF(NFRQ.NE.1) GOTO 78
      WRITE(2,135)
C
C     STANDARD FAR FIELD CALCULATION
C
      GOTO 14
   78 IF(IFAR.EQ.-1) GOTO 113
      PINR=PIN
      PNLR=PNLS
      CALL RDPAT
  113 IF(IXTYP.EQ.0.OR. IXTYP.GE.4) GOTO 119
      NTHIC=NTHIC+1
      INC=INC+1
      XPR1=XPR1+ XPR4
      IF(NTHIC.LE. NTHI) GOTO 54
      NTHIC=1
      XPR1=THETIS
      XPR2=XPR2+ XPR5
      NPHIC=NPHIC+1
      IF(NPHIC.LE. NPHI) GOTO 54
      NPHIC=1
      XPR2=PHISS
C     NORMALIZED RECEIVING PATTERN PRINTED
      IF(IPTFLG.LT.2) GOTO 119
      ITMP1=NTHI* NPHI
      IF(ITMP1.LE. NORMF) GOTO 114
      ITMP1=NORMF
      WRITE(2,181)
  114 TMP1=FNORM(1)
      DO 115  J=2, ITMP1
      IF(FNORM( J).GT. TMP1) TMP1=FNORM( J)
  115 CONTINUE
      WRITE(2,182)  TMP1, XPR3, HPOL( IXTYP), XPR6, ISAVE
      DO 118  J=1, NPHI
      ITMP2=NTHI*( J-1)
      DO 116  I=1, NTHI
      ITMP3=I+ ITMP2
      IF(ITMP3.GT. ITMP1) GOTO 117
      TMP2=FNORM( ITMP3)/ TMP1
```

```
      TMP3=DB20( TMP2)                                                  MA 883
      WRITE(2,183)  XPR1, XPR2, TMP3, TMP2                              MA 884
      XPR1=XPR1+ XPR4                                                   MA 885
  116 CONTINUE                                                          MA 886
  117 XPR1=THETIS                                                       MA 887
      XPR2=XPR2+ XPR5                                                   MA 888
  118 CONTINUE                                                          MA 889
      XPR2=PHISS                                                        MA 890
  119 IF(MHZ.EQ. NFRQ) IFAR=-1                                          MA 891
      IF(NFRQ.NE.1) GOTO 120                                            MA 892
      WRITE(2,135)                                                      MA 893
      GOTO 14                                                           MA 894
  120 MHZ=MHZ+1                                                         MA 895
      IF(MHZ.LE. NFRQ) GOTO 42                                          MA 896
      IF(IPED.EQ.0) GOTO 123                                            MA 897
      IF(NVQD.LT.1) GOTO 199                                            MA 898
      WRITE(2,184)  IVQD( NVQD), ZPNORM                                 MA 899
      GOTO 204                                                          MA 900
  199 WRITE(2,184)  ISANT( NSANT), ZPNORM                               MA 901
  204 ITMP1=NFRQ                                                        MA 902
      IF(ITMP1.LE.( NORMF/4)) GOTO 121                                  MA 903
      ITMP1=NORMF/4                                                     MA 904
      WRITE(2,185)                                                      MA 905
  121 IF(IFRQ.EQ.0) TMP1=FMHZ-( NFRQ-1)* DELFRQ                         MA 906
      IF(IFRQ.EQ.1) TMP1=FMHZ/( DELFRQ**( NFRQ-1))                      MA 907
      DO 122  I=1, ITMP1                                                MA 908
      ITMP2=I+4*( I-1)                                                  MA 909
      TMP2=FNORM( ITMP2)/ ZPNORM                                        MA 910
      TMP3=FNORM( ITMP2+1)/ ZPNORM                                      MA 911
      TMP4=FNORM( ITMP2+2)/ ZPNORM                                      MA 912
      TMP5=FNORM( ITMP2+3)                                              MA 913
      WRITE(2,186)  TMP1, FNORM( ITMP2), FNORM( ITMP2+1), FNORM( ITMP2  MA 914
     *+2), FNORM( ITMP2+3), TMP2, TMP3, TMP4, TMP5                      MA 915
      IF(IFRQ.EQ.0) TMP1= TMP1+ DELFRQ                                  MA 916
      IF(IFRQ.EQ.1) TMP1= TMP1* DELFRQ                                  MA 917
  122 CONTINUE                                                          MA 918
      WRITE(2,135)                                                      MA 919
  123 CONTINUE                                                          MA 920
      NFRQ=1                                                            MA 921
      MHZ=1                                                             MA 922
      GOTO 14                                                           MA 923
  125 FORMAT(A2,19A4)                                                   MA 924
  126 FORMAT('1')                                                       MA 925
  127 FORMAT(///,33X,'******************************************',//,36X,  MA 926
     *'NUMERICAL ELECTROMAGNETICS CODE',//,33X,                         MA 927
     *'******************************************')                     MA 928
  128 FORMAT(////,37X,'- - - - COMMENTS - - - -',//)                    MA 929
C  129 FORMAT(25X,20A4)                                                 MA 930
  129 FORMAT(' ', 20A4)                                                 MA 931
```

```
130 FORMAT(///,10X,'INCORRECT LABEL FOR A COMMENT CARD')
135 FORMAT(/////)
136 FORMAT(A2,I3,3I5,6E10.3)
137 FORMAT(1X,'***** DATA CARD NO.',I3,3X,A2,1X,I3,3(1X,I5),6(1X,1P,E
   *12.5))
138 FORMAT(///,10X,'FAULTY DATA CARD LABEL AFTER GEOMETRY SECTION')
139 FORMAT(///,10X,'NUMBER OF LOADING CARDS EXCEEDS STORAGE ALLOTTED'
   *)
140 FORMAT(///,10X,'DATA FAULT ON LOADING CARD NO.=',I5,5X,'ITAG S',
   *'TEP1=',I5,'  IS GREATER THAN ITAG STEP2=',I5)
141 FORMAT(///,10X,'NUMBER OF EXCITATION CARDS EXCEEDS STORAGE ALLO',
   *'TTED')
142 FORMAT(///,10X,'NUMBER OF NETWORK CARDS EXCEEDS STORAGE ALLOTTED'
   *)
143 FORMAT(///,10X,'WHEN MULTIPLE FREQUENCIES ARE REQUESTED, ONLY ONE
   * NEAR FIELD CARD CAN BE USED -',/,10X,'LAST CARD READ IS USED')
145 FORMAT(////,33X,'- - - - - - FREQUENCY - - - - - -',//,36X,'FR',
   *'EQUENCY=',1P,E11.4,' MHZ',/,36X,'WAVELENGTH=',E11.4,' METERS')
146 FORMAT(///,30X,' - - - STRUCTURE IMPEDANCE LOADING - - -')
147 FORMAT(/,35X,'THIS STRUCTURE IS NOT LOADED')
148 FORMAT(///,34X,'- - - ANTENNA ENVIRONMENT - - -',/)
149 FORMAT(40X,'MEDIUM UNDER SCREEN -')
150 FORMAT(40X,'RELATIVE DIELECTRIC CONST.=',F7.3,/,40X,'CONDUCTIV',
   *'ITY=',1P,E10.3,' MHOS/METER',/,40X,
   *'COMPLEX DIELECTRIC CONSTANT=',2E12.5)
151 FORMAT(42X,'PERFECT GROUND')
152 FORMAT(44X,'FREE SPACE')
153 FORMAT(///,32X,'- - - MATRIX TIMING - - -',//,24X,'FILL=',F9.3,
   *' SEC.,  FACTOR=',F9.3,' SEC.')
154 FORMAT(///,40X,'- - - EXCITATION - - -')
155 FORMAT(/,4X,'PLANE WAVE',4X,'THETA=',F7.2,' DEG,  PHI=',F7.2,
   *' DEG,  ETA=',F7.2,' DEG,  TYPE -',A6,'=  AXIAL RATIO=',F6.3)
156 FORMAT(/,31X,'POSITION (METERS)',14X,'ORIENTATION (DEG)=/',28X,
   *'X',12X,'Y',12X,'Z',10X,'ALPHA',5X,'BETA',4X,'DIPOLE MOMENT',//,4
   *X,'CURRENT SOURCE',1X,3(3X,F10.5),1X,2(3X,F7.2),4X,F8.3)
157 FORMAT(4X,4(I5,1X),1P,6(3X,E11.4),3X,A6,A2)
158 FORMAT(///,44X,'- - - NETWORK DATA - - -')
159 FORMAT(/,6X,'- FROM -    - TO -',11X,'TRANSMISSION LINE',15X,
   *'-  -  SHUNT ADMITTANCES (MHOS)  -  -',14X,'LINE',/,6X,
   *'TAG  SEG.','   TAG  SEG.',6X,'IMPEDANCE',6X,'LENGTH',12X,
   *'- END ONE -',17X,'- END TWO -',12X,'TYPE',/,6X,
   *'NO.   NO.   NO.   NO.',9X,'OHM''S',8X,'METERS',9X,'REAL',10X,
   *'IMAG.',9X,'REAL',10X,'IMAG.')
160 FORMAT(/,6X,'- FROM -',4X,'- TO -',26X,'-  -  ADMITTANCE MATRIX',
   *' ELEMENTS (MHOS)  -  -',/,6X,'TAG  SEG.   TAG  SEG.',13X,'(ON',
   *'E,ONE)',19X,'(ONE,TWO)',19X,'(TWO,TWO)',/,6X,'NO.   NO.   NO.',
   *'   NO.',8X,'REAL',10X,'IMAG.',9X,'REAL',10X,'IMAG.',9X,'REAL',10
   *X,'IMAG.')
161 FORMAT(///,29X,'- - - CURRENTS AND LOCATION - - -',//,33X,'DIS',
```

```
   *'TANCES IN WAVELENGTHS')
162 FORMAT(//,2X,'SEG.',2X,'TAG',4X,'COORD. OF SEG. CENTER',5X,'SEG.'
   *,12X,'- - - CURRENT (AMPS) - - -',/,2X,'NO.',3X,'NO.',5X,'X',8X,
   *'Y',8X,'Z',6X,'LENGTH',5X,'REAL',8X,'IMAG.',7X,'MAG.',8X,'PHASE')
163 FORMAT(///,33X,'- - - RECEIVING PATTERN PARAMETERS - - -',/,43X,
   *'ETA=',F7.2,' DEGREES',/,43X,'TYPE -',A6,/,43X,'AXIAL RATIO=',F6.
   *3,//,11X,'THETA',6X,'PHI',10X,'-  CURRENT  -',9X,'SEG',/,11X,
   *'(DEG)',5X,'(DEG)',7X,'MAGNITUDE',4X,'PHASE',6X,'NO.',/)
164 FORMAT(10X,2(F7.2,3X),1X,1P,E11.4,3X,0P,F7.2,4X,I5)
165 FORMAT(1X,2I5,3F9.4,F9.5,1X,1P,3E12.4,0P,F9.3)
166 FORMAT(///,40X,'- - - POWER BUDGET - - -',//,43X,'INPUT PO',
   *'WER   =',1P,E11.4,' WATTS',/,43X,'RADIATED POWER=',E11.4,
   *' WATTS',/,43X,'STRUCTURE LOSS=',E11.4,' WATTS',/,43X,
   *'NETWORK LOSS  =',E11.4,' WATTS',/,43X,'EFFICIENCY    =',0P,F7.2,
   *' PERCENT')
170 FORMAT(40X,'RADIAL WIRE GROUND SCREEN',/,40X,I5,' WIRES',/,40X,
   *'WIRE LENGTH=',F8.2,' METERS',/,40X,'WIRE RADIUS=',1P,E10.3,
   *' METERS')
181 FORMAT(///,4X,'RECEIVING PATTERN STORAGE TOO SMALL,ARRAY TRUNCA',
   *'TED')
182 FORMAT(///,32X,'- - - NORMALIZED RECEIVING PATTERN - - -',/,41X,
   *'NORMALIZATION FACTOR=',1P,E11.4,/,41X,'ETA=',0P,F7.2,' DEGREES',
   */,41X,'TYPE -',A6,/,41X,'AXIAL RATIO=',F6.3,/,41X,'SEGMENT NO.=',
   *I5,//,21X,'THETA',6X,'PHI',9X,'-  PATTERN  -',/,21X,'(DEG)',5X,
   *'(DEG)',8X,'DB',8X,'MAGNITUDE',/)
183 FORMAT(20X,2(F7.2,3X),1X,F7.2,4X,1P,E11.4)
184 FORMAT(///,36X,'- - - INPUT IMPEDANCE DATA - - -',/,45X,'SO',
   *'URCE SEGMENT NO.',I4,/,45X,'NORMALIZATION FACTOR=',1P,E12.5,//,7
   *X,'FREQ.',13X,'-  -  UNNORMALIZED IMPEDANCE  -  -',21X,'-',
   *' -  NORMALIZED IMPEDANCE  -  -',/,19X,'RESISTANCE',4X,'REACTA',
   *'NCE',6X,'MAGNITUDE',4X,'PHASE',7X,'RESISTANCE',4X,'REACTANCE',6X
   *,'MAGNITUDE',4X,'PHASE',/,8X,'MHZ',11X,'OHMS',10X,'OHMS',11X,
   *'OHMS',5X,'DEGREES',47X,'DEGREES',/)
185 FORMAT(///,4X,'STORAGE FOR IMPEDANCE NORMALIZATION TOO SMALL, A',
   *'RRAY TRUNCATED')
186 FORMAT(3X,F9.3,2X,1P,2(2X,E12.5),3X,E12.5,2X,0P,F7.2,2X,1P,2(2X,E
   *12.5),3X,E12.5,2X,0P,F7.2)
196 FORMAT(////,20X,'APPROXIMATE INTEGRATION EMPLOYED FOR SEGMENT',
   *'S MORE THAN',F8.3,' WAVELENGTHS APART')
197 FORMAT(////,41X,'- - - - SURFACE PATCH CURRENTS - - - -',//,50X,
   *'DISTANCE IN WAVELENGTHS',/,50X,'CURRENT IN AMPS/METER',//,28X,
   *'- - SURFACE COMPONENTS - -',19X,'- - - RECTANGULAR COM',
   *'PONENTS - - -',/,6X,'PATCH CENTER',6X,'TANGENT VECTOR 1',3X,
   *'TANGENT VECTOR 2',11X,'X',19X,'Y',19X,'Z',/,5X,'X',6X,'Y',6X,'Z'
   *,5X,'MAG.',7X,'PHASE',3X,'MAG.',7X,'PHASE',3(4X,'REAL',6X,'IMAG.'
   *))
198 FORMAT(1X,I4,/,1X,3F7.3,2(1P,E11.4,0P,F8.2),1P,6E10.2)
201 FORMAT(/,' RUN TIME =',F10.3)
315 FORMAT(///,34X,'- - - CHARGE DENSITIES - - -',//,36X,
```

```
     *'DISTANCES IN WAVELENGTHS',///,2X,'SEG.',2X,'TAG',4X,              MA1030
     *'COORD. OF SEG. CENTER',5X,'SEG.',10X,                             MA1031
     *'CHARGE DENSITY (COULOMBS/METER)',/,2X,'NO.',3X,'NO.',5X,'X',8X,   MA1032
     *'Y',8X,'Z',6X,'LENGTH',5X,'REAL',8X,'IMAG.',7X,'MAG.',8X,'PHASE')  MA1033
     *                                                                   MA1034
  321 FORMAT(/,20X,'THE EXTENDED THIN WIRE KERNEL WILL BE USED')         MA1035
  303 FORMAT(/,' ERROR - ',A2,' CARD IS NOT ALLOWED WITH N.G.F.')        MA1036
  327 FORMAT(/,35X,' LOADING ONLY IN N.G.F. SECTION')                    MA1037
  302 FORMAT(' ERROR - N.G.F. IN USE.  CANNOT WRITE NEW N.G.F.')         MA1038
  313 FORMAT(/,' NUMBER OF SEGMENTS IN COUPLING CALCULATION (CP) EXCEE'  MA1039
     *,'DS LIMIT')                                                       MA1040
  390 FORMAT(' RADIAL WIRE G. S. APPROXIMATION MAY NOT BE USED WITH SO'  MA1041
     *,'MMERFELD GROUND OPTION')                                         MA1042
  391 FORMAT(40X,'FINITE GROUND.  REFLECTION COEFFICIENT APPROXIMATION'  MA1043
     *)                                                                  MA1044
  392 FORMAT(40X,'FINITE GROUND.  SOMMERFELD SOLUTION')                  MA1045
  393 FORMAT(/,' ERROR IN GROUND PARAMETERS -',/,' COMPLEX DIELECTRIC',  MA1046
     *' CONSTANT FROM FILE IS',1P,2E12.5,/,32X,'REQUESTED',2E12.5)       MA1047
      END                                                                MA1048
```

ARC

PURPOSE

To fill COMMON/DATA/ with segment coordinates for a circular arc of segments.

METHOD

The formal parameters specify the number of segments, radius of the arc, starting angle, final angle and wire radius, segment coordinates are computed for the arc in the x-z plane with a left hand rotation about the y axis.

SYMBOL DICTIONARY

ANG	=	angle of point on the arc (radians, zero on x-axis)
ANG1	=	angle at first end
ANG2	=	angle at second end
DANG	=	angle covered by each segment
IST	=	number of initial segment
ITG	=	tag number assigned to each segment
NS	=	number of segments
RAD	=	wire radius
RADA	=	arc radius
TA	=	$\pi/180$
XS1	=	x coordinate of first end of segment
XS2	=	x coordinate of second end of segment
ZS1	=	z coordinate of first end of segment
ZS2	=	z coordinate of second end of segment

CONSTANTS

.01745329252	=	$\pi/180$
360.00001	=	test for angle greater than 360 degrees

```
      SUBROUTINE ARC(ITG,NS,RADA,ANG1,ANG2,RAD)
C
C     ARC GENERATES SEGMENT GEOMETRY DATA FOR AN ARC OF NS SEGMENTS
C
      COMMON/DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X(NM), Y(NM),
     *Z(NM), SI(NM), BI(NM), ALP(NM), BET(NM), ICON1(N2M), ICON2(
     * N2M), ITAG(N2M), ICONX(NM), WLAM, IPSYM
      DIMENSION  X2(1), Y2(1), Z2(1)
      EQUIVALENCE(X2,SI),(Y2,ALP),(Z2,BET)
      DATA  TA/.01745329252D+0/
      IST=N+1
      N=N+ NS
      NP=N
      MP=M
      IPSYM=0
      IF(NS.LT.1) RETURN
      IF(ABS(ANG2-ANG1).LT.360.00001D+0) GOTO 1
      WRITE(2,3)
      STOP
    1 ANG=ANG1* TA
      DANG=(ANG2- ANG1)* TA/ NS
      XS1=RADA* COS(ANG)
      ZS1=RADA* SIN(ANG)
      DO 2 I=IST, N
      ANG=ANG+DANG
      XS2=RADA*COS(ANG)
      ZS2=RADA*SIN(ANG)
      X(I)=XS1
      Y(I)=0.
      Z(I)=ZS1
      X2(I)=XS2
      Y2(I)=0.
      Z2(I)=ZS2
      XS1=XS2
      ZS1=ZS2
      BI(I)=RAD
    2 ITAG(I)=ITG
C
      RETURN
    3 FORMAT(' ERROR -- ARC ANGLE EXCEEDS 360. DEGREES')
      END
```

ATGN2

PURPOSE

To return zero when both arguments of a two-argument arctangent function are zero. (Most standard arctangent functions give an error return when both arguments are zero.)

METHOD

System function ATAN2 is used except when both arguments are zero, in which case the value zero is returned. The value returned is the angle (in radians) whose sine is X and cosine is Y.

SYMBOL DICTIONARY

X = first argument
Y = second argument

CODE LISTING

```
      FUNCTION ATGN2(X,Y)
C
C     ATGN2 IS ARCTANGENT FUNCTION MODIFIED TO RETURN 0. WHEN X=Y=0.
C
      IF(X) 3,1,3
    1 IF(Y) 3,2,3
    2 ATGN2=0.
      RETURN
    3 ATGN2= ATAN2( X, Y)
      RETURN
      END
```

BLCKOT

PURPOSE

To control the writing and reading of matrix blocks on files for the out-of-core matrix solution. The routine also checks for the end-of-file condition during reading.

METHOD

The routine uses a binary read and write with implied DO loops for reading and writing variable length strings into and out of various core locations. The end-of-file condition is checked by a call to function ENF. If an unexpected end or file is detected (governed by NEOF) the program stops.

CODING

BL9-BL12	Write a record on file NUNIT.
BL14-BL20	Read NBLKS records from NUNIT, and check for end of file.
BL21-BL24	Code if end of file detected.

SYMBOL DICTIONARY

AR	=	matrix array
ENF	=	external function (checks end-of-file condition)
I	=	DO loop index
I1	=	implied DO loop limits, inclusive matrix locations written from
I2		or read into
J	=	implied DO index
NBLKS	=	number of records to be read
NEOF	=	EOF check flag, also used to trace the call to BLCKOT
NUNIT	=	file number

CONSTANT

777 = NEOF when EOF is expected by calling program

```
      SUBROUTINE BLCKOT( AR, NUNIT, IX1, IX2, NBLKS, NEOF)
C
C     BLCKOT CONTROLS THE READING AND WRITING OF MATRIX BLOCKS ON FILES
C     FOR THE OUT-OF-CORE MATRIX SOLUTION.
C
C     LOGICAL  ENF
      COMPLEX  AR
      DIMENSION  AR(1000)
      I1=(IX1+1)/2
      I2=(IX2+1)/2
    1 WRITE(NUNIT) ( AR( J), J= I1, I2)
      RETURN
      ENTRY BLCKIN(AR,NUNIT,IX1,IX2,NBLKS,NEOF)
      I1=(IX1+1)/2
      I2=(IX2+1)/2
      DO 2 I=1, NBLKS
C     IF(ENF(NUNIT)) GO TO 3
      READ(NUNIT,END=3) ( AR( J), J= I1, I2)
    2 CONTINUE
      RETURN
    3 WRITE(2,4)  NUNIT, NBLKS, NEOF
      IF(NEOF.NE.777) STOP
      NEOF=0
C
      RETURN
    4 FORMAT('  EOF ON UNIT',I3,'  NBLKS= ',I3,'  NEOF= ',I5)
      END
```

CABC

PURPOSE

To compute the coefficients in the current function on each segment, given the basis function amplitudes. Surface current components are also computed.

METHOD

The total current on segment i is

$$I_i(s) = A_i + B_i \sin[k(s - s_i)] + C_i \cos[k(s - s_i)] \ ,$$

where s is distance slong the wire, and $s = s_i$ at the center of segment i. The coefficients A_i, B_i, and C_i are the sums of the corresponding coefficients in the portion of each basis function that extends onto segment i.

CODING

CB35	Call to TBF computes components of basis function I.
CB36-CB43	The basis function components are multiplied by the basis function amplitude from array CURX and summed for each segment.
CB45-CB63	For a current slope discontinuity source, the special basis function with discontinuous slope, from which the exciting electric field was computed, is recomputed and added to the current coefficients. The call to TBF, with the second argument zero and ICON1(I) temporarily zero, computes a basis function going to zero with non-zero derivative at end one of segment I.
CB64-CB65	Total current at the center of each segment is computed and stored in place of the basis function amplitudes.
CB68-CB79	The $\hat{t}_1$ and $\hat{t}_2$ components of surface current for each patch are expanded to x-, y-, and z-components.

SYMBOL DICTIONARY

AR,AI	=	real and imaginary parts of the basis function amplitude
CCJ	=	-j/60
CCX	=	-j/60
CS1	=	$\hat{t}_1$ component of surface current on a patch
CS2	=	$\hat{t}_1$ component of surface current on a patch
CURD	=	amplitude of the special basis function for a current slope discontinuity source
CURX	=	input array af basis function amplitudes that are replaced by values of Current at segment centers
J	=	number of a segment onto which a basis function extends
JCO1 JCO2	=	array locations of the $\hat{t}_1$ and $\hat{t}_2$ surface current components for a patch
JX	=	DO loop index; temporary storage of connection number
K	=	array location for patch geometry data
SH	=	(half segment length)/λ
TP	=	2π

```
      SUBROUTINE CABC(CURX)
C
C     CABC COMPUTES COEFFICIENTS OF THE CONSTANT (A), SINE (B), AND
C     COSINE (C) TERMS IN THE CURRENT INTERPOLATION FUNCTIONS FOR THE
C     CURRENT VECTOR CUR.
C
      COMPLEX  CUR, CURX, VQDS, CURD, CCJ, VSANT, VQD, CS1, CS2
      COMMON/DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X(NM), Y(NM),
     *Z(NM), SI(NM), BI(NM), ALP(NM), BET(NM), ICON1(N2M), ICON2(
     * N2M), ITAG(N2M), ICONX(NM), WLAM, IPSYM
      COMMON/CRNT/ AIR(NM), AII(NM), BIR(NM), BII(NM), CIR(NM),
     *CII(NM), CUR(N3M)
      COMMON/SEGJ/ AX(30), BX(30), CX(30), JCO(30), JSNO, ISCON(50),
     *NSCON, IPCON(10), NPCON
      COMMON/VSORC/ VQD(30), VSANT(30), VQDS(30), IVQD(30), ISANT(30)
     *, IQDS(30), NVQD, NSANT, NQDS
      COMMON/ANGL/ SALP(NM)
      DIMENSION  T1X(1), T1Y(1), T1Z(1), T2X(1), T2Y(1), T2Z(1)
      DIMENSION  CURX(1), CCJX(2)
      EQUIVALENCE(T1X,SI),(T1Y,ALP),(T1Z,BET),(T2X,ICON1),(T2Y,ICON2),(
     *T2Z,ITAG)
      EQUIVALENCE(CCJ,CCJX)
      DATA  TP/6.283185308D+0/, CCJX/0.,-0.01666666667D+0/
      IF(N.EQ.0) GOTO 6
      DO 1  I=1, N
      AIR(I)=0.
      AII(I)=0.
      BIR(I)=0.
      BII(I)=0.
      CIR(I)=0.
    1 CII(I)=0.
      DO 2  I=1, N
      AR=REAL(CURX(I))
      AI=AIMAG(CURX(I))
      CALL TBF(I,1)
      DO 2  JX=1, JSNO
      J=JCO(JX)
      AIR(J)=AIR(J)+ AX(JX)* AR
      AII(J)=AII(J)+ AX(JX)* AI
      BIR(J)=BIR(J)+ BX(JX)* AR
      BII(J)=BII(J)+ BX(JX)* AI
      CIR(J)=CIR(J)+ CX(JX)* AR
    2 CII(J)=CII(J)+ CX(JX)* AI
      IF(NQDS.EQ.0) GOTO 4
      DO 3  IS=1, NQDS
      I=IQDS(IS)
      JX=ICON1(I)
      ICON1(I)=0
      CALL TBF(I,0)
```

```
      ICON1(I)=JX
      SH=SI(I)*.5
      CURD=CCJ* VQDS(IS)/((LOG(2.* SH/ BI(I))-1.)*(BX(JSNO)* COS(
     * TP* SH)+ CX(JSNO)* SIN(TP* SH))* WLAM)
      AR=REAL(CURD)
      AI=AIMAG(CURD)
      DO 3  JX=1, JSNO
      J=JCO(JX)
      AIR(J)=AIR(J)+ AX(JX)* AR
      AII(J)=AII(J)+ AX(JX)* AI
      BIR(J)=BIR(J)+ BX(JX)* AR
      BII(J)=BII(J)+ BX(JX)* AI
      CIR(J)=CIR(J)+ CX(JX)* AR
    3 CII(J)=CII(J)+ CX(JX)* AI
    4 DO 5  I=1, N
    5 CURX(I)=CMPLX(AIR(I)+ CIR(I), AII(I)+ CII(I))
C     CONVERT SURFACE CURRENTS FROM T1,T2 COMPONENTS TO X,Y,Z COMPONENTS
    6 IF(M.EQ.0) RETURN
      K=LD- M
      JCO1=N+2* M+1
      JCO2=JCO1+ M
      DO 7  I=1, M
      K=K+1
      JCO1=JCO1-2
      JCO2=JCO2-3
      CS1=CURX(JCO1)
      CS2=CURX(JCO1+1)
      CURX(JCO2)=CS1* T1X(K)+ CS2* T2X(K)
      CURX(JCO2+1)=CS1* T1Y(K)+ CS2* T2Y(K)
    7 CURX(JCO2+2)=CS1* T1Z(K)+ CS2* T2Z(K)
      RETURN
      END
```

<u>CANG</u>

PURPOSE

To calculate the phase angle of a complex number in degrees.

METHOD

z = x + jy

Φ = [arctan (y/x)] 57.29577951

SYMBOL DICTIONARY

AIMAG	=	external routine (imaginary part of complex number)
ATGN2	=	external routine (arctan for all quadrants)
CANG	=	Φ
REAL	=	external routine (real part of a complex number)
Z	=	input complex quantity

CONSTANT

57.29577951 conversion factor for radians to degrees

CODE LISTING

```
      FUNCTION CANG( Z)
C
C     CANG RETURNS THE PHASE ANGLE OF A COMPLEX NUMBER IN DEGREES.
C
      COMPLEX  Z
      CANG=ATGN2(AIMAG(Z),REAL(Z))*57.29577951D+0
      RETURN
      END
```

CMNGF

PURPOSE

To compute and store the matrices B, C and D for the NGF solution.

METHOD

The structure of matrices B, C and D is described in Section VI. The coding to fill these matrices is involved due to their complex structure, as shown in Figure 12 of Section VI. The complexity is increased by the need to divide the matrices into blocks of rows when they are stored on files (see Section VII).

Much of the coding in CMNGF has to do with connections between new and NGF segments and patches. When a new segment or patch connects to a NGF segment the basis function associated with the NGF segment is modified due to the new junction condition. The amplitude of the modified basis function is a new unknown associated with the B' and D' sections of the matrix. The modified basis function may extend onto other NGF segments that may or may not connect directly to new segments. Also, the basis function of the new segment extends onto the NGF segment to which it connects. Hence fields must be computed for the currents on some NGF segments as well as all new segments.

Comments in the code should be of some help in understanding the procedure. The notation D(WS) in the comments corresponds to D_{sw} in Figure 12. Some parts of the code are explained below.

CG61-CG70 TRIO computes the components of all basis functions on segment J, where J is a new segment, and stores the coefficients in COMMON/SEGJ/. The array JCO contains the basis-function numbers which ordinarily are the matrix columns associated with the basis functions. If the basis function is for a new segment then JCO is set at CG66 to the column relative to the beginning of the matrix B. If the basis function is for a NGF segment modified by the connection, then JCO is set at CG68 to the column in B'_{ww} relative to the beginning of B. Thus the calls to CMWW and CMWS may store contributions in B'_{ww} and B'_{sw} as well as B_{ww} and B_{sw}.

CG90-CG108 In this section the fields are evaluated for NGF segments that connect to new segments or patches. TRIO findS all basis functions that contribute to the current on the segment. For a component of a new basis function IR is set to the column in B_{ww} at CG95. For a component of a modified basis function IR is set to the column in B'_{ww}, relative to the start of B, at CG99. If the basis function component is for a NGF basis function that has not been modified the test at CG98 skips to the end of the loop. The arrays in COMMON/SEGJ/ are adjusted from CG101 to CG104 so that CMWW and CMWS will store the matrix element contributions in the correct locations.

CG109-CG119	If a NCF segment connects to a new segment on one end and to a NGF patch on the opposite end the modified basis function extends onto the patch as a singular component of the patch current. The field due to this component on the patch is added to the matrix element of the modified basis function at CG119.
CG122-CG135	This is similar to CG90 to CG108, but evaluates fields of NGF segments that get contributions from modified basis functions, but do not connect directly to new segments. TBF is called, rather than TRIO to compute modified basis function J on all segments on which it exists. New segments and NCF segments for which contributions have already been evaluated are skipped at CG133 and CGl34.
CC165 CG263	Filling C and D is similar to that for B but fields must be evaluated for all NGF segments and patches as well as new segments and patches.

SYMBOL DICTIONARY

CB	=	array for matrix B
CC	=	array for matrix C
CD	=	array for matrix D
IEXKX	=	flag to select extended thin-wire kernel
MIEQ	=	number of patch equations in NGF
MEQ	=	total number of patch equations
NB	=	row dimension of CB. CB will contain only one block of B when ICASX = 3 or 4
NC	=	row dimension of CC (C transposed)
ND	=	row dimension of CD (D transposed)
NEQN	=	starting column of D_{ws}, relative to start of C
NEQF	=	starting column of zeros after D'_{ww}, relative to start of D
NEQS	=	starting column of D'_{ww}, relative to start of D
NEQSP	=	starting column of D'_{ww}, relative to start of C
RKHX	=	minimum range for using the lumped current approximation for the field af a segment

```
      SUBROUTINE CMNGF(CB, CC, CD, NB, NC, ND, RKHX, IEXKX)
C     CMNGF FILLS INTERACTION MATRICIES B, C, AND D FOR N.G.F. SOLUTION
      COMPLEX  CB, CC, CD, ZARRAY, EXK, EYK, EZK, EXS, EYS, EZS, EXC
     *, EYC, EZC
      COMMON  /DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X(NM), Y(NM),
     *Z(NM), SI(NM), BI(NM), ALP(NM), BET(NM), ICON1(N2M), ICON2(
     * N2M), ITAG(N2M), ICONX(NM), WLAM, IPSYM
      COMMON  /ZLOAD/ ZARRAY(NM), NLOAD, NLODF
      COMMON  /SEGJ/ AX(30), BX(30), CX(30), JCO(30), JSNO, ISCON(50),
     *NSCON, IPCON(10), NPCON
      COMMON  /DATAJ/ S, B, XJ, YJ, ZJ, CABJ, SABJ, SALPJ, EXK, EYK,
     *EZK, EXS, EYS, EZS, EXC, EYC, EZC, RKH, IEXK, IND1, INDD1, IND2,
     *INDD2, IPGND
      COMMON  /MATPAR/ ICASE, NBLOKS, NPBLK, NLAST, NBLSYM, NPSYM,
     *NLSYM, IMAT, ICASX, NBBX, NPBX, NLBX, NBBL, NPBL, NLBL
      DIMENSION  CB(NB,1), CC(NC,1), CD(ND,1)
      RKH=RKHX
      IEXK=IEXKX
      M1EQ=2* M1
      M2EQ=M1EQ+1
      MEQ=2* M
      NEQP=ND- NPCON*2
      NEQS=NEQP- NSCON
      NEQSP=NEQS+ NC
      NEQN=NC+ N- N1
      ITX=1
      IF(NSCON.GT.0) ITX=2
      IF(ICASX.EQ.1) GOTO 1
      REWIND 12
      REWIND 14
      REWIND 15
      IF(ICASX.GT.2) GOTO 5
    1 DO 4  J=1, ND
      DO 2  I=1, ND
    2 CD(I, J)=(0.,0.)
      DO 3  I=1, NB
      CB(I, J)=(0.,0.)
    3 CC(I, J)=(0.,0.)
    4 CONTINUE
    5 IST=N- N1+1
      IT=NPBX
C     LOOP THRU 24 FILLS B.  FOR ICASX=1 OR 2 ALSO FILLS D(WW), D(WS)
      ISV=- NPBX
      DO 24  IBLK=1, NBBX
      ISV=ISV+ NPBX
      IF(IBLK.EQ. NBBX) IT=NLBX
      IF(ICASX.LT.3) GOTO 7
      DO 6  J=1, ND
      DO 6  I=1, IT
```

```
    6 CB(I, J)=(0.,0.)
    7 I1=ISV+1
      I2=ISV+ IT
      IN2=I2
      IF(IN2.GT. N1) IN2=N1
      IM1=I1- N1
      IM2=I2- N1
      IF(IM1.LT.1) IM1=1
      IMX=1
      IF(I1.LE. N1) IMX=N1- I1+2
C     FILL B(WW),B(WS).  FOR ICASX=1,2 FILL D(WW),D(WS)
      IF(N2.GT. N) GOTO 12
      DO 11  J=N2, N
      CALL TRIO(J)
      DO 9  I=1, JSNO
      JSS=JCO(I)
C     SET JCO WHEN SOURCE IS NEW BASIS FUNCTION ON NEW SEGMENT
      IF(JSS.LT. N2) GOTO 8
      JCO(I)=JSS- N1
C     SOURCE IS PORTION OF MODIFIED BASIS FUNCTION ON NEW SEGMENT
      GOTO 9
    8 JCO(I)=NEQS+ ICONX(JSS)
    9 CONTINUE
      IF(I1.LE. IN2) CALL CMWW(J, I1, IN2, CB, NB, CB, NB,0)
      IF(IM1.LE. IM2) CALL CMWS(J, IM1, IM2, CB(IMX,1), NB, CB, NB,0
     *)
      IF(ICASX.GT.2) GOTO 11
      CALL CMWW(J, N2, N, CD, ND, CD, ND,1)
C     LOADING IN D(WW)
      IF(M2.LE. M) CALL CMWS(J, M2EQ, MEQ, CD(1, IST), ND, CD, ND,1)
      IF(NLOAD.EQ.0) GOTO 11
      IR=J- N1
      EXK=ZARRAY(J)
      DO 10  I=1, JSNO
      JSS=JCO(I)
   10 CD(JSS, IR)=CD(JSS, IR)-(AX(I)+ CX(I))* EXK
   11 CONTINUE
C     FILL B(WW)PRIME
   12 IF(NSCON.EQ.0) GOTO 20
      DO 19  I=1, NSCON
C     SOURCES ARE NEW OR MODIFIED BASIS FUNCTIONS ON OLD SEGMENTS WHICH
C     CONNECT TO NEW SEGMENTS
      J=ISCON(I)
      CALL TRIO(J)
      JSS=0
      DO 15  IX=1, JSNO
      IR=JCO(IX)
      IF(IR.LT. N2) GOTO 13
      IR=IR- N1
```

```
      GOTO 14
   13 IR=ICONX(IR)
      IF(IR.EQ.0) GOTO 15
      IR=NEQS+ IR
   14 JSS=JSS+1
      JCO(JSS)=IR
      AX(JSS)=AX(IX)
      BX(JSS)=BX(IX)
      CX(JSS)=CX(IX)
   15 CONTINUE
      JSNO=JSS
      IF(I1.LE. IN2) CALL CMWW(J, I1, IN2, CB, NB, CB, NB,0)
C     SOURCE IS SINGULAR COMPONENT OF PATCH CURRENT THAT IS PART OF
C     MODIFIED BASIS FUNCTION FOR OLD SEGMENT THAT CONNECTS TO A NEW
C     SEGMENT ON END OPPOSITE PATCH.
      IF(IM1.LE. IM2) CALL CMWS(J, IM1, IM2, CB(IMX,1), NB, CB, NB,0
     *)
      IF(I1.LE. IN2) CALL CMSW(J, I, I1, IN2, CB, CB,0, NB,-1)
      IF(NLODF.EQ.0) GOTO 17
      JX=J- ISV
      IF(JX.LT.1.OR. JX.GT. IT) GOTO 17
      EXK=ZARRAY(J)
      DO 16  IX=1, JSNO
      JSS=JCO(IX)
C     SOURCES ARE PORTIONS OF MODIFIED BASIS FUNCTION J ON OLD SEGMENTS
C     EXCLUDING OLD SEGMENTS THAT DIRECTLY CONNECT TO NEW SEGMENTS.
   16 CB(JX, JSS)=CB(JX, JSS)-(AX(IX)+ CX(IX))* EXK
   17 CALL TBF(J,1)
      JSX=JSNO
      JSNO=1
      IR=JCO(1)
      JCO(1)=NEQS+ I
      DO 19  IX=1, JSX
      IF(IX.EQ.1) GOTO 18
      IR=JCO(IX)
      AX(1)=AX(IX)
      BX(1)=BX(IX)
      CX(1)=CX(IX)
   18 IF(IR.GT. N1) GOTO 19
      IF(ICONX(IR).NE.0) GOTO 19
      IF(I1.LE. IN2) CALL CMWW(IR, I1, IN2, CB, NB, CB, NB,0)
C     LOADING FOR B(WW)PRIME
      IF(IM1.LE. IM2) CALL CMWS(IR, IM1, IM2, CB(IMX,1), NB, CB, NB,
     *0)
      IF(NLODF.EQ.0) GOTO 19
      JX=IR- ISV
      IF(JX.LT.1.OR. JX.GT. IT) GOTO 19
      EXK=ZARRAY(IR)
      JSS=JCO(1)
```

```
      CB(JX, JSS)=CB(JX, JSS)-(AX(1)+ CX(1))* EXK
   19 CONTINUE
   20 IF(NPCON.EQ.0) GOTO 22
C     FILL B(SS)PRIME TO SET OLD PATCH BASIS FUNCTIONS TO ZERO FOR
C     PATCHES THAT CONNECT TO NEW SEGMENTS
      JSS=NEQP
      DO 21  I=1, NPCON
      IX=IPCON(I)*2+ N1- ISV
      IR=IX-1
      JSS=JSS+1
      IF(IR.GT.0.AND. IR.LE. IT) CB(IR, JSS)=(1.,0.)
      JSS=JSS+1
      IF(IX.GT.0.AND. IX.LE. IT) CB(IX, JSS)=(1.,0.)
   21 CONTINUE
C     FILL B(SW) AND B(SS)
   22 IF(M2.GT. M) GOTO 23
      IF(I1.LE. IN2) CALL CMSW(M2, M, I1, IN2, CB(1, IST), CB, N1, NB
     *,0)
      IF(IM1.LE. IM2) CALL CMSS(M2, M, IM1, IM2, CB(IMX, IST), NB,0)
     *
   23 IF(ICASX.EQ.1) GOTO 24
      WRITE(14) ((CB(I, J), I=1, IT), J=1, ND)
C     FILLING B COMPLETE.  START ON C AND D
   24 CONTINUE
      IT=NPBL
      ISV=- NPBL
      DO 43  IBLK=1, NBBL
      ISV=ISV+ NPBL
      ISVV=ISV+ NC
      IF(IBLK.EQ. NBBL) IT=NLBL
      IF(ICASX.LT.3) GOTO 27
      DO 26  J=1, IT
      DO 25  I=1, NC
   25 CC(I, J)=(0.,0.)
      DO 26  I=1, ND
   26 CD(I, J)=(0.,0.)
   27 I1=ISVV+1
      I2=ISVV+ IT
      IN1=I1- M1EQ
      IN2=I2- M1EQ
      IF(IN2.GT. N) IN2=N
      IM1=I1- N
      IM2=I2- N
      IF(IM1.LT. M2EQ) IM1=M2EQ
      IF(IM2.GT. MEQ) IM2=MEQ
      IMX=1
      IF(IN1.LE. IN2) IMX=NEQN- I1+2
      IF(ICASX.LT.3) GOTO 32
C     SAME AS DO 24 LOOP TO FILL D(WW) FOR ICASX GREATER THAN 2
```

```
      IF(N2.GT. N) GOTO 32
      DO 31  J=N2, N
      CALL TRIO(J)
      DO 29  I=1, JSNO
      JSS=JCO(I)
      IF(JSS.LT. N2) GOTO 28
      JCO(I)=JSS- N1
      GOTO 29
   28 JCO(I)=NEQS+ ICONX(JSS)
   29 CONTINUE
      IF(IN1.LE. IN2) CALL CMWW(J, IN1, IN2, CD, ND, CD, ND,1)
      IF(IM1.LE. IM2) CALL CMWS(J, IM1, IM2, CD(1, IMX), ND, CD, ND,1
     *)
      IF(NLOAD.EQ.0) GOTO 31
      IR=J- N1- ISV
      IF(IR.LT.1.OR. IR.GT. IT) GOTO 31
      EXK=ZARRAY(J)
      DO 30  I=1, JSNO
      JSS=JCO(I)
   30 CD(JSS, IR)=CD(JSS, IR)-(AX(I)+ CX(I))* EXK
   31 CONTINUE
C     FILL D(SW) AND D(SS)
   32 IF(M2.GT. M) GOTO 33
      IF(IN1.LE. IN2) CALL CMSW(M2, M, IN1, IN2, CD(IST,1), CD, N1,
     *ND,1)
      IF(IM1.LE. IM2) CALL CMSS(M2, M, IM1, IM2, CD(IST, IMX), ND,1)
     *
C     FILL C(WW),C(WS), D(WW)PRIME, AND D(WS)PRIME.
   33 IF(N1.LT.1) GOTO 39
      DO 37  J=1, N1
      CALL TRIO(J)
      IF(NSCON.EQ.0) GOTO 36
      DO 35  IX=1, JSNO
      JSS=JCO(IX)
      IF(JSS.LT. N2) GOTO 34
      JCO(IX)=JSS+ M1EQ
      GOTO 35
   34 IR=ICONX(JSS)
      IF(IR.NE.0) JCO(IX)=NEQSP+ IR
   35 CONTINUE
   36 IF(IN1.LE. IN2) CALL CMWW(J, IN1, IN2, CC, NC, CD, ND, ITX)
      IF(IM1.LE. IM2) CALL CMWS(J, IM1, IM2, CC(1, IMX), NC, CD(1,
     *IMX), ND, ITX)
   37 CONTINUE
C     FILL C(WW)PRIME
      IF(NSCON.EQ.0) GOTO 39
      DO 38  IX=1, NSCON
      IR=ISCON(IX)
      JSS=NEQS+ IX- ISV
```

```
      IF(JSS.GT.0.AND. JSS.LE. IT) CC(IR, JSS)=(1.,0.)
   38 CONTINUE
   39 IF(NPCON.EQ.0) GOTO 41
C     FILL C(SS)PRIME
      JSS=NEQP- ISV
      DO 40  I=1, NPCON
      IX=IPCON(I)*2+ N1
      IR=IX-1
      JSS=JSS+1
      IF(JSS.GT.0.AND. JSS.LE. IT) CC(IR, JSS)=(1.,0.)
      JSS=JSS+1
      IF(JSS.GT.0.AND. JSS.LE. IT) CC(IX, JSS)=(1.,0.)
   40 CONTINUE
C     FILL C(SW) AND C(SS)
   41 IF(M1.LT.1) GOTO 42
      IF(IN1.LE. IN2) CALL CMSW(1, M1, IN1, IN2, CC(N2,1), CC,0, NC,1
     *)
      IF(IM1.LE. IM2) CALL CMSS(1, M1, IM1, IM2, CC(N2, IMX), NC,1)
   42 CONTINUE
      IF(ICASX.EQ.1) GOTO 43
      WRITE(12) ((CD(J, I), J=1, ND), I=1, IT)
      WRITE(15) ((CC(J, I), J=1, NC), I=1, IT)
   43 CONTINUE
      IF(ICASX.EQ.1) RETURN
      REWIND 12
      REWIND 14
      REWIND 15
      RETURN
      END
```

CMSET

PURPOSE

To control the filling of the interaction matrix.

METHOD

The linear equations resulting from the moment method solution of equations 13, 14 and the negative of equation 15 in Part I are written an

$$\sum_{j=1}^{N} a_j A_{ij} + \sum_{j=1}^{2M} b_j B_{ij} = E_i \ , \quad i = 1, ..., N$$

$$\sum_{j=1}^{N} c_j C_{kj} + \sum_{j=1}^{2M} d_j D_{kj} = H_k \ , \quad k = 1, ..., 2N$$

where N = number of segments

M = number of patches

$A_{ij} = \hat{s}_i \cdot$ ($\vec{E}$ at $\vec{r}_i$ due to segment basis function j)

$B_{ij} = \hat{s}_i \cdot$ ($\vec{E}$ at $\vec{r}_i$ due to current on patch [(j+1)/2] in direction $\hat{u}_j$)

$C_{kj} = -\hat{v}_k \cdot$ ($\vec{H}$ at $\vec{P}_{[(k+1)/2]}$ due to segment basis function j) $\cdot S_{[(k+1)/2)]}$

$D_{kj} = -\hat{v}_k \cdot$ ($\vec{H}$ at $\vec{P}_{[(k+1)/2]}$ due to current on patch [(j+1)/2] in direction $\hat{u}_j$) $S_{[(k+1)/2)]} + \frac{1}{2}\sigma_{kj}$

$E_i = -\hat{s}_i \cdot$ (incident electric field at $\vec{r}_i$)

$H_k = \hat{v}_k \cdot$(incident magnetic field at $\vec{P}_{[(k+1)/2]}$) $S_{[(k+1)/2]}$

$\vec{r}_i =$ position of the center of segment i

$\vec{P}_i =$ position of the center of patch i

$\hat{s}_i =$ unit vector in the direction of segment i

$\hat{u}_i = \hat{t}_1$ if i is odd for patch [(i+1)/2]
$\hat{u}_i = \hat{t}_2$ if i is even for patch [(i+1)/2]

$\hat{v}_i = \hat{t}_2$ if i is odd for patch [(i+1)/2]
$\hat{v}_i = \hat{t}_1$ if i is even for patch [(i+1)/2]

S_i = 1 if $\hat{t}_1 \times \hat{t}_2 = \hat{n}$ on patch
S_i = -1 if $\hat{t}_1 \times \hat{t}_2 = -\hat{n}$ on patch

$\sigma_{kj} = -1$ if k = j = odd
$\sigma_{kj} = +1$ if k = j = even
$\sigma_{kj} = 0$ if k $\neq$ j

The basis function amplitudes a_j, b_j, c_j and d_j are determined later by solving the matrix equation of order N + 2M.

The matrix elements are computed by calling subroutines CMWW, CMSW, CMWS, and CMSS for the elements of A, B, C and D respectively. For A and C the components of all basis functions that extend across segment J are computed by calling TRIO at CM52. CMWW and CMWS are then called to compute the components of A or C due to these basis function components on segment J.

If segment j, with length Δ_j, is loaded with impedance Z_j. the elements of A are modified as $A_{jk} = A_{jk} - \frac{Z_j}{\Delta j} \times$ (value of basis function at at the center of segment j) for k = the numbers of all basis functions that extend onto segment j. The summation over values of k (k = JSS) for loading on segment J occurs at CM68.

The submatrices are stored in the array CM in transposed form. All references to rows and columns, here, apply to the nontransposed matrices. Thus 'row' in this discussion refers to the second index of CM in the code.

For a structure without symmetry the submatrices are stored in the order

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

If the complete matrix is too large for the array CM then blocks of rows are filled and written into file 11. A block may then contain rows from A and B, rows from C and D or a combination. The row of CM at which C and D start is computed as IST.

For a structure having p symmetric sections the submatrices are stored in the form

$$\begin{bmatrix} A_1 & B_1 & A_2 & B_2 & ... & A_p & B_p \\ C_1 & D_1 & C_2 & D_2 & ... & C_p & D_p \end{bmatrix}$$

where

$$\begin{bmatrix} A_i & B_i \\ C_i & D_i \end{bmatrix}$$

represents A_i in the first row of submatrices in equation 108 of Part I.

Each call to CMWW and CMWS may fill elements of A_i or C_i for any value of i. The column indices in array JCO are adjusted at CM55 to allow for the columns occupied by the B_i and D_i matrices. B_i and D_i are filled for each value of i in the loop from CM75 to CM81. The Fourier transform of the submatrices, or the transform for planar symmetry (equation 116 of Part I) is computed from CM85 to CM100.

SYMBOL DICTIONARY

CM	=	array for the matrix
I1	=	number of first equation in a block (patch equation +N for patches)
I2	=	number of the last equation in a block
IEXKX	=	1 to use extended thin wire kernel on wires, 0 otherwise
IMl	=	number of first patch equation in a block
IM2	=	number of last patch equation in a block
IN2	=	number of the last segment equation in a block
IOUT	=	number of real numbers in a block for output
IPR	=	row in CM (second index) for segment J
IST	=	row in CM of the first patch equation
ISV	=	Il - l
IT	=	number of rows in a block
IXBLK1	=	block number
JMI	=	number of first patch in a symmetric section
JM2	=	number of the last patch in a symmetric section
JST	=	column in GM of the first patch equation for a symmetric block
MP2	=	number of patch equations
NEQ	=	total number of equations
NOP	=	number of symmetric sections
NPEQ	=	number of equations in a symmetric section
NROW	=	row dimensions ot the transposed GM array
RKHX	=	minimum interaction distance at which the infinitesimal dipole approximation is used for the field of a segment
ZAJ	=	Z_j/Δ_j

```
      SUBROUTINE CMSET(NROW,CM,RKHX,IEXKX)
C
C     CMSET SETS UP THE COMPLEX STRUCTURE MATRIX IN THE ARRAY CM
C
      COMPLEX  CM, ZARRAY, ZAJ, EXK, EYK, EZK, EXS,
     *EYS, EZS, EXC, EYC, EZC, SSX, D, DETER
      COMMON/DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X(NM), Y(NM),
     *Z(NM), SI(NM), BI(NM), ALP(NM), BET(NM), ICON1(N2M), ICON2(
     * N2M), ITAG(N2M), ICONX(NM), WLAM, IPSYM
      COMMON/MATPAR/ ICASE, NBLOKS, NPBLK, NLAST, NBLSYM, NPSYM,
     *NLSYM, IMAT, ICASX, NBBX, NPBX, NLBX, NBBL, NPBL, NLBL
      COMMON/SMAT/ SSX(16,16)
      COMMON/SCRATM/ D(N2M)
      COMMON/ZLOAD/ ZARRAY(NM), NLOAD, NLODF
      COMMON/SEGJ/ AX(30), BX(30), CX(30), JCO(30), JSNO, ISCON(50),
     *NSCON, IPCON(10), NPCON
      COMMON/DATAJ/ S, B, XJ, YJ, ZJ, CABJ, SABJ, SALPJ, EXK, EYK,
     *EZK, EXS, EYS, EZS, EXC, EYC, EZC, RKH, IEXK, IND1, INDD1, IND2,
     *INDD2,IPGND
      DIMENSION  CM(NROW,1)
      MP2=2*MP
      NPEQ=NP+ MP2
      NEQ=N+2* M
      NOP=NEQ/ NPEQ
      IF(ICASE.GT.2) REWIND 11
      RKH=RKHX
      IEXK=IEXKX
      IOUT=2* NPBLK* NROW
C
C     CYCLE OVER MATRIX BLOCKS
C
      IT=NPBLK
      DO 13  IXBLK1=1, NBLOKS
      ISV=(IXBLK1-1)* NPBLK
      IF(IXBLK1.EQ. NBLOKS) IT=NLAST
      DO 1  I=1, NROW
      DO 1  J=1, IT
    1 CM(I, J)=(0.,0.)
      I1=ISV+1
      I2=ISV+ IT
      IN2=I2
      IF(IN2.GT. NP) IN2=NP
      IM1=I1- NP
      IM2=I2- NP
      IF(IM1.LT.1) IM1=1
      IST=1
      IF(I1.LE. NP) IST=NP- I1+2
C
C     WIRE SOURCE LOOP
```

```
C
      IF(N.EQ.0) GOTO 5
      DO 4  J=1, N
      CALL TRIO(J)
      DO 2  I=1, JSNO
      IJ=JCO(I)
    2 JCO(I)=((IJ-1)/ NP)* MP2+ IJ
      IF(I1.LE. IN2) CALL CMWW(J, I1, IN2, CM, NROW, CM, NROW,1)
      IF(IM1.LE. IM2) CALL CMWS(J, IM1, IM2, CM(1, IST), NROW, CM,
     *NROW,1)
C
C     MATRIX ELEMENTS MODIFIED BY LOADING
C
      IF(NLOAD.EQ.0) GOTO 4
      IF(J.GT. NP) GOTO 4
      IPR=J- ISV
      IF(IPR.LT.1.OR. IPR.GT. IT) GOTO 4
      ZAJ=ZARRAY(J)
      DO 3  I=1, JSNO
      JSS=JCO(I)
    3 CM(JSS, IPR)=CM(JSS, IPR)-(AX(I)+ CX(I))* ZAJ
    4 CONTINUE
C     MATRIX ELEMENTS FOR PATCH CURRENT SOURCES
    5 IF(M.EQ.0) GOTO 7
      JM1=1- MP
      JM2=0
      JST=1- MP2
      DO 6  I=1, NOP
      JM1=JM1+ MP
      JM2=JM2+ MP
      JST=JST+ NPEQ
      IF(I1.LE. IN2) CALL CMSW(JM1, JM2, I1, IN2, CM(JST,1), CM,0,
     *NROW,1)
      IF(IM1.LE. IM2) CALL CMSS(JM1, JM2, IM1, IM2, CM(JST, IST),
     *NROW,1)
    6 CONTINUE
    7 IF(ICASE.EQ.1) GOTO 13
C     COMBINE ELEMENTS FOR SYMMETRY MODES
      IF(ICASE.EQ.3) GOTO 12
      DO 11  I=1, IT
      DO 11  J=1, NPEQ
      DO 8  K=1, NOP
      KA=J+(K-1)* NPEQ
    8 D(K)=CM(KA, I)
      DETER=D(1)
      DO 9  KK=2, NOP
    9 DETER=DETER+ D(KK)
      CM(J, I)=DETER
      DO 11  K=2, NOP
```

```
      KA=J+(K-1)* NPEQ                                                  CM  99
      DETER=D(1)                                                        CM 100
      DO 10  KK=2, NOP                                                  CM 101
   10 DETER=DETER+ D(KK)* SSX(K, KK)                                    CM 102
      CM(KA, I)=DETER                                                   CM 103
   11 CONTINUE                                                          CM 104
C     WRITE BLOCK FOR OUT-OF-CORE CASES.                                CM 105
      IF(ICASE.LT.3) GOTO 13                                            CM 106
   12 CALL BLCKOT(CM,11,1, IOUT,1,31)                                   CM 107
   13 CONTINUE                                                          CM 108
      IF(ICASE.GT.2) REWIND 11                                          CM 109
      RETURN                                                            CM 110
      END                                                               CM 111
```

CMSS

PURPOSE

To compute and store matrix elements representing the H field at patch centers due to the current on patches.

METHOD

CMSS computes the matrix elements D_{kj} defined in the description of subroutine CMSET. Subroutine HINTG is called to compute the magnetic field at the center of patch I due to current on patch J. H due to the current $\hat{t}_1$ on patch J is stored in EXK, EYK and EZK, while H due to current $\hat{t}_2$ is stored in EXS, EYS and EZS. The term 0.5 σ_{kj} in D_{kj} is added at SS61 and SS62 for odd and even equations. The matrix elements are stored in array CM from SS63 to SS78 in either normal or transposed order. Elements for both the even and odd equations are stored if both equations are within the block.

SYMBOL DICTIONARY

GM	=	array for matrix storage
G11	=	D_{kj} for k odd, j odd
G12	=	D_{kj} for k odd, j even
G21	=	D_{kj} for k even, j odd
G22	=	D_{kj} for k even, j even
I1	=	patch number for first equation
I2	=	patch number for last equation
ICOMP	=	equation number for the odd numbered equation for observation patch I
II1	=	location of the odd numbered equation in CM
II2	=	location of the even numbered equation in CM
IL	=	array location for coordinates at patch I
IM1	=	patch equation number for first equation in block
IM2	=	patch equation number for last equation in block
ITRP	=	0 or 1 to select normal or transposed filling of GM
J1	=	number of first source patch
J2	=	number af last source patch
JJ1	=	column in non-transposed matrix, of the first equation for patch J
JJ2	=	column of second equation for patch J
JL	=	array location for coordinates of patch J
NROW	=	row dimension of GM
T1XI, T1YI, T1ZI T2XI, T2YI, T2ZI T1XJ, T1YJ, T1ZJ T2XJ, T2YJ, T2ZJ	=	x, y and z components of $\hat{t}_1$ or $\hat{t}_2$ for patch I or J
XI,YI,ZI	=	coordinates of center of patch I

```
      SUBROUTINE CMSS(J1, J2, IM1, IM2, CM, NROW, ITRP)
C     CMSS COMPUTES MATRIX ELEMENTS FOR SURFACE-SURFACE INTERACTIONS.
      COMPLEX  G11, G12, G21, G22, CM, EXK, EYK, EZK, EXS, EYS, EZS,
     * EXC, EYC, EZC
      COMMON/DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X(NM), Y(NM),
     *Z(NM), SI(NM), BI(NM), ALP(NM), BET(NM), ICON1(N2M), ICON2(
     * N2M), ITAG(N2M), ICONX(NM), WLAM, IPSYM
      COMMON/ANGL/ SALP(NM)
      COMMON/DATAJ/ S, B, XJ, YJ, ZJ, CABJ, SABJ, SALPJ, EXK, EYK,
     *EZK, EXS, EYS, EZS, EXC, EYC, EZC, RKH, IEXK, IND1, INDD1, IND2,
     *INDD2, IPGND
      DIMENSION  CM(NROW,1)
      DIMENSION  T1X(1), T1Y(1), T1Z(1), T2X(1), T2Y(1), T2Z(1)
      EQUIVALENCE(T1X,SI),(T1Y,ALP),(T1Z,BET),(T2X,ICON1),(T2Y,ICON2),(
     *T2Z,ITAG)
      EQUIVALENCE(T1XJ,CABJ),(T1YJ,SABJ),(T1ZJ,SALPJ),(T2XJ,B),(T2YJ,
     *IND1),(T2ZJ,IND2)
      LDP=LD+1
      I1=(IM1+1)/2
      I2=(IM2+1)/2
      ICOMP=I1*2-3
      II1=-1
C     LOOP OVER OBSERVATION PATCHES
      IF(ICOMP+2.LT. IM1) II1=-2
      DO 5  I=I1, I2
      IL=LDP- I
      ICOMP=ICOMP+2
      II1=II1+2
      II2=II1+1
      T1XI=T1X(IL)* SALP(IL)
      T1YI=T1Y(IL)* SALP(IL)
      T1ZI=T1Z(IL)* SALP(IL)
      T2XI=T2X(IL)* SALP(IL)
      T2YI=T2Y(IL)* SALP(IL)
      T2ZI=T2Z(IL)* SALP(IL)
      XI=X(IL)
      YI=Y(IL)
      ZI=Z(IL)
C     LOOP OVER SOURCE PATCHES
      JJ1=-1
      DO 5  J=J1, J2
      JL=LDP- J
      JJ1=JJ1+2
      JJ2=JJ1+1
      S=BI(JL)
      XJ=X(JL)
      YJ=Y(JL)
      ZJ=Z(JL)
      T1XJ=T1X(JL)
```

```
      T1YJ=T1Y(JL)
      T1ZJ=T1Z(JL)
      T2XJ=T2X(JL)
      T2YJ=T2Y(JL)
      T2ZJ=T2Z(JL)
      CALL HINTG(XI, YI, ZI)
      G11=-(T2XI* EXK+ T2YI* EYK+ T2ZI* EZK)
      G12=-(T2XI* EXS+ T2YI* EYS+ T2ZI* EZS)
      G21=-(T1XI* EXK+ T1YI* EYK+ T1ZI* EZK)
      G22=-(T1XI* EXS+ T1YI* EYS+ T1ZI* EZS)
      IF(I.NE. J) GOTO 1
      G11=G11-.5
      G22=G22+.5
C     NORMAL FILL
    1 IF(ITRP.NE.0) GOTO 3
      IF(ICOMP.LT. IM1) GOTO 2
      CM(II1, JJ1)=G11
      CM(II1, JJ2)=G12
    2 IF(ICOMP.GE. IM2) GOTO 5
      CM(II2, JJ1)=G21
      CM(II2, JJ2)=G22
C     TRANSPOSED FILL
      GOTO 5
    3 IF(ICOMP.LT. IM1) GOTO 4
      CM(JJ1, II1)=G11
      CM(JJ2, II1)=G12
    4 IF(ICOMP.GE. IM2) GOTO 5
      CM(JJ1, II2)=G21
      CM(JJ2, II2)=G22
    5 CONTINUE
      RETURN
      END
```

CMSW

PURPOSE

To compute and store matrix elements representing the electric field at segment centers due to the current on patches.

METHOD

SW30-SW35	Coordinates of observation segment are stored.
SW36-SW42	If either end of the observation segment connects to a surface IPCH is set to the number of the first of the four patches at the connection point.
SW48-SW57	Coordinates of the source patch are stored in COMMON/DATAJ/.
SW61-SW86	If IPCH = J then patch J is the first patch at the point where segment I connects to the surface. Subroutine PCINT is called to integrate the current over the four patches at the connection point. The current on the patches includes the eight basis functions of the four patches and a portion of the basis function from the segment. Hence contributions to nine matrix elements are generated and stored in array EMEL. The field due to the segment basis function extending onto the patches is stored in array CW at SW76 or SW78. The fields due to the first patch basis function, EMEL(l) and EMEL(5), are then stored in array CM at SW80 and SWSI or at SW83 and SW84. ICGO is then incremented. For the next three times through the loop over J the call to PCINT is skipped at SW63 and the remaining values in EMEL are stored.
SW88-SW96	If segment I and patch J are not connected, subroutine UNERE is called to compute the electric field due to the current on the patch with the current treated as Hertzian dipoles in the directions $\hat{t}_1$ and $\hat{t}_2$. The matrix elements are stored in GM.
SW102-SW138	This is a special section of code to compute the electric field due to the component of a segment basis function that extends onto connected patches. It is used at line CCIIZ of subroutine CMNGF for the case where the connected segment and patches are in the NGF file and a new segment is connected to the outer end of the NCF segment modifying its basis function. Subroutine PCINT is called to evaluate the nine matrix elements. Only EMEL(9) is used since the patch basis functions have not been modified.

SYMBOL DICTIONARY

CABI	=	x component of $\hat{i}$ in direction of segment I
CM	=	array for E due to patch basis functions
CW	=	array for E due to Segment basis function extending onto surface at connection point
EMEL	=	array of matrix elements from integrating over surface
FSIGN	=	$\pm I$ depending on which end of segment connects to surface
I1	=	number of first observation segment
I2	=	number of last observation segment
ICGO	=	index for matrix elements at connection point
IL	=	index for segment basis function in CW
IP	=	1 for direct field, 2 for image in ground
IPCH	=	number of first patch connecting to a segment
ITRP	=	0 for normal matrix fill 1 for transposed fill -1 for special NGF case
J	=	source patch
J1	=	first source patch
J2	=	last source patch
JL	=	index for source patch in CM
JS	=	index for patch coordinates
K	=	index in CM or CW for observation segment
NCW	=	index offset for CW
NEQS	=	number of equations excluding NGF
NROW	=	row dimensions of CM and CW
PI	=	π
PX	=	$\sin k(s-s_0)$ for s at the end of the segment
PY	=	$\cos k(s-s_0)$ connected to the surface
SABI	=	y-component of $\hat{i}$ in direction of segment I
SALPI	=	z-component of $\hat{i}$ in direction of segment I
XI,YI,ZI	=	center of observation segment

```
      SUBROUTINE CMSW( J1, J2, I1, I2, CM, CW, NCW, NROW, ITRP)
C     COMPUTES MATRIX ELEMENTS FOR E ALONG WIRES DUE TO PATCH CURRENT
      COMPLEX  CM, ZRATI, ZRATI2, T1, EXK, EYK, EZK, EXS, EYS, EZS,
     *EXC, EYC, EZC, EMEL, CW, FRATI
      COMMON  /DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X( NM), Y( NM),
     *Z( NM), SI( NM), BI( NM), ALP( NM), BET( NM), ICON1( N2M), ICON2(
     * N2M), ITAG( N2M), ICONX( NM), WLAM, IPSYM
      COMMON  /ANGL/ SALP( NM)
      COMMON  /GND/ ZRATI, ZRATI2, FRATI, CL, CH, SCRWL, SCRWR, NRADL,
     *KSYMP, IFAR, IPERF, T1, T2
      COMMON  /DATAJ/ S, B, XJ, YJ, ZJ, CABJ, SABJ, SALPJ, EXK, EYK,
     *EZK, EXS, EYS, EZS, EXC, EYC, EZC, RKH, IEXK, IND1, INDD1, IND2,
     *INDD2, IPGND
      COMMON  /SEGJ/ AX(30), BX(30), CX(30), JCO(30), JSNO, ISCON(50),
     *NSCON, IPCON(10), NPCON
      DIMENSION  CAB(1), SAB(1), CM( NROW,1), CW( NROW,1)
      DIMENSION  T1X(1), T1Y(1), T1Z(1), T2X(1), T2Y(1), T2Z(1), EMEL(9
     *)
      EQUIVALENCE(T1X,SI),(T1Y,ALP),(T1Z,BET),(T2X,ICON1),(T2Y,ICON2),(
     *T2Z,ITAG),(CAB,ALP),(SAB,BET)
      EQUIVALENCE(T1XJ,CABJ),(T1YJ,SABJ),(T1ZJ,SALPJ),(T2XJ,B),(T2YJ,
     *IND1),(T2ZJ,IND2)
      DATA   PI/3.141592654D+0/
      LDP= LD+1
      NEQS= N- N1+2*( M- M1)
      IF(ITRP.LT.0) GOTO 13
      K=0
C     OBSERVATION LOOP
      ICGO=1
      DO 12  I= I1, I2
      K= K+1
      XI= X( I)
      YI= Y( I)
      ZI= Z( I)
      CABI= CAB( I)
      SABI= SAB( I)
      SALPI= SALP( I)
      IPCH=0
      IF(ICON1( I).LT.10000) GOTO 1
      IPCH= ICON1( I)-10000
      FSIGN=-1.
    1 IF(ICON2( I).LT.10000) GOTO 2
      IPCH= ICON2( I)-10000
      FSIGN=1.
C     SOURCE LOOP
    2 JL=0
      DO 12  J= J1, J2
      JS= LDP- J
      JL= JL+2
```

```
      T1XJ= T1X( JS)
      T1YJ= T1Y( JS)
      T1ZJ= T1Z( JS)
      T2XJ= T2X( JS)
      T2YJ= T2Y( JS)
      T2ZJ= T2Z( JS)
      XJ= X( JS)
      YJ= Y( JS)
      ZJ= Z( JS)
C     GROUND LOOP
      S= BI( JS)
      DO 12  IP=1, KSYMP
      IPGND= IP
      IF(IPCH.NE. J.AND. ICGO.EQ.1) GOTO 9
      IF(IP.EQ.2) GOTO 9
      IF(ICGO.GT.1) GOTO 6
      CALL PCINT( XI, YI, ZI, CABI, SABI, SALPI, EMEL)
      PY= PI* SI( I)* FSIGN
      PX= SIN( PY)
      PY= COS( PY)
      EXC= EMEL(9)* FSIGN
      CALL TRIO( I)
      IF(I.GT. N1) GOTO 3
      IL= NEQS+ ICONX( I)
      GOTO 4
    3 IL= I- NCW
      IF(I.LE. NP) IL=(( IL-1)/ NP)*2* MP+ IL
    4 IF(ITRP.NE.0) GOTO 5
      CW( K, IL)= CW( K, IL)+ EXC*( AX( JSNO)+ BX( JSNO)* PX+ CX( JSNO)
     ** PY)
      GOTO 6
    5 CW( IL, K)= CW( IL, K)+ EXC*( AX( JSNO)+ BX( JSNO)* PX+ CX( JSNO)
     ** PY)
    6 IF(ITRP.NE.0) GOTO 7
      CM( K, JL-1)= EMEL( ICGO)
      CM( K, JL)= EMEL( ICGO+4)
      GOTO 8
    7 CM( JL-1, K)= EMEL( ICGO)
      CM( JL, K)= EMEL( ICGO+4)
    8 ICGO= ICGO+1
      IF(ICGO.EQ.5) ICGO=1
      GOTO 11
    9 CALL UNERE( XI, YI, ZI)
C     NORMAL FILL
      IF(ITRP.NE.0) GOTO 10
      CM( K, JL-1)= CM( K, JL-1)+ EXK* CABI+ EYK* SABI+ EZK* SALPI
      CM( K, JL)= CM( K, JL)+ EXS* CABI+ EYS* SABI+ EZS* SALPI
C     TRANSPOSED FILL
      GOTO 11
```

```
   10 CM( JL-1, K)= CM( JL-1, K)+ EXK* CABI+ EYK* SABI+ EZK* SALPI
      CM( JL, K)= CM( JL, K)+ EXS* CABI+ EYS* SABI+ EZS* SALPI
   11 CONTINUE
   12 CONTINUE
C     FOR OLD SEG. CONNECTING TO OLD PATCH ON ONE END AND NEW SEG. ON
C     OTHER END INTEGRATE SINGULAR COMPONENT (9) OF SURFACE CURRENT ONLY
      RETURN
   13 IF(J1.LT. I1.OR. J1.GT. I2) GOTO 16
      IPCH= ICON1( J1)
      IF(IPCH.LT.10000) GOTO 14
      IPCH= IPCH-10000
      FSIGN=-1.
      GOTO 15
   14 IPCH= ICON2( J1)
      IF(IPCH.LT.10000) GOTO 16
      IPCH= IPCH-10000
      FSIGN=1.
   15 IF(IPCH.GT. M1) GOTO 16
      JS= LDP- IPCH
      IPGND=1
      T1XJ= T1X( JS)
      T1YJ= T1Y( JS)
      T1ZJ= T1Z( JS)
      T2XJ= T2X( JS)
      T2YJ= T2Y( JS)
      T2ZJ= T2Z( JS)
      XJ= X( JS)
      YJ= Y( JS)
      ZJ= Z( JS)
      S= BI( JS)
      XI= X( J1)
      YI= Y( J1)
      ZI= Z( J1)
      CABI= CAB( J1)
      SABI= SAB( J1)
      SALPI= SALP( J1)
      CALL PCINT( XI, YI, ZI, CABI, SABI, SALPI, EMEL)
      PY= PI* SI( J1)* FSIGN
      PX= SIN( PY)
      PY= COS( PY)
      EXC= EMEL(9)* FSIGN
      IL= JCO( JSNO)
      K= J1- I1+1
      CW( K, IL)= CW( K, IL)+ EXC*( AX( JSNO)+ BX( JSNO)* PX+ CX( JSNO)
     ** PY)
   16 RETURN
      END
```

CMWS

PURPOSE

To compute and store matrix elements representing the magnetic field at patch centers due to the current on wire segments.

METHOD

Matrix elements are computed for patch equations numbered I1 through I2 with the source segment J. For odd numbered equations the matrix element represents the first term on the right side of equation 14 of Part I. For even numbered equations it is the negative of the first term on the right side of equation 15. For equation 11 and for all odd numbered equations subroutine HSFLD is called to compute the H field at the center of the patch due to constant, sin k(s - s_0) and cos k(s - s_0) currents on segment J. The required component of the field, $-\hat{t}_2 \cdot \vec{H}$ or $-\hat{t}_1 \cdot \vec{H}$ for odd or even equations respectively, is computed from WS49 to WS51. Multiplication by SALP(JS) reverses the sign when ($\hat{t}_1$,$\hat{t}_2$,$\hat{n}$) has a left-hand orientation on a patch formed by reflection. The field component for each basis function component on segment J is computed and stored for WS56 through WS75. Storage of the matrix elements is similar to that in subroutine CMWW.

SYMBOL DICTIONARY

CM	=	array for matrix elements
CN	=	array for matrix elements (NCF' only)
ETK ETS ETC	=	$-\hat{t}_2 \cdot \vec{H}$ or $-\hat{t}_1 \cdot \vec{H}$ due to current of constant, $\sin k(s - s_0)$, or $\cos k(s - s_0)$ respectively
I	=	equation number
I1	=	number of first equation
I2	=	number of second equation
IK	=	0 if I is even, 1 if I is odd
IPATCH	=	patch number for equation I
IPR	=	relative matrix location for equation I. Position in complete matrix depends on the address of CM in the call to CMWS
ITRP	=	0 for non-transposed fill 1 for transposed fill 2 for transposed fill for NGF
J	=	source segment number
JS	=	location in COMMON/DATA/ of paramaters for patch J
JX	=	matrix index for a particular basis function
LDP	=	LD + 1
NR	=	row dimension of GM
NW	=	ZOW dimension of CW
TX	=	x-component of $\hat{t}_1$ or $\hat{t}_2$
TY	=	y-component of $\hat{t}_1$ or $\hat{t}_2$
TZ	=	z-component of $\hat{t}_1$ or $\hat{t}_2$
XI YI ZI	=	x, y and z coordinates of the center of the patch at which the field is computed

```
      SUBROUTINE CMWS(J, I1, I2, CM, NR, CW, NW, ITRP)
C
C     CMWS COMPUTES MATRIX ELEMENTS FOR WIRE-SURFACE INTERACTIONS
C
      COMPLEX  CM, CW, ETK, ETS, ETC, EXK, EYK, EZK, EXS, EYS, EZS,
     *EXC, EYC, EZC
      COMMON  /DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X(NM), Y(NM),
     *Z(NM), SI(NM), BI(NM), ALP(NM), BET(NM), ICON1(N2M), ICON2(
     * N2M), ITAG(N2M), ICONX(NM), WLAM, IPSYM
      COMMON  /ANGL/ SALP(NM)
      COMMON  /SEGJ/ AX(30), BX(30), CX(30), JCO(30), JSNO, ISCON(50),
     *NSCON, IPCON(10), NPCON
      COMMON  /DATAJ/ S, B, XJ, YJ, ZJ, CABJ, SABJ, SALPJ, EXK, EYK,
     *EZK, EXS, EYS, EZS, EXC, EYC, EZC, RKH, IEXK, IND1, INDD1, IND2,
     *INDD2, IPGND
      DIMENSION  CM(NR,1), CW(NW,1), CAB(1), SAB(1)
      DIMENSION  T1X(1), T1Y(1), T1Z(1), T2X(1), T2Y(1), T2Z(1)
      EQUIVALENCE(CAB,ALP),(SAB,BET),(T1X,SI),(T1Y,ALP),(T1Z,BET)
      EQUIVALENCE(T2X,ICON1),(T2Y,ICON2),(T2Z,ITAG)
      LDP=LD+1
      S=SI(J)
      B=BI(J)
      XJ=X(J)
      YJ=Y(J)
      ZJ=Z(J)
      CABJ=CAB(J)
      SABJ=SAB(J)
C
C     OBSERVATION LOOP
C
      SALPJ=SALP(J)
      IPR=0
      DO 9  I=I1, I2
      IPR=IPR+1
      IPATCH=(I+1)/2
      IK=I-(I/2)*2
      IF(IK.EQ.0.AND. IPR.NE.1) GOTO 1
      JS=LDP- IPATCH
      XI=X(JS)
      YI=Y(JS)
      ZI=Z(JS)
      CALL HSFLD(XI, YI, ZI,0.)
      IF(IK.EQ.0) GOTO 1
      TX=T2X(JS)
      TY=T2Y(JS)
      TZ=T2Z(JS)
      GOTO 2
    1 TX=T1X(JS)
      TY=T1Y(JS)
```

```
      TZ=T1Z(JS)
    2 ETK=-(EXK* TX+ EYK* TY+ EZK* TZ)* SALP(JS)
      ETS=-(EXS* TX+ EYS* TY+ EZS* TZ)* SALP(JS)
C
C     FILL MATRIX ELEMENTS.  ELEMENT LOCATIONS DETERMINED BY CONNECTION
C     DATA.
C
      ETC=-(EXC* TX+ EYC* TY+ EZC* TZ)* SALP(JS)
C     NORMAL FILL
      IF(ITRP.NE.0) GOTO 4
      DO 3  IJ=1, JSNO
      JX=JCO(IJ)
    3 CM(IPR, JX)=CM(IPR, JX)+ ETK* AX(IJ)+ ETS* BX(IJ)+ ETC* CX(
     *IJ)
      GOTO 9
C     TRANSPOSED FILL
    4 IF(ITRP.EQ.2) GOTO 6
      DO 5  IJ=1, JSNO
      JX=JCO(IJ)
    5 CM(JX, IPR)=CM(JX, IPR)+ ETK* AX(IJ)+ ETS* BX(IJ)+ ETC* CX(
     *IJ)
C     TRANSPOSED FILL - C(WS) AND D(WS)PRIME (=CW)
      GOTO 9
    6 DO 8  IJ=1, JSNO
      JX=JCO(IJ)
      IF(JX.GT. NR) GOTO 7
      CM(JX, IPR)=CM(JX, IPR)+ ETK* AX(IJ)+ ETS* BX(IJ)+ ETC* CX(
     *IJ)
      GOTO 8
    7 JX=JX- NR
      CW(JX, IPR)=CW(JX, IPR)+ ETK* AX(IJ)+ ETS* BX(IJ)+ ETC* CX(
     *IJ)
    8 CONTINUE
    9 CONTINUE
      RETURN
      END
```

CMWW

PURPOSE

To call subroutines to compute the electric field at segment centers due to current on other segments and to store matrix elements in array locations.

METHOD

WW17-WW24	Parameters of source segment (J) are stored in COMMON/DATAJ/.
WW27-WW43	First end of segment J is tested to determine whether the extended thin wire approximation can be used. It cannot be used at a junction of more than two wires (WW30), at a bend (WW31), at a change in radius (WW38), or at the base of a non-vertical segment connected to the ground (Ww33).
WW44 WW60	Second end of segment J is tested. .
WW66	Loop over observation segments ranges from Il to 12. The index IPR starts at 1 so the matrix element for I1 is stored in the first row or column of the array GM. The location in the complete matrix is determined by the address given for CM when CHMW is called.
WW76	EFLD computes the electric fields at (xi,yi,zi) due to segment J and stores them in COMMON/DATA]/.
HW77-WW79	Electric field tangent to segment I is computed.
WW84-WW103	Matrix elements are formed by combining the field components.
WW86-WW88	Matrix elements are stored in non-transposed order.
WW92-WW94	Matrix elements are stored in transposed order.
WW97-WW104	When the source segment is from a NGF file the matrix elements will normally be stored in submstrix C of the NGF matrix structure. when the segment connects to a new segment, however, contributions to submatrix D result. The C and D contributions are stored in CM and CW, respectively, in transposed order.

SYMBOL DICTIONARY

AI	=	radius of observation segment
CABI	=	x-component of unit vector in direction of segment
CM	=	array for matrix elements
CW	=	array for matrix elements (NGF only)
ETK	=	E field tangent to segment I due to current of
ETS		constant, $\sin k(s-s_0)$ and $\cos k(s-s_0)$
ETC		distribution, respectively, on segment J.
I1	=	first observation segment
I2	=	final observation segment
IJ	=	0 for special treatment when I = J
IPR	=	relative matrix location for observation point
ITRP	=	0 for non-transposed fill
		1 for transposed fill
		2 for transposed fill for NGF
J	=	source segment number
JX	=	matrix index for a particular basis function
NR	=	row dimension of CM
NW	=	row dimension of CW
SABI	=	y-component of unit vector in direction of aegment
SALPI	=	z-component of unit vector in direction of segment
XI,YI,ZI	=	coordinates of center of segment I.

CONSTANTS

0.999999 = test for collinear segments

```
      SUBROUTINE CMWW( J, I1, I2, CM, NR, CW, NW, ITRP)
C
C     CMWW COMPUTES MATRIX ELEMENTS FOR WIRE-WIRE INTERACTIONS
C
      COMPLEX  CM, CW, ETK, ETS, ETC, EXK, EYK, EZK, EXS, EYS, EZS,
     *EXC, EYC, EZC
      COMMON  /DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X( NM), Y( NM),
     *Z( NM), SI( NM), BI( NM), ALP( NM), BET( NM), ICON1( N2M), ICON2(
     * N2M), ITAG( N2M), ICONX( NM), WLAM, IPSYM
      COMMON  /ANGL/ SALP( NM)
      COMMON  /SEGJ/ AX(30), BX(30), CX(30), JCO(30), JSNO, ISCON(50),
     *NSCON, IPCON(10), NPCON
      COMMON  /DATAJ/ S, B, XJ, YJ, ZJ, CABJ, SABJ, SALPJ, EXK, EYK,
     *EZK, EXS, EYS, EZS, EXC, EYC, EZC, RKH, IEXK, IND1, INDD1, IND2,
     *INDD2, IPGND
      DIMENSION  CM( NR,1), CW( NW,1), CAB(1), SAB(1)
C     SET SOURCE SEGMENT PARAMETERS
      EQUIVALENCE(CAB,ALP),(SAB,BET)
      S= SI( J)
      B= BI( J)
      XJ= X( J)
      YJ= Y( J)
      ZJ= Z( J)
      CABJ= CAB( J)
      SABJ= SAB( J)
      SALPJ= SALP( J)
C     DECIDE WETHER EXT. T.W. APPROX. CAN BE USED
      IF(IEXK.EQ.0) GOTO 16
      IPR= ICON1( J)
      IF(IPR) 1,6,2
    1 IPR=- IPR
      IF(- ICON1( IPR).NE. J) GOTO 7
      GOTO 4
    2 IF(IPR.NE. J) GOTO 3
      IF(CABJ* CABJ+ SABJ* SABJ.GT.1.D-8) GOTO 7
      GOTO 5
    3 IF(ICON2( IPR).NE. J) GOTO 7
    4 XI= ABS( CABJ* CAB( IPR)+ SABJ* SAB( IPR)+ SALPJ* SALP( IPR))
      IF(XI.LT.0.999999D+0) GOTO 7
      IF(ABS( BI( IPR)/ B-1.).GT.1.D-6) GOTO 7
    5 IND1=0
      GOTO 8
    6 IND1=1
      GOTO 8
    7 IND1=2
    8 IPR= ICON2( J)
      IF(IPR) 9,14,10
    9 IPR=- IPR
      IF(- ICON2( IPR).NE. J) GOTO 15
```

```
      GOTO 12
   10 IF(IPR.NE. J) GOTO 11
      IF(CABJ* CABJ+ SABJ* SABJ.GT.1.D-8) GOTO 15
      GOTO 13
   11 IF(ICON1( IPR).NE. J) GOTO 15
   12 XI= ABS( CABJ* CAB( IPR)+ SABJ* SAB( IPR)+ SALPJ* SALP( IPR))
      IF(XI.LT.0.999999D+0) GOTO 15
      IF(ABS( BI( IPR)/ B-1.).GT.1.D-6) GOTO 15
   13 IND2=0
      GOTO 16
   14 IND2=1
      GOTO 16
   15 IND2=2
C
C     OBSERVATION LOOP
C
   16 CONTINUE
      IPR=0
      DO 23  I= I1, I2
      IPR= IPR+1
      IJ= I- J
      XI= X( I)
      YI= Y( I)
      ZI= Z( I)
      AI= BI( I)
      CABI= CAB( I)
      SABI= SAB( I)
      SALPI= SALP( I)
      CALL EFLD( XI, YI, ZI, AI, IJ)
      ETK= EXK* CABI+ EYK* SABI+ EZK* SALPI
      ETS= EXS* CABI+ EYS* SABI+ EZS* SALPI
C
C     FILL MATRIX ELEMENTS.  ELEMENT LOCATIONS DETERMINED BY CONNECTION
C     DATA.
C
      ETC= EXC* CABI+ EYC* SABI+ EZC* SALPI
C     NORMAL FILL
      IF(ITRP.NE.0) GOTO 18
      DO 17  IJ=1, JSNO
      JX= JCO( IJ)
   17 CM( IPR, JX)= CM( IPR, JX)+ ETK* AX( IJ)+ ETS* BX( IJ)+ ETC* CX(
     *IJ)
      GOTO 23
C     TRANSPOSED FILL
   18 IF(ITRP.EQ.2) GOTO 20
      DO 19  IJ=1, JSNO
      JX= JCO( IJ)
   19 CM( JX, IPR)= CM( JX, IPR)+ ETK* AX( IJ)+ ETS* BX( IJ)+ ETC* CX(
     *IJ)
```

```
C      TRANS. FILL FOR C(WW) - TEST FOR ELEMENTS FOR D(WW)PRIME.  (=CW)
       GOTO 23
    20 DO 22  IJ=1, JSNO
       JX= JCO( IJ)
       IF(JX.GT. NR) GOTO 21
       CM( JX, IPR)= CM( JX, IPR)+ ETK* AX( IJ)+ ETS* BX( IJ)+ ETC* CX(
      *IJ)
       GOTO 22
    21 JX= JX- NR
       CW( JX, IPR)= CW( JX, IPR)+ ETK* AX( IJ)+ ETS* BX( IJ)+ ETC* CX(
      *IJ)
    22 CONTINUE
    23 CONTINUE
       RETURN
       END
```

CONECT

PURPOSE

To locate segment ends that contact each other or contact the center of a SURFACE patch.

METHOD

The ends of each segment are identified as end 1 and end 2, defined during geometry input. The connection data for segment I is stored in array variables ICON1(I) for end I and ICON2(I) for end 2.

Four conditions are possible at each segment end: (1) no connection (a free end), (2) connection to one or more other segments, (3) connection to a ground plane, or (4) connection to a surface modeled with patches. These conditions are indicated in the following way for end 1 of segment I;

(1) no connection . . . . ICON1(I) = 0

(2) connection to segment J . . . . ICON1(I) = $\pm J$

(3) connection to a ground plane . . . . ICON1(I) = I

(4) connection to patch K . . . . ICON1(I) = 10000+K

In case 2, if segment J has the same reference direction as segment I (end 2 of segment J connected to end 1 of segment I), the sign is positive. For opposed reference directions (end 1 to end 1) the sign is negative. If several segments connect to end 1 of segment I, then J is the number of the next connected segment in sequence.

If segment I connects to patch K, the segment end must coincide with the patch center. Patch K is then divided into four patches numbered K through K+3 by a call to subroutine SUBPH.

The connection data is illustrated in the following listing for the six segments in the structure in figure 3.

ICON1(I)	I	ICON2(I)
10000 + K	1	2
1	2	3
4	3	0
0	4	-5
0	5	6
2	6	0

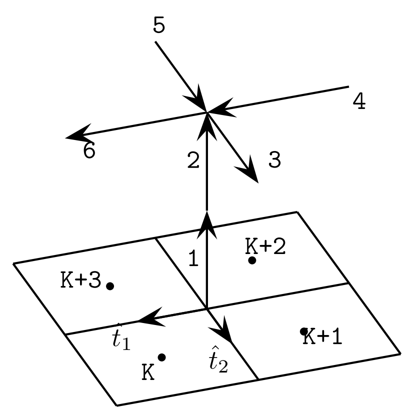


Figure 3. Structure for Illustrating Segment Connection Data.

Connections between patches are not checked, since, except where a wire connects to a surface, the current expansion function on a patch does not extend beyond that patch.

CODING

CN16-CN27	Initialize and adjust symmetry conditions if necessary when ground is present.
CN40 CN46	Check whether end 1 of segment I is below ground plane (error) or contacting ground plane. If the separation of the segment end and the ground is less than SMIN multiplied by the segment length, IOONI is set to I and the z-coordinate of the segment end is set to exactly zero.
CN49-CN60	Check other segments from I+l through N and then 1 through I-1, until a connected end is found. The separation of segment ends is determined by the sum of the separations in x, y, and z to save time.
CN95-CNl26	Search for segments connected to patches. Only new patches (not NGF) are checked. If a connection is found the patch is divided into four patches at its present location in the data arrays and patches following it are shifted up by three locations. This is done by calling SUBPH, an entry point of subroutine PATCH.
CN129-CN162	Search for new segments connected to NGF patches. If a connection is found, four patches covering the area of the original patch, are added to the end of the data arrays by calling SUBPH. The original patch retains its location but the z-coordinate at its center is changed to 10000.
CN182-CN258	The loop through he locates segments connected to junctions.
CN183-CN190	Parameters are initialized to find all segments connected to first end of segment J.
CN191-CN215	Connected segments are located. If the number of any connected segment is less than J the loop is exited at CN200. Thus each junction is processed only once.
CN216-CN230	The connected ends are set to the average of their previous values to ensure that they have identical values.
CN232-CN244	If the junction includes new segments (NSFLG = 1) and IX is a NGF segment an equation number, NSCON, is assigned for the modified basis function of segment IX. The equation number is stored in array ICONX and the segment number is stared in ISCON.
CN245-CN247	Segment numbers are printed for junctions of three or more segments.
CN248-CN257	The loop is initialized for the second end of segment J and the steps from CN191 on are repeated.
CN262-CN275	Equation numbers for modified basis functions are assigned for old segments that connect to new patches.

SYMBOL DICTIONARY

```
    IGND      =  1 to adjust symmetry for ground and set ICON(I)=I;
                 -1 to adjust symmetry only;
                 0 for no ground
    JMAX      =  maximum number of segments connected to a junction
    NPMAX     =  maximum number of NGF patches connecting to new segments
    NSFLG     =  1 if the junction includes any new segments when NGF
                 is in use
    NSMAX     =  maximum number of NGF segments connecting to new segments
    SEP       =  approximate separation of segment ends
    SLEN      =  maximum separation allowed for connection
    SMIN      =  maximum separation as a fraction of segment length
    XI1
    YI1       =  coordinates of end 1 of segment
    ZI1
    XI2
    YI2       =  coordinates of end Z of segment
    ZI2
    XS
    YS        =  coordinates of patch center
    ZS

    CONSTANT

    1.E-3     =  maximum separation tolerance for connected segments
                 as fraction of segment length.
```

```
      SUBROUTINE CONECT(IGND)
C
C     CONNECT SETS UP SEGMENT CONNECTION DATA IN ARRAYS ICON1 AND ICON2
C     BY SEARCHING FOR SEGMENT ENDS THAT ARE IN CONTACT.
C
      COMMON/DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X( NM), Y( NM),
     *Z( NM), SI( NM), BI( NM), ALP( NM), BET( NM), ICON1( N2M), ICON2(
     * N2M), ITAG( N2M), ICONX( NM), WLAM, IPSYM
      COMMON/SEGJ/ AX(30), BX(30), CX(30), JCO(30), JSNO, ISCON(50),
     *NSCON,IPCON(10), NPCON
      DIMENSION  X2(1), Y2(1), Z2(1)
      EQUIVALENCE(X2,SI),(Y2,ALP),(Z2,BET)
      DATA  JMAX/30/, SMIN/1.D-3/, NSMAX/50/, NPMAX/10/
      NSCON=0
      NPCON=0
      IF(IGND.EQ.0) GOTO 3
      WRITE (2,54)
      IF(IGND.GT.0) WRITE (2,55)
      IF(IPSYM.NE.2) GOTO 1
      NP=2*NP
      MP=2*MP
    1 IF(IABS( IPSYM).LE.2) GOTO 2
      NP=N
      MP=M
    2 IF(NP.GT. N) STOP
      IF(NP.EQ. N.AND. MP.EQ. M) IPSYM=0
    3 IF(N.EQ.0) GOTO 26
      DO 15  I=1, N
      ICONX( I)=0
      XI1=X( I)
      YI1=Y( I)
      ZI1=Z( I)
      XI2=X2( I)
      YI2=Y2( I)
      ZI2=Z2( I)
C
C     DETERMINE CONNECTION DATA FOR END 1 OF SEGMENT.
C
      SLEN=SQRT(( XI2- XI1)**2+( YI2- YI1)**2+( ZI2- ZI1)**2)* SMIN
      IF(IGND.LT.1) GOTO 5
      IF(ZI1.GT.- SLEN) GOTO 4
      WRITE (2,56)  I
      STOP
    4 IF(ZI1.GT. SLEN) GOTO 5
      ICON1( I)=I
      Z( I)=0.
      GOTO 9
    5 IC=I
      DO 7  J=2, N
```

```
      IC=IC+1
      IF(IC.GT. N) IC=1
      SEP=ABS( XI1- X( IC))+ ABS( YI1- Y( IC))+ ABS( ZI1- Z( IC))
      IF(SEP.GT. SLEN) GOTO 6
      ICON1( I)=- IC
      GOTO 8
    6 SEP=ABS( XI1- X2( IC))+ ABS( YI1- Y2( IC))+ ABS( ZI1- Z2( IC))
      IF(SEP.GT. SLEN) GOTO 7
      ICON1( I)=IC
      GOTO 8
    7 CONTINUE
      IF(I.LT. N2.AND. ICON1( I).GT.10000) GOTO 8
C
C     DETERMINE CONNECTION DATA FOR END 2 OF SEGMENT.
C
      ICON1( I)=0
    8 IF(IGND.LT.1) GOTO 12
    9 IF(ZI2.GT.- SLEN) GOTO 10
      WRITE (2,56)  I
      STOP
   10 IF(ZI2.GT. SLEN) GOTO 12
      IF(ICON1( I).NE. I) GOTO 11
      WRITE (2,57)  I
      STOP
   11 ICON2( I)=I
      Z2( I)=0.
      GOTO 15
   12 IC=I
      DO 14  J=2, N
      IC=IC+1
      IF(IC.GT. N) IC=1
      SEP=ABS( XI2- X( IC))+ ABS( YI2- Y( IC))+ ABS( ZI2- Z( IC))
      IF(SEP.GT. SLEN) GOTO 13
      ICON2( I)=IC
      GOTO 15
   13 SEP=ABS( XI2- X2( IC))+ ABS( YI2- Y2( IC))+ ABS( ZI2- Z2( IC))
      IF(SEP.GT. SLEN) GOTO 14
      ICON2( I)=- IC
      GOTO 15
   14 CONTINUE
      IF(I.LT. N2.AND. ICON2( I).GT.10000) GOTO 15
      ICON2( I)=0
   15 CONTINUE
C     FIND WIRE-SURFACE CONNECTIONS FOR NEW PATCHES
      IF(M.EQ.0) GOTO 26
      IX=LD+1- M1
      I=M2
   16 IF(I.GT. M) GOTO 20
      IX=IX-1
```

```
      XS=X( IX)
      YS=Y( IX)
      ZS=Z( IX)
      DO 18  ISEG=1, N
      XI1=X( ISEG)
      YI1=Y( ISEG)
      ZI1=Z( ISEG)
      XI2=X2( ISEG)
      YI2=Y2( ISEG)
      ZI2=Z2( ISEG)
C     FOR FIRST END OF SEGMENT
      SLEN=( ABS( XI2- XI1)+ ABS( YI2- YI1)+ ABS( ZI2- ZI1))* SMIN
      SEP=ABS( XI1- XS)+ ABS( YI1- YS)+ ABS( ZI1- ZS)
C     CONNECTION - DIVIDE PATCH INTO 4 PATCHES AT PRESENT ARRAY LOC.
      IF(SEP.GT. SLEN) GOTO 17
      ICON1( ISEG)=10000+ I
      IC=0
      CALL SUBPH( I, IC, XI1, YI1, ZI1, XI2, YI2, ZI2, XA, YA, ZA, XS,
     *YS, ZS)
      GOTO 19
   17 SEP=ABS( XI2- XS)+ ABS( YI2- YS)+ ABS( ZI2- ZS)
      IF(SEP.GT. SLEN) GOTO 18
      ICON2( ISEG)=10000+ I
      IC=0
      CALL SUBPH( I, IC, XI1, YI1, ZI1, XI2, YI2, ZI2, XA, YA, ZA, XS,
     *YS, ZS)
      GOTO 19
   18 CONTINUE
   19 I=I+1
C     REPEAT SEARCH FOR NEW SEGMENTS CONNECTED TO NGF PATCHES.
      GOTO 16
   20 IF(M1.EQ.0.OR. N2.GT. N) GOTO 26
      IX=LD+1
      I=1
   21 IF(I.GT. M1) GOTO 25
      IX=IX-1
      XS=X( IX)
      YS=Y( IX)
      ZS=Z( IX)
      DO 23  ISEG=N2, N
      XI1=X( ISEG)
      YI1=Y( ISEG)
      ZI1=Z( ISEG)
      XI2=X2( ISEG)
      YI2=Y2( ISEG)
      ZI2=Z2( ISEG)
      SLEN=( ABS( XI2- XI1)+ ABS( YI2- YI1)+ ABS( ZI2- ZI1))* SMIN
      SEP=ABS( XI1- XS)+ ABS( YI1- YS)+ ABS( ZI1- ZS)
      IF(SEP.GT. SLEN) GOTO 22
```

```
      ICON1( ISEG)=10001+ M
      IC=1
      NPCON=NPCON+1
      IPCON( NPCON)=I
      CALL SUBPH( I, IC, XI1, YI1, ZI1, XI2, YI2, ZI2, XA, YA, ZA, XS,
     *YS, ZS)
      GOTO 24
   22 SEP=ABS( XI2- XS)+ ABS( YI2- YS)+ ABS( ZI2- ZS)
      IF(SEP.GT. SLEN) GOTO 23
      ICON2( ISEG)=10001+ M
      IC=1
      NPCON=NPCON+1
      IPCON( NPCON)=I
      CALL SUBPH( I, IC, XI1, YI1, ZI1, XI2, YI2, ZI2, XA, YA, ZA, XS,
     *YS, ZS)
      GOTO 24
   23 CONTINUE
   24 I=I+1
      GOTO 21
   25 IF(NPCON.LE. NPMAX) GOTO 26
      WRITE (2,62)  NPMAX
      STOP
   26 WRITE (2,58)  N, NP, IPSYM
      IF(M.GT.0) WRITE (2,61)  M, MP
      ISEG=( N+ M)/( NP+ MP)
      IF(ISEG.EQ.1) GOTO 30
      IF(IPSYM) 28,27,29
   27 STOP
   28 WRITE (2,59)  ISEG
      GOTO 30
   29 IC=ISEG/2
      IF(ISEG.EQ.8) IC=3
      WRITE (2,60)  IC
   30 IF(N.EQ.0) GOTO 48
      WRITE (2,50)
C     ADJUST CONNECTED SEG. ENDS TO EXACTLY COINCIDE.  PRINT JUNCTIONS
C     OF 3 OR MORE SEG.  ALSO FIND OLD SEG. CONNECTING TO NEW SEG.
      ISEG=0
      DO 44  J=1, N
      IEND=-1
      JEND=-1
      IX=ICON1( J)
      IC=1
      JCO(1)=- J
      XA=X( J)
      YA=Y( J)
      ZA=Z( J)
   31 IF(IX.EQ.0) GOTO 43
      IF(IX.EQ. J) GOTO 43
```

```
      IF(IX.GT.10000) GOTO 43                                           CN 197
      NSFLG=0                                                           CN 198
   32 IF(IX) 33,49,34                                                   CN 199
   33 IX=- IX                                                           CN 200
      GOTO 35                                                           CN 201
   34 JEND=- JEND                                                       CN 202
   35 IF(IX.EQ. J) GOTO 37                                              CN 203
      IF(IX.LT. J) GOTO 43                                              CN 204
      IC=IC+1                                                           CN 205
      IF(IC.GT. JMAX) GOTO 49                                           CN 206
      JCO( IC)=IX* JEND                                                 CN 207
      IF(IX.GT. N1) NSFLG=1                                             CN 208
      IF(JEND.EQ.1) GOTO 36                                             CN 209
      XA=XA+ X( IX)                                                     CN 210
      YA=YA+ Y( IX)                                                     CN 211
      ZA=ZA+ Z( IX)                                                     CN 212
      IX=ICON1( IX)                                                     CN 213
      GOTO 32                                                           CN 214
   36 XA=XA+ X2( IX)                                                    CN 215
      YA=YA+ Y2( IX)                                                    CN 216
      ZA=ZA+ Z2( IX)                                                    CN 217
      IX=ICON2( IX)                                                     CN 218
      GOTO 32                                                           CN 219
   37 SEP=IC                                                            CN 220
      XA=XA/ SEP                                                        CN 221
      YA=YA/ SEP                                                        CN 222
      ZA=ZA/ SEP                                                        CN 223
      DO 39  I=1, IC                                                    CN 224
      IX=JCO( I)                                                        CN 225
      IF(IX.GT.0) GOTO 38                                               CN 226
      IX=- IX                                                           CN 227
      X( IX)=XA                                                         CN 228
      Y( IX)=YA                                                         CN 229
      Z( IX)=ZA                                                         CN 230
      GOTO 39                                                           CN 231
   38 X2( IX)=XA                                                        CN 232
      Y2( IX)=YA                                                        CN 233
      Z2( IX)=ZA                                                        CN 234
   39 CONTINUE                                                          CN 235
      IF(N1.EQ.0) GOTO 42                                               CN 236
      IF(NSFLG.EQ.0) GOTO 42                                            CN 237
      DO 41  I=1, IC                                                    CN 238
      IX=IABS( JCO( I))                                                 CN 239
      IF(IX.GT. N1) GOTO 41                                             CN 240
      IF(ICONX( IX).NE.0) GOTO 41                                       CN 241
      NSCON=NSCON+1                                                     CN 242
      IF(NSCON.LE. NSMAX) GOTO 40                                       CN 243
      WRITE (2,62)  NSMAX                                               CN 244
      STOP                                                              CN 245
```

```
   40 ISCON( NSCON)=IX                                                    CN 246
      ICONX( IX)=NSCON                                                    CN 247
   41 CONTINUE                                                            CN 248
   42 IF(IC.LT.3) GOTO 43                                                 CN 249
      ISEG=ISEG+1                                                         CN 250
      WRITE (2,51)  ISEG,( JCO( I), I=1, IC)                              CN 251
   43 IF(IEND.EQ.1) GOTO 44                                               CN 252
      IEND=1                                                              CN 253
      JEND=1                                                              CN 254
      IX=ICON2( J)                                                        CN 255
      IC=1                                                                CN 256
      JCO(1)=J                                                            CN 257
      XA=X2( J)                                                           CN 258
      YA=Y2( J)                                                           CN 259
      ZA=Z2( J)                                                           CN 260
      GOTO 31                                                             CN 261
   44 CONTINUE                                                            CN 262
      IF(ISEG.EQ.0) WRITE (2,52)                                          CN 263
C     FIND OLD SEGMENTS THAT CONNECT TO NEW PATCHES                       CN 264
      IF(N1.EQ.0.OR. M1.EQ. M) GOTO 48                                    CN 265
      DO 47  J=1, N1                                                      CN 266
      IX=ICON1( J)                                                        CN 267
      IF(IX.LT.10000) GOTO 45                                             CN 268
      IX=IX-10000                                                         CN 269
      IF(IX.GT. M1) GOTO 46                                               CN 270
   45 IX=ICON2( J)                                                        CN 271
      IF(IX.LT.10000) GOTO 47                                             CN 272
      IX=IX-10000                                                         CN 273
      IF(IX.LT. M2) GOTO 47                                               CN 274
   46 IF(ICONX( J).NE.0) GOTO 47                                          CN 275
      NSCON=NSCON+1                                                       CN 276
      ISCON( NSCON)=J                                                     CN 277
      ICONX( J)=NSCON                                                     CN 278
   47 CONTINUE                                                            CN 279
   48 CONTINUE                                                            CN 280
      RETURN                                                              CN 281
   49 WRITE (2,53)  IX                                                    CN 282
C                                                                         CN 283
      STOP                                                                CN 284
   50 FORMAT(//,9X,'- MULTIPLE WIRE JUNCTIONS -',/,1X,'JUNCTION',4X,      CN 285
     *'SEGMENTS  (- FOR END 1, + FOR END 2)')                             CN 286
   51 FORMAT(1X,I5,5X,20I5,/,(11X,20I5))                                  CN 287
   52 FORMAT(2X,'NONE')                                                   CN 288
   53 FORMAT(' CONNECT - SEGMENT CONNECTION ERROR FOR SEGMENT',I5)        CN 289
   54 FORMAT(/,3X,'GROUND PLANE SPECIFIED.')                              CN 290
   55 FORMAT(/,3X,'WHERE WIRE ENDS TOUCH GROUND, CURRENT WILL BE ',       CN 291
     *'INTERPOLATED TO IMAGE IN GROUND PLANE.',/)                         CN 292
   56 FORMAT(' GEOMETRY DATA ERROR-- SEGMENT',I5,' EXTENDS BELOW GRO',    CN 293
     *'UND')                                                              CN 294
```

```
57 FORMAT(' GEOMETRY DATA ERROR--SEGMENT',I5,' LIES IN GROUND ',      CN 295
  *'PLANE.')                                                          CN 296
58 FORMAT(/,3X,'TOTAL SEGMENTS USED=',I5,5X,'NO. SEG. IN ','A SY',    CN 297
  *'MMETRIC CELL=',I5,5X,'SYMMETRY FLAG=',I3)                         CN 298
59 FORMAT(' STRUCTURE HAS',I4,' FOLD ROTATIONAL SYMMETRY',/)          CN 299
60 FORMAT(' STRUCTURE HAS',I2,' PLANES OF SYMMETRY',/)                CN 300
61 FORMAT(3X,'TOTAL PATCHES USED=',I5,6X,'NO. PATCHES IN A SYMMET',   CN 301
  *'RIC CELL=',I5)                                                    CN 302
62 FORMAT(' ERROR - NO. NEW SEGMENTS CONNECTED TO N.G.F. SEGMENTS',   CN 303
  *'OR PATCHES EXCEEDS LIMIT OF',I5)                                  CN 304
   END                                                                CN 305
```

COUPLE

PURPOSE

To compute the maximum coupling between pairs of segments.

METHOD

If a coupling calculation has been requested (CP card) subroutine COUPLE is called each time that the current is computed for a new excitation. The code from CP10 to CP12 checks that the excitation is a single applied-field voltage source on the segment specified in NCTAG and NCSEG. If the excitation is correct the input admittance and mutual admittances to all other segments specified in NCTAG and NCSEG are stored in Y11A and Y12A from CP13 to CP22.

When all segments have been excited (ICOUP = NCOUP) the second part of the code, from CP24 to CP58 is executed to evaluate the equations in section V.6 of Part I.

SYMBOL DICTIONARY

C	=	L (see part I, section V.6)
CUR	=	array of values of current at the centers of segments
DBC	=	$10log(G_{MAX})$
GMAX	=	G_{MAX}
ISG1	=	segment number
ISG2	=	segment number
J1	=	index of Y_{12} in array Y12A
J2	=	index of Y_{21} in array Y12A
K	=	segment number
RHO	=	ρ
WLAM	=	wavelength
Y11	=	Y_{11}
Y12	=	$(Y_{12} + Y_{21})/2$
Y22	=	Y22
YIN	=	YIN
YL	=	YL
ZIN	=	$1/Y_{IN}$
ZL	=	$1/Y_L$

```
      SUBROUTINE COUPLE( CUR, WLAM)
C
C     COUPLE COMPUTES THE MAXIMUM COUPLING BETWEEN PAIRS OF SEGMENTS.
C
      COMPLEX  Y11A, Y12A, CUR, Y11, Y12, Y22, YL, YIN, ZL, ZIN, RHO
     *, VQD, VSANT, VQDS
      COMMON  /YPARM/ NCOUP, ICOUP, NCTAG(5), NCSEG(5), Y11A(5), Y12A(
     *20)
      COMMON  /VSORC/ VQD(30), VSANT(30), VQDS(30), IVQD(30), ISANT(30)
     *, IQDS(30), NVQD, NSANT, NQDS
      DIMENSION  CUR(1)
      IF(NSANT.NE.1.OR. NVQD.NE.0) RETURN
      J= ISEGNO( NCTAG( ICOUP+1), NCSEG( ICOUP+1))
      IF(J.NE. ISANT(1)) RETURN
      ICOUP= ICOUP+1
      ZIN= VSANT(1)
      Y11A( ICOUP)= CUR( J)* WLAM/ ZIN
      L1=( ICOUP-1)*( NCOUP-1)
      DO 1  I=1, NCOUP
      IF(I.EQ. ICOUP) GOTO 1
      K= ISEGNO( NCTAG( I), NCSEG( I))
      L1= L1+1
      Y12A( L1)= CUR( K)* WLAM/ ZIN
    1 CONTINUE
      IF(ICOUP.LT. NCOUP) RETURN
      WRITE (2,6)
      NPM1= NCOUP-1
      DO 5  I=1, NPM1
      ITT1= NCTAG( I)
      ITS1= NCSEG( I)
      ISG1= ISEGNO( ITT1, ITS1)
      L1= I+1
      DO 5  J= L1, NCOUP
      ITT2= NCTAG( J)
      ITS2= NCSEG( J)
      ISG2= ISEGNO( ITT2, ITS2)
      J1= J+( I-1)* NPM1-1
      J2= I+( J-1)* NPM1
      Y11= Y11A( I)
      Y22= Y11A( J)
      Y12=.5*( Y12A( J1)+ Y12A( J2))
      YIN= Y12* Y12
      DBC= ABS( YIN)
      C= DBC/(2.* REAL( Y11)* REAL( Y22)- REAL( YIN))
      IF(C.LT.0..OR. C.GT.1.) GOTO 4
      IF(C.LT..01) GOTO 2
      GMAX=(1.- SQRT(1.- C* C))/ C
      GOTO 3
    2 GMAX=.5*( C+.25* C* C* C)
```

```
    3 RHO= GMAX* CONJG( YIN)/ DBC
      YL=((1.- RHO)/(1.+ RHO)+1.)* REAL( Y22)- Y22
      ZL=1./ YL
      YIN= Y11- YIN/( Y22+ YL)
      ZIN=1./ YIN
      DBC= DB10( GMAX)
      WRITE (2,7)  ITT1, ITS1, ISG1, ITT2, ITS2, ISG2, DBC, ZL, ZIN
      GOTO 5
    4 WRITE (2,8)  ITT1, ITS1, ISG1, ITT2, ITS2, ISG2, C
    5 CONTINUE
C
      RETURN
    6 FORMAT(///,36X,'- - - ISOLATION DATA - - -',//,6X,'- - COUPLIN',
     *'G BETWEEN - -',8X,'MAXIMUM',15X,'- - - FOR MAXIMUM COUPLING - ',
     *'- -',/,12X,'SEG.',14X,'SEG.',3X,'COUPLING',4X,'LOAD IMPEDANCE ',
     *'(2ND SEG.)',7X,'INPUT IMPEDANCE',/,2X,'TAG/SEG.',3X,'NO.',4X,
     *'TAG/''SEG.',3X,'NO.',6X,'(DB)',8X,'REAL',9X,'IMAG.',9X,'REAL',9X
     *,'IMAG.')
    7 FORMAT(2(1X,I4,1X,I4,1X,I5,2X),F9.3,2X,1P,2(2X,E12.5,1X,E12.5))
    8 FORMAT(2(1X,I4,1X,I4,1X,I5,2X),'**ERROR** COUPLING IS NOT BETWE',
     *'EN 0 AND 1. (=',1P,E12.5,')')
      END
```

DATAGN

PURPOSE

To read structure input data and set segment and patch data.

METHOD

The main READ statement is at DA35. The READ statement at DA65 is for the continuation of wire data (GC card following GW), and the use at DA133 is for the continuation of surface patch data (SC following SP or SM).

The first input parameter GM determines the function of the card as indicated in the following table.

GM	GO TO	FUNCTION
GA	8	define wire arc
GC	6	continuation of wire data
GE	29	end of geometry data
GF	27	read NGF file
GM	26	rotate or translate structure
GR	19	rotate about z-axis (symmetry)
GS	21	scale structure
GW	3	define straight wire
GX	18	reflect in coordinate planes (symmetry)
SC	10	continuation of patch data
SM	13	define multiple surface patches
SP	9	define surface patch

The functions of the other input parameters depend on the type of data card and can be determined from the data card descriptions in Part III of this manual.

Subroutines are called to perform many of the operations requested by the data cards. Coding in DATAGN performs other operations, prints information and checks for input errors. After a GE card is read subroutine CONECT is called at DA211 to find electrical connections of segments. Segment and patch data is printed from DA217 to DA256. Line DA241 tests for segments of zero length ($< 10^{-20}$) or zero radius ($< 10^{-101}$).

SYMBOL DICTIONARY

Variables have multiple uses which depend an the type of input card being processed.

```
      SUBROUTINE DATAGN
C
C     DATAGN IS THE MAIN ROUTINE FOR INPUT OF GEOMETRY DATA.
C
      CHARACTER *2  GM, ATST
      CHARACTER *1 IFX,IFY,IFZ,IPT
      COMMON/DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X( NM), Y( NM),
     *Z( NM), SI( NM), BI( NM), ALP( NM), BET( NM), ICON1( N2M), ICON2(
     * N2M), ITAG( N2M), ICONX( NM), WLAM, IPSYM
      COMMON/ANGL/ SALP( NM)
      COMMON  /PLOT/ IPLP1, IPLP2, IPLP3, IPLP4
      DIMENSION  X2(1), Y2(1), Z2(1), T1X(1), T1Y(1), T1Z(1), T2X(1),
     *T2Y(1), T2Z(1), ATST(13), IFX(2), IFY(2), IFZ(2), CAB(1), SAB(1),
     * IPT(4)
      EQUIVALENCE(T1X,SI),(T1Y,ALP),(T1Z,BET),(T2X,ICON1),(T2Y,ICON2),(
     *T2Z,ITAG),(X2,SI),(Y2,ALP),(Z2,BET),(CAB,ALP),(SAB,BET)
      DATA ATST/'GW','GX','GR','GS','GE','GM','SP','SM','GF','GA',
     *          'SC','GC','GH'/
*      DATA   ATST/2HGW,2HGX,2HGR,2HGS,2HGE,2HGM,2HSP,2HSM,2HGF,2HGA,
*     *2HSC,2HGC,2HGH/
      DATA   IFX/1H ,1HX/, IFY/1H ,1HY/, IFZ/1H ,1HZ/
      DATA   TA/0.01745329252D+0/, TD/57.29577951D+0/, IPT/1HP,1HR,1HT,
     *1HQ/
      IPSYM=0
      NWIRE=0
      N=0
      NP=0
      M=0
      MP=0
      N1=0
      N2=1
      M1=0
      M2=1
      ISCT=0
C
C     READ GEOMETRY DATA CARD AND BRANCH TO SECTION FOR OPERATION
C     REQUESTED
C
C***
C 1     READ (5,42) GM,ITG,NS,XW1,YW1,ZW1,XW2,YW2,ZW2,RAD
      IPHD=0
    1 CALL READGM( GM, ITG, NS, XW1, YW1, ZW1, XW2, YW2, ZW2, RAD)
      IF(N+ M.GT. LD) GOTO 37
      IF(GM.EQ. ATST(9)) GOTO 27
      IF(IPHD.EQ.1) GOTO 2
      WRITE (2,40)
      WRITE (2,41)
      IPHD=1
    2 IF(GM.EQ. ATST(11)) GOTO 10
```

```
      ISCT=0
      IF(GM.EQ. ATST(1)) GOTO 3
      IF(GM.EQ. ATST(2)) GOTO 18
      IF(GM.EQ. ATST(3)) GOTO 19
      IF(GM.EQ. ATST(4)) GOTO 21
      IF(GM.EQ. ATST(7)) GOTO 9
      IF(GM.EQ. ATST(8)) GOTO 13
      IF(GM.EQ. ATST(5)) GOTO 29
      IF(GM.EQ. ATST(6)) GOTO 26
      IF(GM.EQ. ATST(10)) GOTO 8
      IF(GM.EQ. ATST(13)) GOTO 123
C
C     GENERATE SEGMENT DATA FOR STRAIGHT WIRE.
      GOTO 36
    3 NWIRE= NWIRE+1
      I1= N+1
      I2= N+ NS
      WRITE (2,43)  NWIRE, XW1, YW1, ZW1, XW2, YW2, ZW2, RAD, NS, I1,
     *I2, ITG
      IF(RAD.EQ.0) GOTO 4
      XS1=1.
      YS1=1.
      GOTO 7
C 4     READ (5,42) GM,IX,IY,XS1,YS1,ZS1
    4 CALL READGM( GM, IX, IY, XS1, YS1, ZS1, DUMMY, DUMMY, DUMMY,
     *DUMMY)
      IF(GM.EQ. ATST(12)) GOTO 6
    5 WRITE (2,48)
      STOP
    6 WRITE (2,61)  XS1, YS1, ZS1
      IF(YS1.EQ.0.OR. ZS1.EQ.0) GOTO 5
      RAD= YS1
      YS1=( ZS1/ YS1)**(1./( NS-1.))
    7 CALL WIRE( XW1, YW1, ZW1, XW2, YW2, ZW2, RAD, XS1, YS1, NS, ITG)
C
C     GENERATE SEGMENT DATA FOR WIRE ARC
C
      GOTO 1
    8 NWIRE= NWIRE+1
      I1= N+1
      I2= N+ NS
      WRITE (2,38)  NWIRE, XW1, YW1, ZW1, XW2, NS, I1, I2, ITG
      CALL ARC( ITG, NS, XW1, YW1, ZW1, XW2)
C     GENERATE HELIX
      GOTO 1
  123 NWIRE= NWIRE+1
      I1= N+1
      I2= N+ NS
      WRITE (2,124)  XW1, YW1, NWIRE, ZW1, XW2, YW2, ZW2, RAD, NS, I1,
```

```
     *I2, ITG                                                           DA  99
      CALL HELIX( XW1, YW1, ZW1, XW2, YW2, ZW2, RAD, NS, ITG)           DA 100
      GOTO 1                                                            DA 101
C                                                                       DA 102
C     GENERATE SINGLE NEW PATCH                                         DA 103
C                                                                       DA 104
  124 FORMAT(5X,'HELIX STRUCTURE-   AXIAL SPACING BETWEEN TURNS =',F8.3 DA 105
     *,' TOTAL AXIAL LENGTH =',F8.3/1X,I5,2X,'RADIUS OF HELIX =',4(2X,F DA 106
     *8.3),7X,F11.5,I8,4X,I5,1X,I5,3X,I5)                               DA 107
    9 I1= M+1                                                           DA 108
      NS= NS+1                                                          DA 109
      IF(ITG.NE.0) GOTO 17                                              DA 110
      WRITE (2,51)  I1, IPT( NS), XW1, YW1, ZW1, XW2, YW2, ZW2          DA 111
      IF(NS.EQ.2.OR. NS.EQ.4) ISCT=1                                    DA 112
      IF(NS.GT.1) GOTO 14                                               DA 113
      XW2= XW2* TA                                                      DA 114
      YW2= YW2* TA                                                      DA 115
      GOTO 16                                                           DA 116
   10 IF(ISCT.EQ.0) GOTO 17                                             DA 117
      I1= M+1                                                           DA 118
      NS= NS+1                                                          DA 119
      IF(ITG.NE.0) GOTO 17                                              DA 120
      IF(NS.NE.2.AND. NS.NE.4) GOTO 17                                  DA 121
      XS1= X4                                                           DA 122
      YS1= Y4                                                           DA 123
      ZS1= Z4                                                           DA 124
      XS2= X3                                                           DA 125
      YS2= Y3                                                           DA 126
      ZS2= Z3                                                           DA 127
      X3= XW1                                                           DA 128
      Y3= YW1                                                           DA 129
      Z3= ZW1                                                           DA 130
      IF(NS.NE.4) GOTO 11                                               DA 131
      X4= XW2                                                           DA 132
      Y4= YW2                                                           DA 133
      Z4= ZW2                                                           DA 134
   11 XW1= XS1                                                          DA 135
      YW1= YS1                                                          DA 136
      ZW1= ZS1                                                          DA 137
      XW2= XS2                                                          DA 138
      YW2= YS2                                                          DA 139
      ZW2= ZS2                                                          DA 140
      IF(NS.EQ.4) GOTO 12                                               DA 141
      X4= XW1+ X3- XW2                                                  DA 142
      Y4= YW1+ Y3- YW2                                                  DA 143
      Z4= ZW1+ Z3- ZW2                                                  DA 144
   12 WRITE (2,51)  I1, IPT( NS), XW1, YW1, ZW1, XW2, YW2, ZW2          DA 145
      WRITE (2,39)  X3, Y3, Z3, X4, Y4, Z4                              DA 146
C                                                                       DA 147
```

```
C     GENERATE MULTIPLE-PATCH SURFACE                                   DA 148
C                                                                       DA 149
      GOTO 16                                                           DA 150
   13 I1= M+1                                                           DA 151
      WRITE (2,59)  I1, IPT(2), XW1, YW1, ZW1, XW2, YW2, ZW2, ITG, NS    DA 152
      IF(ITG.LT.1.OR. NS.LT.1) GOTO 17                                  DA 153
C 14    READ (5,42) GM,IX,IY,X3,Y3,Z3,X4,Y4,Z4                           DA 154
   14 CALL READGM( GM, IX, IY, X3, Y3, Z3, X4, Y4, Z4, DUMMY)           DA 155
      IF(NS.NE.2.AND. ITG.LT.1) GOTO 15                                 DA 156
      X4= XW1+ X3- XW2                                                  DA 157
      Y4= YW1+ Y3- YW2                                                  DA 158
      Z4= ZW1+ Z3- ZW2                                                  DA 159
   15 WRITE (2,39)  X3, Y3, Z3, X4, Y4, Z4                              DA 160
      IF(GM.NE. ATST(11)) GOTO 17                                       DA 161
   16 CALL PATCH( ITG, NS, XW1, YW1, ZW1, XW2, YW2, ZW2, X3, Y3, Z3, X4  DA 162
     *, Y4, Z4)                                                         DA 163
      GOTO 1                                                            DA 164
   17 WRITE (2,60)                                                      DA 165
C                                                                       DA 166
C     REFLECT STRUCTURE ALONG X,Y, OR Z AXES OR ROTATE TO FORM CYLINDER. DA 167
C                                                                       DA 168
      STOP                                                              DA 169
   18 IY= NS/10                                                         DA 170
      IZ= NS- IY*10                                                     DA 171
      IX= IY/10                                                         DA 172
      IY= IY- IX*10                                                     DA 173
      IF(IX.NE.0) IX=1                                                  DA 174
      IF(IY.NE.0) IY=1                                                  DA 175
      IF(IZ.NE.0) IZ=1                                                  DA 176
      WRITE (2,44)  IFX( IX+1), IFY( IY+1), IFZ( IZ+1), ITG              DA 177
      GOTO 20                                                           DA 178
   19 WRITE (2,45)  NS, ITG                                             DA 179
      IX=-1                                                             DA 180
   20 CALL REFLC( IX, IY, IZ, ITG, NS)                                  DA 181
C                                                                       DA 182
C     SCALE STRUCTURE DIMENSIONS BY FACTOR XW1.                         DA 183
C                                                                       DA 184
      GOTO 1                                                            DA 185
   21 IF(N.LT. N2) GOTO 23                                              DA 186
      DO 22  I= N2, N                                                   DA 187
      X( I)= X( I)* XW1                                                 DA 188
      Y( I)= Y( I)* XW1                                                 DA 189
      Z( I)= Z( I)* XW1                                                 DA 190
      X2( I)= X2( I)* XW1                                               DA 191
      Y2( I)= Y2( I)* XW1                                               DA 192
      Z2( I)= Z2( I)* XW1                                               DA 193
   22 BI( I)= BI( I)* XW1                                               DA 194
   23 IF(M.LT. M2) GOTO 25                                              DA 195
      YW1= XW1* XW1                                                     DA 196
```

```
      IX= LD+1- M
      IY= LD- M1
      DO 24  I= IX, IY
      X( I)= X( I)* XW1
      Y( I)= Y( I)* XW1
      Z( I)= Z( I)* XW1
   24 BI( I)= BI( I)* YW1
   25 WRITE (2,46)  XW1
C
C     MOVE STRUCTURE OR REPRODUCE ORIGINAL STRUCTURE IN NEW POSITIONS.
C
      GOTO 1
   26 WRITE (2,47)  ITG, NS, XW1, YW1, ZW1, XW2, YW2, ZW2, RAD
      XW1= XW1* TA
      YW1= YW1* TA
      ZW1= ZW1* TA
      CALL MOVE( XW1, YW1, ZW1, XW2, YW2, ZW2, INT( RAD+.5), NS, ITG)
C
C     READ NUMERICAL GREEN'S FUNCTION TAPE
C
      GOTO 1
   27 IF(N+ M.EQ.0) GOTO 28
      WRITE (2,52)
      STOP
   28 CALL GFIL( ITG)
      NPSAV= NP
      MPSAV= MP
      IPSAV= IPSYM
C
C     TERMINATE STRUCTURE GEOMETRY INPUT.
C
      GOTO 1
   29 IF(NS.EQ.0) GOTO 290
      IPLP1=1
      IPLP2=1
  290 IX= N1+ M1
      IF(IX.EQ.0) GOTO 30
      NP= N
      MP= M
      IPSYM=0
   30 CALL CONECT( ITG)
      IF(IX.EQ.0) GOTO 31
      NP= NPSAV
      MP= MPSAV
      IPSYM= IPSAV
   31 IF(N+ M.GT. LD) GOTO 37
      IF(N.EQ.0) GOTO 33
```

```
      WRITE (2,53)
      WRITE (2,54)
      DO 32  I=1, N
      XW1= X2( I)- X( I)
      YW1= Y2( I)- Y( I)
      ZW1= Z2( I)- Z( I)
      X( I)=( X( I)+ X2( I))*.5
      Y( I)=( Y( I)+ Y2( I))*.5
      Z( I)=( Z( I)+ Z2( I))*.5
      XW2= XW1* XW1+ YW1* YW1+ ZW1* ZW1
      YW2= SQRT( XW2)
      YW2=( XW2/ YW2+ YW2)*.5
      SI( I)= YW2
      CAB( I)= XW1/ YW2
      SAB( I)= YW1/ YW2
      XW2= ZW1/ YW2
      IF(XW2.GT.1.) XW2=1.
      IF(XW2.LT.-1.) XW2=-1.
      SALP( I)= XW2
      XW2= ASIN( XW2)* TD
      YW2= ATGN2( YW1, XW1)* TD

      WRITE (2,55)  I, X( I), Y( I), Z( I), SI( I), XW2, YW2, BI( I),
     *ICON1( I), I, ICON2( I), ITAG( I)
      IF(IPLP1.NE.1) GOTO 320
      WRITE( 8,*)  X( I), Y( I), Z( I), SI( I), XW2, YW2, BI( I), ICON1
     *( I), I, ICON2( I)

  320 CONTINUE
      IF(SI( I).GT.1.D-20.AND. BI( I).GT.0.) GOTO 32
      WRITE (2,56)
      STOP
   32 CONTINUE
   33 IF(M.EQ.0) GOTO 35
      WRITE (2,57)
      J= LD+1
      DO 34  I=1, M
      J= J-1
      XW1=( T1Y( J)* T2Z( J)- T1Z( J)* T2Y( J))* SALP( J)
      YW1=( T1Z( J)* T2X( J)- T1X( J)* T2Z( J))* SALP( J)
      ZW1=( T1X( J)* T2Y( J)- T1Y( J)* T2X( J))* SALP( J)
      WRITE (2,58)  I, X( J), Y( J), Z( J), XW1, YW1, ZW1, BI( J), T1X(
     * J), T1Y( J), T1Z( J), T2X( J), T2Y( J), T2Z( J)
   34 CONTINUE
   35 RETURN
   36 WRITE (2,48)
      WRITE (2,49)  GM, ITG, NS, XW1, YW1, ZW1, XW2, YW2, ZW2, RAD
      STOP
   37 WRITE (2,50)
```

```
C                                                                       DA 295
      STOP                                                              DA 296
   38 FORMAT(1X,I5,2X,'ARC RADIUS =',F9.5,2X,'FROM',F8.3,' TO',F8.3,    DA 297
     *' DEGREES',11X,F11.5,2X,I5,4X,I5,1X,I5,3X,I5)                     DA 298
   39 FORMAT(6X,3F11.5,1X,3F11.5)                                       DA 299
   40 FORMAT(////,33X,'- - - STRUCTURE SPECIFICATION - - -',//,37X,     DA 300
     *'COORDINATES MUST BE INPUT IN',/,37X,                             DA 301
     *'METERS OR BE SCALED TO METERS',/,37X,                            DA 302
     *'BEFORE STRUCTURE INPUT IS ENDED',//)                             DA 303
   41 FORMAT(2X,'WIRE',79X,'NO. OF',4X,'FIRST',2X,'LAST',5X,'TAG',/,2X, DA 304
     *'NO.',8X,'X1',9X,'Y1',9X,'Z1',10X,'X2',9X,'Y2',9X,'Z2',6X,        DA 305
     *'RADIUS',3X,'SEG.',5X,'SEG.',3X,'SEG.',5X,'NO.')                  DA 306
   42 FORMAT(A2, I3, I5, 7F10.5)                                        DA 307
   43 FORMAT(1X,I5,3F11.5,1X,4F11.5,2X,I5,4X,I5,1X,I5,3X,I5)            DA 308
   44 FORMAT(6X,'STRUCTURE REFLECTED ALONG THE AXES',3(1X,A1),'.  TA',  DA 309
     *'GS INCREMENTED BY',I5)                                           DA 310
   45 FORMAT(6X,'STRUCTURE ROTATED ABOUT Z-AXIS',I3,' TIMES.  LABELS',  DA 311
     *' INCREMENTED BY',I5)                                             DA 312
   46 FORMAT(6X,'STRUCTURE SCALED BY FACTOR',F10.5)                     DA 313
   47 FORMAT(6X,'THE STRUCTURE HAS BEEN MOVED, MOVE DATA CARD IS -/6X', DA 314
     *I3,I5,7F10.5)                                                     DA 315
   48 FORMAT(' GEOMETRY DATA CARD ERROR')                               DA 316
   49 FORMAT(1X,A2,I3,I5,7F10.5)                                        DA 317
   50 FORMAT(' NUMBER OF WIRE SEGMENTS AND SURFACE PATCHES EXCEEDS DI', DA 318
     *'MENSION LIMIT.')                                                 DA 319
   51 FORMAT(1X,I5,A1,F10.5,2F11.5,1X,3F11.5)                           DA 320
   52 FORMAT(' ERROR - GF MUST BE FIRST GEOMETRY DATA CARD')            DA 321
   53 FORMAT(////33X,'- - - - SEGMENTATION DATA - - - -',//,40X,'COO',  DA 322
     *'RDINATES IN METERS',//,25X,                                      DA 323
     *'I+ AND I- INDICATE THE SEGMENTS BEFORE AND AFTER I',//)          DA 324
   54 FORMAT(2X,'SEG.',3X,'COORDINATES OF SEG. CENTER',5X,'SEG.',5X,    DA 325
     *'ORIENTATION ANGLES',4X,'WIRE',4X,'CONNECTION DATA',3X,'TAG',/,2X DA 326
     *,'NO.',7X,'X',9X,'Y',9X,'Z',7X,'LENGTH',5X,'ALPHA',5X,'BETA',6X,  DA 327
     *'RADIUS',4X,'I-',3X,'I',4X,'I+',4X,'NO.')                         DA 328
   55 FORMAT(1X,I5,4F10.5,1X,3F10.5,1X,3I5,2X,I5)                       DA 329
   56 FORMAT(' SEGMENT DATA ERROR')                                     DA 330
   57 FORMAT(////,44X,'- - - SURFACE PATCH DATA - - -',//,49X,'COORD',  DA 331
     *'INATES IN METERS',//,1X,'PATCH',5X,'COORD. OF PATCH CENTER',7X,  DA 332
     *'UNIT NORMAL VECTOR',6X,'PATCH',12X,                              DA 333
     *'COMPONENTS OF UNIT TANGENT V''ECTORS',/,2X,'NO.',6X,'X',9X,'Y',9 DA 334
     *X,'Z',9X,'X',7X,'Y',7X,'Z',7X,'AREA',7X,'X1',6X,'Y1',6X,'Z1',7X,  DA 335
     *'X2',6X,'Y2',6X,'Z2')                                             DA 336
   58 FORMAT(1X,I4,3F10.5,1X,3F8.4,F10.5,1X,3F8.4,1X,3F8.4)             DA 337
   59 FORMAT(1X,I5,A1,F10.5,2F11.5,1X,3F11.5,5X,'SURFACE -',I4,' BY',I3 DA 338
     *,' PATCHES')                                                      DA 339
   60 FORMAT(' PATCH DATA ERROR')                                       DA 340
   61 FORMAT(9X,'ABOVE WIRE IS TAPERED.  SEG. LENGTH RATIO =',F9.5,/,33 DA 341
     *X,'RADIUS FROM',F9.5,' TO',F9.5)                                  DA 342
      END                                                               DA 343
```

DB10

PURPOSE

To convert an input magnitude quantity (field) or magnitude squared quantity (power) into decibels.

METHOD

For a squared quantity, the decibel conversion is

$$Q_{db} = 10 \log_{10} Q^2 \qquad (Q^2 input),$$

and for an unsquared quantity,

$$Q = 20\, log_{10}\, Q \ .$$

DB10 is used for the squared quantity while the entry DB20 is used for the quantity which is not squared.

SYMBOL DICTIONARY

ALOG10	=	external routine (log to the base 10)
DB10	=	Q_{db}
F	=	scaling term
X	=	input quantity

CONSTANT

-999.99 = returned for an input less than 10^{-20}

```
      FUNCTION DB10( X)
C
C     FUNCTION DB-- RETURNS DB FOR MAGNITUDE (FIELD) OR MAG**2 (POWER) I
C
      IMPLICIT REAL (A-H,O-Z)
      F=10.
      GOTO 1
      ENTRY DB20 (X)
      F=20.
    1 IF(X.LT.1.D-20) GOTO 2
      DB10= F* LOG10( X)
      RETURN
    2 DB10=-999.99
      RETURN
      END
```

EFLD

PURPOSE

To compute the near electric field due to constant, sine, and cosine current distributions on a segment in free space or over ground.

METHOD

The electric field is computed at the point XI, YI, ZI due to the segment defined by parameters in COMMON/DATAJ/. Either the thin wire or extended thin wire formulas may be used. When a ground is present, the code is executed twice in a loop. In the second pass, the field of the image of the segment is computed, multiplied by the reflection coefficients, and added to the direct field. The reflection coefficients for the reflected ray from the center of the source segment are used for the entire segment.

The field is evaluated in a cylindrical coordinate system with the source segment at the origin, along the z axis. The ρ coordinate of the field evaluation point is computed for the surface of the observation segment

$$\rho' = (\rho^2 + a^2)^{1/2},$$

where ρ is the distance from the axis or the source segment to (XI,YI,ZI) and a is the radius of the observation segment. The field is computed in ρ and z components as

$$\vec{E} = E_\rho(\vec{\rho}/\rho') + E_z\hat{z} \ .$$

Use of ρ' avoids a singularity when (XI,YI,ZI) is the center of the source segment. In the addition of field components, ρ/ρ' is used rather than ρ, since E_ρ is the field in the direction ρ' to one side of the observation segment.

When the Sommerfeld/Norton option is used for an antenna over ground the electric field at $\hat{r}$ due to the current on a segment is evaluated in three terms as

$$\vec{E}(\vec{r}) = \vec{E}_D(\vec{r}) + \frac{k_1^2 - k_2^2}{k_1^2 + k_2^2}\vec{E}_I(\vec{r}) + \vec{E}_S(\vec{r})$$

$\vec{E}_D$ is the direct field of the segment in the absence or ground, and $\vec{E}_I$ is the field of the image of the segment reflected in a perfectly conducting ground. These field camonents are evaluated in EFLD between EF19 and EF150. The factor $(k_1^2 - k_2^2)/(k_1^2 + k_2^2)$ is contained in the variable FRATI.

The field $\vec{E}_S$, due to the Sommerfeld integrals is evaluated from EF155 to EF227. If the separation of the observation point and the center of the source segment is less than one wavelength, subroutine ROM2 is called at EE191 to integrate over the segment. DMIN is set to the magnitude of the first two terms in $\vec{E}$ divided by 100 as a lower limit on the denominator of the relative error test in the numerical integration. This relaxes the relative accuracy requirement when $\vec{E}_S$ is small compared to the first two terms.

If the separation of the source segment and observation point is greater than a wavelength, SFLDS is called at EF197 to evaluate $\vec{E}_s$ by the Norton approximation.

To compute $\vec{E}_S$ with the thin wire approximation applied in a manner consistent with that for $\vec{E}_I$, the field is evaluated at a point displaced normal to the image of the source segment and normal to the separation $\vec{R}$. If the direction of the image of the source segment is $\hat{j}$ the displacement is $\vec{D}$ where

$$\begin{aligned} \vec{D} &= +a\hat{d} \quad \text{for} \quad \hat{z}\cdot\hat{d} > 0 \\ \vec{D} &= -a\hat{d} \quad \text{for} \quad \hat{z}\cdot\hat{d} < 0 \\ \hat{d} &= (\hat{j}\times\vec{R})/|\hat{j}\times\vec{R}| \\ a &= \text{radius of observation segment} \end{aligned}$$

This displaced observation point (X0,Y0,Z0) is computed from EF166 to EF181. Some of the complexity is needed to make the result independent of orientation of segments relative to the coordinate axes.

To adjust the ρ component of field for the factor $|\vec{\rho}/\rho'|$ the field $\vec{E}'$ is computed as

$$\vec{E}' = F\vec{E} + (1-F)(\vec{E}\cdot\hat{j})\hat{j}$$

where

$$F = [\rho^2/(\rho^2+a^2)]^{1/2}$$

$$\rho^2 = |\vec{R}|^2 - (\vec{R}\cdot\hat{j})^2$$

This is done from EF204 to EF218 but is skipped if F(DMIN) is greater than 0.95.

CODING

EF23	Loop over direct and image fields.
EF29-EF31	Components of ρ.
EF33-EF40	Components of ρ/ρ' computed.
EF46-EF62	Electric field of the segment computed by infinitesimal dipole approximation.
EF68	Field computed by thin-wire approximation.
EF70	Field computed by extended thin-wire approximation.
EF72-EF80	Field converted to x-, y-, and z-components.
EF89-EF111	Reflection coefficients computed.
EF112-EF129	Image fields modified by reflection coefficients.
EF130-EF138	Reflected fields added to direct fields.

SYMBOL DICTIONARY

AI	=	radius of segment on which field is evaluated
CTH	=	cos θ; θ = angle from axis of infinitesimal dipole or angle between the reflecting ray and vertical
EGND	=	components of $\vec{E}_S$ (see EQUIVALENCE statement)
EPX	=	
EPY	=	
ETA	=	
IJ	=	IJX = flag to indicate field evaluation point is on the source segment (IJ = 0)
PI	=	π
PX	=	x and y components of unit vector normal to the plane of
PY		incidence of the reflected wave ($\hat{\rho}$)
R	=	distance from field evaluation point to the center of nne source segment

REFPS = reflection coefficient for a horizontally polarized field
REFS = reflection coefficient for a vertically polarized field
RPL = +1 for direct field, -1 for reflected field
RH = ρ'
RHOSPC = distance from coordinate origin to the point where the ray from the source to (XI,YI,ZI) reflects from the ground
RHOX
RHOY = x, y, and z components of $\vec{\rho}$ or $\vec{\rho}/\rho'$
or $\hat{j} \times \vec{R}$
RHOZ
RMAG = $2\pi R$ or R or dipole moment for sin ks current
SALPR = z-component of unit vector in the direction of the source segment or its image
SHAF = half of segment length
TERC = ρ component of field due to cos ks, sin ks,
TERS and constant currents, respectively
TERK
TEZC = z-component of field due to cos ks, sin ks, and
TEZS constant current, respectively
TEZK
TP = 2π
TXC
TYC
TZC
TXS
TYS = x, y, and z components of field due to cos ks,
TLS sin ks, and constant current
TXK
TYK
TZK
XI
YI = x, y, z coordinates of field evaluation point
ZI
XIJ = cmnpunents of distance from source to observation
YIJ point
ZIJ
XO
YO = coordinates of field evaluation point for E_S
ZO
XSPEC = x, y coordinates of ground plane reflection point
YSPEC
XYMAG = horizontal distance from center of source segment to observation point
ZP = projection of the vector from the source segment (XI,YI,ZI) onto the axis of the source Segment
ZRATX = temporary storage for ZRATI
ZRSIN = $(1 - Z_R^2 sin^2\theta)^{1/2}$ for ground
ZSCRN = quantity used in computing reflection coefficient for radial wire ground screen

```
      SUBROUTINE EFLD(XI,YI,ZI,AI,IJ)
C
C     COMPUTE NEAR E FIELDS OF A SEGMENT WITH SINE, COSINE, AND
C     CONSTANT CURRENTS.  GROUND EFFECT INCLUDED.
C
      COMPLEX  TXK, TYK, TZK, TXS, TYS, TZS, TXC, TYC, TZC, EXK, EYK
     *, EZK, EXS, EYS, EZS, EXC, EYC, EZC, EPX, EPY, ZRATI, REFS, REFPS
     *, ZRSIN, ZRATX, T1, ZSCRN, ZRATI2, TEZS, TERS, TEZC, TERC, TEZK,
     *TERK, EGND, FRATI
      COMMON/DATAJ/ S, B, XJ, YJ, ZJ, CABJ, SABJ, SALPJ, EXK, EYK,
     *EZK, EXS, EYS, EZS, EXC, EYC, EZC, RKH, IEXK, IND1, INDD1, IND2,
     *INDD2, IPGND
      COMMON/GND/ ZRATI, ZRATI2, FRATI, CL, CH, SCRWL, SCRWR, NRADL,
     *KSYMP, IFAR, IPERF, T1, T2
      COMMON/INCOM/ XO, YO, ZO, SN, XSN, YSN, ISNOR
      DIMENSION  EGND(9)
      EQUIVALENCE(EGND(1),TXK),(EGND(2),TYK),(EGND(3),TZK),(EGND(4),TXS
     *),(EGND(5),TYS),(EGND(6),TZS),(EGND(7),TXC),(EGND(8),TYC),(EGND(9
     *),TZC)
      DATA  ETA/376.73/, PI/3.141592654D+0/, TP/6.283185308D+0/
      XIJ=XI- XJ
      YIJ=YI- YJ
      IJX=IJ
      RFL=-1.
      DO 12  IP=1, KSYMP
      IF(IP.EQ.2) IJX=1
      RFL=- RFL
      SALPR=SALPJ* RFL
      ZIJ=ZI- RFL* ZJ
      ZP=XIJ* CABJ+ YIJ* SABJ+ ZIJ* SALPR
      RHOX=XIJ- CABJ* ZP
      RHOY=YIJ- SABJ* ZP
      RHOZ=ZIJ- SALPR* ZP
      RH=SQRT( RHOX* RHOX+ RHOY* RHOY+ RHOZ* RHOZ+ AI* AI)
      IF(RH.GT.1.D-10) GOTO 1
      RHOX=0.
      RHOY=0.
      RHOZ=0.
      GOTO 2
    1 RHOX=RHOX/ RH
      RHOY=RHOY/ RH
      RHOZ=RHOZ/ RH
    2 R=SQRT( ZP* ZP+ RH* RH)
C
C     LUMPED CURRENT ELEMENT APPROX. FOR LARGE SEPARATIONS
C
      IF(R.LT. RKH) GOTO 3
      RMAG=TP* R
      CTH=ZP/ R
```

```
      PX=RH/ R
      TXK=CMPLX( COS( RMAG),- SIN( RMAG))
      PY=TP* R* R
      TYK=ETA* CTH* TXK* CMPLX(1.D+0,-1.D+0/ RMAG)/ PY
      TZK=ETA* PX* TXK* CMPLX(1.D+0, RMAG-1.D+0/ RMAG)/(2.* PY)
      TEZK=TYK* CTH- TZK* PX
      TERK=TYK* PX+ TZK* CTH
      RMAG=SIN( PI* S)/ PI
      TEZC=TEZK* RMAG
      TERC=TERK* RMAG
      TEZK=TEZK* S
      TERK=TERK* S
      TXS=(0.,0.)
      TYS=(0.,0.)
      TZS=(0.,0.)
      GOTO 6
C
C     EKSC FOR THIN WIRE APPROX. OR EKSCX FOR EXTENDED T.W. APPROX.
C
    3 IF(IEXK.EQ.1) GOTO 4
      CALL EKSC( S, ZP, RH, TP, IJX, TEZS, TERS, TEZC, TERC, TEZK, TERK
     *)
      GOTO 5
    4 CALL EKSCX( B, S, ZP, RH, TP, IJX, IND1, IND2, TEZS, TERS, TEZC,
     *TERC, TEZK, TERK)
    5 TXS=TEZS* CABJ+ TERS* RHOX
      TYS=TEZS* SABJ+ TERS* RHOY
      TZS=TEZS* SALPR+ TERS* RHOZ
    6 TXK=TEZK* CABJ+ TERK* RHOX
      TYK=TEZK* SABJ+ TERK* RHOY
      TZK=TEZK* SALPR+ TERK* RHOZ
      TXC=TEZC* CABJ+ TERC* RHOX
      TYC=TEZC* SABJ+ TERC* RHOY
      TZC=TEZC* SALPR+ TERC* RHOZ
      IF(IP.NE.2) GOTO 11
      IF(IPERF.GT.0) GOTO 10
      ZRATX=ZRATI
      RMAG=R
C
C     SET PARAMETERS FOR RADIAL WIRE GROUND SCREEN.
C
      XYMAG=SQRT( XIJ* XIJ+ YIJ* YIJ)
      IF(NRADL.EQ.0) GOTO 7
      XSPEC=( XI* ZJ+ ZI* XJ)/( ZI+ ZJ)
      YSPEC=( YI* ZJ+ ZI* YJ)/( ZI+ ZJ)
      RHOSPC=SQRT( XSPEC* XSPEC+ YSPEC* YSPEC+ T2* T2)
      IF(RHOSPC.GT. SCRWL) GOTO 7
      ZSCRN=T1* RHOSPC* LOG( RHOSPC/ T2)
      ZRATX=( ZSCRN* ZRATI)/( ETA* ZRATI+ ZSCRN)
```

```
C
C     CALCULATION OF REFLECTION COEFFICIENTS WHEN GROUND IS SPECIFIED.
C
    7 IF(XYMAG.GT.1.D-6) GOTO 8
      PX=0.
      PY=0.
      CTH=1.
      ZRSIN=(1.,0.)
      GOTO 9
    8 PX=- YIJ/ XYMAG
      PY=XIJ/ XYMAG
      CTH=ZIJ/ RMAG
      ZRSIN=SQRT(1.- ZRATX* ZRATX*(1.- CTH* CTH))
    9 REFS=( CTH- ZRATX* ZRSIN)/( CTH+ ZRATX* ZRSIN)
      REFPS=-( ZRATX* CTH- ZRSIN)/( ZRATX* CTH+ ZRSIN)
      REFPS=REFPS- REFS
      EPY=PX* TXK+ PY* TYK
      EPX=PX* EPY
      EPY=PY* EPY
      TXK=REFS* TXK+ REFPS* EPX
      TYK=REFS* TYK+ REFPS* EPY
      TZK=REFS* TZK
      EPY=PX* TXS+ PY* TYS
      EPX=PX* EPY
      EPY=PY* EPY
      TXS=REFS* TXS+ REFPS* EPX
      TYS=REFS* TYS+ REFPS* EPY
      TZS=REFS* TZS
      EPY=PX* TXC+ PY* TYC
      EPX=PX* EPY
      EPY=PY* EPY
      TXC=REFS* TXC+ REFPS* EPX
      TYC=REFS* TYC+ REFPS* EPY
      TZC=REFS* TZC
   10 EXK=EXK- TXK* FRATI
      EYK=EYK- TYK* FRATI
      EZK=EZK- TZK* FRATI
      EXS=EXS- TXS* FRATI
      EYS=EYS- TYS* FRATI
      EZS=EZS- TZS* FRATI
      EXC=EXC- TXC* FRATI
      EYC=EYC- TYC* FRATI
      EZC=EZC- TZC* FRATI
      GOTO 12
   11 EXK=TXK
      EYK=TYK
      EZK=TZK
      EXS=TXS
      EYS=TYS
```

```
      EZS=TZS                                                           EF 148
      EXC=TXC                                                           EF 149
      EYC=TYC                                                           EF 150
      EZC=TZC                                                           EF 151
   12 CONTINUE                                                          EF 152
      IF(IPERF.EQ.2) GOTO 13                                            EF 153
C                                                                       EF 154
C     FIELD DUE TO GROUND USING SOMMERFELD/NORTON                       EF 155
C                                                                       EF 156
      RETURN                                                            EF 157
   13 SN=SQRT( CABJ* CABJ+ SABJ* SABJ)                                  EF 158
      IF(SN.LT.1.D-5) GOTO 14                                           EF 159
      XSN=CABJ/ SN                                                      EF 160
      YSN=SABJ/ SN                                                      EF 161
      GOTO 15                                                           EF 162
   14 SN=0.                                                             EF 163
      XSN=1.                                                            EF 164
C                                                                       EF 165
C     DISPLACE OBSERVATION POINT FOR THIN WIRE APPROXIMATION            EF 166
C                                                                       EF 167
      YSN=0.                                                            EF 168
   15 ZIJ=ZI+ ZJ                                                        EF 169
      SALPR=- SALPJ                                                     EF 170
      RHOX=SABJ* ZIJ- SALPR* YIJ                                        EF 171
      RHOY=SALPR* XIJ- CABJ* ZIJ                                        EF 172
      RHOZ=CABJ* YIJ- SABJ* XIJ                                         EF 173
      RH=RHOX* RHOX+ RHOY* RHOY+ RHOZ* RHOZ                             EF 174
      IF(RH.GT.1.D-10) GOTO 16                                          EF 175
      XO=XI- AI* YSN                                                    EF 176
      YO=YI+ AI* XSN                                                    EF 177
      ZO=ZI                                                             EF 178
      GOTO 17                                                           EF 179
   16 RH=AI/ SQRT( RH)                                                  EF 180
      IF(RHOZ.LT.0.) RH=- RH                                            EF 181
      XO=XI+ RH* RHOX                                                   EF 182
      YO=YI+ RH* RHOY                                                   EF 183
      ZO=ZI+ RH* RHOZ                                                   EF 184
   17 R=XIJ* XIJ+ YIJ* YIJ+ ZIJ* ZIJ                                    EF 185
C                                                                       EF 186
C     FIELD FROM INTERPOLATION IS INTEGRATED OVER SEGMENT               EF 187
C                                                                       EF 188
      IF(R.GT..95) GOTO 18                                              EF 189
      ISNOR=1                                                           EF 190
      DMIN=EXK* CONJG( EXK)+ EYK* CONJG( EYK)+ EZK* CONJG( EZK)         EF 191
      DMIN=.01* SQRT( DMIN)                                             EF 192
      SHAF=.5* S                                                        EF 193
      CALL ROM2(- SHAF, SHAF, EGND, DMIN)                               EF 194
C                                                                       EF 195
C     NORTON FIELD EQUATIONS AND LUMPED CURRENT ELEMENT APPROXIMATION   EF 196
```

```
C
      GOTO 19
   18 ISNOR=2
      CALL SFLDS(0., EGND)
      GOTO 22
   19 ZP=XIJ* CABJ+ YIJ* SABJ+ ZIJ* SALPR
      RH=R- ZP* ZP
      IF(RH.GT.1.D-10) GOTO 20
      DMIN=0.
      GOTO 21
   20 DMIN=SQRT( RH/( RH+ AI* AI))
   21 IF(DMIN.GT..95) GOTO 22
      PX=1.- DMIN
      TERK=( TXK* CABJ+ TYK* SABJ+ TZK* SALPR)* PX
      TXK=DMIN* TXK+ TERK* CABJ
      TYK=DMIN* TYK+ TERK* SABJ
      TZK=DMIN* TZK+ TERK* SALPR
      TERS=( TXS* CABJ+ TYS* SABJ+ TZS* SALPR)* PX
      TXS=DMIN* TXS+ TERS* CABJ
      TYS=DMIN* TYS+ TERS* SABJ
      TZS=DMIN* TZS+ TERS* SALPR
      TERC=( TXC* CABJ+ TYC* SABJ+ TZC* SALPR)* PX
      TXC=DMIN* TXC+ TERC* CABJ
      TYC=DMIN* TYC+ TERC* SABJ
      TZC=DMIN* TZC+ TERC* SALPR
   22 EXK=EXK+ TXK
      EYK=EYK+ TYK
      EZK=EZK+ TZK
      EXS=EXS+ TXS
      EYS=EYS+ TYS
      EZS=EZS+ TZS
      EXC=EXC+ TXC
      EYC=EYC+ TYC
      EZC=EZC+ TZC
      RETURN
      END
```

EKSC

PURPOSE

To compute the electric field due to current filaments with sin kz, cos kz and constant distributions.

METHOD

Equations 71 through 74 in Part I are used. The current filament is located at the origin of a cylindrical coordinate system, oriented along the z-axis, and extending from $-\Delta/2$ to $\Delta/2$. The field is computed in ρ and z components.

SYMBOL DICTIONARY

CINT	=	$\int_{-\Delta/2}^{\Delta/2} cos(kr)/rdz$
CON	=	CONX = $j\eta/(8\pi^2), \eta = \sqrt{\mu_0/\epsilon_0}$
CS	=	$cos(k\Delta/2)$
ERS EZS ERC EZC ERK EZK	=	ρ and z components of field due to sin kz, cos kz, and constant (S, C, K, respectively) current distributions extending from z = -Δ/2 to z = Δ/2
GP1 GP2	=	-(1 + jkr) G_0/r^2 for z = -Δ/2 and Δ/2, respectively, where G_0 = exp(-jkr)/r
GZ1 GZ2	=	G_0 for z = -Δ/2 and Δ/2, respectively
GZP1 GZP2	=	$\partial G_0/\partial z$ at EK22 and $\partial G_0/\partial \rho$ at EK28, EK29 for z = -Δ/Z and Δ/2, respectively
IJ	=	IJX = 0 to indicate that the field point is an the source segment
RH	=	ρ coordinate of field point
RHK	=	kρ (k = $2\pi/\lambda$, λ = 1)
RKB2	=	$(k\rho)^2$
S	=	Δ
SH	=	Δ/2
SHK	=	kΔ/2
SINT	=	$\int_{-\Delta/2}^{\Delta/2} sin(kr)/r\,dz$
SS	=	sin (kΔ/2)
XK	=	k = $2\pi/\lambda$, where λ = 1
Z	=	z-coordinate of field point
Z1	=	-Δ/2 - z
Z2	=	Δ/2 - z
ZPK	=	kz

CONSTANT

4.771341189	=	$\eta/(8\pi^2)$

```
      SUBROUTINE EKSC( S, Z, RH, XK, IJ, EZS, ERS, EZC, ERC, EZK, ERK)
      IMPLICIT REAL (A-H,O-Z)
C     COMPUTE E FIELD OF SINE, COSINE, AND CONSTANT CURRENT FILAMENTS BY
C     THIN WIRE APPROXIMATION.
      COMPLEX  CON, GZ1, GZ2, GP1, GP2, GZP1, GZP2, EZS, ERS, EZC,
     *ERC, EZK, ERK
      COMMON  /TMI/ ZPK, RKB2, IJX
      DIMENSION  CONX(2)
      EQUIVALENCE(CONX,CON)
      DATA   CONX/0.,4.771341189D+0/
      IJX= IJ
      ZPK= XK* Z
      RHK= XK* RH
      RKB2= RHK* RHK
      SH=.5* S
      SHK= XK* SH
      SS= SIN( SHK)
      CS= COS( SHK)
      Z2= SH- Z
      Z1=-( SH+ Z)
      CALL GX( Z1, RH, XK, GZ1, GP1)
      CALL GX( Z2, RH, XK, GZ2, GP2)
      GZP1= GP1* Z1
      GZP2= GP2* Z2
      EZS= CON*(( GZ2- GZ1)* CS* XK-( GZP2+ GZP1)* SS)
      EZC=- CON*(( GZ2+ GZ1)* SS* XK+( GZP2- GZP1)* CS)
      ERK= CON*( GP2- GP1)* RH
      CALL INTX(- SHK, SHK, RHK, IJ, CINT, SINT)
      EZK=- CON*( GZP2- GZP1+ XK* XK* CMPLX( CINT,- SINT))
      GZP1= GZP1* Z1
      GZP2= GZP2* Z2
      IF(RH.LT.1.D-10) GOTO 1
      ERS=- CON*(( GZP2+ GZP1+ GZ2+ GZ1)* SS-( Z2* GZ2- Z1* GZ1)* CS*
     *XK)/ RH
      ERC=- CON*(( GZP2- GZP1+ GZ2- GZ1)* CS+( Z2* GZ2+ Z1* GZ1)* SS*
     *XK)/ RH
      RETURN
    1 ERS=(0.,0.)
      ERC=(0.,0.)
      RETURN
      END
```

EKSCX

PURPOSE

To compute the electric field due to current distributions of sin kz, cos kz, and constant on the surface of a cylinder by the extended thin wire approximation.

METHOD

Equations 84 through 87 in Part I are used. The current tube is centered on the origin of a cylindrical coordinate system, oriented along the z-axis and extending from $-\Delta/2$ to $\Delta/2$. The field is computed in ρ and z components.

If INX1 = 2, the field contributions from end 1 of the segment ($z = -\Delta/2$) are evaluated by the thin wire approximation for a current filament on the cylinder axis. INX2 has the same meaning for end 2 of the segment ($z = \Delta/2$). Then thin-wire approximation is used at an end when there is a bend or change in radius from that end to the next segment.

When the ρ coordinate of the field point (RHX) is less than the radius of the current tube (BX), then RHX and BX are interchanged and a flag, IRA, is set to 1 to cause alternate forms for G_1 and its derivatives to be used in routine GXX.

SYMBOL DICTIONARY

A2	=	B^2
B	=	radius of the current tube
BK	=	kB, where $k = 2\pi/\lambda$, $\lambda = 1$
BK2	=	$(BK)^2/4$
BX	=	radius of the current tube
CINT	=	$\int_{-\Delta/2}^{\Delta/2} cos(kr)/r\,dz$
CON	=	CONX = $j\eta/(8\pi^2)$, where $\eta = \sqrt{\mu_0/\epsilon_0}$
CS	=	cos $(k\Delta/2)$
ERS		
EZS	=	ρ and z components of field due to sin kz, cos kz, and constant (S, C, K, respectively) current distributions
EZC		
ERK		extending from $z = -\Delta/2$ to $z = \Delta/2$.
GR1	=	G_2 for $z = -\Delta/2$ and $\Delta/2$, respectively
GR2		
GRK1	=	$\partial G_1/\partial\rho$
GRK2		
GZ1	=	G_1
GZ2		
GZP1	=	$\partial G_1/\partial\rho$
GZP2		
GZZ1	=	
GZZ2		
IJ	=	IJX = 0 to indicate that the field point is on the source segment
INX1	=	2 to use the thin wire form at end 1 or end 2,
INX2		respectively
IRA	=	1 to indicate RHX < BX

RH	=	ρ coordinate of the field point or wire radius
RHK	=	k(RH)
RHX	=	ρ coordinate of the field point
RKB2	=	$(\text{RHK})^2$
S	=	Δ
SH	=	$\Delta/2$
SHK	=	$k\Delta/2$
SINT	=	$\int_{-\Delta/2}^{\Delta/2} sin(kr)/r\,dz$
SS	=	sin $(k\Delta/2)$
XK	=	k = $2\pi/\lambda$, λ = 1
Z	=	z-coordinate of field point
Z1	=	$-\Delta/2 - z$
Z2	=	$\Delta/2 - z$
ZPK	=	kz

CONSTANT

4.77134118	=	$\eta/(8\pi^2)$

```
      SUBROUTINE EKSCX( BX, S, Z, RHX, XK, IJ, INX1, INX2, EZS, ERS,
     *EZC, ERC, EZK, ERK)
C     COMPUTE E FIELD OF SINE, COSINE, AND CONSTANT CURRENT FILAMENTS BY
C     EXTENDED THIN WIRE APPROXIMATION.
      IMPLICIT REAL (A-H,O-Z)
      COMPLEX  CON, GZ1, GZ2, GZP1, GZP2, GR1, GR2, GRP1, GRP2, EZS,
     * EZC, ERS, ERC, GRK1, GRK2, EZK, ERK, GZZ1, GZZ2
      COMMON  /TMI/ ZPK, RKB2, IJX
      DIMENSION  CONX(2)
      EQUIVALENCE(CONX,CON)
      DATA   CONX/0.,4.771341189D+0/
      IF(RHX.LT. BX) GOTO 1
      RH= RHX
      B= BX
      IRA=0
      GOTO 2
    1 RH= BX
      B= RHX
      IRA=1
    2 SH=.5* S
      IJX= IJ
      ZPK= XK* Z
      RHK= XK* RH
      RKB2= RHK* RHK
      SHK= XK* SH
      SS= SIN( SHK)
      CS= COS( SHK)
      Z2= SH- Z
      Z1=-( SH+ Z)
      A2= B* B
      IF(INX1.EQ.2) GOTO 3
      CALL GXX( Z1, RH, B, A2, XK, IRA, GZ1, GZP1, GR1, GRP1, GRK1,
     *GZZ1)
      GOTO 4
    3 CALL GX( Z1, RHX, XK, GZ1, GRK1)
      GZP1= GRK1* Z1
      GR1= GZ1/ RHX
      GRP1= GZP1/ RHX
      GRK1= GRK1* RHX
      GZZ1=(0.,0.)
    4 IF(INX2.EQ.2) GOTO 5
      CALL GXX( Z2, RH, B, A2, XK, IRA, GZ2, GZP2, GR2, GRP2, GRK2,
     *GZZ2)
      GOTO 6
    5 CALL GX( Z2, RHX, XK, GZ2, GRK2)
      GZP2= GRK2* Z2
      GR2= GZ2/ RHX
      GRP2= GZP2/ RHX
      GRK2= GRK2* RHX
```

```
      GZZ2=(0.,0.)
    6 EZS= CON*(( GZ2- GZ1)* CS* XK-( GZP2+ GZP1)* SS)
      EZC=- CON*(( GZ2+ GZ1)* SS* XK+( GZP2- GZP1)* CS)
      ERS=- CON*(( Z2* GRP2+ Z1* GRP1+ GR2+ GR1)* SS-( Z2* GR2- Z1* GR1
     *)* CS* XK)
      ERC=- CON*(( Z2* GRP2- Z1* GRP1+ GR2- GR1)* CS+( Z2* GR2+ Z1* GR1
     *)* SS* XK)
      ERK= CON*( GRK2- GRK1)
      CALL INTX(- SHK, SHK, RHK, IJ, CINT, SINT)
      BK= B* XK
      BK2= BK* BK*.25
      EZK=- CON*( GZP2- GZP1+ XK* XK*(1.- BK2)* CMPLX( CINT,- SINT)-
     *BK2*( GZZ2- GZZ1))
      RETURN
      END
```

ENF

PURPOSE

To check for an end of file.

METHOD

ENF uses the standard Fortran end-of-file test and returns the logical values .TRUE. or .FALSE. This separate function is used for convenience in adapting the code to particular computers, since the Fortran end-of-file test statements often differ between computers. The form of ENF here is for CDC computers.

SYMBOL DICTIONARY

ENF	=	logical value: .TRUE. if end of file was encountered; .FALSE. otherwise
NUNIT	=	logical unit number

CODE LISTING

```
C ***
C     DOUBLE PRECISION 6/4/85
C
      LOGICAL FUNCTION ENF( NUNIT)
C ***
C*********** THIS ROUTINE NOT USED ON VAX **************
C     IF (EOF,NUNIT) 1,2
      IMPLICIT REAL*8 (A-H,O-Z)
    1 ENF=.TRUE.
      RETURN
    2 ENF=.FALSE.
      RETURN
      END
```

```
C ***
C     DOUBLE PRECISION 6/4/85
C
C     IMPLICIT REAL*8(A-H,O-Z)
C ***
      SUBROUTINE ERROR (stat)
      IMPLICIT none
      CHARACTER   MSG*80
 integer stat

 print *, 'ERROR - open error encountered.'
 print *, '        stat = lib-',stat
CJCB      CALL SYS$GETMSG(%VAL(RMSSTS),MSGLEN,MSG,,,)
CJCB      CALL ERRSNS( FNUM, RMSSTS, RMSSTV, IUNIT, CNDVAL)
c     CALL STROPC(MSG)
c     IND= INDEX( MSG,',')
c     PRINT1 , MSG( IND+2:MSGLEN )
c   1 FORMAT(//,'  ****  ERROR  ****   ',//,5X,A,//)
      RETURN
      END
```

ETMNS

PURPOSE

To fill the array representing the right-hand side of the matrix equation with the negative of the electric field tangent to the segments and with the tangential magnetic field on the surfaces.

METHOD

The array E represents the right-hand side of the matrix equation. For the i-th segment, the right-hand side is the negative of the applied electric field component tangent to the segment, and is stored in location i in array E. For the i-th surface patch, there are two rows in the matrix equation (from the two components of the vector equations) with locations N+2i-l and N+2i, where N is the total number of wire segments. The contents of E for these locations are

$$E(N+2i-1) = -\hat{t}_1 \cdot (\hat{n} \times \vec{H}_i) = \pm\hat{t}_2 \cdot \vec{H}_i$$

$$E(N+2i) = \hat{t}_2 \cdot (\hat{n} \times \vec{H}_i) = \pm\hat{t}_1 \cdot \vec{H}_i$$

where $\vec{H}_i$ is the magnetic field applied to patch i. The forms on the right are used in the code with the plus sign applying when ($\hat{t}_1$,$\hat{t}_2$,$\hat{n}$) terms a right-hand system and the minus sign when left-hand. To avoid the need to check ($\hat{t}_1$,$\hat{t}_2$,$\hat{n}$), the sign is stored in array SALP where, for patch i, SALP (LD + l - l) = $\pm$1 according to ($\hat{t}_1$,$\hat{t}_2$,$\hat{n}$), with up the length of the arrays an COMMON/DATA/. If the structure has symmetry, the entries in E are reordered by subroutine SOLVES.

The parameter IPR selects the type of excitation; the meanings of other parameters depend on the option selected by IPR and are explained below. The excitations associated with IPR values are:

IPR = 0 applied field voltage source
1 incident plane wave, linear polarization
2 incident plane wave, right-hand elliptic polarization
3 incident plane wave, left-hand elliptic polarization
4 infinitesimal current element source
5 current slope discontinuity voltage source

CODING

ET29-ET34	Applied field voltage source (IPR = 0).
ET36-ET38	QDSRC is called for each current slope discontinuity voltage source (IPR = 5).
ET44-ET160	Incident plane wave. The direction of propagation and polarization of the wave are illustrated in figure 4 in which $\hat{\rho}$ is the unit vector normal to $\hat{k}$ in the plane defined by $\hat{k}$ and $\hat{z}$. The plane wave as a function of position $\vec{r}$ is

$$\vec{E}^I(\vec{r}) = \vec{E}_0 exp(-j\vec{k} \cdot \vec{r})$$

$$\vec{H}^I(\vec{r}) = \frac{1}{\eta}\hat{k} \times \vec{E}_0 exp(-j\vec{k} \cdot \vec{r})$$

where

$\vec{k} = (2\pi/\lambda)\,\hat{k}$
$\hat{k}$ = unit vector in direction of propagation
$\vec{E}_0 = \vec{E}_1$ for linear polarization
= $(\vec{E}_1 - jA\hat{E}_2)$ for right-hand elliptical polarization
= $(\vec{E}_1 + jA\hat{E}_2)$ for left-hand elliptical polarization
A = ellipse axes ration
$\hat{E}_2 = \hat{k} \times \hat{E}_1$

ET44-ET58 P1 = θ
P2 = Φ
P3 = ξ
PX,PY,PZ = x,y,z components of $\hat{E}_1$
WX,WY,WZ = $\hat{k}$
QX,QY,QZ = $\hat{E}_2$ = $\hat{k} \times \hat{E}_1$

ET61-ET68 Ground reflection coefficients computed:
RRH = reflection coefficient for E normal to the plane of incidence
RRV = reflection coefficient for E normal in the plane of incidence

ET70-ET108 Linearly polarized wave (IPR = 1).

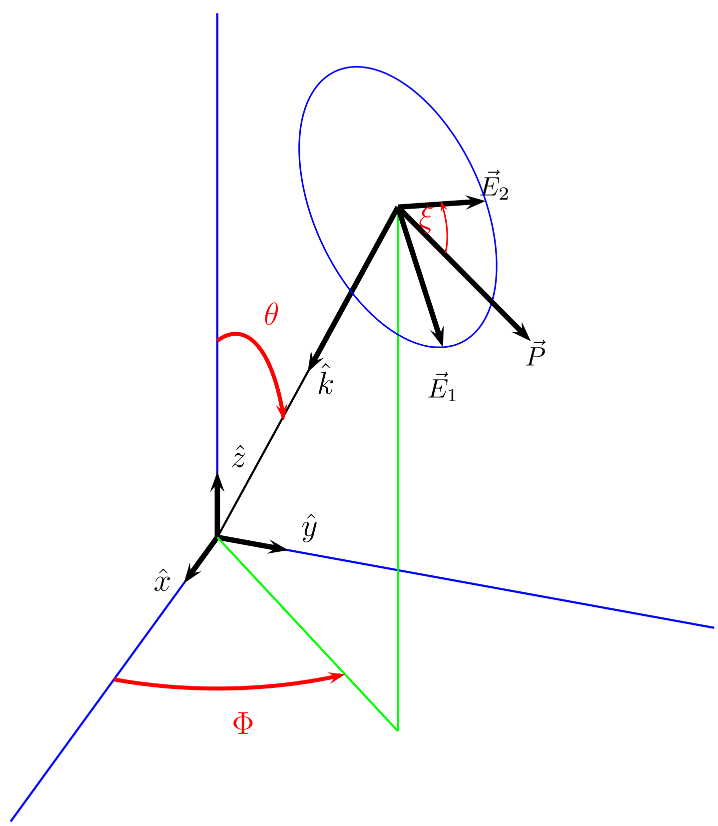


Figure 4. Coordinate Parameters for the Incident Plane Wave.

ET71-ET73 Direct illumination of segments by E field. ARG = -$\hat{k} \cdot \vec{r}_i$, where $\vec{r}_i$ = center point of segment I.
$E(I) = -(\hat{E}_i \cdot \hat{i}) exp(-j\vec{k} \cdot \vec{r}_i)$,
where $\hat{i}$ = unit vector in the direction of segment I.

ET75-ET82 Illumination of segments by the ground reflected field.
CX,CY,CZ = reflected E field

ET84-ET93 Direct H field illumination of patches.

ET95-ET108 Illumination of patches by the ground reflected field.
CX,CY,CZ = reflected H field

ET113-ET159 Elliptically polarized wave (IPR = 2 or 3).
P6 = ellipse axes ratio = A.

ET116-ET121 Direct E field illumination of segments.
CX,CY,CZ = $\vec{E}_1 \pm jA\hat{E}_2$ (+ for left-hand polarization, - for right-hand)

ET123-ET130 Illumination of segments by the ground reflected E field.

ET132-ET144 Illumination of patches by the direct H field.
CX,CY,CZ = $\hat{k} \times \vec{E}_0$

ET146-ET159 Illumination of patches by ground reflected H field.

ET164-ET225 Infinitesimal current element source (IPR = 4). A current element of moment $I_0\, l$ at the origin of a spherical coordinate system, as shown in figure 5, produces field components

$$\vec{E}_R(\vec{R}) = I_0 l \frac{\eta}{2\pi} exp(-jkR) \left(1 - \frac{j}{kR}\right) \frac{1}{R^2} cos\theta \hat{R}$$

$$\vec{E}_\theta(\vec{R}) = I_0 l \frac{\eta}{4\pi} exp(-jkR) \left[\frac{jk}{R} + \left(1 - \frac{j}{kR}\right) \frac{1}{R^2}\right] sin\theta \hat{\theta}$$

If the location and orientation of segment i and the current element with respect to the x,y,z coordinate system are
$\vec{r}_i$ = location of segment i
$\hat{i}$ = orientation of segment i
$\vec{D}$ = location of current element
$\hat{d}$ = orientation of current element
then
$\vec{R} = \vec{r}_i - \vec{D}$
$\hat{R} = \vec{R}/|\vec{R}|$
cos θ = $\hat{R} \cdot \hat{d}$
sin θ = $(1 - cos^2\theta)^{1/2}$

The orientation of the current element is defined by its angle of elevation above the x-y plane, a, and the angle from the x axis to its projection on the x-y plane, b. Thus, $\hat{d}$ = cos a cos b $\hat{x}$ + cos a sin b $\hat{y}$ sin a $\hat{z}$.
The $\vec{R}$ and $\hat{\theta}$ field components are converted to $\hat{\rho}$ and $\hat{d}$ components E_ρ and E_d , where
$E_d = E_R cos\theta - E_\theta sin\theta$
$E_\rho = E_R sin\theta + E_\theta cos\theta$
and the excitation computed as
$E(I) = -\hat{i} \cdot (E_d\, \hat{d} + E_\rho\, \hat{\rho})$.

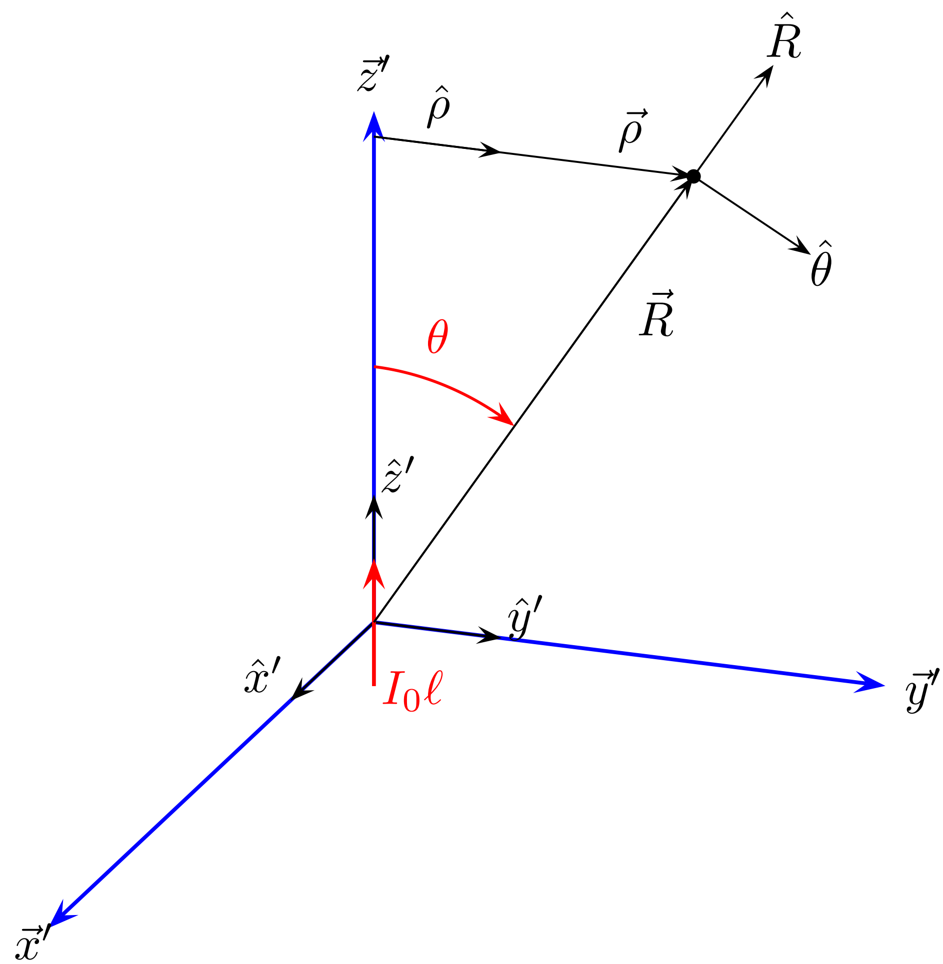


Figure 5. Coordinate Parameters for Current Element.

ET164-ET225	P1,P2,P3 = x,y,z coordinates of current element ($\vec{D}$)
	P4 = a
	P5 = b
	P6 = $I_0\, l/\lambda^2$
ET164-ET169	WX,WY,WZ = x,y and z components of $\hat{d}$
	DS = $(\eta/2\pi)I_0\, l/\lambda^2$
	DSH = $(1/4\pi)I_0\, l/\lambda^2$
ET173	Start of loop over all segments and patches.
ET176-ET179	For patches,
	IS = location of patch data in geometry arrays
	I1,I2 = locations to be filled in E
ET180-ET182	PX,PY,PZ = $\vec{R}/\lambda$
ET183-ET193	R = $\lvert\vec{R}/\lambda\rvert$
	PX,PY,PZ = $\hat{R}$
	GTH = cos θ
	STH = sin θ
	QX,QY QZ = $\hat{R} - (\hat{d}\cdot\hat{R})\hat{d}$
ET196-ET204	QX,QY,QZ = $\hat{\rho}$
	T1 = exp(-jkR)
ET206-ET215	E field on segments
	T2 = (1 - j/kR)λ^2/R^2
	ER = E_R
	ET = E_θ
	ERH = E_ρ
	EZH = E_Z
	CX,CY,CZ = x,y,z components of total E field
ET216-ET224	H field on patches
	PX,PY,PZ = $\hat{d}\times\hat{\rho} = \hat{\Phi}$
	T2 = $\pm H_\Phi$
	CX,CY,CZ = $\pm H^I$

1.E-30	= tolerance in test for zero
2.654420938E-3	= $1/\eta = \sqrt{\epsilon_o/\mu_0}$
59.958	= $\eta/2\pi$
6.283185308	= 2π

```
      SUBROUTINE ETMNS( P1, P2, P3, P4, P5, P6, IPR, E)
C
C     ETMNS FILLS THE ARRAY E WITH THE NEGATIVE OF THE ELECTRIC FIELD
C     INCIDENT ON THE STRUCTURE.  E IS THE RIGHT HAND SIDE OF THE MATRIX
C     EQUATION.
C
      IMPLICIT REAL (A-H,O-Z)
      COMPLEX  E, CX, CY, CZ, VSANT, ER, ET, EZH, ERH, VQD
     *, VQDS, ZRATI, ZRATI2, RRV, RRH, T1, TT1, TT2, FRATI
      COMMON  /DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X( NM), Y( NM),
     *Z( NM), SI( NM), BI( NM), ALP( NM), BET( NM), ICON1( N2M), ICON2(
     * N2M), ITAG( N2M), ICONX( NM), WLAM, IPSYM
      COMMON  /ANGL/ SALP( NM)
      COMMON  /VSORC/ VQD(30), VSANT(30), VQDS(30), IVQD(30), ISANT(30)
     *, IQDS(30), NVQD, NSANT, NQDS
      COMMON  /GND/ ZRATI, ZRATI2, FRATI, CL, CH, SCRWL, SCRWR, NRADL,
     *KSYMP, IFAR, IPERF, T1, T2
      DIMENSION  CAB(1), SAB(1), E( N2M)
      DIMENSION  T1X(1), T1Y(1), T1Z(1), T2X(1), T2Y(1), T2Z(1)
      EQUIVALENCE(CAB,ALP),(SAB,BET)
      EQUIVALENCE(T1X,SI),(T1Y,ALP),(T1Z,BET),(T2X,ICON1),(T2Y,ICON2),(
     *T2Z,ITAG)
      DATA   TP/6.283185308D+0/, RETA/2.654420938D-3/
      NEQ= N+2* M
      NQDS=0
C
C     APPLIED FIELD OF VOLTAGE SOURCES FOR TRANSMITTING CASE
C
      IF(IPR.GT.0.AND. IPR.NE.5) GOTO 5
      DO 1  I=1, NEQ
    1 E( I)=(0.,0.)
      IF(NSANT.EQ.0) GOTO 3
      DO 2  I=1, NSANT
      IS= ISANT( I)
    2 E( IS)=- VSANT( I)/( SI( IS)* WLAM)
    3 IF(NVQD.EQ.0) RETURN
      DO 4  I=1, NVQD
      IS= IVQD( I)
    4 CALL QDSRC( IS, VQD( I), E)
      RETURN
C
C     INCIDENT PLANE WAVE, LINEARLY POLARIZED.
C
    5 IF(IPR.GT.3) GOTO 19
      CTH= COS( P1)
      STH= SIN( P1)
      CPH= COS( P2)
      SPH= SIN( P2)
      CET= COS( P3)
```

```
      SET= SIN( P3)
      PX= CTH* CPH* CET- SPH* SET
      PY= CTH* SPH* CET+ CPH* SET
      PZ=- STH* CET
      WX=- STH* CPH
      WY=- STH* SPH
      WZ=- CTH
      QX= WY* PZ- WZ* PY
      QY= WZ* PX- WX* PZ
      QZ= WX* PY- WY* PX
      IF(KSYMP.EQ.1) GOTO 7
      IF(IPERF.EQ.1) GOTO 6
      RRV= SQRT(1.- ZRATI* ZRATI* STH* STH)
      RRH= ZRATI* CTH
      RRH=( RRH- RRV)/( RRH+ RRV)
      RRV= ZRATI* RRV
      RRV=-( CTH- RRV)/( CTH+ RRV)
      GOTO 7
    6 RRV=-(1.,0.)
      RRH=-(1.,0.)
    7 IF(IPR.GT.1) GOTO 13
      IF(N.EQ.0) GOTO 10
      DO 8  I=1, N
      ARG=- TP*( WX* X( I)+ WY* Y( I)+ WZ* Z( I))
    8 E( I)=-( PX* CAB( I)+ PY* SAB( I)+ PZ* SALP( I))* CMPLX( COS( ARG
     *), SIN( ARG))
      IF(KSYMP.EQ.1) GOTO 10
      TT1=( PY* CPH- PX* SPH)*( RRH- RRV)
      CX= RRV* PX- TT1* SPH
      CY= RRV* PY+ TT1* CPH
      CZ=- RRV* PZ
      DO 9  I=1, N
      ARG=- TP*( WX* X( I)+ WY* Y( I)- WZ* Z( I))
    9 E( I)= E( I)-( CX* CAB( I)+ CY* SAB( I)+ CZ* SALP( I))* CMPLX(
     *COS( ARG), SIN( ARG))
   10 IF(M.EQ.0) RETURN
      I= LD+1
      I1= N-1
      DO 11  IS=1, M
      I= I-1
      I1= I1+2
      I2= I1+1
      ARG=- TP*( WX* X( I)+ WY* Y( I)+ WZ* Z( I))
      TT1= CMPLX( COS( ARG), SIN( ARG))* SALP( I)* RETA
      E( I2)=( QX* T1X( I)+ QY* T1Y( I)+ QZ* T1Z( I))* TT1
   11 E( I1)=( QX* T2X( I)+ QY* T2Y( I)+ QZ* T2Z( I))* TT1
      IF(KSYMP.EQ.1) RETURN
      TT1=( QY* CPH- QX* SPH)*( RRV- RRH)
      CX=-( RRH* QX- TT1* SPH)
```

```
      CY=-( RRH* QY+ TT1* CPH)
      CZ= RRH* QZ
      I= LD+1
      I1= N-1
      DO 12  IS=1, M
      I= I-1
      I1= I1+2
      I2= I1+1
      ARG=- TP*( WX* X( I)+ WY* Y( I)- WZ* Z( I))
      TT1= CMPLX( COS( ARG), SIN( ARG))* SALP( I)* RETA
      E( I2)= E( I2)+( CX* T1X( I)+ CY* T1Y( I)+ CZ* T1Z( I))* TT1
   12 E( I1)= E( I1)+( CX* T2X( I)+ CY* T2Y( I)+ CZ* T2Z( I))* TT1
C
C     INCIDENT PLANE WAVE, ELLIPTIC POLARIZATION.
C
      RETURN
   13 TT1=-(0.,1.)* P6
      IF(IPR.EQ.3) TT1=- TT1
      IF(N.EQ.0) GOTO 16
      CX= PX+ TT1* QX
      CY= PY+ TT1* QY
      CZ= PZ+ TT1* QZ
      DO 14  I=1, N
      ARG=- TP*( WX* X( I)+ WY* Y( I)+ WZ* Z( I))
   14 E( I)=-( CX* CAB( I)+ CY* SAB( I)+ CZ* SALP( I))* CMPLX( COS( ARG
     *), SIN( ARG))
      IF(KSYMP.EQ.1) GOTO 16
      TT2=( CY* CPH- CX* SPH)*( RRH- RRV)
      CX= RRV* CX- TT2* SPH
      CY= RRV* CY+ TT2* CPH
      CZ=- RRV* CZ
      DO 15  I=1, N
      ARG=- TP*( WX* X( I)+ WY* Y( I)- WZ* Z( I))
   15 E( I)= E( I)-( CX* CAB( I)+ CY* SAB( I)+ CZ* SALP( I))* CMPLX(
     *COS( ARG), SIN( ARG))
   16 IF(M.EQ.0) RETURN
      CX= QX- TT1* PX
      CY= QY- TT1* PY
      CZ= QZ- TT1* PZ
      I= LD+1
      I1= N-1
      DO 17  IS=1, M
      I= I-1
      I1= I1+2
      I2= I1+1
      ARG=- TP*( WX* X( I)+ WY* Y( I)+ WZ* Z( I))
      TT2= CMPLX( COS( ARG), SIN( ARG))* SALP( I)* RETA
      E( I2)=( CX* T1X( I)+ CY* T1Y( I)+ CZ* T1Z( I))* TT2
   17 E( I1)=( CX* T2X( I)+ CY* T2Y( I)+ CZ* T2Z( I))* TT2
```

```
      IF(KSYMP.EQ.1) RETURN                                             ET 148
      TT1=( CY* CPH- CX* SPH)*( RRV- RRH)                               ET 149
      CX=-( RRH* CX- TT1* SPH)                                          ET 150
      CY=-( RRH* CY+ TT1* CPH)                                          ET 151
      CZ= RRH* CZ                                                       ET 152
      I= LD+1                                                           ET 153
      I1= N-1                                                           ET 154
      DO 18  IS=1, M                                                    ET 155
      I= I-1                                                            ET 156
      I1= I1+2                                                          ET 157
      I2= I1+1                                                          ET 158
      ARG=- TP*( WX* X( I)+ WY* Y( I)- WZ* Z( I))                       ET 159
      TT1= CMPLX( COS( ARG), SIN( ARG))* SALP( I)* RETA                 ET 160
      E( I2)= E( I2)+( CX* T1X( I)+ CY* T1Y( I)+ CZ* T1Z( I))* TT1      ET 161
   18 E( I1)= E( I1)+( CX* T2X( I)+ CY* T2Y( I)+ CZ* T2Z( I))* TT1      ET 162
C                                                                       ET 163
C     INCIDENT FIELD OF AN ELEMENTARY CURRENT SOURCE.                   ET 164
C                                                                       ET 165
      RETURN                                                            ET 166
   19 WZ= COS( P4)                                                      ET 167
      WX= WZ* COS( P5)                                                  ET 168
      WY= WZ* SIN( P5)                                                  ET 169
      WZ= SIN( P4)                                                      ET 170
      DS= P6*59.958                                                     ET 171
      DSH= P6/(2.* TP)                                                  ET 172
      NPM= N+ M                                                         ET 173
      IS= LD+1                                                          ET 174
      I1= N-1                                                           ET 175
      DO 24  I=1, NPM                                                   ET 176
      II= I                                                             ET 177
      IF(I.LE. N) GOTO 20                                               ET 178
      IS= IS-1                                                          ET 179
      II= IS                                                            ET 180
      I1= I1+2                                                          ET 181
      I2= I1+1                                                          ET 182
   20 PX= X( II)- P1                                                    ET 183
      PY= Y( II)- P2                                                    ET 184
      PZ= Z( II)- P3                                                    ET 185
      RS= PX* PX+ PY* PY+ PZ* PZ                                        ET 186
      IF(RS.LT.1.D-30) GOTO 24                                          ET 187
      R= SQRT( RS)                                                      ET 188
      PX= PX/ R                                                         ET 189
      PY= PY/ R                                                         ET 190
      PZ= PZ/ R                                                         ET 191
      CTH= PX* WX+ PY* WY+ PZ* WZ                                       ET 192
      STH= SQRT(1.- CTH* CTH)                                           ET 193
      QX= PX- WX* CTH                                                   ET 194
      QY= PY- WY* CTH                                                   ET 195
      QZ= PZ- WZ* CTH                                                   ET 196
```

```
      ARG= SQRT( QX* QX+ QY* QY+ QZ* QZ)                                ET 197
      IF(ARG.LT.1.D-30) GOTO 21                                         ET 198
      QX= QX/ ARG                                                       ET 199
      QY= QY/ ARG                                                       ET 200
      QZ= QZ/ ARG                                                       ET 201
      GOTO 22                                                           ET 202
   21 QX=1.                                                             ET 203
      QY=0.                                                             ET 204
      QZ=0.                                                             ET 205
   22 ARG=- TP* R                                                       ET 206
      TT1= CMPLX( COS( ARG), SIN( ARG))                                 ET 207
      IF(I.GT. N) GOTO 23                                               ET 208
      TT2= CMPLX(1.D+0,-1.D+0/( R* TP))/ RS                             ET 209
      ER= DS* TT1* TT2* CTH                                             ET 210
      ET=.5* DS* TT1*((0.,1.)* TP/ R+ TT2)* STH                         ET 211
      EZH= ER* CTH- ET* STH                                             ET 212
      ERH= ER* STH+ ET* CTH                                             ET 213
      CX= EZH* WX+ ERH* QX                                              ET 214
      CY= EZH* WY+ ERH* QY                                              ET 215
      CZ= EZH* WZ+ ERH* QZ                                              ET 216
      E( I)=-( CX* CAB( I)+ CY* SAB( I)+ CZ* SALP( I))                  ET 217
      GOTO 24                                                           ET 218
   23 PX= WY* QZ- WZ* QY                                                ET 219
      PY= WZ* QX- WX* QZ                                                ET 220
      PZ= WX* QY- WY* QX                                                ET 221
      TT2= DSH* TT1* CMPLX(1./ R, TP)/ R* STH* SALP( II)                ET 222
      CX= TT2* PX                                                       ET 223
      CY= TT2* PY                                                       ET 224
      CZ= TT2* PZ                                                       ET 225
      E( I2)= CX* T1X( II)+ CY* T1Y( II)+ CZ* T1Z( II)                  ET 226
      E( I1)= CX* T2X( II)+ CY* T2Y( II)+ CZ* T2Z( II)                  ET 227
   24 CONTINUE                                                          ET 228
      RETURN                                                            ET 229
      END                                                               ET 230
```

FACGF

PURPOSE

To perform the steps in the NGF solution that do not depend on the excitation vector.

METHOD

The NGF solution procedure is discussed in Section VI. The steps performed in FACGF are to evaluate $A^{-1}B$ and D - $CA^{-1}B$. The matrix D - $CA^{-1}B$ is then factored into triangular matrices L and U. The procedure is complicated by the possible need to use file storage for the matrices. The comments in the code and the tables for ICASX = 2, 3 and 4 in Section VII offer a fairly complete description of the procedure.

SYMBOL DICTIONARY

A	=	array for matrix A (L U factors) or block of A if file storage is used
B	=	array for B or block of B
BX	=	array for B when $A^{-l}B$ is being computed with ICASX = 2. The array B starts at the beginning of GM in this case. BX leaves room for AF at the beginning of CM
C	=	array for C or block of C (matrix transposed)
D	=	array for D or block of D (matrix transposed)
IBFL	=	file in which B is stored
ICASS	=	saved value of ICASE
IP	=	pivot index array
IX	=	data on row interchanges in LFACTR.
M1	=	number of patches in the NGF
MP	=	number of patches in a symmetric section in the NGF
N1	=	number of segments in the NGF
N1C	=	number of columns in C (same as order of A)
N1CP	=	N1C + 1
N2C	=	order of matrix D
NBLSYS	=	saved value of NBLSYM
NIC	=	index increment
NLSYS	=	saved value of NLSYM
NP	=	number of segments in a symmetric section in the NGF
SYS	=	saved value of NPSYM
JM	=	summation variable for matrix products

```
      SUBROUTINE FACGF( A, B, C, D, BX, IP, IX, NP, N1, MP, M1, N1C,
     *N2C)
C     FACGF COMPUTES AND FACTORS D-C(INV(A)B).
      IMPLICIT REAL (A-H,O-Z)
      COMPLEX  A, B, C, D, BX, SUM
      COMMON  /MATPAR/ ICASE, NBLOKS, NPBLK, NLAST, NBLSYM, NPSYM,
     *NLSYM, IMAT, ICASX, NBBX, NPBX, NLBX, NBBL, NPBL, NLBL
      DIMENSION  A(1), B( N1C,1), C( N1C,1), D( N2C,1), BX( N1C,1), IP(
     *1), IX(1)
      IF(N2C.EQ.0) RETURN
      IBFL=14
C     CONVERT B FROM BLOCKS OF ROWS ON T14 TO BLOCKS OF COL. ON T16
      IF(ICASX.LT.3) GOTO 1
      CALL REBLK( B, C, N1C, NPBX, N2C)
      IBFL=16
    1 NPB= NPBL
C     COMPUTE INV(A)B AND WRITE ON TAPE14
      IF(ICASX.EQ.2) REWIND 14
      DO 2  IB=1, NBBL
      IF(IB.EQ. NBBL) NPB= NLBL
      IF(ICASX.GT.1) READ(IBFL) (( BX( I, J), I=1, N1C), J=1, NPB)
      CALL SOLVES( A, IP, BX, N1C, NPB, NP, N1, MP, M1,13,13)
      IF(ICASX.EQ.2) REWIND 14
      IF(ICASX.GT.1) WRITE( 14) (( BX( I, J), I=1, N1C), J=1, NPB)
    2 CONTINUE
      IF(ICASX.EQ.1) GOTO 3
      REWIND 11
      REWIND 12
      REWIND 15
      REWIND IBFL
C     COMPUTE D-C(INV(A)B) AND WRITE ON TAPE11
    3 NPC= NPBL
      DO 8  IC=1, NBBL
      IF(IC.EQ. NBBL) NPC= NLBL
      IF(ICASX.EQ.1) GOTO 4
      READ(15) (( C( I, J), I=1, N1C), J=1, NPC)
      READ(12) (( D( I, J), I=1, N2C), J=1, NPC)
      REWIND 14
    4 NPB= NPBL
      NIC=0
      DO 7  IB=1, NBBL
      IF(IB.EQ. NBBL) NPB= NLBL
      IF(ICASX.GT.1) READ(14) (( B( I, J), I=1, N1C), J=1, NPB)
      DO 6  I=1, NPB
      II= I+ NIC
      DO 6  J=1, NPC
      SUM=(0.,0.)
      DO 5  K=1, N1C
    5 SUM= SUM+ B( K, I)* C( K, J)
```

```
    6 D( II, J)= D( II, J)- SUM
    7 NIC= NIC+ NPBL
      IF(ICASX.GT.1) WRITE( 11) (( D( I, J), I=1, N2C), J=1, NPBL)
    8 CONTINUE
      IF(ICASX.EQ.1) GOTO 9
      REWIND 11
      REWIND 12
      REWIND 14
      REWIND 15
C     FACTOR D-C(INV(A)B)
    9 N1CP= N1C+1
      IF(ICASX.GT.1) GOTO 10
      CALL FACTR( N2C, D, IP( N1CP), N2C)
      GOTO 13
   10 IF(ICASX.EQ.4) GOTO 12
      NPB= NPBL
      IC=0
      DO 11  IB=1, NBBL
      IF(IB.EQ. NBBL) NPB= NLBL
      II= IC+1
      IC= IC+ N2C* NPB
   11 READ(11) ( B( I,1), I= II, IC)
      REWIND 11
      CALL FACTR( N2C, B, IP( N1CP), N2C)
      NIC= N2C* N2C
      WRITE( 11) ( B( I,1), I=1, NIC)
      REWIND 11
      GOTO 13
   12 NBLSYS= NBLSYM
      NPSYS= NPSYM
      NLSYS= NLSYM
      ICASS= ICASE
      NBLSYM= NBBL
      NPSYM= NPBL
      NLSYM= NLBL
      ICASE=3
      CALL FACIO( B, N2C,1, IX( N1CP),11,12,16,11)
      CALL LUNSCR( B, N2C,1, IP( N1CP), IX( N1CP),12,11,16)
      NBLSYM= NBLSYS
      NPSYM= NPSYS
      NLSYM= NLSYS
      ICASE= ICASS
   13 RETURN
      END
```

FACIO

PURPOSE

To read and write matrix blocks needed for the LU decomposition;

METHOD

Sequential access is used on all files. The matrix is initially stored in file IU1 in blocks of columns of the transposed matrix. The block size is such that two blocks will fit into the array A for the Gauss elimination process. If the matrix were divided into four blocks, the order for reading the blocks into core would be

Blocks	
1, 2	1 and 2 will be completely factored
1, 3	3 and 4 partially factored
1, 4	
2, 3	factorization of 3 completed
2, 4	4 partially factored
3, 4	factorization complete

IU1 is the initial input file. Partially factored blocks are read from file IFILE3 and written to IFILE4 where FILE3 = IU3 and IFILE4 = IU4 when IXBLK1 is odd, and IFILE3 = IU4 and IFILE4 = IU3 when IXBLK1 is even. Completed blocks are written to file IU2. Although the last block may be shorter than other blocks the same number of words is read or written. The excess words are ignored in subroutine LFACTR.

Subroutine LFACTR is called to perform the Gauss elimination. For a symmetric structure the loop from F018 to F043 factors each submatrix.

SYMBOL DICTIONARY

A	=	array for matrix storage
I1	=	location in A of beginning of block 1
I2	=	location in A of end of block 1
I3	=	location in A of beginning of block 2
I4	=	location in A of end of block 2
IFILE3	=	input file
IFILE4	=	output file
IP	=	array for pivot element indices
IT	=	number of words in a matrix block
IU1,IU2,IU3,IU4	=	file numbers
IXBLK1	=	number of first block stored in A
IXBLK2	=	number of second block stared in A
KA	=	first Location in IP for submatrix KK
NBM	=	number of blocks minus one
NOP	=	number of submatrices for symmetry
NROW	=	number of rows in a block
T1,T2,TIME	=	variables to sum total time spent in LFACTR

```
      SUBROUTINE FACIO( A, NROW, NOP, IP, IU1, IU2, IU3, IU4)
C
C     FACIO CONTROLS I/O FOR OUT-OF-CORE FACTORIZATION
C
      IMPLICIT REAL (A-H,O-Z)
      COMPLEX  A
      COMMON  /MATPAR/ ICASE, NBLOKS, NPBLK, NLAST, NBLSYM, NPSYM,
     *NLSYM, IMAT, ICASX, NBBX, NPBX, NLBX, NBBL, NPBL, NLBL
      DIMENSION  A( NROW,1), IP( NROW)
      IT=2* NPSYM* NROW
      NBM= NBLSYM-1
      I1=1
      I2= IT
      I3= I2+1
      I4=2* IT
      TIME=0.
      REWIND IU1
      REWIND IU2
      DO 3  KK=1, NOP
      KA=( KK-1)* NROW+1
      IFILE3= IU1
      IFILE4= IU3
      DO 2  IXBLK1=1, NBM
      REWIND IU3
      REWIND IU4
      CALL BLCKIN( A, IFILE3, I1, I2,1,17)
      IXBP= IXBLK1+1
      DO 1  IXBLK2= IXBP, NBLSYM
      CALL BLCKIN( A, IFILE3, I3, I4,1,18)
      CALL SECONDS( T1)
      CALL LFACTR( A, NROW, IXBLK1, IXBLK2, IP( KA))
      CALL SECONDS( T2)
      TIME= TIME+ T2- T1
      IF(IXBLK2.EQ. IXBP) CALL BLCKOT( A, IU2, I1, I2,1,19)
      IF(IXBLK1.EQ. NBM.AND. IXBLK2.EQ. NBLSYM) IFILE4= IU2
      CALL BLCKOT( A, IFILE4, I3, I4,1,20)
    1 CONTINUE
      IFILE3= IU3
      IFILE4= IU4
      IF(( IXBLK1/2)*2.NE. IXBLK1) GOTO 2
      IFILE3= IU4
      IFILE4= IU3
    2 CONTINUE
    3 CONTINUE
      REWIND IU1
      REWIND IU2
      REWIND IU3
      REWIND IU4
      WRITE (2,4)  TIME
```

```
C                                                                 FO  50
      RETURN                                                      FO  51
    4 FORMAT(' CP TIME TAKEN FOR FACTORIZATION = ',1P,E12.5)      FO  52
      END                                                         FO  53
```

FACTR

PURPOSE

To factor a complex matrix into a lower triangular and an upper triangular matrix using the Gauss-Doolittle technique. The matrix in this case is a transposed matrix. The factored matrix is used by subroutine SOLVE to determine the solution of the matrix equation Ax = B.

METHOD

The algorithm used in this routine is presented by A. Kalston (ref. 1). The decomposition of the matrix A is such that A = LU, where L is a lower triangular matrix with l's down the diagonal, and U is an upper triangular matrix. The L and U matrices overwrite the matrix A. The computations to obtain L and U are done using one complex scratch vector (D) and one integer vector (IF) that keep track of row interchanges when elements are positioned for size. If positioning for size is not taken into account, the general procedure is

$$a_{11} = u_{11}$$

$$a_{i1} = \ell_{il}\, u_{11} \qquad \text{i=2,...,n}$$

which gives the first column of the L and U matrices. Then

$$a_{12} = u_{12}$$

$$a_{22} = \ell_{21}\, u_{12} + u_{22}$$

$$a_{i2} = \ell_{i1}\, u_{12} + \ell_{12}\, u_{22} \qquad \text{i=3,...,n}$$

gives the second column. The computations for the successive columns continue in this way. The general equations for the r-th column are

$$a_{1r} = u_{1r}$$

$$a_{2r} = \ell_{21}\, u_{1r} + u_{2r}$$

$$\vdots$$

$$a_{rr} = \ell_{rl} u_{1r} + \ell_{r2} u_{2r} + ... + \ell_{r,r-1} u_{r-1,r} + u_{rr}$$

$$a_{ir} = \ell_{ir} u_{1r} + \ell_{ir} u_{2r} + ... + \ell_{ir} u_{rr}, \qquad i = r+1, ..., n$$

There are only two differences in the coding used in FACTR and the coding suggested by Ralston. The first is that double precision variables are not used for the accumulation of sums, since for the size and conditioning of the matrices anticipated in core, the computer word length is sufficient to insure accuracy. The second difference is that the row and column indices of the A matrix in the routine have been interchanged to handle the transposed matrix.

CODING

The coding is divided into five steps which correspond to the steps given by Ralston.

FA14	Loop over columns (rows with the interchanged indices used in the routine).
FA18-FA20	Fill D vector with column (row) of A.
FA24-FA35	Solution for u_{ir} (i = 1,...,r) in the above equations taking into account positioning.
FA40-FA54	Selecting largest value for positioning.
FA56-FA62	Solution for l_{ir} (i = r + 1,...,n) in the above equations.
FA64-FA66	Printing of small pivot elements.

SYMBOL DICTIONARY

A	=	input transposed matrix overwritten with calculated L^T and U^T matrices
CONJG	=	external routine (conjugate of a complex number)
D	=	scratch vector
DMAX	=	maximum value in D
ELMAG	=	intermediate variable
I	=	DO loop index
IFLG	=	small pivot flag
IP	=	integer vector storing positioning information
J	=	DO loop index
JPI	=	J + 1
K	=	DO loop index
N	=	order of matrix being factored
NDIM	=	dimensions of the array where the matrix is stored. NDIM $\geq$ N
PJ	=	intermediate variable
PR	=	intermediate variable
R	=	DO loop index
REAL	=	external routine (real part of complex number)
RM1	=	R - 1
RP1	=	R + 1

```
      SUBROUTINE FACTR( N, A, IP, NDIM)
C
C     SUBROUTINE TO FACTOR A MATRIX INTO A UNIT LOWER TRIANGULAR MATRIX
C     AND AN UPPER TRIANGULAR MATRIX USING THE GAUSS-DOOLITTLE ALGORITHM
C     PRESENTED ON PAGES 411-416 OF A. RALSTON--A FIRST COURSE IN
C     NUMERICAL ANALYSIS.  COMMENTS BELOW REFER TO COMMENTS IN RALSTONS
C     TEXT.    (MATRIX TRANSPOSED.
C
      IMPLICIT REAL (A-H,O-Z)
      COMPLEX  A, D, ARJ
      DIMENSION  A( NDIM, NDIM), IP( NDIM)
      COMMON  /SCRATM/ D( N2M)
      INTEGER  R, RM1, RP1, PJ, PR
      IFLG=0
C
C     STEP 1
C
      DO 9  R=1, N
      DO 1  K=1, N
      D( K)= A( R, K)
C
C     STEPS 2 AND 3
C
    1 CONTINUE
      RM1= R-1
      IF(RM1.LT.1) GOTO 4
      DO 3  J=1, RM1
      PJ= IP( J)
      ARJ= D( PJ)
      A( R, J)= ARJ
      D( PJ)= D( J)
      JP1= J+1
      DO 2  I= JP1, N
      D( I)= D( I)- A( J, I)* ARJ
    2 CONTINUE
    3 CONTINUE
C
C     STEP 4
C
    4 CONTINUE
      DMAX= REAL( D( R)* CONJG( D( R)))
      IP( R)= R
      RP1= R+1
      IF(RP1.GT. N) GOTO 6
      DO 5  I= RP1, N
      ELMAG= REAL( D( I)* CONJG( D( I)))
      IF(ELMAG.LT. DMAX) GOTO 5
      DMAX= ELMAG
      IP( R)= I
```

```
    5 CONTINUE
    6 CONTINUE
      IF(DMAX.LT.1.D-10) IFLG=1
      PR= IP( R)
      A( R, R)= D( PR)
C
C     STEP 5
C
      D( PR)= D( R)
      IF(RP1.GT. N) GOTO 8
      ARJ=1./ A( R, R)
      DO 7  I= RP1, N
      A( R, I)= D( I)* ARJ
    7 CONTINUE
    8 CONTINUE
      IF(IFLG.EQ.0) GOTO 9
      WRITE (2,10)  R, DMAX
      IFLG=0
    9 CONTINUE
C
      RETURN
   10 FORMAT(1H ,'PIVOT(',I3,')=',1P,E16.8)
      END
```

<u>FACTRS</u>

PURPOSE

To call the appropriate subroutines for the LU decomposition of a matrix.

METHOD

The operation of FACTRS depends on the mode of storage of the matrix as determined by the value of ICASE (see COMMON/MATPAR/ in Section III). For ICASE = 1 subroutine FACTR is called at FS16 to factor the matrix. For ICASE = 2 FACTR is called for each of the NOP submstrices. If ICASE = 3 FACIO and LUNSCR are called at FS23 and FS24. FACIO reads the matrix from file IU1 and writes the result on file IU2. LUNSCR leaves the final result on file IU3.

For ICASE = 4 (symmetry, submatrices fit in core) or ICASE = 5 (symmetry, submatrices do not fit in core) the matrix elements on file IU1 are written in a new order on file IU2 from FS29 to FS46. The sequence of data an file IU1 is

```
      column 1 of submatrix 1
      column 1 of submatrix 2
                 .
                 .
                 .
     column 1 of submatrix NOP
      column 2 of submatrix 1
                 .
                 .
                 .
     column 2 of submatrix NOP
      column 3 of submatrix I
                 .
                 .
                 .
   column NPBLK of submatrix NOP
```

The matrices are written onto file IU2 in the sequence

```
      column 1 of submatrix 1
      column 2 of submatrix 1
                 .
                 .
                 .
    column NPBLK of submatrix 1
      column 1 of submatrix 2
                 .
                 .
                 .
   column NPBLK of submatrix NOP
```

For ICASE = 4 each submatrix is then read into memory at FS58 and decomposed into

LU factors by calling FACTR at FS60. The factored matrices are written to file IU3 at FS61.

For ICASE = 5 the matrices are transferred from file IU2 to IU1 at FS76 to FS77. Subroutine FACIO is then called to factor all of the NOP submatrices. The result is left on file IU2. LUNSCR reorders the rows of each matrix and leaves the result on IU3.

SYMBOL DICTIONARY

A	=	array for matrix storage
I2	=	number of words in a block
ICOLS	=	number oE columns in a block
IP	=	array for pivot element indices
IRl,IR2,IRR1,IRR2	=	row indices for reordering columns
IU1,IU2,IU3,IU4	=	file numbers
IX	=	array of pivot element data
KA	=	starting location of a submetrix in the array
NOP	=	number of symmetric sections
NP	=	number of equations for each symmetric section (order of submatrix)
NROW	=	total number of equations (NP x NOP)

```
      SUBROUTINE FACTRS( NP, NROW, A, IP, IX, IU1, IU2, IU3, IU4)
C
C     FACTRS, FOR SYMMETRIC STRUCTURE, TRANSFORMS SUBMATRICIES TO FORM
C     MATRICIES OF THE SYMMETRIC MODES AND CALLS ROUTINE TO FACTOR
C     MATRICIES.  IF NO SYMMETRY, THE ROUTINE IS CALLED TO FACTOR THE
C     COMPLETE MATRIX.
C
      IMPLICIT REAL (A-H,O-Z)
      COMPLEX  A
      COMMON  /MATPAR/ ICASE, NBLOKS, NPBLK, NLAST, NBLSYM, NPSYM,
     *NLSYM, IMAT, ICASX, NBBX, NPBX, NLBX, NBBL, NPBL, NLBL
      DIMENSION  A(1), IP( NROW), IX( NROW)
      NOP= NROW/ NP
      IF(ICASE.GT.2) GOTO 2
      DO 1  KK=1, NOP
      KA=( KK-1)* NP+1
    1 CALL FACTR( NP, A( KA), IP( KA), NROW)
      RETURN
C
C     FACTOR SUBMATRICIES, OR FACTOR COMPLETE MATRIX IF NO SYMMETRY
C     EXISTS.
C
    2 IF(ICASE.GT.3) GOTO 3
      CALL FACIO( A, NROW, NOP, IX, IU1, IU2, IU3, IU4)
      CALL LUNSCR( A, NROW, NOP, IP, IX, IU2, IU3, IU4)
C
C     REWRITE THE MATRICES BY COLUMNS ON TAPE 13
C
      RETURN
    3 I2=2* NPBLK* NROW
      REWIND IU2
      DO 5  K=1, NOP
      REWIND IU1
      ICOLS= NPBLK
      IR2= K* NP
      IR1= IR2- NP+1
      DO 5  L=1, NBLOKS
      IF(NBLOKS.EQ.1.AND. K.GT.1) GOTO 4
      CALL BLCKIN( A, IU1,1, I2,1,602)
      IF(L.EQ. NBLOKS) ICOLS= NLAST
    4 IRR1= IR1
      IRR2= IR2
      DO 5  ICOLDX=1, ICOLS
      WRITE( IU2) ( A( I), I= IRR1, IRR2)
      IRR1= IRR1+ NROW
      IRR2= IRR2+ NROW
    5 CONTINUE
      REWIND IU1
      REWIND IU2
```

```
      IF(ICASE.EQ.5) GOTO 8                                             FS  50
      REWIND IU3                                                        FS  51
      IRR1= NP* NP                                                      FS  52
      DO 7  KK=1, NOP                                                   FS  53
      IR1=1- NP                                                         FS  54
      IR2=0                                                             FS  55
      DO 6  I=1, NP                                                     FS  56
      IR1= IR1+ NP                                                      FS  57
      IR2= IR2+ NP                                                      FS  58
    6 READ(IU2) ( A( J), J= IR1, IR2)                                   FS  59
      KA=( KK-1)* NP+1                                                  FS  60
      CALL FACTR( NP, A, IP( KA), NP)                                   FS  61
      WRITE( IU3) ( A( I), I=1, IRR1)                                   FS  62
    7 CONTINUE                                                          FS  63
      REWIND IU2                                                        FS  64
      REWIND IU3                                                        FS  65
      RETURN                                                            FS  66
    8 I2=2* NPSYM* NP                                                   FS  67
      DO 10  KK=1, NOP                                                  FS  68
      J2= NPSYM                                                         FS  69
      DO 10  L=1, NBLSYM                                                FS  70
      IF(L.EQ. NBLSYM) J2= NLSYM                                        FS  71
      IR1=1- NP                                                         FS  72
      IR2=0                                                             FS  73
      DO 9  J=1, J2                                                     FS  74
      IR1= IR1+ NP                                                      FS  75
      IR2= IR2+ NP                                                      FS  76
    9 READ(IU2) ( A( I), I= IR1, IR2)                                   FS  77
   10 CALL BLCKOT( A, IU1,1, I2,1,193)                                  FS  78
      REWIND IU1                                                        FS  79
      CALL FACIO( A, NP, NOP, IX, IU1, IU2, IU3, IU4)                   FS  80
      CALL LUNSCR( A, NP, NOP, IP, IX, IU2, IU3, IU4)                   FS  81
      RETURN                                                            FS  82
      END                                                               FS  83
```

FBAR

PURPOSE

To compute the Sommerfeld attenuation function for Norton's asymptotic field approximations.

METHOD

The value returned for FBAR is

$$F(P) = 1 - j\sqrt{\pi P}\exp(-P)[1 - \mathtt{erf}(j\sqrt{P})]$$

where erf(z) is the error function. If $|j\sqrt{P}| \leq 3$ the value of erf(j$\sqrt{P}$) is computed from the series

$$\mathtt{erf}(z) = \frac{w}{\sqrt{\pi}}\sum_{n=0}^{\infty}\frac{(-1)^n z^{2n+1}}{n!(2n+1)}$$

For $|j\sqrt{P}|$ > 3, F(P) is evaluated from the first six terms of the asymptotic expansion

$$\sqrt{\pi}z\exp(z^2)(1 - \mathtt{erf}(z)) \approx 1 + \sum_{M=1}^{\infty}(-1)^M\frac{1\cdot 3\ldots(2M-1)}{(2z^2)^M}$$

for $z \to \infty$, $|\mathtt{arg}(z)| < 3\pi/4$.

SYMBOL DICTIONARY

ACCS	=	relative convergence test value
FJ	=	j = $\sqrt{-1}$
MINUS	=	1 if Re(z) < 0
P	=	P
POW	=	$(-1)^n z^{2n+1}/n!$
SMS	=	magnitude squared of series
SP	=	$\sqrt{\pi}$
SUM	=	series value
TERM	=	term in the series
TMS	=	$\lvert TERM\rvert^2$
TOSP	=	2/$\sqrt{\pi}$
Z	=	j$\sqrt{P}$
ZS	=	z^2

```
C       COMPLEX FUNCTION FBAR( P)
      FUNCTION FBAR( P)
C
C     FBAR IS SOMMERFELD ATTENUATION FUNCTION FOR NUMERICAL DISTANCE P
C
      COMPLEX  Z, ZS, SUM, POW, TERM, P, FJ, FBAR
      DIMENSION  FJX(2)
      EQUIVALENCE(FJ,FJX)
      DATA   TOSP/1.128379167D+0/, ACCS/1.D-12/, SP/1.772453851D+0/,
     *FJX/0.,1./
      Z= FJ* SQRT( P)
C
C     SERIES EXPANSION
C
      IF(ABS( Z).GT.3.) GOTO 3
      ZS= Z* Z
      SUM= Z
      POW= Z
      DO 1  I=1,100
      POW=- POW* ZS/ DFLOAT( I)
      TERM= POW/(2.* I+1.)
      SUM= SUM+ TERM
      TMS= REAL( TERM* CONJG( TERM))
      SMS= REAL( SUM* CONJG( SUM))
      IF(TMS/ SMS.LT. ACCS) GOTO 2
    1 CONTINUE
    2 FBAR=1.-(1.- SUM* TOSP)* Z* EXP( ZS)* SP
C
C     ASYMPTOTIC EXPANSION
C
      RETURN
    3 IF(REAL( Z).GE.0.) GOTO 4
      MINUS=1
      Z=- Z
      GOTO 5
    4 MINUS=0
    5 ZS=.5/( Z* Z)
      SUM=(0.,0.)
      TERM=(1.,0.)
      DO 6  I=1,6
      TERM=- TERM*(2.* I-1.)* ZS
    6 SUM= SUM+ TERM
      IF(MINUS.EQ.1) SUM= SUM-2.* SP* Z* EXP( Z* Z)
      FBAR=- SUM
      RETURN
      END
```

FBLOCK

PURPOSE

To set parameters for storage of the interaction matrix.

METHOD

FBLOCK sets values of the parameters ICASE through NLSYM in COMMON/MATPAR/. The input psrameters NROW and NCOL are the number of rows and columns in the non-transposed matrix. IMAX is the number of matrix elements that can be stored in the array in COMMON/CMB/. If a NGF file will be written (WG card) then IRNGF complex locations are reserved for future use. If a NGF file has not been requested then IRNGF is zero.

If (NROW)(NCOL) $\leq$ IMAX - IRNGF the complete matrix can be stored in COMMON/CMB/. ICASE is then 1 for no symmetry or 2 for symmetry. If the structure has symmetry and one submatrix fits in core but not the complete matrix,

$$\text{(NROW)(NCOL)} > \text{IMAX - IRNGF}$$
$$\text{NROW}^2 \leq \text{IMAX - IRNGF},$$

then ICASE is 4.

If the matrix cannot fit in core for the LU decomposition then it is divided into blocks of rows (columns of the transposed matrix) for transfer between core and file storage. The blocks are made as large as possible so that one block fits into IMAX - IRNGF locations and two blocks fit into IMAX locations. Since two blocks are needed in core only during the Gauss elimination process this makes at least IRNGF locations available during the NGF solution.

CODING

FB10-RB17	ICASE = 1 or 2
FB20-FB32	ICASE = 3
FB34-FB40	ICASE = 4 or 5, block parameters for whole matrix
FB42-FB48	ICASE 4, block parameters for submatrices
FB49-FB58	ICASE = 5, block parameters for submatrices
FB65-FB71	S matrix for rotational symmetry (Equation III of Part I)
FB75-FE88	S matrix for plane symmetry

SYMBOL DICTIONARY

ARG	=	2π(I - 1)(J - 1)/NOP
IMAX	=	number of complex numbers that can be stored in COMMON/CMB/
IMX1	=	IMAX - IRNGF
IPSYM	=	parameter from COMMON/DATA/
IRNGF	=	array storage reserved for NGF
KA	=	number of planes of symmetry
NCOL	=	number of columns in matrix
NOP	=	number of symmetric sections
NROW	=	number of rows in matrix
PHAZ	=	$2\pi/NOP$

```
      SUBROUTINE FBLOCK( NROW, NCOL, IMAX, IRNGF, IPSYM)
C     FBLOCK SETS PARAMETERS FOR OUT-OF-CORE SOLUTION FOR THE PRIMARY
C     MATRIX (A)
      COMPLEX  SSX, DETER
      COMMON  /MATPAR/ ICASE, NBLOKS, NPBLK, NLAST, NBLSYM, NPSYM,
     *NLSYM, IMAT, ICASX, NBBX, NPBX, NLBX, NBBL, NPBL, NLBL
      COMMON  /SMAT/ SSX(16,16)
      IMX1= IMAX- IRNGF
      IF(NROW* NCOL.GT. IMX1) GOTO 2
      NBLOKS=1
      NPBLK= NROW
      NLAST= NROW
      IMAT= NROW* NCOL
      IF(NROW.NE. NCOL) GOTO 1
      ICASE=1
      RETURN
    1 ICASE=2
      GOTO 5
    2 IF(NROW.NE. NCOL) GOTO 3
      ICASE=3
      NPBLK= IMAX/(2* NCOL)
      NPSYM= IMX1/ NCOL
      IF(NPSYM.LT. NPBLK) NPBLK= NPSYM
      IF(NPBLK.LT.1) GOTO 12
      NBLOKS=( NROW-1)/ NPBLK
      NLAST= NROW- NBLOKS* NPBLK
      NBLOKS= NBLOKS+1
      NBLSYM= NBLOKS
      NPSYM= NPBLK
      NLSYM= NLAST
      IMAT= NPBLK* NCOL
      WRITE (2,14)  NBLOKS, NPBLK, NLAST
      GOTO 11
    3 NPBLK= IMAX/ NCOL
      IF(NPBLK.LT.1) GOTO 12
      IF(NPBLK.GT. NROW) NPBLK= NROW
      NBLOKS=( NROW-1)/ NPBLK
      NLAST= NROW- NBLOKS* NPBLK
      NBLOKS= NBLOKS+1
      WRITE (2,14)  NBLOKS, NPBLK, NLAST
      IF(NROW* NROW.GT. IMX1) GOTO 4
      ICASE=4
      NBLSYM=1
      NPSYM= NROW
      NLSYM= NROW
      IMAT= NROW* NROW
      WRITE (2,15)
      GOTO 5
    4 ICASE=5
```

```
      NPSYM= IMAX/(2* NROW)
      NBLSYM= IMX1/ NROW
      IF(NBLSYM.LT. NPSYM) NPSYM= NBLSYM
      IF(NPSYM.LT.1) GOTO 12
      NBLSYM=( NROW-1)/ NPSYM
      NLSYM= NROW- NBLSYM* NPSYM
      NBLSYM= NBLSYM+1
      WRITE (2,16)  NBLSYM, NPSYM, NLSYM
      IMAT= NPSYM* NROW
    5 NOP= NCOL/ NROW
      IF(NOP* NROW.NE. NCOL) GOTO 13
C
C     SET UP SSX MATRIX FOR ROTATIONAL SYMMETRY.
C
      IF(IPSYM.GT.0) GOTO 7
      PHAZ=6.2831853072D+0/ NOP
      DO 6  I=2, NOP
      DO 6  J= I, NOP
      ARG= PHAZ* DFLOAT( I-1)* DFLOAT( J-1)
      SSX( I, J)= CMPLX( COS( ARG), SIN( ARG))
    6 SSX( J, I)= SSX( I, J)
C
C     SET UP SSX MATRIX FOR PLANE SYMMETRY
C
      GOTO 11
    7 KK=1
      SSX(1,1)=(1.,0.)
      IF(( NOP.EQ.2).OR.( NOP.EQ.4).OR.( NOP.EQ.8)) GOTO 8
      STOP
    8 KA= NOP/2
      IF(NOP.EQ.8) KA=3
      DO 10  K=1, KA
      DO 9  I=1, KK
      DO 9  J=1, KK
      DETER= SSX( I, J)
      SSX( I, J+ KK)= DETER
      SSX( I+ KK, J+ KK)=- DETER
    9 SSX( I+ KK, J)= DETER
   10 KK= KK*2
   11 RETURN
   12 WRITE (2,17)  NROW, NCOL
      STOP
   13 WRITE (2,18)  NROW, NCOL
C
      STOP
   14 FORMAT(//' MATRIX FILE STORAGE -  NO. BLOCKS=',I5,' COLUMNS PE',
     *'R BLOCK=',I5,' COLUMNS IN LAST BLOCK=',I5)
   15 FORMAT(' SUBMATRICIES FIT IN CORE')
   16 FORMAT(' SUBMATRIX PARTITIONING -  NO. BLOCKS=',I5,' COLUMNS P',
```

```
  *'ER BLOCK=',I5,' COLUMNS IN LAST BLOCK=',I5)
17 FORMAT(' ERROR - INSUFFICIENT STORAGE FOR MATRIX',2I5)
18 FORMAT(' SYMMETRY ERROR - NROW,NCOL=',2I5)
   END
```

FBNGF

PURPOSE

To set parameters for storage of the matrices B, C and D for the NGF solution.

METHOD

The modes of matrix storage for the NGF solution are described in Section VIII. FBNGF choses the smallest ICASX (1 through 4) possible given the size of the matrices A, B, C and D and the space available in the array GM in COMMON/CMB/. If B, C and D must be divided into blocks (ICASX = 3 or 4) the blocks are chosen are large as possible to minimize the number of input and output requests. Parameters specifying the number and size of blocks are stored in COMMON/MATPAR/ (see Section III).

FBNGF also sets the locations in GM at which storage of B, C and D start. For example, CM(IC11) is passed from the main program to subroutines CMNGF and FACGF as the starting location of array C.

SYMBOL DICTIONARY

IB11	=	location in CM at which storage of B starts
IC11	=	location in CM at which storage of C starts
ID11	=	location in CM at which storage of D starts
IMAT	=	number of complex numbers in A_F
IR	=	space available (complex numbers) in CM when A_F is not being used.
IRESRV	=	total length of GM
IRESX	=	space available in CM when A_F is being used
IX11	=	location in GM at which storage of B starts when $A^{-1}B$ is computed (A_F occupies space in CM)
NBCD	=	number of complex numbers in B, C and D combined
NBLN	=	number of complex numbers in B or C
NDLN	=	length of D
NEQ	=	number of rows in B, columns in C
NEQ2	=	number of columns in B or D, rows in C or D

```
      SUBROUTINE FBNGF( NEQ, NEQ2, IRESRV, IB11, IC11, ID11, IX11)
C     FBNGF SETS THE BLOCKING PARAMETERS FOR THE B, C, AND D ARRAYS FOR
C     OUT-OF-CORE STORAGE.
      COMMON  /MATPAR/ ICASE, NBLOKS, NPBLK, NLAST, NBLSYM, NPSYM,
     *NLSYM, IMAT, ICASX, NBBX, NPBX, NLBX, NBBL, NPBL, NLBL
      IRESX= IRESRV- IMAT
      NBLN= NEQ* NEQ2
      NDLN= NEQ2* NEQ2
      NBCD=2* NBLN+ NDLN
      IF(NBCD.GT. IRESX) GOTO 1
      ICASX=1
      IB11= IMAT+1
      GOTO 2
    1 IF(ICASE.LT.3) GOTO 3
      IF(NBCD.GT. IRESRV.OR. NBLN.GT. IRESX) GOTO 3
      ICASX=2
      IB11=1
    2 NBBX=1
      NPBX= NEQ
      NLBX= NEQ
      NBBL=1
      NPBL= NEQ2
      NLBL= NEQ2
      GOTO 5
    3 IR= IRESRV
      IF(ICASE.LT.3) IR= IRESX
      ICASX=3
      IF(NDLN.GT. IR) ICASX=4
      NBCD=2* NEQ+ NEQ2
      NPBL= IR/ NBCD
      NLBL= IR/(2* NEQ2)
      IF(NLBL.LT. NPBL) NPBL= NLBL
      IF(ICASE.LT.3) GOTO 4
      NLBL= IRESX/ NEQ
      IF(NLBL.LT. NPBL) NPBL= NLBL
    4 IF(NPBL.LT.1) GOTO 6
      NBBL=( NEQ2-1)/ NPBL
      NLBL= NEQ2- NBBL* NPBL
      NBBL= NBBL+1
      NBLN= NEQ* NPBL
      IR= IR- NBLN
      NPBX= IR/ NEQ2
      IF(NPBX.GT. NEQ) NPBX= NEQ
      NBBX=( NEQ-1)/ NPBX
      NLBX= NEQ- NBBX* NPBX
      NBBX= NBBX+1
      IB11=1
      IF(ICASE.LT.3) IB11= IMAT+1
    5 IC11= IB11+ NBLN
```

```
      ID11= IC11+ NBLN
      IX11= IMAT+1
      WRITE (2,11)  NEQ2
      IF(ICASX.EQ.1) RETURN
      WRITE (2,8)  ICASX
      WRITE (2,9)  NBBX, NPBX, NLBX
      WRITE (2,10)  NBBL, NPBL, NLBL
      RETURN
    6 WRITE (2,7)  IRESRV, IMAT, NEQ, NEQ2
C
      STOP
    7 FORMAT(55H ERROR - INSUFFICIENT STORAGE FOR INTERACTION MATRICIES
     *,'  IRESRV,IMAT,NEQ,NEQ2 =',4I5)
    8 FORMAT(' FILE STORAGE FOR NEW MATRIX SECTIONS -  ICASX =', I2)
    9 FORMAT(' B FILLED BY ROWS -',15X,'NO. BLOCKS =',I3,3X,
     *  'ROWS PER BLOCK =', I3, '   ROWS IN LAST BLOCK =', I3)
   10 FORMAT(' B BY COLUMNS, C AND D BY ROWS -  NO. BLOCKS =',I3,
     * '    R/C PER BLOCK =', I3, '    R/C IN LAST BLOCK =', I3)
   11 FORMAT(//,' N.G.F. - NUMBER OF NEW UNKNOWNS IS', I4)
      END
```

FFLD

PURPOSE

To calculate the radiated electric field due to the currents on wires and surfaces in free space or over ground. The range factor exp(-jkr0)/(r_0/λ) is omitted.

METHOD

Equation (126 of Part I is used to evaluate the radiated field of wires and surfaces. The surface part of the equation is evaluated in subroutine FFLDS, however. For wires, the field equation is

$$\vec{E}(\vec{r}_0) = \frac{j\eta\exp(-jkr_0)}{4\pi r_0/\lambda}(\hat{k}\hat{k} - \bar{\bar{I}}) \cdot \vec{F}(\vec{r}_0)$$

$$\vec{F}(\vec{r}_0) = 2\pi \int_L \exp(j\vec{k}\cdot\vec{r})\left[\vec{I}(s)/\lambda\right] ds/\lambda$$

where

r_0	=	$\lvert\vec{r}_0\rvert$
$\hat{k}$	=	$\vec{r}_0/\lvert\vec{r}_0\rvert$
k	=	$2\pi/\lambda$
$\vec{k}$	=	k$\hat{k}$
$\vec{I}$(s)	=	current on the wire at s
$\bar{\bar{I}}$	=	identity dyad
L	=	contour of the wire
$\vec{r}$	=	position of the point at s on the wire

The dot product with the dyad $\hat{k}\hat{k}-\bar{\bar{I}}$ results in the component of $\vec{F}$ transverse to $\hat{k}$. This is accomplished in the code by computing the dot products with the unit vectors θ and Φ, normal to $\hat{k}$.

For a wire structure consisting of N straight segments, $\vec{r}$ on segment i is replaced by

$$\vec{r} = \vec{r}_i + \lambda\, t\, \hat{u}_i,$$

where

$\vec{r}_i$ = location of the center of segment i

$\hat{u}_i$ = unit vector in the direction of segment i

Then, $\vec{F}$ is evaluated as

$$\vec{F}(\vec{r}_0) = \sum_{i=I}^{N} \exp(j\vec{k}\cdot\vec{r}_i)\vec{Q}_i$$

$$Q_i = 2\pi\hat{u}_i \int_{-\Delta_i/2}^{\Delta_i/2} \exp[j2\pi\, t(\hat{k}\cdot\hat{u}_i)]\, I_i(t)/\lambda\, dt$$

where Δ_i is the length of segment i normalized to λ. With

$$I_i(t)/\lambda = A_i + B_i\sin(2\pi t) + C_i\cos(2\pi t)\,,$$

the integral can be evaluated as

$$\vec{Q}_i = \hat{u}_i\left(A_i\frac{2\sin(\pi w_i\Delta_i)}{w_i} - jB_i\left[\frac{\sin[\pi(1-w_i)\Delta_i]}{(1-w_i)} - \frac{\sin[\pi(1+w_i)\Delta_i]}{(1+w_i)}\right] + C_i\left[\frac{\sin[\pi(1-w_i)\Delta_i]}{(1-w_i)} + \frac{\sin[\pi(1+w_i)\Delta_i]}{(1+w_i)}\right]\right)$$

where $w_i = -\hat{k}\cdot\hat{u}_i$.

The effect of a ground is included by computing the field of the image of each segment and modifying it by the Fresnel reflection coefficients. The coding here differs from section II-4 of Part I in some respects. Rather than reflecting each segment in the ground plane, the direction of observation, $\hat{k}$, is reflected for the image calculation. Thus, the sign of the z component of $\hat{k}$ is changed at the start of the image calculation. The z component of the image field must also be changed in sign at the end of the calculation. Also, the change in sign of the image field due to the change in sign of charge on the image is combined with the reflection coefficients. Thus, the reflection coefficients are the negative of those in Part I.

The code allows (or a change in ground height and electrical parameters at a fixed radial distance from the origin (circular cliff) or at a fixed distance in x (linear cliff). In these cases, the reflection point of the ray from the center of each segment is computed, and the reflection coefficients and phase lag are computed for the appropriate ground. Effects from the region of change, such as diffraction from the edge, are not included, however. A radial wire ground screen may also be included by the reflection coefficient approximation described in section II-4 of Part I.

CODING

FF30-FF164	Calculation of field due to segments.
FF34-FF164	Loop over direct and image fields.
FF38-FF63	Reflection coefficients computed.
FF64	$\hat{k}$ reflected in ground for image.
FF65-FF70	Direct fields saved, and CIX,CIY,CIZ initialized before image calculation.
FF75-FF96	Field of segment I computed.
FF102-FF104	Summation of fields for direct field or uniform ground.
FF110-FF149	Appropriate reflection coefficient determined and field summed for reflected field from two-medium ground or radial-wire ground screen.
FF156-FF159	Image field multiplied by reflection coefficients for uniform ground and added to direct field.
FF161-FF163	Reflected field added to direct field for two-medium ground or radial wire ground.
FF166-FF167	Dot products of $\vec{P}$ with θ and Φ for wires only.
FF169-FF208	Calculation of field due to surface patches.
FF177-FF203	Loop over direct and image fields.
FF179	$\hat{k}$ reflected for image.
FF180	FFLDS calculates field.
FF186-FF202	Field multiplied by reflection coefficients for uniform ground only.

SYMBOL DICTIONARY

A	=	$2\sin(\pi w_i\Delta_i)/w_i$ (a series is used for small w_i)
ARG	=	$\hat{k}\cdot\hat{r}_i$
B	=	coefficient of B_i in $\vec{Q}_i$
BOO	=	$\sin[\pi(1-w_i)\Delta_i]/[\pi(l-w_i)\Delta_i]$
BOT	=	$\pi(1-w_i)\Delta_i$
C	=	coefficient of C_i in $\vec{Q}_i$
CAB		
SAB	=	x,y z-components of $\hat{u}_i$
SALP		
CCX		
CCY	=	variables for summation of x,y,and z-components of $\vec{F}$
CCZ		
CDP	=	$(\vec{F}\cdot\hat{\Phi})(R_V-R_H)$
CIX		
CIY	=	variables for summation of x,y, and z-components of $\vec{F}$
CIZ		
CONST	=	CONSX = $-j\eta/4\pi$
D	=	distance of ray reflection point from origin
DARC	=	phase increment due to change in ground level
EL	=	$\pi\Delta_i$
EPH	=	Φ component of $(r_0/\lambda)\exp(jkr_0)\vec{E}(\vec{r_0})$
ETH	=	θ component of $(r_0/\lambda)\exp(jkr_0)\vec{E}(\vec{r_0})$
ETA	=	$\eta = \sqrt{\mu/\epsilon}$
EX		
EY	=	$(r_0/\lambda)\exp(jkr_0)\vec{E}(\vec{r_0})$ for patches
EZ		
EXA	=	Q_i
GX		
GY	=	$(r_0/\lambda)\exp(jkr_0)\vec{E}(\vec{r_0})$ for direct and reflected fields of patches
GZ		
I	=	segment number
OMEGA	=	w_i
PHI	=	Φ
PHX,PHY	=	x and y components of Φ
PI	=	π
RFL	=	± 1 for direct or image field of patch
RI	=	imaginary part of Q_i
ROX		
ROY	=	x,y, and z-components of $\hat{k}$
ROZ		
ROZS	=	saved value of ROZ

RR	=	real part of Q_i
RRH	=	$-R_H$
RRH1	=	$-R_H$ for first ground medium
RRH2	=	$-R_H$ for second ground medium
RRV	=	$-R_V$
RRV1	=	$-R_V$ for first ground medium
RRV2	=	$-R_V$ for second ground medium
RRZ	=	z component of $\hat{k}$
SILL	=	$\pi w_i \Delta_i$
THET	=	θ (angle from vertical to $\hat{k}$)
THX		
THY	=	= θ
THZ		
TIX I		
TIY *	=	Q_i for image in ground
TIZ		
TOO	=	$\sin[\pi(1+w_i)\Delta_i]/[\pi(1+w_i)\Delta_i]$
TOP	=	$\pi(1+w_i)\Delta_i$
TP	=	2π
TTHET	=	tan θ
ZRATI	=	$[\epsilon_r - j\sigma/(\omega\epsilon_0)]^{-1/2}$ ϵ_r, σ = ground parameters
ZRSIN	=	$[1-(ZRAT1)^2 \sin^2\theta]^{1/2}$
ZSCRN	=	surface impedance of ground with radial wire ground screen

-29.91922085	=	$-j\eta/(4\pi)$
3.141592654	=	π
376.73	=	η]
6.283185308	=	2π

```
      SUBROUTINE FFLD( THET, PHI, ETH, EPH)
C
C     FFLD CALCULATES THE FAR ZONE RADIATED ELECTRIC FIELDS,
C     THE FACTOR EXP(J*K*R)/(R/LAMDA) NOT INCLUDED
C
      COMPLEX  CIX, CIY, CIZ, EXA, ETH, EPH, CONST, CCX, CCY, CCZ,
     *CDP, CUR
      COMPLEX  ZRATI, ZRSIN, RRV, RRH, RRV1, RRH1, RRV2, RRH2,
     *ZRATI2, TIX, TIY, TIZ, T1, ZSCRN, EX, EY, EZ, GX, GY, GZ, FRATI
      COMMON  /DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X( NM), Y( NM),
     *Z( NM), SI( NM), BI( NM), ALP( NM), BET( NM), ICON1( N2M), ICON2(
     * N2M), ITAG( N2M), ICONX( NM), WLAM, IPSYM
      COMMON  /ANGL/ SALP( NM)
      COMMON  /CRNT/ AIR( NM), AII( NM), BIR( NM), BII( NM), CIR( NM),
     *CII( NM), CUR( N3M)
      COMMON  /GND/ ZRATI, ZRATI2, FRATI, CL, CH, SCRWL, SCRWR, NRADL,
     *KSYMP, IFAR, IPERF, T1, T2
      DIMENSION  CAB(1), SAB(1), CONSX(2)
      EQUIVALENCE(CAB,ALP),(SAB,BET),(CONST,CONSX)
      DATA   PI, TP, ETA/3.141592654D+0,6.283185308D+0,376.73/
      DATA   CONSX/0.,-29.97922085D+0/
      PHX=- SIN( PHI)
      PHY= COS( PHI)
      ROZ= COS( THET)
      ROZS= ROZ
      THX= ROZ* PHY
      THY=- ROZ* PHX
      THZ=- SIN( THET)
      ROX=- THZ* PHY
      ROY= THZ* PHX
C
C     LOOP FOR STRUCTURE IMAGE IF ANY
C
      IF(N.EQ.0) GOTO 20
C
C     CALCULATION OF REFLECTION COEFFECIENTS
C
      DO 19  K=1, KSYMP
      IF(K.EQ.1) GOTO 4
C
C     FOR PERFECT GROUND
C
      IF(IPERF.NE.1) GOTO 1
      RRV=-(1.,0.)
      RRH=-(1.,0.)
C
C     FOR INFINITE PLANAR GROUND
C
      GOTO 2
```

```
    1 ZRSIN= SQRT(1.- ZRATI* ZRATI* THZ* THZ)
      RRV=-( ROZ- ZRATI* ZRSIN)/( ROZ+ ZRATI* ZRSIN)
      RRH=( ZRATI* ROZ- ZRSIN)/( ZRATI* ROZ+ ZRSIN)
C
C     FOR THE CLIFF PROBLEM, TWO REFLCTION COEFFICIENTS CALCULATED
C
    2 IF(IFAR.LE.1) GOTO 3
      RRV1= RRV
      RRH1= RRH
      TTHET= TAN( THET)
      IF(IFAR.EQ.4) GOTO 3
      ZRSIN= SQRT(1.- ZRATI2* ZRATI2* THZ* THZ)
      RRV2=-( ROZ- ZRATI2* ZRSIN)/( ROZ+ ZRATI2* ZRSIN)
      RRH2=( ZRATI2* ROZ- ZRSIN)/( ZRATI2* ROZ+ ZRSIN)
      DARG=- TP*2.* CH* ROZ
    3 ROZ=- ROZ
      CCX= CIX
      CCY= CIY
      CCZ= CIZ
    4 CIX=(0.,0.)
      CIY=(0.,0.)
C
C     LOOP OVER STRUCTURE SEGMENTS
C
      CIZ=(0.,0.)
      DO 17  I=1, N
      OMEGA=-( ROX* CAB( I)+ ROY* SAB( I)+ ROZ* SALP( I))
      EL= PI* SI( I)
      SILL= OMEGA* EL
      TOP= EL+ SILL
      BOT= EL- SILL
      IF(ABS( OMEGA).LT.1.D-7) GOTO 5
      A=2.* SIN( SILL)/ OMEGA
      GOTO 6
    5 A=(2.- OMEGA* OMEGA* EL* EL/3.)* EL
    6 IF(ABS( TOP).LT.1.D-7) GOTO 7
      TOO= SIN( TOP)/ TOP
      GOTO 8
    7 TOO=1.- TOP* TOP/6.
    8 IF(ABS( BOT).LT.1.D-7) GOTO 9
      BOO= SIN( BOT)/ BOT
      GOTO 10
    9 BOO=1.- BOT* BOT/6.
   10 B= EL*( BOO- TOO)
      C= EL*( BOO+ TOO)
      RR= A* AIR( I)+ B* BII( I)+ C* CIR( I)
      RI= A* AII( I)- B* BIR( I)+ C* CII( I)
      ARG= TP*( X( I)* ROX+ Y( I)* ROY+ Z( I)* ROZ)
      IF(K.EQ.2.AND. IFAR.GE.2) GOTO 11
```

```
C
C     SUMMATION FOR FAR FIELD INTEGRAL
C
      EXA= CMPLX( COS( ARG), SIN( ARG))* CMPLX( RR, RI)
      CIX= CIX+ EXA* CAB( I)
      CIY= CIY+ EXA* SAB( I)
      CIZ= CIZ+ EXA* SALP( I)
C
C     CALCULATION OF IMAGE CONTRIBUTION IN CLIFF AND GROUND SCREEN
C     PROBLEMS.
C
      GOTO 17
C
C     SPECULAR POINT DISTANCE
C
   11 DR= Z( I)* TTHET
      D= DR* PHY+ X( I)
      IF(IFAR.EQ.2) GOTO 13
      D= SQRT( D* D+( Y( I)- DR* PHX)**2)
      IF(IFAR.EQ.3) GOTO 13
C
C     RADIAL WIRE GROUND SCREEN REFLECTION COEFFICIENT
C
      IF(( SCRWL- D).LT.0.) GOTO 12
      D= D+ T2
      ZSCRN= T1* D* LOG( D/ T2)
      ZSCRN=( ZSCRN* ZRATI)/( ETA* ZRATI+ ZSCRN)
      ZRSIN= SQRT(1.- ZSCRN* ZSCRN* THZ* THZ)
      RRV=( ROZ+ ZSCRN* ZRSIN)/(- ROZ+ ZSCRN* ZRSIN)
      RRH=( ZSCRN* ROZ+ ZRSIN)/( ZSCRN* ROZ- ZRSIN)
      GOTO 16
   12 IF(IFAR.EQ.4) GOTO 14
      IF(IFAR.EQ.5) D= DR* PHY+ X( I)
   13 IF(( CL- D).LE.0.) GOTO 15
   14 RRV= RRV1
      RRH= RRH1
      GOTO 16
   15 RRV= RRV2
      RRH= RRH2
      ARG= ARG+ DARG
C
C     CONTRIBUTION OF EACH IMAGE SEGMENT MODIFIED BY REFLECTION COEF. ,
C     FOR CLIFF AND GROUND SCREEN PROBLEMS
C
   16 EXA= CMPLX( COS( ARG), SIN( ARG))* CMPLX( RR, RI)
      TIX= EXA* CAB( I)
      TIY= EXA* SAB( I)
      TIZ= EXA* SALP( I)
      CDP=( TIX* PHX+ TIY* PHY)*( RRH- RRV)
```

```
      CIX= CIX+ TIX* RRV+ CDP* PHX
      CIY= CIY+ TIY* RRV+ CDP* PHY
      CIZ= CIZ- TIZ* RRV
   17 CONTINUE
      IF(K.EQ.1) GOTO 19
C
C     CALCULATION OF CONTRIBUTION OF STRUCTURE IMAGE FOR INFINITE GROUND
C
      IF(IFAR.GE.2) GOTO 18
      CDP=( CIX* PHX+ CIY* PHY)*( RRH- RRV)
      CIX= CCX+ CIX* RRV+ CDP* PHX
      CIY= CCY+ CIY* RRV+ CDP* PHY
      CIZ= CCZ- CIZ* RRV
      GOTO 19
   18 CIX= CIX+ CCX
      CIY= CIY+ CCY
      CIZ= CIZ+ CCZ
   19 CONTINUE
      IF(M.GT.0) GOTO 21
      ETH=( CIX* THX+ CIY* THY+ CIZ* THZ)* CONST
      EPH=( CIX* PHX+ CIY* PHY)* CONST
      RETURN
   20 CIX=(0.,0.)
      CIY=(0.,0.)
      CIZ=(0.,0.)
C
C     ELECTRIC FIELD COMPONENTS
C
   21 ROZ= ROZS
      RFL=-1.
      DO 25  IP=1, KSYMP
      RFL=- RFL
      RRZ= ROZ* RFL
      CALL FFLDS( ROX, ROY, RRZ, CUR( N+1), GX, GY, GZ)
      IF(IP.EQ.2) GOTO 22
      EX= GX
      EY= GY
      EZ= GZ
      GOTO 25
   22 IF(IPERF.NE.1) GOTO 23
      GX=- GX
      GY=- GY
      GZ=- GZ
      GOTO 24
   23 RRV= SQRT(1.- ZRATI* ZRATI* THZ* THZ)
      RRH= ZRATI* ROZ
      RRH=( RRH- RRV)/( RRH+ RRV)
      RRV= ZRATI* RRV
      RRV=-( ROZ- RRV)/( ROZ+ RRV)
```

```
      ETH=( GX* PHX+ GY* PHY)*( RRH- RRV)                               FF 197
      GX= GX* RRV+ ETH* PHX                                             FF 198
      GY= GY* RRV+ ETH* PHY                                             FF 199
      GZ= GZ* RRV                                                       FF 200
   24 EX= EX+ GX                                                        FF 201
      EY= EY+ GY                                                        FF 202
      EZ= EZ- GZ                                                        FF 203
   25 CONTINUE                                                          FF 204
      EX= EX+ CIX* CONST                                                FF 205
      EY= EY+ CIY* CONST                                                FF 206
      EZ= EZ+ CIZ* CONST                                                FF 207
      ETH= EX* THX+ EY* THY+ EZ* THZ                                    FF 208
      EPH= EX* PHX+ EY* PHY                                             FF 209
      RETURN                                                            FF 210
      END                                                               FF 211
```

FFLDS

PURPOSE

To calculate the x,y,z components of the far electric field due to surface currents. The term exp(-jkr$_0$)/(r$_0$/λ) is omitted.

METHOD

The field is computed using the surface portion of equation (126) in Part I. With lengths normalized to the wavelength, the field equation is

$$\vec{E}(\vec{r}_0) = \frac{j\eta}{2}\frac{j\eta\exp(-jkr_0)}{r_0/\lambda}(\hat{k}\hat{k} - \bar{\bar{I}}) \cdot \vec{F}(\vec{r}_0)$$

$$\vec{F}(\vec{r}_0) = \int_S \vec{J}_S(\vec{r})\exp(j\vec{k}\cdot\vec{r})\,dA/\lambda^2$$

where

r$_0$	=	$\lvert\vec{r}_0\rvert$
$\hat{k}$	=	$\vec{r}_0/\lvert\vec{r}_0\rvert$
k	=	$2\pi/\lambda$
$\vec{k}$	=	k$\hat{k}$
$\bar{\bar{I}}$	=	identity dyad
$\vec{J}_S$	=	surface current on surface S

The dot product with the dyad $\hat{k}\hat{k} - \bar{\bar{I}}$ results in the component of the integral

$$\vec{F}(\vec{r}_0) = \int_S \vec{J}_S(\vec{r})\exp(j\vec{k}\cdot\vec{r})\,dA/\lambda^2$$

transverse to $\hat{k}$. The integral is evaluated by summation over the patches with the current assumed constant over each patch.

SYMBOL DICTIONARY

ARG	=	$\hat{k}\cdot\hat{r}_i$, $\vec{r}_i$ = center of patch I
CONS	=	CONSX = $j\eta/2$
CT	=	$\exp(j\vec{k}\cdot\vec{r}_i)\,dA/\lambda^2$ at FL18
	=	$\hat{k}\cdot\vec{F}(\vec{r}_0)$ at FL24
EX,EY,EZ	=	x,y,z components of $\vec{F}(\vec{r}_0)$ at FL22
		$(r_0/\lambda)\exp(jkr_0)\vec{E}(\vec{r}_0)$ at FL27
I	=	array location of patch data
J	=	patch number
K	=	current array index
ROX,ROY,ROZ	=	x,y, and z-components of $\hat{k}$
S(I)	=	(area of patch I)/λ^2
SCUR	=	array containing surface current components
TPI	=	2π
XS,YS,ZS	=	arrays containing center point coordinates of patches normalized to wavelenth.

```
      SUBROUTINE FFLDS( ROX, ROY, ROZ, SCUR, EX, EY, EZ)
C     CALCULATES THE XYZ COMPONENTS OF THE ELECTRIC FIELD DUE TO
C     SURFACE CURRENTS
      COMPLEX  CT, CONS, SCUR, EX, EY, EZ
      COMMON  /DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X( NM), Y( NM),
     *Z( NM), SI( NM), BI( NM), ALP( NM), BET( NM), ICON1( N2M), ICON2(
     * N2M), ITAG( N2M), ICONX( NM), WLAM, IPSYM
      DIMENSION  XS(1), YS(1), ZS(1), S(1), SCUR(1), CONSX(2)
      EQUIVALENCE(XS,X),(YS,Y),(ZS,Z),(S,BI),(CONS,CONSX)
      DATA   TPI/6.283185308D+0/, CONSX/0.,188.365/
      EX=(0.,0.)
      EY=(0.,0.)
      EZ=(0.,0.)
      I= LD+1
      DO 1  J=1, M
      I= I-1
      ARG= TPI*( ROX* XS( I)+ ROY* YS( I)+ ROZ* ZS( I))
      CT= CMPLX( COS( ARG)* S( I), SIN( ARG)* S( I))
      K=3* J
      EX= EX+ SCUR( K-2)* CT
      EY= EY+ SCUR( K-1)* CT
      EZ= EZ+ SCUR( K)* CT
    1 CONTINUE
      CT= ROX* EX+ ROY* EY+ ROZ* EZ
      EX= CONS*( CT* ROX- EX)
      EY= CONS*( CT* ROY- EY)
      EZ= CONS*( CT* ROZ- EZ)
      RETURN
      END
```

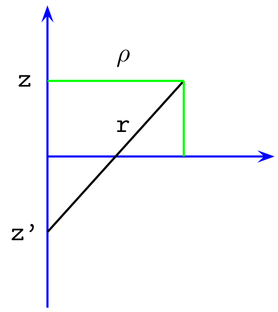
<u>GF</u>

PURPOSE

To supply values of the integrated function exp(jkr)/(kr) to the numerical integration routine INTX.

METHOD

The geometry parameters for integration aver a segment are shown in the following diagram.



in which

$$r(z') = [\rho^2 + (z' - z)^2]^{1/2}.$$

if the field point (ρ,z) is not on the source segment, the integrand value is

$$G(z') = \frac{\exp[jkr(z')]}{kr(z')}.$$

if the field point is on the source Segment (ρ = 0, z = 0), the integrand value is

$$G(z') = \frac{\exp[jkr(z')]}{kr(z')}.$$

In the latter case, if kr is less than 0.2, then (cos kr)/kr is evaluated by the first three terms of its Taylor's series to reduce numerical error.

SYMBOL DICTIONARY

CO	=	real part of G(z')
IJ	=	flag to indicate when field point is on source segment (by IJ = 0)
RK	=	kr
RKB2	=	$(k\rho)^2$
SI	=	imaginary part of G(z')
ZDK	=	kz' - kz
ZK	=	kz'
ZPK	=	kz
-1.388888889E-3	=	constant in series for (cos kr - 1)/kr
4.166666667E-2	=	constant in series for (cos kr - 1)/kr
0.5	=	constant in series for (cos kr - 1)/kr

```
      SUBROUTINE GF( ZK, CO, SI)
C
C     GF COMPUTES THE INTEGRAND EXP(JKR)/(KR) FOR NUMERICAL INTEGRATION.
C
      COMMON  /TMI/ ZPK, RKB2, IJ
      ZDK= ZK- ZPK
      RK= SQRT( RKB2+ ZDK* ZDK)
      SI= SIN( RK)/ RK
      IF(IJ) 1,2,1
    1 CO= COS( RK)/ RK
      RETURN
    2 IF(RK.LT..2) GOTO 3
      CO=( COS( RK)-1.)/ RK
      RETURN
    3 RKS= RK* RK
      CO=((-1.38888889D-3* RKS+4.16666667D-2)* RKS-.5)* RK
      RETURN
      END
```

GFIL

PURPOSE

To read the NGF file and store parameters in the proper arrays.

METHOD

GI22	Miscellaneous parameters are read.
GI30-GI48	Segment coordinates were converted to the form involving the segment center, segment length, and orientatian (see Section III, COMMON/DATA/) with dimensions of wavelength. They must be converted back to the coordinates of the segment ends so that subroutine CONNECT can locate connections. Dimensions are converted to meters.
GI52-GI62	Patch coordinates are converted from units of wavelength to meters since they will be scaled back to wavelengths along with the new segments and patches.
GI63	Matrix blocking parameters are read.
GI64	Interpolation tables for the Sommerfeld integrals are read if the Sommerfeld/Norton ground treatment was used.
GI74	Matrix A_F is read for in-core storage (ICASE = 1 or 2).
GI78-GI81	A_F is read for ICASE = 4.
GI83-GI88	A_F is read for ICASE1 = 3 or 5.
GI92-GI113	A heading summarizing the NGF file is printed.

SYMBOL DICTIONARY

DX	=	half segment length (meters)
IGFL	=	file number for NGF file
IOUT	=	number of elements in matrix
IPRT	=	1 to print coordinates of ends of segments
NBL2	=	two times number of blocks in matrix A_F (since A_F is stored twice, in ascending and descending order)
NEQ	=	order of the NGF matrix
NOP	=	number of symmetric sections
NPEQ	=	number of unknowns for a symmetric section
XI,YI,ZI	=	coordinates of the center of a segment or patch

```
      SUBROUTINE GFIL( IPRT)
C
C     GFIL READS THE N.G.F. FILE
C
      INTEGER*4 COM
      COMPLEX  CM, SSX, ZRATI, ZRATI2, T1, ZARRAY, AR1, AR2, AR3,
     *EPSCF, FRATI
      COMMON  /DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X( NM), Y( NM),
     *Z( NM), SI( NM), BI( NM), ALP( NM), BET( NM), ICON1( N2M), ICON2(
     * N2M), ITAG( N2M), ICONX( NM), WLAM, IPSYM
      COMMON  /CMB/ CM(90000)
      COMMON  /ANGL/ SALP( NM)
      COMMON  /GND/ ZRATI, ZRATI2, FRATI, CL, CH, SCRWL, SCRWR, NRADL,
     *KSYMP, IFAR, IPERF, T1, T2
      COMMON  /GGRID/ AR1(11,10,4), AR2(17,5,4), AR3(9,8,4), EPSCF, DXA
     *(3), DYA(3), XSA(3), YSA(3), NXA(3), NYA(3)
      COMMON  /MATPAR/ ICASE, NBLOKS, NPBLK, NLAST, NBLSYM, NPSYM,
     *NLSYM, IMAT, ICASX, NBBX, NPBX, NLBX, NBBL, NPBL, NLBL
      COMMON  /SMAT/ SSX(16,16)
      COMMON  /ZLOAD/ ZARRAY( NM), NLOAD, NLODF
      COMMON  /SAVE/ IP( N2M), KCOM, COM(20,5), EPSR, SIG, SCRWLT,
     *SCRWRT, FMHZ
      DATA   IGFL/20/
      REWIND IGFL
      READ(IGFL)  N1, NP, M1, MP, WLAM, FMHZ, IPSYM, KSYMP, IPERF,
     *NRADL, EPSR, SIG, SCRWLT, SCRWRT, NLODF, KCOM
      N= N1
      M= M1
      N2= N1+1
      M2= M1+1
C     READ SEG. DATA AND CONVERT BACK TO END COORD. IN UNITS OF METERS
      IF(N1.EQ.0) GOTO 2
      READ(IGFL) ( X( I), I=1, N1),( Y( I), I=1, N1),( Z( I), I=1, N1)
     *
      READ(IGFL) ( SI( I), I=1, N1),( BI( I), I=1, N1),( ALP( I), I=1,
     * N1)
      READ(IGFL) ( BET( I), I=1, N1),( SALP( I), I=1, N1)
      READ(IGFL) ( ICON1( I), I=1, N1),( ICON2( I), I=1, N1)
      READ(IGFL) ( ITAG( I), I=1, N1)
      IF(NLODF.NE.0) READ(IGFL) ( ZARRAY( I), I=1, N1)
      DO 1  I=1, N1
      XI= X( I)* WLAM
      YI= Y( I)* WLAM
      ZI= Z( I)* WLAM
      DX= SI( I)*.5* WLAM
      X( I)= XI- ALP( I)* DX
      Y( I)= YI- BET( I)* DX
      Z( I)= ZI- SALP( I)* DX
      SI( I)= XI+ ALP( I)* DX
```

```
      ALP( I)= YI+ BET( I)* DX
      BET( I)= ZI+ SALP( I)* DX
      BI( I)= BI( I)* WLAM
    1 CONTINUE
    2 IF(M1.EQ.0) GOTO 4
C     READ PATCH DATA AND CONVERT TO METERS
      J= LD- M1+1
      READ(IGFL) ( X( I), I= J, LD),( Y( I), I= J, LD),( Z( I), I= J,
     *LD)
      READ(IGFL) ( SI( I), I= J, LD),( BI( I), I= J, LD),( ALP( I), I=
     * J, LD)
      READ(IGFL) ( BET( I), I= J, LD),( SALP( I), I= J, LD)
      READ(IGFL) ( ICON1( I), I= J, LD),( ICON2( I), I= J, LD)
      READ(IGFL) ( ITAG( I), I= J, LD)
      DX= WLAM* WLAM
      DO 3  I= J, LD
      X( I)= X( I)* WLAM
      Y( I)= Y( I)* WLAM
      Z( I)= Z( I)* WLAM
    3 BI( I)= BI( I)* DX
    4 READ(IGFL)  ICASE, NBLOKS, NPBLK, NLAST, NBLSYM, NPSYM, NLSYM,
     *IMAT
      IF(IPERF.EQ.2) READ(IGFL)  AR1, AR2, AR3, EPSCF, DXA, DYA, XSA,
     * YSA, NXA, NYA
      NEQ= N1+2* M1
      NPEQ= NP+2* MP
      NOP= NEQ/ NPEQ
      IF(NOP.GT.1) READ(IGFL) (( SSX( I, J), I=1, NOP), J=1, NOP)
C     READ MATRIX A AND WRITE TAPE13 FOR OUT OF CORE
      READ(IGFL) ( IP( I), I=1, NEQ), COM
      IF(ICASE.GT.2) GOTO 5
      IOUT= NEQ* NPEQ
      READ(IGFL) ( CM( I), I=1, IOUT)
      GOTO 10
    5 REWIND 13
      IF(ICASE.NE.4) GOTO 7
      IOUT= NPEQ* NPEQ
      DO 6  K=1, NOP
      READ(IGFL) ( CM( J), J=1, IOUT)
    6 WRITE( 13) ( CM( J), J=1, IOUT)
      GOTO 9
    7 IOUT= NPSYM* NPEQ*2
      NBL2=2* NBLSYM
      DO 8  IOP=1, NOP
      DO 8  I=1, NBL2
      CALL BLCKIN( CM, IGFL,1, IOUT,1,206)
    8 CALL BLCKOT( CM,13,1, IOUT,1,205)
    9 REWIND 13
C     WRITE(6,N) G.F. HEADING
```

```
   10 REWIND IGFL                                                       GI  99
      WRITE (2,16)                                                      GI 100
      WRITE (2,14)                                                      GI 101
      WRITE (2,14)                                                      GI 102
      WRITE (2,17)                                                      GI 103
      WRITE (2,18)  N1, M1                                              GI 104
      IF(NOP.GT.1) WRITE (2,19)  NOP                                    GI 105
      WRITE (2,20)  IMAT, ICASE                                         GI 106
      IF(ICASE.LT.3) GOTO 11                                            GI 107
      NBL2= NEQ* NPEQ                                                   GI 108
      WRITE (2,21)  NBL2                                                GI 109
   11 WRITE (2,22)  FMHZ                                                GI 110
      IF(KSYMP.EQ.2.AND. IPERF.EQ.1) WRITE (2,23)                       GI 111
      IF(KSYMP.EQ.2.AND. IPERF.EQ.0) WRITE (2,27)                       GI 112
      IF(KSYMP.EQ.2.AND. IPERF.EQ.2) WRITE (2,28)                       GI 113
      IF(KSYMP.EQ.2.AND. IPERF.NE.1) WRITE (2,24)  EPSR, SIG            GI 114
      WRITE (2,17)                                                      GI 115
      DO 12  J=1, KCOM                                                  GI 116
   12 WRITE (2,15) ( COM( I, J), I=1,19)                                GI 117
      WRITE (2,17)                                                      GI 118
      WRITE (2,14)                                                      GI 119
      WRITE (2,14)                                                      GI 120
      WRITE (2,16)                                                      GI 121
      IF(IPRT.EQ.0) RETURN                                              GI 122
      WRITE (2,25)                                                      GI 123
      DO 13  I=1, N1                                                    GI 124
   13 WRITE (2,26)  I, X( I), Y( I), Z( I), SI( I), ALP( I), BET( I)    GI 125
C                                                                       GI 126
      RETURN                                                            GI 127
   14 FORMAT(5X,'**************************************************',   GI 128
     *'**********************************')                             GI 129
   15 FORMAT(5X,3H** ,19A4,3H **)                                       GI 130
   16 FORMAT(////)                                                      GI 131
   17 FORMAT(5X,2H**,80X,2H**)                                          GI 132
   18 FORMAT(5X,'** NUMERICAL GREEN S FUNCTION',53X,2H**,/,5X,'** NO',  GI 133
     *'. SEGMENTS =',I4,10X,'NO. PATCHES =',I4,34X,2H**)                GI 134
   19 FORMAT(5X,'** NO. SYMMETRIC SECTIONS =',I4,51X,2H**)              GI 135
   20 FORMAT(5X,'** N.G.F. MATRIX -  CORE STORAGE =',I7,' COMPLEX NU',  GI 136
     *'MBERS,  CASE',I2,16X,2H**)                                       GI 137
   21 FORMAT(5X,2H**,19X,'MATRIX SIZE =',I7,' COMPLEX NUMBERS',25X,'**') GI 138
   22 FORMAT(5X,'** FREQUENCY =',1P,E12.5,' MHZ.',51X,2H**)             GI 139
   23 FORMAT(5X,'** PERFECT GROUND',65X,2H**)                           GI 140
   24 FORMAT(5X,'** GROUND PARAMETERS - DIELECTRIC CONSTANT =',1P,E12.5, GI 141
     *26X,'**',/,5X,'**',21X,'CONDUCTIVITY =',E12.5,' MHOS/M.',25X,'**') GI 142
   25 FORMAT(39X,'NUMERICAL GREEN S FUNCTION DATA',/,41X,'COORDINATES',  GI 143
     *' OF SEGMENT ENDS',/,51X,'(METERS)',/,5X,'SEG.',11X,              GI 144
     *'- - - END ON''E - - -',26X,'- - - END TWO - - -',/,6X,3HNO.,6X,1 GI 145
     *HX,14X,1HY,14X,1HZ,14X,1HX,14X,1HY,14X,1HZ)                       GI 146
   26 FORMAT(1X,I7,1P,6E15.6)                                           GI 147
```

```
27 FORMAT(5X,'** FINITE GROUND.  REFLECTION COEFFICIENT APPROXIMAT',  GI 148
  *'ION',27X,2H**)                                                    GI 149
28 FORMAT(5X,'** FINITE GROUND.  SOMMERFELD SOLUTION',44X,'**')       GI 150
   END                                                                GI 151
```

GFLD

PURPOSE

To compute the electric field at intermediate distances from a radiating structure over ground, including the surface-wave field component.

METHOD

Approximate expressions for the field of a horizontal or vertical current element over a ground plane were derived by K. A. Norton (ref. 2). These expressions are used to evaluate the field of each segment in a structure and the components summed for the total field of the structure. To evaluate Norton's expressions for segment i, a local coordinate system (x',y',z') is defined (fig. 6a) with origin on the ground plane and the vertical z-axis passing through segment i. In the (x,y,z) coordinate system (fig.6b) the location and orientation of segment i are

$\vec{r}_i = x_i\hat{x} + y_i\hat{y} + z_i\hat{z}$

$\hat{i} = \cos\alpha\cos\beta\hat{x} + \cos\alpha\sin\beta\hat{y} + \sin\alpha\hat{z}$

and the field observation point is at (ρ,Φ,z). The origin of the primed coordinate system is at (x_i,y_i,0) in the unprimed coordinates, and the x' axis is along the projection of the segment an the ground plane.

Norton's expressions give the electric field in ρ', Φ', and z' components for infinitesimal current elements either vertical or horizontal, and directed along the x' axis. To evaluate the field of a segment, the segment current is decomposed into horizontal and vertical components, and the fields of the infinitesimal current elements are integrated over the segment. Each field component for the infinitesimal current element has the form

$$E_A(\rho', \Phi', z') = F_1(\rho', \Phi', z')\exp(-jkR_1) + F_2(\rho', \Phi', z')\exp(-jkR_2),$$

for

$R_1 = |\vec{R}_1|$

$R_2 = |\vec{R}_2|$

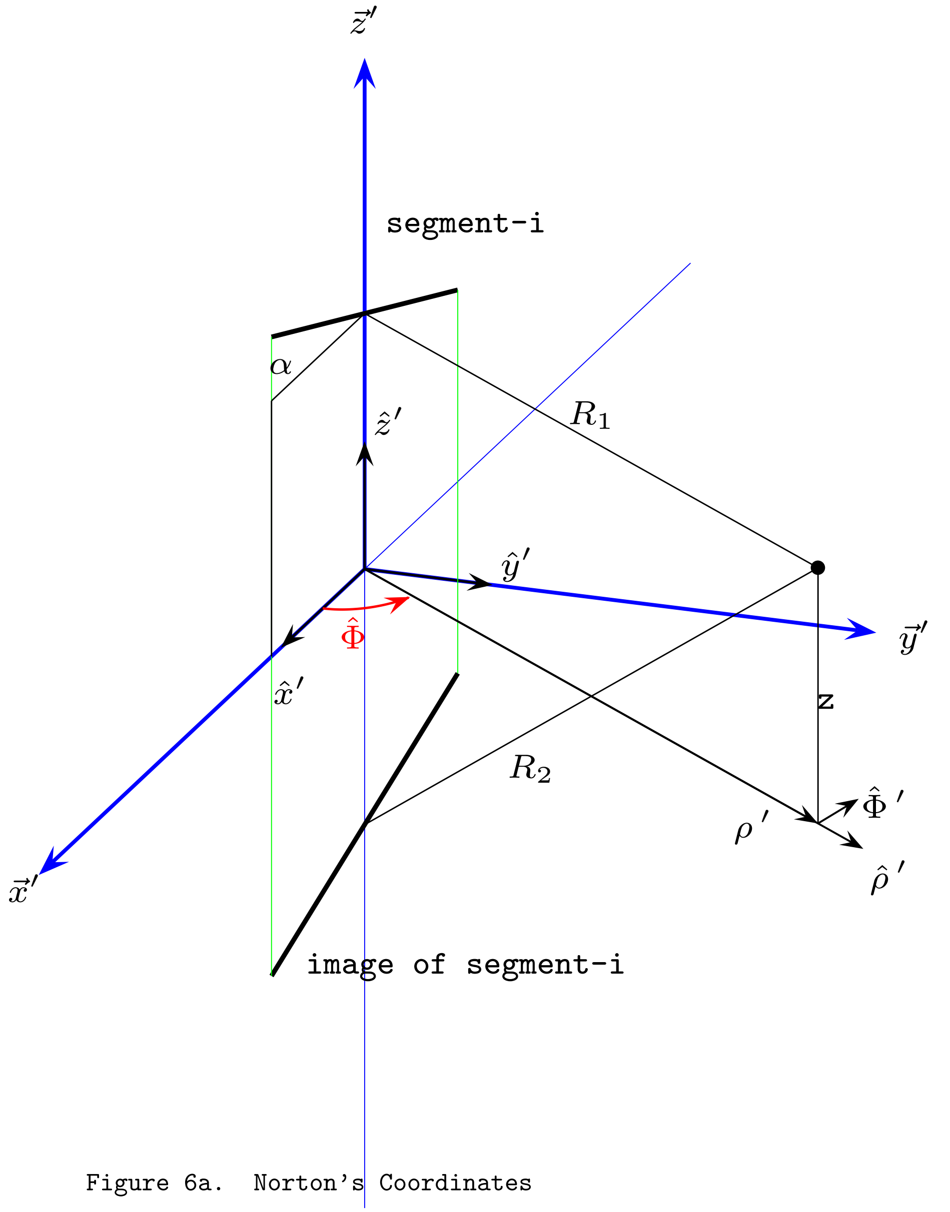


Figure 6a. Norton's Coordinates

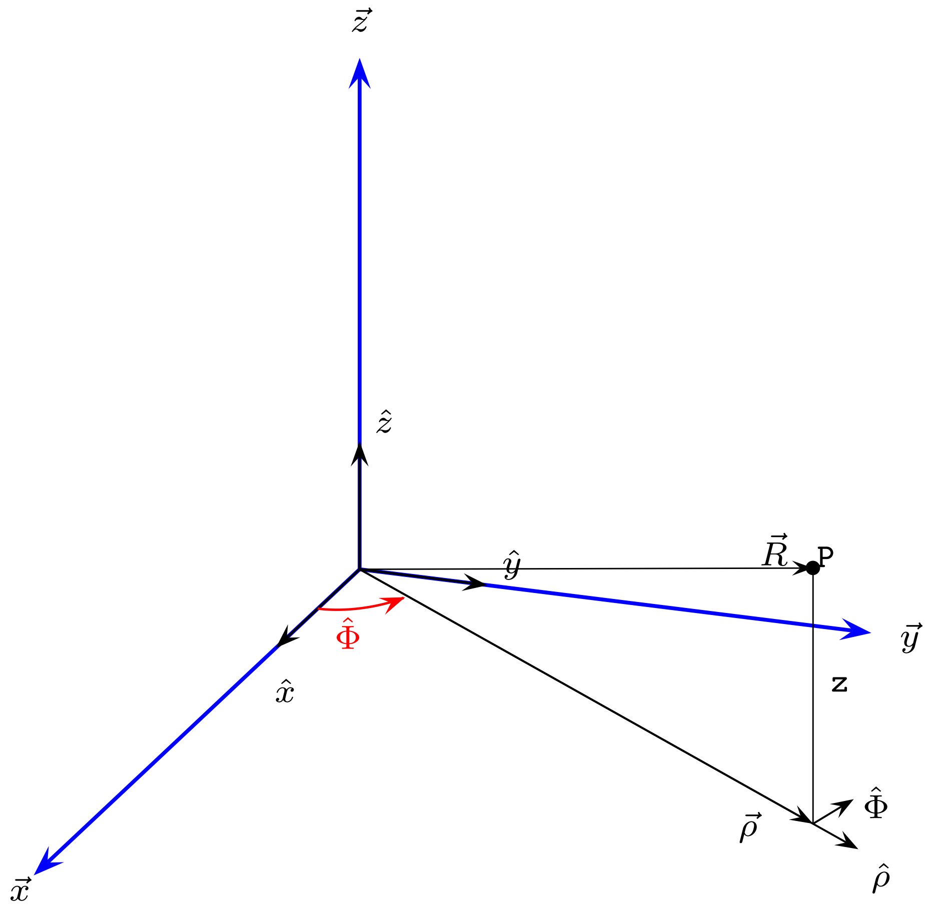


Figure 6b. NEC Coordinates

Figure 6. Coordinate Systems Used to Evaluate Norton's Expressions for the Ground Wave Fields in the NEC Program.

where F_l and F_2 are algebraic functions of R_1 and R_2 and can be considered constant for integration over the segment as long as R_1 and R_2 are much greater than the segment length. To integrate the exponential factors over the segment, R_1 and R_2 are approximated as

$$R_1 \approx R - \hat{R}_1 \cdot (\vec{r}_i + \hat{i}\, s)$$

$$R_2 \approx R - \hat{R}_2 \cdot (\vec{r}_i\,' + \hat{i}'\, s)$$

where R = |R|, $\hat{R}_1 = \vec{R}_1/|\vec{R}_1|$; $\vec{r}_i\,', \hat{i}'$ = position and orientation of image of segment i, and s = variable of length along the segment (s = 0 at segment center). The current on the segment is

$$I_i(s) = A_i + B_i \sin ks + C_i \cos ks.$$

With F_l and F_2 considered constant, each vector component of the field produced by segment i involves an integral of the form

$$E = F_1' \int_{-\Delta/2\lambda}^{\Delta/2\lambda} \frac{I_i(s)}{\lambda} \exp(-jks\omega) d(s/\lambda) + F_2' \int_{-\Delta/2\lambda}^{\Delta/2\lambda} \frac{I_i(s)}{\lambda} \exp(-jks\omega') d(s/\lambda),$$

where

$$F_1' = \lambda^2 F_1 \exp[-jk(R - \hat{R}_1 \cdot \vec{r}_i)]$$

$$F_2' = \lambda^2 F_2 \exp[-jk(R - \hat{R}_2 \cdot \vec{r}_i\,')]$$

$$\omega = -\hat{R}_1 \cdot \hat{i}$$

$$\omega' = -\hat{R}_2 \cdot \hat{i}'$$

Δ = segment length

The integrals can be evaluated as

$$G_1 = \int_{-\Delta/2\lambda}^{\Delta/2\lambda} \frac{I_i(s)}{\lambda} \exp(-j2\pi\omega s/\lambda) d(s/\lambda)$$

$$2\pi G_1 = \frac{A_i}{\lambda} \frac{2 \sin \pi\omega d}{\omega} - j \frac{B_i}{\lambda} \left(\frac{\sin[\pi(1-\omega)d]}{(1-\omega)} - \frac{\sin[\pi(1+\omega)d]}{(1+\omega)} \right) + \frac{C_i}{\lambda} \left(\frac{\sin[\pi(1-\omega)d]}{(1-\omega)} + \frac{\sin[\pi(1+\omega)d]}{(1+\omega)} \right)$$

where d = Δ/λ. The integral far G_2 (the coefficient or F_2') is the same with $\vec{r}_i$ and $\hat{i}$ reflected in the ground plane. The terms G_1 and G_2 and other necessary quantities are passed to subroutine GWAVE through COMMON/GWAV/. GWAVE returns the field components

$E_\rho^V = \rho'$ component of field due to vertical current component

$E_z^V = z$ component of field due tD vertical current component

$E_\rho^h = \rho'$ component Of field due to horizontal Current component

$E_\Phi^h = \Phi'$ component of field due to horizontal current Component

$E_z^h = z$ component of field due to horizontal current component

The common factor exp(-jkR) occurring in F_1' and F_2' is omitted from the field components and included in the total field after summation.

These field componente are then combined to form the total field in x, y, z-components and summed for each segment. The field is finally converted to r, θ, Φ components in a spherical coordinate system coinciding with the x, y, z-coordinate system.

The approximations involved in the calculation of the surface wave are valid to second order in u^2, where

u = k/k_2

k = wave number in free space

k_2 = wave number in ground medium

The approximations are valid for practical ground parameters. To ensure that the expressions are not used in an invalid range, however, the surface wave is not computed if |u| is greater than 0.5. Rather, subroutine FFLD is called, and the resulting space wave is multiplied by the range factor exp(-jkR)/(R/λ). The radial field component will be zero in this case. FFLD is also called if R/λ is greater than 10^5, or if there is no ground present.

SYMBOL DICTIONARY

A	=	coefficient of A_i/λ in $2\pi G_1$ and $2\pi G_2$
ABS	=	external routine (absolute value)
ARG	=	argument of exp( ) for phase factor
ATAN	=	external routine (arctangent)
B	=	coefficient of B_i/λ in $2\pi G_1$ and $2\pi G_2$
BOO	=	sin (BOT)/BOT
BOT	=	$\pi(1-\omega)d$
C	=	coefficient of C_i/λ in $2\pi G_1$ and $2\pi G_2$
CAB(I)	=	cos α cos β for segment I
CABS	=	external routine (magnitude of complex number)
CALP	=	cos α
CBET	=	cos β
CIX	=	x-component in summation for field
CIY	=	y-component in summation for field
CIZ	=	z-component in summation for field
CMPLX	=	external routine (forms complex number)
COS	=	external routine (cosine)
CPH	=	cos Φ'
DX DY DZ	=	x, y, z components of $\hat{i}$
EL	=	πd
EPH	=	E^h_Φ or E^h_Φ cos α (Φ' component of total field of segment I
EPI	=	Φ component of field of structure
ERD	=	R component of field of structure
ERH	=	E^h_ρ and ρ' component of total field of segment I
ERV	=	E^V_ρ
ETH	=	θ component of field of structure
EX	=	x component of field for segment I

EXA = phase factor at GD30 and GD130:
$G_1 \exp(jk\hat{R}_1 \cdot \vec{r}_i)$ or $G_2 \exp(jk\hat{R}_2 \cdot \vec{r}_i\,')$ at GD109
EY = y component of field for segment I
EZH = E_z^h and z component of total field of segment I
EZV = E_z^V
FFLD = external routine (computes space wave)
GWAVE = external routine (computes E_ρ^V, E_ρ^h, ...)
I = DO loop index (I)
K = DO loop index (loop over segment and image)
KSYMP = 1 if ground is present; 0 otherwise
OMEGA = ω
PHI = Φ
PHX = x component of $\hat{\Phi}$
PHY = y component of $\hat{\Phi}$
PI = π
R = R/λ
RFL = sign factor to reflect segment coordinates in ground
RHO = ρ/λ
RHP = ρ'/λ
RHS = $(\rho'/\lambda)^2$
RHX = x component of $\hat{\rho}$'
RHY = y component of $\hat{\rho}$'
RI = imaginary part of $2\pi G_1$ or $2\pi G_2$
RIX = x component of $\vec{R}_1/\lambda$ or $\vec{R}_2/\lambda$
RIY = y component of $\vec{R}_1/\lambda$ or $\vec{R}_2/\lambda$
RIZ = z component of $\vec{R}_1/\lambda$ or $\vec{R}_2/\lambda$
RNX
RNY = x, y, z components of $\hat{R}_1$ or $\hat{R}_2$ or $\hat{R}$
RNZ
RR = real part of $2\pi G_1$ or $2\pi G_2$
RX = x component of $\vec{\rho}/\lambda$
RXYZ = R_1/λ or R_2/λ (for s = 0)
RY = y component of $\vec{\rho}/\lambda$
RZ = z/λ
SAB(I) = cos α sin β
SBET = sin β
SILL = $\pi d\omega$
SIN = external routine (sine)
SPH = sin Φ'

SQRT	=	external routine (square root)
THET	=	θ in spherical coordinate system
THX	=	x component of $\hat{\theta}$
THY	=	y component of $\hat{\theta}$
THZ	=	z component of $\hat{\theta}$
TOO	=	sin(TOP)/TOP
TOP	=	$\pi(1+\omega)d$
TP	=	2π
U	=	u
UX	=	u
UZ	=	u^2
XX1	=	$G_l \exp(jk\hat{R}_1 \cdot \vec{r}_i)$
XXZ	=	$G_2 \exp(jk\hat{R}_2 \cdot \vec{r}_i)$
1.E-20	=	tolerance in test for zero
1.E-7	=	toierance in test for zero
1.E-6	=	tolerance in test for zero
0.5	=	upper limit for |u|
3.141592654	=	π
6.283185308	=	2π
1.5+5	=	upper limit for R/λ

```
      SUBROUTINE GFLD( RHO, PHI, RZ, ETH, EPI, ERD, UX, KSYMP)
C
C     GFLD COMPUTES THE RADIATED FIELD INCLUDING GROUND WAVE.
C
      COMPLEX  CUR, EPI, CIX, CIY, CIZ, EXA, XX1, XX2, U, U2, ERV,
     *EZV, ERH, EPH
      COMPLEX  EZH, EX, EY, ETH, UX, ERD
      COMMON  /DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X( NM), Y( NM),
     *Z( NM), SI( NM), BI( NM), ALP( NM), BET( NM), ICON1( N2M), ICON2(
     * N2M), ITAG( N2M), ICONX( NM), WLAM, IPSYM
      COMMON  /ANGL/ SALP( NM)
      COMMON  /CRNT/ AIR( NM), AII( NM), BIR( NM), BII( NM), CIR( NM),
     *CII( NM), CUR( N3M)
      COMMON  /GWAV/ U, U2, XX1, XX2, R1, R2, ZMH, ZPH
      DIMENSION  CAB(1), SAB(1)
      EQUIVALENCE(CAB(1),ALP(1)),(SAB(1),BET(1))
      DATA   PI, TP/3.141592654D+0,6.283185308D+0/
      R= SQRT( RHO* RHO+ RZ* RZ)
      IF(KSYMP.EQ.1) GOTO 1
      IF(ABS( UX).GT..5) GOTO 1
      IF(R.GT.1.E5) GOTO 1
C
C     COMPUTATION OF SPACE WAVE ONLY
C
      GOTO 4
    1 IF(RZ.LT.1.D-20) GOTO 2
      THET= ATAN( RHO/ RZ)
      GOTO 3
    2 THET= PI*.5
    3 CALL FFLD( THET, PHI, ETH, EPI)
      ARG=- TP* R
      EXA= CMPLX( COS( ARG), SIN( ARG))/ R
      ETH= ETH* EXA
      EPI= EPI* EXA
      ERD=(0.,0.)
C
C     COMPUTATION OF SPACE AND GROUND WAVES.
C
      RETURN
    4 U= UX
      U2= U* U
      PHX=- SIN( PHI)
      PHY= COS( PHI)
      RX= RHO* PHY
      RY=- RHO* PHX
      CIX=(0.,0.)
      CIY=(0.,0.)
C
C     SUMMATION OF FIELD FROM INDIVIDUAL SEGMENTS
```

```
C
      CIZ=(0.,0.)
      DO 17  I=1, N
      DX= CAB( I)
      DY= SAB( I)
      DZ= SALP( I)
      RIX= RX- X( I)
      RIY= RY- Y( I)
      RHS= RIX* RIX+ RIY* RIY
      RHP= SQRT( RHS)
      IF(RHP.LT.1.D-6) GOTO 5
      RHX= RIX/ RHP
      RHY= RIY/ RHP
      GOTO 6
    5 RHX=1.
      RHY=0.
    6 CALP=1.- DZ* DZ
      IF(CALP.LT.1.D-6) GOTO 7
      CALP= SQRT( CALP)
      CBET= DX/ CALP
      SBET= DY/ CALP
      CPH= RHX* CBET+ RHY* SBET
      SPH= RHY* CBET- RHX* SBET
      GOTO 8
    7 CPH= RHX
      SPH= RHY
    8 EL= PI* SI( I)
C
C     INTEGRATION OF (CURRENT)*(PHASE FACTOR) OVER SEGMENT AND IMAGE FOR
C     CONSTANT, SINE, AND COSINE CURRENT DISTRIBUTIONS
C
      RFL=-1.
      DO 16  K=1,2
      RFL=- RFL
      RIZ= RZ- Z( I)* RFL
      RXYZ= SQRT( RIX* RIX+ RIY* RIY+ RIZ* RIZ)
      RNX= RIX/ RXYZ
      RNY= RIY/ RXYZ
      RNZ= RIZ/ RXYZ
      OMEGA=-( RNX* DX+ RNY* DY+ RNZ* DZ* RFL)
      SILL= OMEGA* EL
      TOP= EL+ SILL
      BOT= EL- SILL
      IF(ABS( OMEGA).LT.1.D-7) GOTO 9
      A=2.* SIN( SILL)/ OMEGA
      GOTO 10
    9 A=(2.- OMEGA* OMEGA* EL* EL/3.)* EL
   10 IF(ABS( TOP).LT.1.D-7) GOTO 11
      TOO= SIN( TOP)/ TOP
```

```
      GOTO 12
   11 TOO=1.- TOP* TOP/6.
   12 IF(ABS( BOT).LT.1.D-7) GOTO 13
      BOO= SIN( BOT)/ BOT
      GOTO 14
   13 BOO=1.- BOT* BOT/6.
   14 B= EL*( BOO- TOO)
      C= EL*( BOO+ TOO)
      RR= A* AIR( I)+ B* BII( I)+ C* CIR( I)
      RI= A* AII( I)- B* BIR( I)+ C* CII( I)
      ARG= TP*( X( I)* RNX+ Y( I)* RNY+ Z( I)* RNZ* RFL)
      EXA= CMPLX( COS( ARG), SIN( ARG))* CMPLX( RR, RI)/ TP
      IF(K.EQ.2) GOTO 15
      XX1= EXA
      R1= RXYZ
      ZMH= RIZ
      GOTO 16
   15 XX2= EXA
      R2= RXYZ
      ZPH= RIZ
C
C     CALL SUBROUTINE TO COMPUTE THE FIELD OF SEGMENT INCLUDING GROUND
C     WAVE.
C
   16 CONTINUE
      CALL GWAVE( ERV, EZV, ERH, EZH, EPH)
      ERH= ERH* CPH* CALP+ ERV* DZ
      EPH= EPH* SPH* CALP
      EZH= EZH* CPH* CALP+ EZV* DZ
      EX= ERH* RHX- EPH* RHY
      EY= ERH* RHY+ EPH* RHX
      CIX= CIX+ EX
      CIY= CIY+ EY
   17 CIZ= CIZ+ EZH
      ARG=- TP* R
      EXA= CMPLX( COS( ARG), SIN( ARG))
      CIX= CIX* EXA
      CIY= CIY* EXA
      CIZ= CIZ* EXA
      RNX= RX/ R
      RNY= RY/ R
      RNZ= RZ/ R
      THX= RNZ* PHY
      THY=- RNZ* PHX
      THZ=- RHO/ R
      ETH= CIX* THX+ CIY* THY+ CIZ* THZ
      EPI= CIX* PHX+ CIY* PHY
      ERD= CIX* RNX+ CIY* RNY+ CIZ* RNZ
      RETURN
```

END GD 148

GFOUT

PURPOSE

To write the NGF file.

METHOD

The contents of the COMMON blocks in GFOUT are written to file 20. If ICASE is 3 or 5 the blocks of the LU decomposition of matrix A are on file 13 in ascending order and on file 14 in descending order. Both files are written to file 20.

SYMBOL DICTIONARY

IGFL	=	NGF file number
IOUT	=	number of elements in matrix
NEQ	=	order of matrix A
NOP	=	number of symmetric sections
NPEQ	=	number of unknowns for a symmetric section

```
      SUBROUTINE GFOUT
C
C     WRITE N.G.F. FILE
C
      INTEGER*4 COM
      COMPLEX  CM, SSX, ZRATI, ZRATI2, T1, ZARRAY, AR1, AR2, AR3,
     *EPSCF, FRATI
      COMMON  /DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X( NM), Y( NM),
     *Z( NM), SI( NM), BI( NM), ALP( NM), BET( NM), ICON1( N2M), ICON2(
     * N2M), ITAG( N2M), ICONX( NM), WLAM, IPSYM
      COMMON  /CMB/ CM(90000)
      COMMON  /ANGL/ SALP( NM)
      COMMON  /GND/ ZRATI, ZRATI2, FRATI, CL, CH, SCRWL, SCRWR, NRADL,
     *KSYMP, IFAR, IPERF, T1, T2
      COMMON  /GGRID/ AR1(11,10,4), AR2(17,5,4), AR3(9,8,4), EPSCF, DXA
     *(3), DYA(3), XSA(3), YSA(3), NXA(3), NYA(3)
      COMMON  /MATPAR/ ICASE, NBLOKS, NPBLK, NLAST, NBLSYM, NPSYM,
     *NLSYM, IMAT, ICASX, NBBX, NPBX, NLBX, NBBL, NPBL, NLBL
      COMMON  /SMAT/ SSX(16,16)
      COMMON  /ZLOAD/ ZARRAY( NM), NLOAD, NLODF
      COMMON  /SAVE/ IP( N2M), KCOM, COM(20,5), EPSR, SIG, SCRWLT,
     *SCRWRT, FMHZ
      DATA   IGFL/20/
      NEQ= N+2* M
      NPEQ= NP+2* MP
      NOP= NEQ/ NPEQ
      WRITE( IGFL)  N, NP, M, MP, WLAM, FMHZ, IPSYM, KSYMP, IPERF,
     *NRADL, EPSR, SIG, SCRWLT, SCRWRT, NLOAD, KCOM
      IF(N.EQ.0) GOTO 1
      WRITE( IGFL) ( X( I), I=1, N),( Y( I), I=1, N),( Z( I), I=1, N)
      WRITE( IGFL) ( SI( I), I=1, N),( BI( I), I=1, N),( ALP( I), I=1,
     *N)
      WRITE( IGFL) ( BET( I), I=1, N),( SALP( I), I=1, N)
      WRITE( IGFL) ( ICON1( I), I=1, N),( ICON2( I), I=1, N)
      WRITE( IGFL) ( ITAG( I), I=1, N)
      IF(NLOAD.GT.0) WRITE( IGFL) ( ZARRAY( I), I=1, N)
    1 IF(M.EQ.0) GOTO 2
      J= LD- M+1
      WRITE( IGFL) ( X( I), I= J, LD),( Y( I), I= J, LD),( Z( I), I= J,
     * LD)
      WRITE( IGFL) ( SI( I), I= J, LD),( BI( I), I= J, LD),( ALP( I), I
     *= J, LD)
      WRITE( IGFL) ( BET( I), I= J, LD),( SALP( I), I= J, LD)
      WRITE( IGFL) ( ICON1( I), I= J, LD),( ICON2( I), I= J, LD)
      WRITE( IGFL) ( ITAG( I), I= J, LD)
    2 WRITE( IGFL)  ICASE, NBLOKS, NPBLK, NLAST, NBLSYM, NPSYM, NLSYM,
     *IMAT
      IF(IPERF.EQ.2) WRITE( IGFL)  AR1, AR2, AR3, EPSCF, DXA, DYA, XSA
     *, YSA, NXA, NYA
```

```
      IF(NOP.GT.1) WRITE( IGFL) (( SSX( I, J), I=1, NOP), J=1, NOP)      GO  50
      WRITE( IGFL) ( IP( I), I=1, NEQ), COM                              GO  51
      IF(ICASE.GT.2) GOTO 3                                              GO  52
      IOUT= NEQ* NPEQ                                                    GO  53
      WRITE( IGFL) ( CM( I), I=1, IOUT)                                  GO  54
      GOTO 12                                                            GO  55
    3 IF(ICASE.NE.4) GOTO 5                                              GO  56
      REWIND 13                                                          GO  57
      I= NPEQ* NPEQ                                                      GO  58
      DO 4  K=1, NOP                                                     GO  59
      READ(13) ( CM( J), J=1, I)                                         GO  60
    4 WRITE( IGFL) ( CM( J), J=1, I)                                     GO  61
      REWIND 13                                                          GO  62
      GOTO 12                                                            GO  63
    5 REWIND 13                                                          GO  64
      REWIND 14                                                          GO  65
      IF(ICASE.EQ.5) GOTO 8                                              GO  66
      IOUT= NPBLK* NEQ*2                                                 GO  67
      DO 6  I=1, NBLOKS                                                  GO  68
      CALL BLCKIN( CM,13,1, IOUT,1,201)                                  GO  69
    6 CALL BLCKOT( CM, IGFL,1, IOUT,1,202)                               GO  70
      DO 7  I=1, NBLOKS                                                  GO  71
      CALL BLCKIN( CM,14,1, IOUT,1,203)                                  GO  72
    7 CALL BLCKOT( CM, IGFL,1, IOUT,1,204)                               GO  73
      GOTO 12                                                            GO  74
    8 IOUT= NPSYM* NPEQ*2                                                GO  75
      DO 11  IOP=1, NOP                                                  GO  76
      DO 9  I=1, NBLSYM                                                  GO  77
      CALL BLCKIN( CM,13,1, IOUT,1,205)                                  GO  78
    9 CALL BLCKOT( CM, IGFL,1, IOUT,1,206)                               GO  79
      DO 10  I=1, NBLSYM                                                 GO  80
      CALL BLCKIN( CM,14,1, IOUT,1,207)                                  GO  81
   10 CALL BLCKOT( CM, IGFL,1, IOUT,1,208)                               GO  82
   11 CONTINUE                                                           GO  83
      REWIND 13                                                          GO  84
      REWIND 14                                                          GO  85
   12 REWIND IGFL                                                        GO  86
      WRITE (2,13)  IGFL, IMAT                                           GO  87
C                                                                        GO  88
      RETURN                                                             GO  89
   13 FORMAT(///,' ****NUMERICAL GREEN S FUNCTION FILE ON TAPE',I3,      GO  90
     *'****',/,5X,'MATRIX STORAGE -',I7,' COMPLEX NUMBERS',///)          GO  91
      END                                                                GO  92
```

GH

PURPOSE

To compute the function that is numerically integrated for the near H Field of a segment.

METHOD

The value returned by GH is

$$G = \left[\frac{1}{(kr)^3} + \frac{j}{(kr)^2} \right] \exp(-jkr),$$

where

r	=	$\left[\rho'^2 + (z - z')^2 \right]^{1/2}$
ρ'	=	ρ coordinate of the field observation point in a cylindrical coordinate system with origin at the center of the source segment and z-axis oriented along the source segment
z'	=	z coordinate of the field observation point in the cylindrical coordinate system
z	=	z coordinate of the integration point an the source segment
k	=	$2\pi/\lambda$

SYMBOL DICTIONARY

CKR	=	cos kr
HR	=	real part of G
HI	=	imaginary part of G
R	=	kr
RHKS	=	(kρ')2
RR2	=	1/(kr)^2
RR3	=	1/(kr)^3
RS	=	(kr)^2
SKR	=	sin kr
ZK	=	kz
ZPK	=	kz'

```
      SUBROUTINE GH(ZK,HR,HI)
C     INTEGRAND FOR H FIELD OF A WIRE
      IMPLICIT REAL (A-H,O-Z)
      COMMON/TMH/ ZPK,RHKS
      RS=ZK-ZPK
      RS=RHKS+RS*RS
      R=SQRT(RS)
      CKR=COS(R)
      SKR=SIN(R)
      RR2=1./RS
      RR3=RR2/R
      HR=SKR*RR2+CKR*RR3
      HI=CKR*RR2-SKR*RR3
      RETURN
      END
```

GWAVE

PURPOSE

To compute the components of electric field due to an electric current element over a ground plane at intermediate distances, including the surface wave field.

METHOD

Approximate expressions for the electric field of a vertical or horizontal infinitesimal current element above a ground plane, including surface wave, were derived by K. A. Norton (ref. 2). The geometry is shown in figure 6a for a current element at height a above the ground plane and field observation point at p. The current element is located on the z' axis, and the horizontal current element is directed along the x' axis. The vertical current element produces z' and ρ' field components given by

$$E_z^V = -\frac{j\eta I d\ell}{2\lambda}\Bigg\{\cos^2\psi'\frac{\exp(-jkR_1)}{R_1} + R_V\cos^2\psi\frac{\exp(-jkR_2)}{R_2}$$

$$+(1-R_V)\cos^2\psi \;\; F\,\frac{\exp(-jkR_2)}{R_2}$$

$$+\;u\sqrt{1-u^2\cos^2\psi}\;\;\sin\;\;\psi\;\;2\;\;\frac{\exp(-jkR_2)}{jkR_2^2}i$$

$$+\;\frac{\exp(-jkR_1)}{R_1}\;\;\left(\frac{1}{jkR_1}+\frac{1}{(jkR_1)^2}\right)(1-3\sin^2\psi')$$

$$+\;\frac{\exp(-jkR_2)}{R_2}\;\;\left(\frac{1}{jkR_2}+\frac{1}{(jkR_2)^2}\right)(1-3\sin^2\psi)\;\;,$$

$$E_\rho^V = \frac{j\eta I d\ell}{2\lambda}\Bigg\{\sin\psi'\cos\psi'\frac{\exp(-jkR_1)}{R_1} + R_V\sin\psi\cos\psi\frac{\exp(-jkR_2)}{R_2}$$

$$-\;\;\cos\psi(1-R_V)u\sqrt{1-u^2\cos^2\psi}\;\;F\;\;\frac{\exp(-jkR_2)}{R_2}$$

$$-\sin\psi\cos\psi(1-R_V)\;\;\frac{\exp(-jkR_2)}{jkR_2^2}$$

$$+\;3\;\;\sin\psi'\cos\psi'\left(\frac{1}{jkR_1}+\frac{1}{(jkR_1)^2}\right)\frac{\exp(-jkR_1)}{R_1}$$

$$-\;\;\cos\psi(1-R_V)u\sqrt{1-u^2\cos^2\psi}\;\;\frac{\exp(-jkR_2)}{jkR_2^2}$$

$$+\;3\;\;\sin\psi\cos\psi\left(\frac{1}{jkR_2}+\frac{1}{(jkR_2)^2}\right)\frac{\exp(-jkR_2)}{R_2}$$

where

F	=	1 - j$\sqrt{\pi w}\exp(-w)\ erfc(j\sqrt{w})$
erfc(z)	=	1 - erf(z)
erf(z)	=	2/$\sqrt{\pi}\ \int_0^z \exp(-t^2)dt$ (error function)
w	=	$4p_1/(1 - R_V)^2$
p_1	=	$-jkR_2 u^2(1-u^2\cos^2\psi)/(2\cos^2\psi)$
R_V	=	$\frac{\sin\psi - u\sqrt{1-u^2\cos^2\psi}}{\sin\psi + u\sqrt{1-u^2\cos^2\psi}}$
u	=	k/k_2
k	=	wave number in free space
k_2	=	wave number in lower medium
$\sin\psi$	=	$(z + a)/R_2$
$\sin\psi'$	=	$(z - a)/R_1$

The horizontal current element directed along the x' axis produces ρ', Φ', and z' field components given by

$$E_z^h = \frac{j\eta I d\ell}{2\lambda}\ \cos\Phi' \Bigg\{ \sin\psi'\cos\psi' \frac{\exp(-jkR_1)}{R_1}$$

$$-R_v \sin\psi\cos\psi\cdot\frac{\exp(-jkR_2)}{R_2}$$

$$+\cos\psi(1-R_v)u\sqrt{1-u^2\cos^2\psi}\ \ F\ \ \frac{\exp(-jkR_2)}{R_2}$$

$$+\sin\psi\cos\psi(1-R_v)\frac{\exp(-jkR_2)}{jkR_2^2}$$

$$+3\sin\psi'\cos\psi'\left(\frac{1}{jkR_1}+\frac{1}{(jkR_1)^2}\right)\frac{\exp(-jkR_1)}{R_1}$$

$$+\cos\psi(1-R_v)u\sqrt{1-u^2\cos^2\psi}\ \ \frac{\exp(-jkR_2)}{2jkR_2^2}$$

$$-3\sin\psi\cos\psi\left(\frac{1}{jkR_2}+\frac{1}{(jkR_2)^2}\right)\frac{\exp(-jkR_2)}{2jkR_2^2}\Bigg\}\ ,$$

$$E_\rho^h = \frac{-j\eta I d\ell}{2\lambda}\cos\Phi'\Bigg\{\sin^2\psi'\frac{\exp(-jkR_1)}{R_1} - R_v\sin^2\psi\frac{\exp(-jkR_2)}{R_2}$$

$$-(1-u^2\cos^2\psi)u^2(1-R_v)F\frac{\exp(-jkR_2)}{R_2}$$

$$+\left(\frac{1}{jkR_1}+\frac{1}{(jkR_1)^2}\right)(1-3\cos^2\psi')\frac{\exp(-jkR_1)}{R_1}$$

$$-\left(\frac{1}{jkR_2}+\frac{1}{(jkR_2)^2}\right)(1-3\cos^2\psi)\Bigg[1-u^2(1+R_v)-u^2(1-R_v)$$

$$\times\frac{\exp(-jkR_2)}{R_2}+u^2\cos^2(1-R_v)\left(1+\frac{1}{jkR_2}\right)$$

$$\times\left[F\left(u^2(1-u^2\cos^2\psi)-\sin^2\psi+\frac{1}{jkR_2}\right)-\frac{1}{jkR_2}\right]\frac{\exp(-jkR_2)}{R_2}\Bigg\}$$

$$E_\Phi^h = \frac{j\eta I d\ell}{2\lambda}\sin\Phi'\Bigg\{\frac{\exp(-jkR_1)}{R_1} - R_h\frac{\exp(-jkR_2)}{R_2}$$

$$+(R_h+1)G\frac{\exp(-jkR_2)}{R_2}+\left(1+\frac{1}{jkR_1}\right)\frac{\exp(-jkR_1)}{jkR_1^2}$$

$$-\left(1+\frac{1}{jkR_2}\right)\left[1-u^2(1+R_v)-u^2(1-R_v)F\right]\frac{\exp(-jkR_2)}{jkR_2^2}$$

$$-\frac{u^2(1-R_v)}{2}\left[F\left(u^2(1-u^2\cos^2\psi)-\sin^2\psi+\frac{1}{jkR_2}\right)-\frac{1}{jkR_2}\right]$$

$$\times\frac{\exp(-jkR_2)}{jkR_2^2}\Bigg\}\ ,$$

where

$$G = [1 - j\sqrt{\pi v}\exp(-v) erfc(j\sqrt{v}] \ ,$$

$$v = 4q_1/(1 + R_h)^2$$

$$q_1 = -jkR_2(1 - u^2\cos^2\psi)/(2u^2\cos^2\psi)$$

$$R_h = \frac{\sqrt{1 - u^2\cos^2\psi} - u\sin\psi}{\sqrt{1 - u^2\cos^2\psi} + u\sin\psi}$$

The approximations in these expressions are valid for E_l and R_2 greater than about a wavelength and to second order in u^2. In each equation, the first term represents the direct space wave field of the current element, the second term is the space wave field reflected from the ground, and the following higher order terms involving F and G represent the ground wave. It may be noted that the coefficients R_v and R_h are the Fresnel reflection coefficients for vertical and horizontal polarization, respectively.

To obtain the field due to a structure, these expressions are integrated over each segment and the fields of the segments are summed in subroutine GFLD. For integration, R_1 and R_2 are the distances from the integration point ℓ on the segment to point p. Since R_1 and R_2 are assumed large compared to the segment length, R_1, R_2, ψ, and ψ' are considered constant during integration over the segment except where jkR_1 and jkR_2 occur in exponential functions. Thus, if s represents distance along the segment, the integral of each expression over the segment is obtained by replacing $(Id\ell/\lambda^2\exp(-jkR_1)$ and $(Id\ell/\lambda^2\exp(-jkR_2)$ by XX1 and XX2 from subroutine GFLD. A factor of exp(-jkR) is omitted from the fields and is included after summation in GFLD. Including a factor of 1/λ^2 in XX1 and XX2 makes a factor of λ available to normalize R_1 and R_2 in the denominators of the field expressions. The factors sin Φ' or cos Φ' are omitted from the fields due to a horizontal current element in GMAVE and are supplied later.

SYMBOL DICTIONARY

CPP	=	cos ψ
CPPP	=	cos ψ'
CPPP2	=	$\cos^2\psi'$
CPP2	=	$\cos^2\psi$
ECON	=	-jη/2 (η = impedance of free space)
EPH	=	$E^h_\Phi/\sin\Phi'$
ERH	=	$E^h_\rho/\cos\Phi'$
ERV	=	E^v_ρ
EZH	=	$E^h_z/\cos\Phi'$
EZV	=	E^v_z
F	=	F
FJ	=	j = $\sqrt{-1}$
G	=	G
OMR	=	1 - R_v
PI	=	π
P1	=	p_1
Q1	=	q_1
RH	=	R_n
RK1	=	-jkR_l
RK2	=	-jkR_2
RV	=	R_v
R1	=	R_1/λ
R2	=	R_2/λ
SPP	=	sin ψ
SPPP	=	sin ψ'
SPPP2	=	$\sin^2\psi'$
SPP2	=	$\sin^2\psi$
TPJ	=	2πj
T1	=	1 - $u^2\cos^2\psi$
T2	=	$\sqrt{T1}$
T3	=	-[1/(jkR_1)+ 1/(jkR_1)^2]
T4	=	-[1/(jkR_2)+ 1/(jkR_2)^2]
U	=	u
U2	=	u^2
V	=	v
W	=	w
XR1	=	XX1/(R/λ)
XR2	=	XX2/(R/λ)
XX1	=	$G_1\exp(jk\hat{R}_1\cdot\vec{r}_i)$
XX2	=	$G_2\exp(jk\hat{R}_2\cdot\vec{r}'_i)$
X1,X2,...,X7	=	first, second, ..., seventh term in each field expression
ZMH	=	z - a
ZPH	=	z + a

```
      SUBROUTINE GWAVE( ERV, EZV, ERH, EZH, EPH)
C
C     GWAVE COMPUTES THE ELECTRIC FIELD, INCLUDING GROUND WAVE, OF A
C     CURRENT ELEMENT OVER A GROUND PLANE USING FORMULAS OF K.A. NORTON
C     (PROC. IRE, SEPT., 1937, PP.1203,1236.)
C
      COMPLEX  FJ, TPJ, U2, U, RK1, RK2, T1, T2, T3, T4, P1, RV, OMR
     *, W, F, Q1, RH, V, G, XR1, XR2, X1, X2, X3, X4, X5, X6, X7, EZV,
     *ERV, EZH, ERH, EPH, XX1, XX2, ECON, FBAR
      COMMON  /GWAV/ U, U2, XX1, XX2, R1, R2, ZMH, ZPH
      DIMENSION  FJX(2), TPJX(2), ECONX(2)
      EQUIVALENCE(FJ,FJX),(TPJ,TPJX),(ECON,ECONX)
      DATA   PI/3.141592654D+0/, FJX/0.,1./, TPJX/0.,6.283185308D+0/
      DATA   ECONX/0.,-188.367/
      SPPP= ZMH/ R1
      SPPP2= SPPP* SPPP
      CPPP2=1.- SPPP2
      IF(CPPP2.LT.1.D-20) CPPP2=1.D-20
      CPPP= SQRT( CPPP2)
      SPP= ZPH/ R2
      SPP2= SPP* SPP
      CPP2=1.- SPP2
      IF(CPP2.LT.1.D-20) CPP2=1.D-20
      CPP= SQRT( CPP2)
      RK1=- TPJ* R1
      RK2=- TPJ* R2
      T1=1.- U2* CPP2
      T2= SQRT( T1)
      T3=(1.-1./ RK1)/ RK1
      T4=(1.-1./ RK2)/ RK2
      P1= RK2* U2* T1/(2.* CPP2)
      RV=( SPP- U* T2)/( SPP+ U* T2)
      OMR=1.- RV
      W=1./ OMR
      W=(4.,0.)* P1* W* W
      F= FBAR( W)
      Q1= RK2* T1/(2.* U2* CPP2)
      RH=( T2- U* SPP)/( T2+ U* SPP)
      V=1./(1.+ RH)
      V=(4.,0.)* Q1* V* V
      G= FBAR( V)
      XR1= XX1/ R1
      XR2= XX2/ R2
      X1= CPPP2* XR1
      X2= RV* CPP2* XR2
      X3= OMR* CPP2* F* XR2
      X4= U* T2* SPP*2.* XR2/ RK2
      X5= XR1* T3*(1.-3.* SPPP2)
      X6= XR2* T4*(1.-3.* SPP2)
```

```
      EZV=( X1+ X2+ X3- X4- X5- X6)* ECON
      X1= SPPP* CPPP* XR1
      X2= RV* SPP* CPP* XR2
      X3= CPP* OMR* U* T2* F* XR2
      X4= SPP* CPP* OMR* XR2/ RK2
      X5=3.* SPPP* CPPP* T3* XR1
      X6= CPP* U* T2* OMR* XR2/ RK2*.5
      X7=3.* SPP* CPP* T4* XR2
      ERV=-( X1+ X2- X3+ X4- X5+ X6- X7)* ECON
      EZH=-( X1- X2+ X3- X4- X5- X6+ X7)* ECON
      X1= SPPP2* XR1
      X2= RV* SPP2* XR2
      X4= U2* T1* OMR* F* XR2
      X5= T3*(1.-3.* CPPP2)* XR1
      X6= T4*(1.-3.* CPP2)*(1.- U2*(1.+ RV)- U2* OMR* F)* XR2
      X7= U2* CPP2* OMR*(1.-1./ RK2)*( F*( U2* T1- SPP2-1./ RK2)+1./
     *RK2)* XR2
      ERH=( X1- X2- X4- X5+ X6+ X7)* ECON
      X1= XR1
      X2= RH* XR2
      X3=( RH+1.)* G* XR2
      X4= T3* XR1
      X5= T4*(1.- U2*(1.+ RV)- U2* OMR* F)* XR2
      X6=.5* U2* OMR*( F*( U2* T1- SPP2-1./ RK2)+1./ RK2)* XR2/ RK2
      EPH=-( X1- X2+ X3- X4+ X5+ X6)* ECON
      RETURN
      END
```

GX

PURPOSE

To evaluate terms for the field contribution due to segment ends in the thin wire kernel.

SYMBOL DICTIONARY

GZ = exp(-jkr)/r = G_0
GZP = -(1 + jkr) exp(-jkr)/r^3
R = r
R2 = $r^2 = \rho^2 + z^2$
RH = ρ
RK = kR
XK = $2\pi/\lambda$
ZZ = z

CODE LISTING

```
      SUBROUTINE GX(ZZ,RH,XK,GZ,GZP)
C     SEGMENT END CONTRIBUTIONS FOR THIN WIRE APPROX.
      COMPLEX GZ,GZP
      R2=ZZ*ZZ+RH*RH
      R=SQRT(R2)
      RKZ=XK*R
      GZ=CMPLX(COS(RKZ),-SIN(RKZ))/ R
      GZP=-CMPLX(1.0,RKZ)*GZ/ R2
      RETURN
      END
```

GXX

PURPOSE

To evaluate terms for the field contribution due to segment ends in the extended thin wire kernel.

METHOD

Equations 59 through 94 in Part I are evaluated for ρ > a, and equations 99 through 103 for ρ < a. Several variables are used for storage of intermediate results before being set to their final values.

SYMBOL DICTIONARY

A	=	radius of source segment, a
A2	=	a^2
Cl	=	1 + jkr_0
C2	=	3(1 + jkr_0) - $k^2r_0^2$
C3	=	(6 + jkr_0)$k^2r_0^2$ - 15(1 + jkr_0)
G1	=	G_1
G1P	=	$\partial G_1/\partial z'$
G2	=	G_2
G2P	=	$\partial G_2/\partial z'$
G3	=	$\partial G_1/\partial \rho$p
GZ	=	G_0
GZP	=	$\partial G_0/\partial z'$
IRA	=	1 to indicate ρ < a
R	=	r_0
R2	=	r_0^2
R4	=	r_0^4
RH	=	ρ
RH2	=	ρ^2
RK	=	kr_0
RK2	=	$k^2r_0^2$
T1	=	$a^2\rho^2/4r^4$
T2	=	$a^2/2r^2$
XK	=	k = $2\pi/\lambda$
ZZ	=	z' = z

```
      SUBROUTINE GXX(ZZ,RH,A,A2,XK,IRA,G1,G1P,G2,G2P,G3,GZP)
C     SEGMENT END CONTRIBUTIONS FOR EXT. THIN WIRE APPROX.
      COMPLEX  GZ,C1,C2,C3,G1,G1P,G2,G2P,G3,GZP
      R2=ZZ*ZZ+RH*RH
      R=SQRT(R2)
      R4=R2*R2
      RK=XK*R
      RK2=RK*RK
      RH2=RH*RH
      T1=.25*A2*RH2/ R4
      T2=.5*A2/R2
      C1=CMPLX(1.0,RK)
      C2=3.0*C1- RK2
      C3=CMPLX(6.0,RK)*RK2-15.*C1
      GZ=CMPLX(COS(RK),-SIN(RK))/R
      G2=GZ*(1.+T1*C2)
      G1=G2-T2*C1*GZ
      GZ=GZ/R2
      G2P=GZ*(T1*C3-C1)
      GZP=T2*C2*GZ
      G3=G2P+GZP
      G1P=G3*ZZ
      IF(IRA.EQ.1) GOTO 2
      G3=(G3+GZP)*RH
      GZP=-ZZ*C1*GZ
      IF(RH.GT.1.D-10) GOTO 1
      G2=0.0
      G2P=0.0
      RETURN
    1 G2=G2/RH
      G2P=G2P*ZZ/RH
      RETURN
    2 T2=.5*A
      G2=-T2*C1*GZ
      G2P=T2*GZ*C2/ R2
      G3=RH2*G2P-A*GZ*C1
      G2P=G2P*ZZ
      GZP=-ZZ*C1*GZ
      RETURN
      END
```

```
      SUBROUTINE HELIX(S,HL,A1,B1,A2,B2,RAD,NS,ITG)
C     SUBROUTINE HELIX GENERATES SEGMENT GEOMETRY DATA FOR A HELIX OF NS
C     SEGMENTS
      COMMON/DATA/ LD,N1,N2,N,NP,M1,M2,M,MP,X(NM),Y(NM),
     *Z(NM),SI(NM),BI(NM),ALP(NM),BET(NM),ICON1(N2M),ICON2(
     * N2M),ITAG(N2M),ICONX(NM),WLAM,IPSYM
      DIMENSION  X2(1),Y2(1),Z2(1)
      EQUIVALENCE (X2(1),SI(1)),(Y2(1),ALP(1)),(Z2(1),BET(1))
      DATA PI/3.1415926D+0/
      IST=N+1
      N=N+NS
      NP=N
      MP=M
      IPSYM=0
      IF(NS.LT.1) RETURN
      TURNS=ABS(HL/S)
      ZINC=ABS(HL/NS)
      Z(IST)=0.
      DO 25 I=IST,N
      BI(I)=RAD
      ITAG(I)=ITG
      IF(I.NE.IST) Z(I)= Z(I-1)+ ZINC
      Z2(I)=Z(I)+ ZINC
      IF(A2.NE.A1) GOTO 10
      IF(B1.EQ.0) B1= A1
      X(I)=A1*COS(2.* PI* Z(I)/ S)
      Y(I)=B1*SIN(2.* PI* Z(I)/ S)
      X2(I)=A1*COS(2.* PI* Z2(I)/ S)
      Y2(I)=B1*SIN(2.* PI* Z2(I)/ S)
      GOTO 20
   10 IF(B2.EQ.0) B2= A2
      X(I)=(A1+(A2-A1)*Z(I)/ABS(HL))*COS(2.*PI*Z(I)/S)
      Y(I)=(B1+(B2-B1)*Z(I)/ABS(HL))*SIN(2.*PI*Z(I)/S)
      X2(I)=(A1+(A2-A1)*Z2(I)/ABS(HL))*COS(2.*PI*Z2(I)/S)
      Y2(I)=(B1+(B2-B1)*Z2(I)/ABS(HL))*SIN(2.*PI*Z2(I)/S)
   20 IF(HL.GT.0) GOTO 25
      COPY=X(I)
      X(I)=Y(I)
      Y(I)=COPY
      COPY=X2(I)
      X2(I)=Y2(I)
      Y2(I)=COPY
   25 CONTINUE
      IF(A2.EQ.A1) GOTO 21
      SANGLE=ATAN(A2/(ABS(HL)+(ABS(HL)*A1)/(A2-A1)))
      WRITE (2,104) SANGLE
  104 FORMAT(5X,'THE CONE ANGLE OF THE SPIRAL IS',F10.4)
      RETURN
   21 IF(A1.NE.B1) GOTO 30
```

```
      HDIA=2.0*A1
      TURN=HDIA*PI
      PITCH=ATAN(S/(PI*HDIA))
      TURN=TURN/COS(PITCH)
      PITCH=180.*PITCH/PI
      GOTO 40
   30 IF(A1.LT.B1) GOTO 34
      HMAJ=2.*A1
      HMIN=2.*B1
      GOTO 35
   34 HMAJ=2.*B1
      HMIN=2.*A1
   35 HDIA=SQRT((HMAJ**2+ HMIN**2)/2*HMAJ)
      TURN=2.*PI*HDIA
      PITCH=(180./PI)*ATAN(S/(PI*HDIA))
   40 WRITE (2,105) PITCH,TURN
  105 FORMAT(5X,'THE PITCH ANGLE IS',F10.4/5X,
     *'THE LENGTH OF WIRE/TURN ''IS',F10.4)
      RETURN
      END
```

HFK

PURPOSE

To compute the near H field of a uniform current filament by numerical integration.

METHOD

The H field of a current filament of length Δ with uniform current distribution of magnitude I = λ is

$$H_\Phi = \frac{k\rho'}{2}\int_{-k\Delta/2}^{k\Delta/2}\left[\frac{1}{(kr)^3}+\frac{1}{(kr)^2}\right]\exp(-jkr)\,d(kz),$$

where r, ρ', z' and z are defined in the description of subroutine GH. The numerical integration is performed by the method of Romberg quadrature with variable interval width, which is described in the discussion of subroutine INTX. The integral is multiplied by kρ'/2 at HF79 and HF80 in the Code.

SYMBOL DICTIONARY

This listing excludes those variables used in the numerical quadrature algorithm, which are defined under subroutin INTX.

RHK	=	kρ'
RHKS	=	(kρ')
SGI	=	imaginary part of H_Φ
SGR	=	real part of H_Φ
ZPK	=	kz' (z' = z coordinate of observation point)
ZPKX	=	ZPK

```
      SUBROUTINE HFK(EL1,EL2,RHK,ZPKX,SGR,SGI)
C     HFK COMPUTES THE H FIELD OF A UNIFORM CURRENT FILAMENT BY
C     NUMERICAL INTEGRATION
      COMMON/TMH/ ZPK,RHKS
      DATA  NX,NM,NTS,RX/1,65536,4,1.D-4/
      ZPK=ZPKX
      RHKS=RHK* RHK
      Z=EL1
      ZE=EL2
      S=ZE- Z
      EP=S/(10.* NM)
      ZEND=ZE- EP
      SGR=0.0
      SGI=0.0
      NS=NX
      NT=0
      CALL GH( Z, G1R, G1I)
    1 DZ=S/ NS
      ZP=Z+ DZ
      IF(ZP- ZE) 3,3,2
    2 DZ=ZE- Z
      IF(ABS( DZ)- EP) 17,17,3
    3 DZOT=DZ*.5
      ZP=Z+ DZOT
      CALL GH( ZP, G3R, G3I)
      ZP=Z+ DZ
      CALL GH( ZP, G5R, G5I)
    4 T00R=( G1R+ G5R)* DZOT
      T00I=( G1I+ G5I)* DZOT
      T01R=( T00R+ DZ* G3R)*0.5
      T01I=( T00I+ DZ* G3I)*0.5
      T10R=(4.0* T01R- T00R)/3.0
      T10I=(4.0* T01I- T00I)/3.0
      CALL TEST( T01R, T10R, TE1R, T01I, T10I, TE1I,0.)
      IF(TE1I- RX) 5,5,6
    5 IF(TE1R- RX) 8,8,6
    6 ZP=Z+ DZ*0.25
      CALL GH( ZP, G2R, G2I)
      ZP=Z+ DZ*0.75
      CALL GH( ZP, G4R, G4I)
      T02R=( T01R+ DZOT*( G2R+ G4R))*0.5
      T02I=( T01I+ DZOT*( G2I+ G4I))*0.5
      T11R=(4.0* T02R- T01R)/3.0
      T11I=(4.0*T02I-T01I)/3.0
      T20R=(16.0*T11R-T10R)/15.0
      T20I=(16.0*T11I-T10I)/15.0
      CALL TEST(T11R,T20R,TE2R,T11I,T20I,TE2I,0.0)
      IF(TE2I-RX) 7,7,14
    7 IF(TE2R-RX) 9,9,14
```

```
    8 SGR=SGR+T10R
      SGI=SGI+T10I
      NT=NT+2
      GOTO 10
    9 SGR=SGR+T20R
      SGI=SGI+T20I
      NT=NT+1
   10 Z=Z+DZ
      IF(Z-ZEND) 11,17,17
   11 G1R=G5R
      G1I=G5I
      IF(NT-NTS) 1,12,12
   12 IF(NS-NX) 1,1,13
   13 NS=NS/2
      NT=1
      GOTO 1
   14 NT=0
      IF(NS-NM) 16,15,15
   15 WRITE(2,18)  Z
      GOTO 9
   16 NS=NS*2
      DZ=S/NS
      DZOT=DZ*0.5
      G5R=G3R
      G5I=G3I
      G3R=G2R
      G3I=G2I
      GOTO 4
   17 CONTINUE
      SGR=SGR* RHK*.5
      SGI=SGI* RHK*.5
C
      RETURN
   18 FORMAT(' STEP SIZE LIMITED AT Z = ',F10.5)
      END
```

HINTG

PURPOSE

To compute the near magnetic field due to a single patch in free space or over ground.

METHOD

The magnetic field is computed at the point, XI,YI,ZI due to the patch defined by parameters in COMMON/DATAJ/. The H field at $\vec{r} = (XI)\hat{x} + (YI)\hat{y} + (ZI)\hat{z}$ due to patch i, centered at $\vec{r}_i$, is approximated as:

$$\vec{H}(r) = -\frac{1}{4\pi}\left[(1 + jkR)\frac{\exp(-jkR)}{(R/\lambda)^3}\right]\left[(\vec{R}/\lambda) \times \vec{J}_i\right]A_i/\lambda^2$$

where $\vec{R} = \vec{r} - \vec{r}_i$, and A_i is the area of patch i. This expression treats the surface currents as lumped at the center of the patch. H is computed for unit currents along the surface vectors $\hat{t}_{1i}$ and $\hat{t}_{2i}$.

When a ground is present, the code is executed twice in a loop. In the second pass, the field of the image of the patch is computed, multiplied by the reflection coefficients, and added to the direct field.

SYMBOL DICTIONARY

CR = cos(kR)

CTH = cos θ, θ = angle between the reflected ray and the normal to the ground

EXC
EYC = x,y and z-components of H excluding ($\times\vec{J}_i$) term
EZC

EXK
EYK = $\vec{H}$ for $\vec{J}_i = \hat{t}_{1i}$
EZK

EXS
EYS = $\vec{H}$ for I$\vec{J}_i = \hat{t}_{2i}$
EZS

F1X
F1Y = $\vec{H}$ for $\vec{J}_i = \hat{t}_{1i}$; direct or reflected field contribution
F1Z

F2X		
F2Y	=	$\vec{H}$ for $\vec{J_i}$ = $\hat{t}_{2i}$; direct or reflected field contribution
F2Z		
FPI	=	4π
GAM	=	H excluding the term $(\vec{R}/\lambda) \times \vec{J_i}$
IP	=	1 for direct field, 2 for reflected field
IPERF	=	1 for perfect ground, 0 otherwise
KSYMP	=	1 for free space, 2 for ground
PX		
PY	=	unit vector normal to plane of incidence for reflected ray $\hat{\rho}$
R	=	R/λ
RFL	=	+1 for direct field, -1 for reflected field
RK	=	kR; k = $2\pi/lambda$
RRH	=	R_H
RRV	=	R_V
RSQ	=	R^2/λ^2
RX		
RY	=	$\vec{R}/\lambda$
RZ		
S	=	A_i/λ^2
SR	=	sin(kR)
T1XJ		
T1YJ	=	$\hat{t}_{1i}$
T1ZJ		
T2XJ		
T2YJ	=	$\hat{t}_{2i}$
T2ZJ		
T1ZR	=	z component of $\hat{t}_{1i}$ for patch i or for the image of patch i reflected in the ground
T2ZR	=	same as T12R for $\hat{t}_{2i}$
TP	=	2π
XI		
YI	=	field evaluation point $\vec{r}/\lambda$
ZI		
XJ		
YJ	=	position of center of patch $\vec{r}_i/\lambda$
ZJ		
XYMAG	=	magnitude of $\vec{R}/\lambda$ projected on the x-y plame
12.56637062	=	4π
6.283185308	=	2π

```
      SUBROUTINE HINTG( XI, YI, ZI)
C     HINTG COMPUTES THE H FIELD OF A PATCH CURRENT
      COMPLEX  EXK, EYK, EZK, EXS, EYS, EZS, EXC, EYC, EZC, ZRATI,
     *ZRATI2, GAM, F1X, F1Y, F1Z, F2X, F2Y, F2Z, RRV, RRH, T1, FRATI
      COMMON  /DATAJ/ S, B, XJ, YJ, ZJ, CABJ, SABJ, SALPJ, EXK, EYK,
     *EZK, EXS, EYS, EZS, EXC, EYC, EZC, RKH, IEXK, IND1, INDD1, IND2,
     *INDD2, IPGND
      COMMON  /GND/ ZRATI, ZRATI2, FRATI, CL, CH, SCRWL, SCRWR, NRADL,
     *KSYMP, IFAR, IPERF, T1, T2
      EQUIVALENCE(T1XJ,CABJ),(T1YJ,SABJ),(T1ZJ,SALPJ),(T2XJ,B),(T2YJ,
     *IND1),(T2ZJ,IND2)
      DATA   FPI/12.56637062D+0/, TP/6.283185308D+0/
      RX= XI- XJ
      RY= YI- YJ
      RFL=-1.
      EXK=(0.,0.)
      EYK=(0.,0.)
      EZK=(0.,0.)
      EXS=(0.,0.)
      EYS=(0.,0.)
      EZS=(0.,0.)
      DO 5  IP=1, KSYMP
      RFL=- RFL
      RZ= ZI- ZJ* RFL
      RSQ= RX* RX+ RY* RY+ RZ* RZ
      IF(RSQ.LT.1.D-20) GOTO 5
      R= SQRT( RSQ)
      RK= TP* R
      CR= COS( RK)
      SR= SIN( RK)
      GAM=-( CMPLX( CR,- SR)+ RK* CMPLX( SR, CR))/( FPI* RSQ* R)* S
      EXC= GAM* RX
      EYC= GAM* RY
      EZC= GAM* RZ
      T1ZR= T1ZJ* RFL
      T2ZR= T2ZJ* RFL
      F1X= EYC* T1ZR- EZC* T1YJ
      F1Y= EZC* T1XJ- EXC* T1ZR
      F1Z= EXC* T1YJ- EYC* T1XJ
      F2X= EYC* T2ZR- EZC* T2YJ
      F2Y= EZC* T2XJ- EXC* T2ZR
      F2Z= EXC* T2YJ- EYC* T2XJ
      IF(IP.EQ.1) GOTO 4
      IF(IPERF.NE.1) GOTO 1
      F1X=- F1X
      F1Y=- F1Y
      F1Z=- F1Z
      F2X=- F2X
      F2Y=- F2Y
```

```
      F2Z=- F2Z
      GOTO 4
    1 XYMAG= SQRT( RX* RX+ RY* RY)
      IF(XYMAG.GT.1.D-6) GOTO 2
      PX=0.
      PY=0.
      CTH=1.
      RRV=(1.,0.)
      GOTO 3
    2 PX=- RY/ XYMAG
      PY= RX/ XYMAG
      CTH= RZ/ R
      RRV= SQRT(1.- ZRATI* ZRATI*(1.- CTH* CTH))
    3 RRH= ZRATI* CTH
      RRH=( RRH- RRV)/( RRH+ RRV)
      RRV= ZRATI* RRV
      RRV=-( CTH- RRV)/( CTH+ RRV)
      GAM=( F1X* PX+ F1Y* PY)*( RRV- RRH)
      F1X= F1X* RRH+ GAM* PX
      F1Y= F1Y* RRH+ GAM* PY
      F1Z= F1Z* RRH
      GAM=( F2X* PX+ F2Y* PY)*( RRV- RRH)
      F2X= F2X* RRH+ GAM* PX
      F2Y= F2Y* RRH+ GAM* PY
      F2Z= F2Z* RRH
    4 EXK= EXK+ F1X
      EYK= EYK+ F1Y
      EZK= EZK+ F1Z
      EXS= EXS+ F2X
      EYS= EYS+ F2Y
      EZS= EZS+ F2Z
    5 CONTINUE
      RETURN
      END
```

HSFLD

PURPOSE

To compute the near magnetic field due to constant, sine, and cosine current distributions on a segment in free space or over ground.

METHOD

The magnetic field is computed at the point XI, YI, ZI due to the segment defined by parameters in COMMON/DATAJ/. The fields computed by routine HSFLX are stored in /DATAJ/. When a ground is present, the code is executed twice in a loop. In the second pass, the field of the image of the segment is computed, multiplied by the reflection coefficients, and added to the direct field.

The field is evaluated in a cylindrical coordinate system with the source segment at the origin. The radius of a segment on which the field is evaluated is treated in the same way as for the electric field in subroutine EFLD. When the field evaluation point is not on a segment, the observation segment radius is set to zero in the call to HSFLD. Thus, as for the electric field, the ρ coordinate of the field evaluation point is computed for the surface of the observation segment as $\rho' = (\rho^2 + a^2)^{1/2}$, where ρ is the distance from the axis of the source segment to (XI, YI, ZI) and a is the radius of the observation segment. The resulting H field is multiplied by ρ/ρ'.

SYMBOL DICTIONARY

AI	=	radius of observation segment, if any
CTH	=	cos θ, θ = angle between the ray reflected from the ground and vertical
ETA	=	$\eta = \sqrt{\mu/\epsilon}$
HPC HPK HPS	=	H_Φ due to cosine, constant, and sine current, respectively
PHX PHY PHZ	=	$(\rho/\rho')\hat{\Phi}$ in the cylindrical coordinates of the source segment or its image
PX PY	=	unit vector normal to the plane of incidence of the reflected ray, $\hat{p}$
QX QY QZ	=	$\rho/\rho'[R_H\hat{\Phi} + (R_V - R_H)(\hat{\Phi}\cdot\hat{p})\hat{p}] for reflected ray$
RFL	=	+1 for direct field, -1 for reflected field
RH	=	ρ'
RHOSPC	=	distance from coordinate origin to the point where the ray from the source to (XI,YI,ZI) reflects from the ground
RHOX RHOY RHOZ	=	$\vec{\rho}$ or $\vec{\rho}/\rho'$
RMAC	=	distance from the field evaluation point to the ceter of the source segment
RRH	=	R_H
RRV	=	R_V

```
SALPR  =  z component of unit vector in the direction of the source
          segment or its image
XI
YI     =  x, y, z coordinates of the field evaluation point
ZI
XIJ
YIJ    =  x, y, z components of distance from center of source
ZIJ       segment to field observation point
XSPEC  =  x coordinate of the ground plane reflection point
YSPEC  =  y coordinate of the ground plane reflection point
XYMAG  =  horizontal distance from the source segment to the
          field observation point
ZP     =  projection of the vector (XIJ,YIJ,ZIJ) on the axis of the
          source segment
ZRATX  =  temporary storage for ZRATI
```

```
      SUBROUTINE HSFLD(XI,YI,ZI,AI)
C     HSFLD COMPUTES THE H FIELD FOR CONSTANT, SINE, AND COSINE CURRENT
C     ON A SEGMENT INCLUDING GROUND EFFECTS.
      COMPLEX  EXK, EYK, EZK, EXS, EYS, EZS, EXC, EYC, EZC, ZRATI,
     *ZRATI2, T1, HPK, HPS, HPC, QX, QY, QZ, RRV, RRH, ZRATX, FRATI
      COMMON/DATAJ/ S, B, XJ, YJ, ZJ, CABJ, SABJ, SALPJ, EXK, EYK,
     *EZK, EXS, EYS, EZS, EXC, EYC, EZC, RKH, IEXK, IND1, INDD1, IND2,
     *INDD2, IPGND
      COMMON/GND/ ZRATI, ZRATI2, FRATI, CL, CH, SCRWL, SCRWR, NRADL,
     *KSYMP, IFAR, IPERF, T1, T2
      DATA  ETA/376.73/
      XIJ=XI- XJ
      YIJ=YI- YJ
      RFL=-1.
      DO 7 IP=1, KSYMP
      RFL=-RFL
      SALPR=SALPJ* RFL
      ZIJ= ZI-RFL* ZJ
      ZP= XIJ*CABJ+ YIJ* SABJ+ ZIJ* SALPR
      RHOX= XIJ-CABJ* ZP
      RHOY= YIJ-SABJ* ZP
      RHOZ= ZIJ-SALPR* ZP
      RH= SQRT( RHOX* RHOX+ RHOY* RHOY+ RHOZ* RHOZ+ AI* AI)
      IF(RH.GT.1.D-10) GOTO 1
      EXK=0.
      EYK=0.
      EZK=0.
      EXS=0.
      EYS=0.
      EZS=0.
      EXC=0.
      EYC=0.
      EZC=0.
      GOTO 7
    1 RHOX=RHOX/ RH
      RHOY=RHOY/ RH
      RHOZ=RHOZ/ RH
      PHX=SABJ* RHOZ- SALPR* RHOY
      PHY=SALPR* RHOX- CABJ* RHOZ
      PHZ=CABJ* RHOY- SABJ* RHOX
      CALL HSFLX(S,RH,ZP,HPK,HPS,HPC)
      IF(IP.NE.2) GOTO 6
      IF(IPERF.EQ.1) GOTO 5
      ZRATX= ZRATI
      RMAG= SQRT( ZP* ZP+ RH* RH)
C
C     SET PARAMETERS FOR RADIAL WIRE GROUND SCREEN.
C
      XYMAG= SQRT( XIJ* XIJ+ YIJ* YIJ)
```

```
      IF(NRADL.EQ.0) GOTO 2
      XSPEC=( XI* ZJ+ ZI* XJ)/( ZI+ ZJ)
      YSPEC=( YI* ZJ+ ZI* YJ)/( ZI+ ZJ)
      RHOSPC= SQRT( XSPEC* XSPEC+ YSPEC* YSPEC+ T2* T2)
      IF(RHOSPC.GT. SCRWL) GOTO 2
      RRV= T1* RHOSPC* LOG( RHOSPC/ T2)
      ZRATX=( RRV* ZRATI)/( ETA* ZRATI+ RRV)
C
C     CALCULATION OF REFLECTION COEFFICIENTS WHEN GROUND IS SPECIFIED.
C
    2 IF(XYMAG.GT.1.D-6) GOTO 3
      PX=0.
      PY=0.
      CTH=1.
      RRV=(1.,0.)
      GOTO 4
    3 PX=- YIJ/ XYMAG
      PY=XIJ/ XYMAG
      CTH=ZIJ/ RMAG
      RRV=SQRT(1.- ZRATX* ZRATX*(1.- CTH* CTH))
    4 RRH=ZRATX* CTH
      RRH=-( RRH- RRV)/( RRH+ RRV)
      RRV=ZRATX* RRV
      RRV=( CTH- RRV)/( CTH+ RRV)
      QY=( PHX* PX+ PHY* PY)*( RRV- RRH)
      QX=QY* PX+ PHX* RRH
      QY=QY* PY+ PHY* RRH
      QZ=PHZ* RRH
      EXK=EXK-HPK* QX
      EYK=EYK-HPK* QY
      EZK=EZK-HPK* QZ
      EXS=EXS-HPS* QX
      EYS=EYS-HPS* QY
      EZS=EZS-HPS* QZ
      EXC=EXC-HPC* QX
      EYC=EYC-HPC* QY
      EZC=EZC-HPC* QZ
      GOTO 7
    5 EXK=EXK-HPK* PHX
      EYK=EYK-HPK* PHY
      EZK=EZK-HPK* PHZ
      EXS=EXS-HPS* PHX
      EYS=EYS-HPS* PHY
      EZS=EZS-HPS* PHZ
      EXC=EXC-HPC* PHX
      EYC=EYC-HPC* PHY
      EZC=EZC-HPC* PHZ
      GOTO 7
    6 EXK=HPK* PHX
```

```
      EYK=HPK* PHY                                                      HS  99
      EZK=HPK* PHZ                                                      HS 100
      EXS=HPS* PHX                                                      HS 101
      EYS=HPS* PHY                                                      HS 102
      EZS=HPS* PHZ                                                      HS 103
      EXC=HPC* PHX                                                      HS 104
      EYC=HPC* PHY                                                      HS 105
      EZC=HPC* PHZ                                                      HS 106
    7 CONTINUE                                                          HS 107
      RETURN                                                            HS 108
      END                                                               HS 109
```

HSFLX

PURPOSE

To compute the near H field of filamentary currents of sine, cosine, and constant distribution on a segment.

METHOD

The wire segment is considered to be located at the origin of a local cylindrical coordinate system with the point at which the H field is computed being (ρ,Φ,z). The coordinate geometry for a filament of current of length Δ is shown in figure 7. For a sine or cosine current distribution, the field can be written in closed form. For a current

$$I_0\left[\begin{array}{c}\sin kz'\\ \cos kz'\end{array}\right],$$

the field is

$$H_\Phi(\rho,z)=\frac{-jI_0/\lambda}{2k\rho}\left\{\exp(-jkr_2)\left[\begin{array}{c}cos(k\Delta/2)\\ -\sin(k\Delta/2)\end{array}\right]-\exp(-jkr_2)\left[\begin{array}{c}cos(k\Delta/2)\\ \sin(k\Delta/2)\end{array}\right]\right.$$

$$-j(kz-k\Delta/2)\frac{\exp(-jkr_2)}{kr_2}\left[\begin{array}{c}\sin(k\Delta/2)\\ \cos(k\Delta/2)\end{array}\right]$$

$$\left.+j(kz+k\Delta/2)\frac{\exp(-jkr_1)}{kr_1}\left[\begin{array}{c}-\sin(k\Delta/2)\\ \cos(k\Delta/2)\end{array}\right]\right\}$$

$I_0/\lambda=1$ is assumed in this routine.

For small values of ρ with |z| > Δ/2, this equation may produce large numerical errors due to cancellation of large terms. Hence, for z > 0 and $\rho/(z-\Delta/2)<10^{-3}$, a more stable approximation for small $\rho/(z\pm\Delta/2)$ is used:

$$H_\Phi=\frac{(\rho/\lambda)(I_0/\lambda)}{8\pi}\exp(-jkz)\left\{\left[\frac{2\pi}{(z+\Delta/2)/\lambda}-\frac{2\pi}{(z-\Delta/2)/\lambda}\left[\begin{array}{c}1\\ -j\end{array}\right]\right.\right.$$

$$\left.\left.+\left[\frac{\exp(jk\Delta/2}{(z-\Delta/2)^2/\lambda^2}\left(\begin{array}{c}\sin(k\Delta/2)\\ \cos(k\Delta/2)\end{array}\right)-\frac{\exp(-jk\Delta/2}{(z+\Delta/2)^2/\lambda^2}\left(\begin{array}{c}-\sin(k\Delta/2)\\ \cos(k\Delta/2)\end{array}\right)\right]\right]\right\}$$

For z<0, the above equation is evaluated for $H_\Phi(\rho,-z)$. The field of a sin kz' current is multiplied by -1 in this case, since it is an odd function of z.

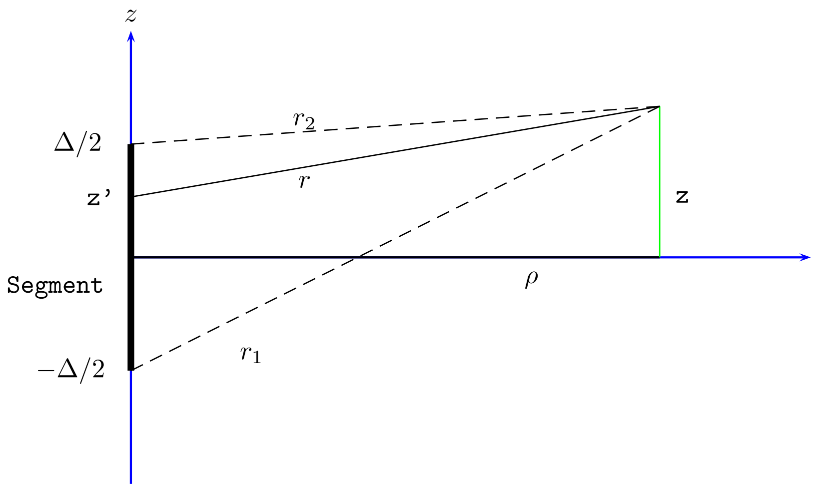


Figure 7. Coordinates for Evaluating H Field of a Segment.

The field due to a constant current is obtained by numerical integration, which is performed by subroutine HFK. If ρ is zero, all field quantities are set to zero, since H_Φ is undefined.

SYMBOL DICTIONARY

CDK	=	cos(kΔ/2)
CONS	=	-j/(2kρ)
DH	=	$\Delta/2$
DK	=	kΔ/2
EKR1	=	exp(-jkr$_1$)
EKR2	=	exp(-jkr$_2$)
FJ	=	j
FJK	=	-j2π
HKR,HKI	=	real and imaginary parts of H due to a constant current
HPC HPK HPS	=	H_Φ due to cosine, constant, and sine currents, respectively
HSS	=	sign of z
PI8	=	8π
R1	=	r$_1$
R2	=	r$_2$
RH	=	ρ
RH2	=	ρ^2
RHZ	=	$\rho/(z-\Delta/2)$
S	=	Δ
SDK	=	sin(kΔ/2)
TP	=	2π
Z1	=	z + Δ/2
Z2	=	z - Δ/2
ZP	=	z

```
      SUBROUTINE HSFLX( S, RH, ZPX, HPK, HPS, HPC)
C     CALCULATES H FIELD OF SINE COSINE, AND CONSTANT CURRENT OF SEGMENT
      COMPLEX  FJ, FJK, EKR1, EKR2, T1, T2, CONS, HPS, HPC, HPK
      DIMENSION  FJX(2), FJKX(2)
      EQUIVALENCE(FJ,FJX),(FJK,FJKX)
      DATA   TP/6.283185308D+0/, FJX/0.,1./, FJKX/0.,-6.283185308D+0/
      DATA   PI8/25.13274123D+0/
      IF(RH.LT.1.D-10) GOTO 6
      IF(ZPX.LT.0.) GOTO 1
      ZP= ZPX
      HSS=1.
      GOTO 2
    1 ZP=- ZPX
      HSS=-1.
    2 DH=.5* S
      Z1= ZP+ DH
      Z2= ZP- DH
      IF(Z2.LT.1.D-7) GOTO 3
      RHZ= RH/ Z2
      GOTO 4
    3 RHZ=1.
    4 DK= TP* DH
      CDK= COS( DK)
      SDK= SIN( DK)
      CALL HFK(- DK, DK, RH* TP, ZP* TP, HKR, HKI)
      HPK= CMPLX( HKR, HKI)
      IF(RHZ.LT.1.D-3) GOTO 5
      RH2= RH* RH
      R1= SQRT( RH2+ Z1* Z1)
      R2= SQRT( RH2+ Z2* Z2)
      EKR1= EXP( FJK* R1)
      EKR2= EXP( FJK* R2)
      T1= Z1* EKR1/ R1
      T2= Z2* EKR2/ R2
      HPS=( CDK*( EKR2- EKR1)- FJ* SDK*( T2+ T1))* HSS
      HPC=- SDK*( EKR2+ EKR1)- FJ* CDK*( T2- T1)
      CONS=- FJ/(2.* TP* RH)
      HPS= CONS* HPS
      HPC= CONS* HPC
      RETURN
    5 EKR1= CMPLX( CDK, SDK)/( Z2* Z2)
      EKR2= CMPLX( CDK,- SDK)/( Z1* Z1)
      T1= TP*(1./ Z1-1./ Z2)
      T2= EXP( FJK* ZP)* RH/ PI8
      HPS= T2*( T1+( EKR1+ EKR2)* SDK)* HSS
      HPC= T2*(- FJ* T1+( EKR1- EKR2)* CDK)
      RETURN
    6 HPS=(0.,0.)
      HPC=(0.,0.)
```

```
      HPK=(0.,0.)
      RETURN
      END
```

INTRP

PURPOSE

To evaluate the Sommerfeld integral contributions to the field of a source over ground by interpolation in precomputed tables.

METHOD

The interpolation region in R_1 and θ is covered by three grids as shown in Figure 12 of Part I. The interpolation tables and the number of data points and the boundaries of each grid are read from file 21 and stored in COMMON/GGRID/ by the main program. In subroutine INTRP the variable x corresponds to R_1 and y to θ.

The three interpolation tables are stored in the arrays AR1, AR2 and AR3 in COMMON/GGRID/. For grid i, ARi(I,J,K) is the value at

$$x_I = s_i + (I-1)\Delta x_i, \qquad I = 1, ..., N_i$$

$$y_J = t_i + (J-1)\Delta y_i, \qquad J = 1, ..., M_i$$

where

$$s_i = XSA(i), \Delta x_i = DXA(i), N_i = NXA(i)$$

$$t_i = YSA(i), \Delta y_i = DYA(i), M_i = NYA(i)$$

Each array contains values for I_ρ^V, I_z^H, I_ρ^H and I_Φ^H from equations 156 through 159 of Part I for K equal to 1 through 4, respectively. The grid boundaries and density of points can be varied but the relative positions of the three grids must be as shown in Figure 12 of Part I for the logic for choosing the correct grid to work correctly. In particular, XSA(l), YSA(I) and YSA(2) must be zero; and XSA(2) and XSA(3) must be equal.

For a given x and y the values of I_ρ^V, I_z^H, I_ρ^H and I_Φ^H are found by bivariate cubic interpolation and returned in the variables Fl, F2, F3 and F4. The grid containing (x,y) is determined and a four by four point region containing (x,y) is selected. If x_i and y_k are the minimum values of x and y in the four by four point region then four interpolation polynomials in x are computed for y = y_j with j = k, k+1, k+2, k+3. These are

$$f_{ij}(x) = a_{ij}\xi^3 + b_{ij}\xi^2 + c_{ij}\xi + d_{ij}$$

where $\xi_i = (x - x_{i+1})/\Delta x$

$$a_{ij} = \frac{1}{6}[F_{i+3,j} - F_{i,j} + 3(F_{i+1,j} - F_{i+2,j}]$$

$$b_{ij} = \frac{1}{2}[F_{i,j} - 2F_{i+1,j} + F_{i+2,j}]$$

$$c_{ij} = F_{i+2,j} - \frac{1}{6}[2F_{i,j} + 3F_{i+1,j} + F_{i+3,j}]$$

$$d_{ij} = F_{i+1,j}$$

$$F_{i,j} = F(x_i, y_j)$$

A cubic polynomial in y, fit to the points f_{ij}(x) for j = k, ..., k + 3 is then evaluated for the given y to obtain the interpolated value $\hat{F}$(x,y)

$$
\begin{aligned}
\hat{F}(x,y) &= \tfrac{1}{6}(p_1\eta^3 + p_2\eta_k^2 + p_3\eta_k) + p_4 \\
\eta_k &= (y - y_{k+1})/\Delta y \\
p_1 &= f_{i,k+3}(x) - f_{ik}(x) + 3[f_{i,k+1}(x) - f_{i,k+2}(x)] \\
p_2 &= 3[f_{i,k}(x) - 2f_{i,k+1}(x) + f_{i,k+2}(x)] \\
p_3 &= 6f_{i,k+2}(x) - 2f_{i,k}(x) - 3f_{i,k+1}(x) - f_{i,k+3}(x) \\
p_4 &= f_{i,k+1}
\end{aligned}
$$

To reduce computation time the coefficients a_{ij}, b_{ij}, c_{ij} and d_{ij} are saved as long as successive points (x,y) fall in the same four by four point region of a grid. In addition the four by four point interpolation regions are restricted to starting indices i and k with values 3n+1, n=0, 1 .... Thus the regions do not overlap. This is less accurate than centering the region on each x,y point but requires less frequent computation of the coefficients. At the outer edges of a grid the regions are chosen to extend to the edge but not beyond. If x,y is out of the entire three grid region the nearest four by four point region in used for extrapolation.

The coefficients a_{ij}, b_{ij}, c_{ij} and d_{ij} are stored in two dimensional arrays from IT 106 to IT 109. When they are used, from IT 118 to IT 149 they ar3 used as simple variables (A(1,1) $\equiv$ All) to save time. Also the three dimensional arrays AR1, AR2, and AR3 are used as linear arrays from IT 92 an IT 105. The equivalent three subscripts are shown in the comment at IT 91.

SYMBOL DICTIONARY

A_{ij}	=	A(i,j) = a_{ij}
AR1	=	ARL1 = grid 1
AR2	=	ARL2 = grid 2
AR3	=	ARL3 = grid 3
B_{ij}	=	B(i,j) = b_{ij}
C_{ij}	=	C(i,j) = c_{ij}
D_{ij}	=	D(i,j) = d_{ij}
DX	=	Δx for grid being used
DXA	=	array of Δx values for the three grids
DY	=	Δy for grid being used
DYA	=	array of Δy values
EPSCF	=	$\epsilon_1 - j\sigma/\omega\epsilon_0$
Fl	=	I_ρ^V
F2	=	I_z^V
F3	=	I_ρ^H
F4	=	I_Φ^H
FX1	=	$f_{i,j}$(x)
FX2	=	$f_{i,j+1}$(x)
FX3	=	$f_{i,j+2}$(x)
FX4	=	$f_{i,j+3}$(x)
IADD	=	index for linear arrays ARL1, etc.
IADZ	=	initial value far IADD
IGR	=	grid number for present x,y
IGRS	=	grid number for last x,y

IX	=	x index of the grid coordinate just less than x
IXEG	=	x index of the upper edge of the last normally located interpolation patch when a patch out of the normal locations is sued at the outer edge of a grid, -10000 otherwise
IXS	=	1 plus the x index of the lower edge of 4 by 4 point interpolation patch
IY,IYEG,IYS	=	same for y as IX, IXEG and IXS
K	=	1, 2, 3, 4 for I_ρ^V, I_z^V, I_ρ^H, I_Φ^H
ND	=	NDA for the particular grid
NDA	=	array containing the first dimensions of AR1, AR2 and AR3
NDP	=	NDPA for a particular grid
NDPA	=	array containing the product of the first two dimensions in AR1, AR2 and AR3
NXA	=	number of x values in each grid
NXM2	=	NXA-2 for a particular grid
NXMS	=	upper x index of the last normally located patch at the edge of a grid
NYA,NYM2,NYMS	=	same for y as NXA, NXM2 and NXMS
Pl,P2,P3,P4	=	p_1, p_2, P_3, p_4
X	=	x
XS	=	XSA for the present grid
XS2	=	XSA(2) through equivalence
XSA	=	array of values of x at lower edge of each grid (s_i)
XX	=	ξ_i
XZ	=	x_{i+1} for computing ξ_i
Y	=	y
YS	=	YSA for present grid
YS3	=	YSA(3) through equivalence
YSA	=	array af values of y at lower edge of each grid (t_i)
YY	=	η_k
YZ	=	y_{k+1} for computing η_k

```
      SUBROUTINE INTRP( X, Y, F1, F2, F3, F4)
C
C     INTRP USES BIVARIATE CUBIC INTERPOLATION TO OBTAIN THE VALUES OF
C     4 FUNCTIONS AT THE POINT (X,Y).
C
      COMPLEX  F1, F2, F3, F4, A, B, C, D, FX1, FX2, FX3, FX4, P1,
     *P2, P3, P4, A11, A12, A13, A14, A21, A22, A23, A24, A31, A32, A33
     *, A34, A41, A42, A43, A44, B11, B12, B13, B14, B21, B22, B23, B24
     *, B31, B32, B33, B34, B41, B42, B43, B44, C11, C12, C13, C14, C21
     *, C22, C23, C24, C31, C32, C33, C34, C41, C42, C43, C44, D11, D12
     *, D13, D14, D21, D22, D23, D24, D31, D32, D33, D34, D41, D42, D43
     *, D44
      COMPLEX  AR1, AR2, AR3, ARL1, ARL2, ARL3, EPSCF
      COMMON  /GGRID/ AR1(11,10,4), AR2(17,5,4), AR3(9,8,4), EPSCF, DXA
     *(3), DYA(3), XSA(3), YSA(3), NXA(3), NYA(3)
      DIMENSION  NDA(3), NDPA(3)
      DIMENSION  A(4,4), B(4,4), C(4,4), D(4,4), ARL1(1), ARL2(1), ARL3
     *(1)
      EQUIVALENCE(A(1,1),A11),(A(1,2),A12),(A(1,3),A13),(A(1,4),A14)
      EQUIVALENCE(A(2,1),A21),(A(2,2),A22),(A(2,3),A23),(A(2,4),A24)
      EQUIVALENCE(A(3,1),A31),(A(3,2),A32),(A(3,3),A33),(A(3,4),A34)
      EQUIVALENCE(A(4,1),A41),(A(4,2),A42),(A(4,3),A43),(A(4,4),A44)
      EQUIVALENCE(B(1,1),B11),(B(1,2),B12),(B(1,3),B13),(B(1,4),B14)
      EQUIVALENCE(B(2,1),B21),(B(2,2),B22),(B(2,3),B23),(B(2,4),B24)
      EQUIVALENCE(B(3,1),B31),(B(3,2),B32),(B(3,3),B33),(B(3,4),B34)
      EQUIVALENCE(B(4,1),B41),(B(4,2),B42),(B(4,3),B43),(B(4,4),B44)
      EQUIVALENCE(C(1,1),C11),(C(1,2),C12),(C(1,3),C13),(C(1,4),C14)
      EQUIVALENCE(C(2,1),C21),(C(2,2),C22),(C(2,3),C23),(C(2,4),C24)
      EQUIVALENCE(C(3,1),C31),(C(3,2),C32),(C(3,3),C33),(C(3,4),C34)
      EQUIVALENCE(C(4,1),C41),(C(4,2),C42),(C(4,3),C43),(C(4,4),C44)
      EQUIVALENCE(D(1,1),D11),(D(1,2),D12),(D(1,3),D13),(D(1,4),D14)
      EQUIVALENCE(D(2,1),D21),(D(2,2),D22),(D(2,3),D23),(D(2,4),D24)
      EQUIVALENCE(D(3,1),D31),(D(3,2),D32),(D(3,3),D33),(D(3,4),D34)
      EQUIVALENCE(D(4,1),D41),(D(4,2),D42),(D(4,3),D43),(D(4,4),D44)
      EQUIVALENCE(ARL1,AR1),(ARL2,AR2),(ARL3,AR3),(XS2,XSA(2)),(YS3,YSA
     *(3))
      DATA   IXS, IYS, IGRS/-10,-10,-10/, DX, DY, XS, YS/1.,1.,0.,0./
      DATA   NDA/11,17,9/, NDPA/110,85,72/, IXEG, IYEG/0,0/
      IF(X.LT. XS.OR. Y.LT. YS) GOTO 1
      IX= INT(( X- XS)/ DX)+1
C
C     IF POINT LIES IN SAME 4 BY 4 POINT REGION AS PREVIOUS POINT, OLD
C     VALUES ARE REUSED
C
      IY= INT(( Y- YS)/ DY)+1
      IF(IX.LT. IXEG.OR. IY.LT. IYEG) GOTO 1
C
C     DETERMINE CORRECT GRID AND GRID REGION
C
```

```
      IF(IABS( IX- IXS).LT.2.AND. IABS( IY- IYS).LT.2) GOTO 12
    1 IF(X.GT. XS2) GOTO 2
      IGR=1
      GOTO 3
    2 IGR=2
      IF(Y.GT. YS3) IGR=3
    3 IF(IGR.EQ. IGRS) GOTO 4
      IGRS= IGR
      DX= DXA( IGRS)
      DY= DYA( IGRS)
      XS= XSA( IGRS)
      YS= YSA( IGRS)
      NXM2= NXA( IGRS)-2
      NYM2= NYA( IGRS)-2
      NXMS=(( NXM2+1)/3)*3+1
      NYMS=(( NYM2+1)/3)*3+1
      ND= NDA( IGRS)
      NDP= NDPA( IGRS)
      IX= INT(( X- XS)/ DX)+1
      IY= INT(( Y- YS)/ DY)+1
    4 IXS=(( IX-1)/3)*3+2
      IF(IXS.LT.2) IXS=2
      IXEG=-10000
      IF(IXS.LE. NXM2) GOTO 5
      IXS= NXM2
      IXEG= NXMS
    5 IYS=(( IY-1)/3)*3+2
      IF(IYS.LT.2) IYS=2
      IYEG=-10000
      IF(IYS.LE. NYM2) GOTO 6
      IYS= NYM2
C
C     COMPUTE COEFFICIENTS OF 4 CUBIC POLYNOMIALS IN X FOR THE 4 GRID
C     VALUES OF Y FOR EACH OF THE 4 FUNCTIONS
C
      IYEG= NYMS
    6 IADZ= IXS+( IYS-3)* ND- NDP
      DO 11  K=1,4
      IADZ= IADZ+ NDP
      IADD= IADZ
      DO 11  I=1,4
      IADD= IADD+ ND
C     P1=AR1(IXS-1,IYS-2+I,K)
      GOTO (7,8,9), IGRS
    7 P1= ARL1( IADD-1)
      P2= ARL1( IADD)
      P3= ARL1( IADD+1)
      P4= ARL1( IADD+2)
      GOTO 10
```

```
    8 P1= ARL2( IADD-1)
      P2= ARL2( IADD)
      P3= ARL2( IADD+1)
      P4= ARL2( IADD+2)
      GOTO 10
    9 P1= ARL3( IADD-1)
      P2= ARL3( IADD)
      P3= ARL3( IADD+1)
      P4= ARL3( IADD+2)
   10 A( I, K)=( P4- P1+3.*( P2- P3))*.1666666667D+0
      B( I, K)=( P1-2.* P2+ P3)*.5
      C( I, K)= P3-(2.* P1+3.* P2+ P4)*.1666666667D+0
   11 D( I, K)= P2
      XZ=( IXS-1)* DX+ XS
C
C     EVALUATE POLYMOMIALS IN X AND THEN USE CUBIC INTERPOLATION IN Y
C     FOR EACH OF THE 4 FUNCTIONS.
C
      YZ=( IYS-1)* DY+ YS
   12 XX=( X- XZ)/ DX
      YY=( Y- YZ)/ DY
      FX1=(( A11* XX+ B11)* XX+ C11)* XX+ D11
      FX2=(( A21* XX+ B21)* XX+ C21)* XX+ D21
      FX3=(( A31* XX+ B31)* XX+ C31)* XX+ D31
      FX4=(( A41* XX+ B41)* XX+ C41)* XX+ D41
      P1= FX4- FX1+3.*( FX2- FX3)
      P2=3.*( FX1-2.* FX2+ FX3)
      P3=6.* FX3-2.* FX1-3.* FX2- FX4
      F1=(( P1* YY+ P2)* YY+ P3)* YY*.1666666667D+0+ FX2
      FX1=(( A12* XX+ B12)* XX+ C12)* XX+ D12
      FX2=(( A22* XX+ B22)* XX+ C22)* XX+ D22
      FX3=(( A32* XX+ B32)* XX+ C32)* XX+ D32
      FX4=(( A42* XX+ B42)* XX+ C42)* XX+ D42
      P1= FX4- FX1+3.*( FX2- FX3)
      P2=3.*( FX1-2.* FX2+ FX3)
      P3=6.* FX3-2.* FX1-3.* FX2- FX4
      F2=(( P1* YY+ P2)* YY+ P3)* YY*.1666666667D+0+ FX2
      FX1=(( A13* XX+ B13)* XX+ C13)* XX+ D13
      FX2=(( A23* XX+ B23)* XX+ C23)* XX+ D23
      FX3=(( A33* XX+ B33)* XX+ C33)* XX+ D33
      FX4=(( A43* XX+ B43)* XX+ C43)* XX+ D43
      P1= FX4- FX1+3.*( FX2- FX3)
      P2=3.*( FX1-2.* FX2+ FX3)
      P3=6.* FX3-2.* FX1-3.* FX2- FX4
      F3=(( P1* YY+ P2)* YY+ P3)* YY*.1666666667D+0+ FX2
      FX1=(( A14* XX+ B14)* XX+ C14)* XX+ D14
      FX2=(( A24* XX+ B24)* XX+ C24)* XX+ D24
      FX3=(( A34* XX+ B34)* XX+ C34)* XX+ D34
      FX4=(( A44* XX+ B44)* XX+ C44)* XX+ D44
```

```
P1= FX4- FX1+3.*( FX2- FX3)                                             IT 148
P2=3.*( FX1-2.* FX2+ FX3)                                               IT 149
P3=6.* FX3-2.* FX1-3.* FX2- FX4                                         IT 150
F4=(( P1* YY+ P2)* YY+ P3)* YY*.1666666667D+0+ FX2                      IT 151
RETURN                                                                  IT 152
END                                                                     IT 153
```

INTX

PURPOSE

To numerically compute the integral of the function exp(jkr)/kr.

METHOD

For evaluation of the field due to a segment, a local cylindrical coordinate system is defined with origin at the center of the segment and z-axis in the segment direction. This geometry is illustrated in the discussion of subroutine GF. Subroutine INTX is called by subroutine EFLD to evaluate the integral

$$G = \int_{-k\Delta/2}^{k\Delta/2} \frac{exp(-jkr)}{kr} d(kz),$$

where

$$r = [\rho'^{\,2} + (z - z')^2]^{1/2},$$

and other symbols are defined in the discussion of subroutine GF.

The numerical integration technique of Romberg integration with variable interval width is used (refs. 3 and 4). The Romberg integration formula is obtained from the trapezoidal formula by an iterative procedure (ref. 1). The trapezoidal rule for integration of the function f(x) over an interval (a, b) using 2 subintervals is

$$T_{0k} = [(b - a)/N][(1/2)f_0 + f_1 + ... + f_{N-1} + (1/2)f_N],$$

where

N = 2^k

f_i = f(x_i)

x_i = a + i(b - a)/N

These trapezoidal rule answers are then used in the iterative formula

$$T_{m,n} = \left(4^m T_{m-1,n+1} - T_{m-1,n}\right)/(4^m - 1).$$

The results $T_{m,n}$ may be arranged in a triangular matrix of the form

$$\begin{array}{lll} T_{0,0} & & \\ T_{0,0} & T_{0,0} & \\ T_{0,0} & T_{0,0} & T_{0,0} \\ \vdots & \vdots & \vdots \end{array}$$

where the elements in the first column, T_{0k}, represent the trapezoidal rule results, and the elements in the diagonal, T_{k0}, are the Romberg integration results for 2^k subintervals.

Convergence to increasingly more accurate answers takes place down the first column and the diagonal, as well as towards the right along the rows. The row convergence generally provides a more realistic indication of error magnitude than two successive trapezoidal-rule or Romberg answers.

This convergence along the rows is used to determine the interval width in the variable interval-width scheme. The complete integration interval is first divided into a minimum number of subintervals (presently set to 1) and T_{00}, T_{01}, and T_{10} are computed on the first subinterval. The relative difference of T_{01} and T_{10} is then computed, and if less than the error criterion, R_x, T_{10} is accepted as the integral over that interval, and integration proceeds to the next interval. If the difference of T_{01} and T_{10} is too great, T_{02}, T_{11} nd T_{20} are computed. The relative difference of T_{11} and T_{20} is then computed, and if less than R_x, T_{20} is accepted as the integral over the subinterval. If the difference of T_{11} and T_{20} is too great, the subinterval is divided in half and the process repeated starting with T_{00} for the left hand, new subinterval. The subinterval is repeatedly halved until convergence to less than R_x is found. The process is repeated for successive subintervals until the right-hand side of the integration interval is reached. When convergence has been obtained with a given subinterval size for a few times, the routine attempts doubling the subinterval size to maintain the largest subinterval size that will give the required accuracy. Thus, the routine will use many points in a rapidly changing region of a function and fewer points where the function is smoothly varying.

Since the function to be integrated is complex, the convergence of both real and imaginary parts is tested and both must be less than R_x. The same subinterval sizes are used for real and imaginary parts.

When the field of a segment is being computed at the segment's own center, the length r becomes

$$r = [b^2 + (z - z')^2]^{1/2},$$

where b is the wire radius. For small values of b, the real part of the integrand is sharply peaked and, hence, difficult to integrate numerically. Hence, the integral is divided into the components

$$G' = \int_{-k\Delta/2}^{k\Delta/2} \frac{\exp(-jkr)-1}{kr} d(kz)$$

$$G'' = \int_{-k\Delta/2}^{k\Delta/2} \frac{1}{kr} d(kz)$$

$$G = G' + G''$$

G' must be computed numerically; however, the integrand is no longer peaked. G'', which contains the sharp peak, can be computed as

$$G'' = 2\log\left(\frac{\sqrt{b^2+\Delta^2}=\Delta}{b}\right)$$

To further reduce integration time for the self term, the integral of G' is computed from -kΔ/2 to 0, and the result doubled to obtain G'.

SYMBOL DICTIONARY

ABS	=	external routine (absolute value)
ALOG	=	external routine (natural log)
B	=	wire radius, b/λ
DZ	=	subinterval size on which T_{00}, T_{01}, ... are computed
DZOT	=	0.5 DZ
EL1	=	-kΔ/2
EL2	=	kΔ/2
EP	=	tolerance for ending the integration interval
FNM	=	real number equivalent of NM
FNS	=	real number equivalent of NS
GF	=	external routine (integrand)
G1I	=	imaginary part of f_1
G1R	=	real part of f_1
G2I	=	imaginary part of f_2
G2R	=	real part of f_2
G3I	=	imaginary part of f_3
G3R	=	real part of f_3
G4I	=	imaginary part of f_4
G4R	=	real part of f_4
G5I	=	imaginary part of f_5
G5R	=	real part of f_5
IJ	=	indication af self term integration when equal to zero
NM	=	minimum allowed subinterval size is kΔ/NM
NS	=	present subinterval size is kΔ/NS
NT	=	counter to control increasing of subinterval size
NTS	=	larger values retard increasing of subinterval size
NX	=	maximum allowed subinterval size is kΔ/NX
RX	=	R_x
S	=	Δ/λ
SGI	=	imaginary part of G
SGR	=	real part of G
SQRT	=	external routine (square root)
TEST	=	external routine (computes relative convergence)
TE1I	=	relative difference of T_{01} and T_{10} for imaginary part
TE1R	=	relative difference of T_{01} and T_{10} for real part
TE2I	=	relative difference of T_{11} and T_{20} for imaginary part
TE2R	=	relative difference of T_{11} and T_{20} for real part
T00I	=	imaginary part T_{00}
T00R	=	real part T_{00}
T01I	=	imaginary part T_{01}.
T01R	=	real part T_{01}
T02I	=	imaginary part T_{02}
T02R	=	real part T_{02}
T10I	=	imaginary part T_{10}
T10R	=	real part of T_{10}

T11I	=	imaginary part of T_{11}
T11R	=	real part of T_{11}
T20I	=	imaginary part of T_{20}
T20R	=	real part of T_{20}
Z	=	integration variable at left-hand side of subinterval
ZE	=	kΔ/2
ZEND	=	kΔ/2 - EP; EP = tolerance term
ZP	=	integration variable
65536	=	2^{16} = limit of minimum subinterval size (NM)
1.E-4	=	error criterion, R_x

```
      SUBROUTINE INTX( EL1, EL2, B, IJ, SGR, SGI)
C
C     INTX PERFORMS NUMERICAL INTEGRATION OF EXP(JKR)/R BY THE METHOD OF
C     VARIABLE INTERVAL WIDTH ROMBERG INTEGRATION.  THE INTEGRAND VALUE
C     IS SUPPLIED BY SUBROUTINE GF.
C
      DATA   NX, NM, NTS, RX/1,65536,4,1.D-4/
      Z= EL1
      ZE= EL2
      IF(IJ.EQ.0) ZE=0.
      S= ZE- Z
      FNM= NM
      EP= S/(10.* FNM)
      ZEND= ZE- EP
      SGR=0.
      SGI=0.
      NS= NX
      NT=0
      CALL GF( Z, G1R, G1I)
    1 FNS= NS
      DZ= S/ FNS
      ZP= Z+ DZ
      IF(ZP- ZE) 3,3,2
    2 DZ= ZE- Z
      IF(ABS( DZ)- EP) 17,17,3
    3 DZOT= DZ*.5
      ZP= Z+ DZOT
      CALL GF( ZP, G3R, G3I)
      ZP= Z+ DZ
      CALL GF( ZP, G5R, G5I)
    4 T00R=( G1R+ G5R)* DZOT
      T00I=( G1I+ G5I)* DZOT
      T01R=( T00R+ DZ* G3R)*0.5
      T01I=( T00I+ DZ* G3I)*0.5
      T10R=(4.0* T01R- T00R)/3.0
C
C     TEST CONVERGENCE OF 3 POINT ROMBERG RESULT.
C
      T10I=(4.0* T01I- T00I)/3.0
      CALL TEST( T01R, T10R, TE1R, T01I, T10I, TE1I,0.)
      IF(TE1I- RX) 5,5,6
    5 IF(TE1R- RX) 8,8,6
    6 ZP= Z+ DZ*0.25
      CALL GF( ZP, G2R, G2I)
      ZP= Z+ DZ*0.75
      CALL GF( ZP, G4R, G4I)
      T02R=( T01R+ DZOT*( G2R+ G4R))*0.5
      T02I=( T01I+ DZOT*( G2I+ G4I))*0.5
      T11R=(4.0* T02R- T01R)/3.0
```

```
      T11I=(4.0* T02I- T01I)/3.0
      T20R=(16.0* T11R- T10R)/15.0
C
C     TEST CONVERGENCE OF 5 POINT ROMBERG RESULT.
C
      T20I=(16.0* T11I- T10I)/15.0
      CALL TEST( T11R, T20R, TE2R, T11I, T20I, TE2I,0.)
      IF(TE2I- RX) 7,7,14
    7 IF(TE2R- RX) 9,9,14
    8 SGR= SGR+ T10R
      SGI= SGI+ T10I
      NT= NT+2
      GOTO 10
    9 SGR= SGR+ T20R
      SGI= SGI+ T20I
      NT= NT+1
   10 Z= Z+ DZ
      IF(Z- ZEND) 11,17,17
   11 G1R= G5R
      G1I= G5I
      IF(NT- NTS) 1,12,12
C
C     DOUBLE STEP SIZE
C
   12 IF(NS- NX) 1,1,13
   13 NS= NS/2
      NT=1
      GOTO 1
   14 NT=0
      IF(NS- NM) 16,15,15
   15 WRITE (2,20)  Z
C
C     HALVE STEP SIZE
C
      GOTO 9
   16 NS= NS*2
      FNS= NS
      DZ= S/ FNS
      DZOT= DZ*0.5
      G5R= G3R
      G5I= G3I
      G3R= G2R
      G3I= G2I
      GOTO 4
   17 CONTINUE
C
C     ADD CONTRIBUTION OF NEAR SINGULARITY FOR DIAGONAL TERM
C
      IF(IJ) 19,18,19
```

```
   18 SGR=2.*( SGR+ LOG(( SQRT( B* B+ S* S)+ S)/ B))
      SGI=2.* SGI
   19 CONTINUE
C
      RETURN
   20 FORMAT(' STEP SIZE LIMITED AT Z=',F10.5)
      END
```

ISEGNO

PURPOSE

To determine the segment number of the m-th segment ordered by increasing segment numbers in the set of segments with tag numbers equal to the given tag number. With a given tag of zero, segment number m is returned.

METHOD

Search segments consecutively and check their tag numbers against a given tag.

SYMBOL DICTIONARY

I	=	DO loop index
ICNT	=	counter
ITAG1	=	input tag number (given tag)
M	=	input quantity specifying the position in the set of segments with the given tag

```
      FUNCTION ISEGNO( ITAGI, MX)
C
C     ISEGNO RETURNS THE SEGMENT NUMBER OF THE MTH SEGMENT HAVING THE
C     TAG NUMBER ITAGI.  IF ITAGI=0 SEGMENT NUMBER M IS RETURNED.
C
      COMMON  /DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X( NM), Y( NM),
     *Z( NM), SI( NM), BI( NM), ALP( NM), BET( NM), ICON1( N2M), ICON2(
     * N2M), ITAG( N2M), ICONX( NM), WLAM, IPSYM
      IF(MX.GT.0) GOTO 1
      WRITE (2,6)
      STOP
    1 ICNT=0
      IF(ITAGI.NE.0) GOTO 2
      ISEGNO= MX
      RETURN
    2 IF(N.LT.1) GOTO 4
      DO 3  I=1, N
      IF(ITAG( I).NE. ITAGI) GOTO 3
      ICNT= ICNT+1
      IF(ICNT.EQ. MX) GOTO 5
    3 CONTINUE
    4 WRITE (2,7)  ITAGI
      STOP
    5 ISEGNO= I
C
      RETURN
    6 FORMAT(4X,'CHECK DATA, PARAMETER SPECIFYING SEGMENT POSITION IN',
     *' A GROUP OF EQUAL TAGS MUST NOT BE ZERO')
    7 FORMAT(///,10X,'NO SEGMENT HAS AN ITAG OF ',I5)
      END
```

LFACTR

PURPOSE

To perform the Gauss-Doolittle factorization calculations on two blocks of the matrix in core storage. This routine in conjunction with FACIO factors a matrix that is too large for core storage into an upper and lower triangular matrix using the Gauss-Doolittle technique. The factored matrix is used by LUNSCR and LTSOLV to determine the solution of the transposed matrix equation $x^T A^T = B^T$.

METHOD

The basic algorithm used in this routine is presented by Ralston in ref. 1 on pages 411-416. A brief discussion is also given under FACTR in this manual. The main difference between LFACTR and FACTR is that LFACTR is set up to perform the calculations on two blocks of columns of the transposed matrix that reside in core storage. This situation arises when the matrix is too large to fit in core at one time; thus, the matrix is divided into blocks of columns and stored on files. This matrix is then factored into a lower triangular matrix and an upper triangular matrix by the subroutines FACIO and LFACTR. The function of these two subroutines is closely tied together: LFACTR performs the mathematical computations involved in the factorization, while FACIO controls the input and output of matrix blocks in core storage, and, thus, controls the necessary block ordering input to LFACTR. For clarification of the ordering of matrix blocks during factorization, refer to FACIO.

The computations performed in LFACTR are slightly different for three matrix block conditions: (1) block numbers 1 and 2, (2) adjacent matrix blocks, and (3) non-adjacent matrix blocks. If the blocks are numbers 1 and 2, both blocks are factored, and the computations proceed exactly as in FACTR. The only difference between LFACTR and FACTR here is that the two blocks do not form a square matrix, and the row and column indices in LFACTR have not been interchanged as in FACTR. At the end of this stage, both blocks 1 and 2 are completely factored. For case 2, where the blacks are adjacent in the matrix and other than 1 and 2, the first block is assumed factored and is used to complete the factorization of the partially factored second block. The computations start with the first column of the second block and proceed as in FACTR (with the exceptions noted above). If the blocks are not adjacent (case 3), the first block is assumed factored and is used to partially factor the second black. Computations start with the first column of the second block. Factorization cannot be completed, since values from the intervening columns are necessary.

CODING

LF20-LF39	Initialization of loop parameters for the various matrix block conditions.
LF40-LF99	Loop over columns to be factored or partially factored.
LF44-LF46	Write column of A in scratch vector D.
LF49-LF62	Computations for u_{ir} (see FACTR), where positioning for size is taken into account. The range of i is determined by the matrix blocks used.
LF69-LF71	For case 3, the partially factored column is stored in A, and a jump to LF100 is made.
LF73-LF87	For cases 1 and 2, the maximum value in the column is found for positioning.

LF92-LF94 For cases 1 and 2, ℓ_{ir} (see FACTR) is calculated; limits on i are dependent on blocks.

SYMBOL DICTIONARY

A	=	array which contains the two blocks of columns of the transposed matrix in some state of factorization
CONJG	=	external routine (conjugate of complex numbers)
D	=	scratch vector, temporary storage of one column
DMAX	=	maximum value in column
ELMAG	=	intermediate variable
I	=	DO loop lndex
IFLG	=	small pivot value flag
IP	=	array containing positioning information
IXJ	=	index
IX1	=	first block number, input
IX2	=	second block number, input
J	=	DO loop index
JP1	=	J + 1
J1 J2	=	DO loop limits
J2P1	=	J2 + 1
J2P2	=	J2 + 2
K	=	DO loop index
L1 L2 L3	=	logical variables for testing
NCOL	=	number of columns
NROW	=	number of rows
PJ PR	=	intermediate variables
R	=	DO loop index
REAL	=	external routine (real part of a complex number)
R1 R2	=	DO loop limits, relative column number limits for calculations

In programs using double precision accumulation in the matrix solution, the following double precision variables are used in LFACTR.

DAR1 DAI1 DAR2 DAI2	=	real and imaginary parts of a number for temporary storage
DR DI	=	real and imaginary vectors replacing the complex vector D in single precision programs
1.E-10	=	small value test

```
      SUBROUTINE LFACTR( A, NROW, IX1, IX2, IP)
C
C     LFACTR PERFORMS GAUSS-DOOLITTLE MANIPULATIONS ON THE TWO BLOCKS OF
C     THE TRANSPOSED MATRIX IN CORE STORAGE.  THE GAUSS-DOOLITTLE
C     ALGORITHM IS PRESENTED ON PAGES 411-416 OF A. RALSTON -- A FIRST
C     COURSE IN NUMERICAL ANALYSIS.  COMMENTS BELOW REFER TO COMMENTS IN
C     RALSTONS TEXT.
C
      COMPLEX  A, D, AJR
      INTEGER  R, R1, R2, PJ, PR
      LOGICAL  L1, L2, L3
      COMMON  /MATPAR/ ICASE, NBLOKS, NPBLK, NLAST, NBLSYM, NPSYM,
     *NLSYM, IMAT, ICASX, NBBX, NPBX, NLBX, NBBL, NPBL, NLBL
      COMMON  /SCRATM/ D( N2M)
      DIMENSION  A( NROW,1), IP( NROW)
C
C     INITIALIZE R1,R2,J1,J2
C
      IFLG=0
      L1= IX1.EQ.1.AND. IX2.EQ.2
      L2=( IX2-1).EQ. IX1
      L3= IX2.EQ. NBLSYM
      IF(L1) GOTO 1
      GOTO 2
    1 R1=1
      R2=2* NPSYM
      J1=1
      J2=-1
      GOTO 5
    2 R1= NPSYM+1
      R2=2* NPSYM
      J1=( IX1-1)* NPSYM+1
      IF(L2) GOTO 3
      GOTO 4
    3 J2= J1+ NPSYM-2
      GOTO 5
    4 J2= J1+ NPSYM-1
    5 IF(L3) R2= NPSYM+ NLSYM
C
C     STEP 1
C
      DO 16  R= R1, R2
      DO 6  K= J1, NROW
      D( K)= A( K, R)
C
C     STEPS 2 AND 3
C
    6 CONTINUE
      IF(L1.OR. L2) J2= J2+1
```

```
      IF(J1.GT. J2) GOTO 9
      IXJ=0
      DO 8  J= J1, J2
      IXJ= IXJ+1
      PJ= IP( J)
      AJR= D( PJ)
      A( J, R)= AJR
      D( PJ)= D( J)
      JP1= J+1
      DO 7  I= JP1, NROW
      D( I)= D( I)- A( I, IXJ)* AJR
    7 CONTINUE
    8 CONTINUE
C
C     STEP 4
C
    9 CONTINUE
      J2P1= J2+1
      IF(L1.OR. L2) GOTO 11
      IF(NROW.LT. J2P1) GOTO 16
      DO 10  I= J2P1, NROW
      A( I, R)= D( I)
   10 CONTINUE
      GOTO 16
   11 DMAX= REAL( D( J2P1)* CONJG( D( J2P1)))
      IP( J2P1)= J2P1
      J2P2= J2+2
      IF(J2P2.GT. NROW) GOTO 13
      DO 12  I= J2P2, NROW
      ELMAG= REAL( D( I)* CONJG( D( I)))
      IF(ELMAG.LT. DMAX) GOTO 12
      DMAX= ELMAG
      IP( J2P1)= I
   12 CONTINUE
   13 CONTINUE
      IF(DMAX.LT.1.D-10) IFLG=1
      PR= IP( J2P1)
      A( J2P1, R)= D( PR)
C
C     STEP 5
C
      D( PR)= D( J2P1)
      IF(J2P2.GT. NROW) GOTO 15
      AJR=1./ A( J2P1, R)
      DO 14  I= J2P2, NROW
      A( I, R)= D( I)* AJR
   14 CONTINUE
   15 CONTINUE
      IF(IFLG.EQ.0) GOTO 16
```

```
      WRITE (2,17)  J2, DMAX
      IFLG=0
   16 CONTINUE
C
      RETURN
   17 FORMAT(' ','PIVOT(,I3,2H)=',1P,E16.8)
      END
```

LOAD

PURPOSE

To compute the impedances at a given frequency for the loading specified by LD cards.

METHOD

The value of λZ/Δ, where Z is the total impedance on a segment and Δ is the length of the segment, is computed for each loaded segment and stored in the array ZARRAY. The proper impedance formula is chosen by the value of the input quantity LDTYP. These computations are performed from the sequence LO74 to LO96 of the program, and the formulae are:

LDTYP = 0 (series R, L, and C):

$Z = R + j\omega L + \frac{1}{j\omega C}$

$Z' = \frac{\lambda Z}{\Delta} = \frac{R}{(\Delta/\lambda)} + j2\pi c(L/\Delta) + \frac{1}{j2\pi c(\Delta/\lambda)^2(C/\Delta)}$

where c is the speed of light and R, L, and C are input.

LDTYP = 1 (parallel R, L, and C; R, L, and C input):

$$Z' = \frac{1}{(\Delta/\lambda)(1/R) + \frac{\Delta}{j2\pi cL} + j2\pi c(\Delta/\lambda)^2(C/\Delta)}$$

LDTYP = 2 and 3 (same as above, but R/Δ L/Δ, C/Δ are input)

LDTYP = 4 (resistance and reactance input);

$$Z' = \frac{\texttt{resistance} + j\texttt{reactance}}{(\Delta/\lambda)}$$

LDTYP = 5 (call another subroutine for wire conductivity calculation)

SYMBOL DICTIONARY

ABS	=	external routine (absolute value of a real number)
AIMAG	=	external routine (imaginary part of a complex number)
CMPLX	=	external routine (forms a complex number)
ICHK	=	check flag in diagnosing data errors
ISTEP	=	loading card subscript
IWARN	=	flag checking for multiply loaded segments
JUMP	=	LDTYP + 1
LDTAG	=	tag number, input quantity
LDTAGF	=	input quantity
LDTAGS	=	LDTAG(ISTEP)
LDTAGT	=	input quantity
LDTYP	=	input quantity specifying loading type
NLOAD	=	number of input loading data cards
PRNT	=	external routine (prints the impedance data in a table)
REAL	=	external routine (takes the real part of a complex number)
TPCJ	=	j2π, where c is the speed of light
ZARRAY	=	array containing λZ/Δ for each segment, dimensioned to the maximum number of segments
ZINT	=	external routine (calculates the internal impedance of a finitely conducting wire)
ZLC ZLI ZLR	=	input quantities, the definitions are a function of the type of loading specified. For the case of series RLC (LDTYP = 0): ZLC = capacitance (farads), ZLI = inductance (henrys), and ZLR = resistance (ohms). For the remaining cases, see Part III.
ZT	=	Z' = λZ/Δ for one segment; however, variable name is used during the calculation of this quantity
1.E-20	=	Floating point zero test:
(0.,1.88365371E+9)	=	j2πc, where c is the velocity of light

```
      SUBROUTINE LOAD(LDTYP,LDTAG,LDTAGF,LDTAGT,ZLR,ZLI,ZLC)
C
C     LOAD CALCULATES THE IMPEDANCE OF SPECIFIED SEGMENTS FOR VARIOUS
C     TYPES OF LOADING
C
      COMPLEX  ZARRAY, ZT, TPCJ, ZINT
      COMMON/DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X( NM), Y( NM),
     *Z( NM), SI( NM), BI( NM), ALP( NM), BET( NM), ICON1( N2M), ICON2(
     * N2M), ITAG( N2M), ICONX( NM), WLAM, IPSYM
      COMMON/ZLOAD/ ZARRAY( NM), NLOAD, NLODF
      DIMENSION  LDTYP(1), LDTAG(1), LDTAGF(1), LDTAGT(1), ZLR(1), ZLI(
     *1), ZLC(1), TPCJX(2)
      EQUIVALENCE(TPCJ,TPCJX)
C
C     WRITE(6,HEADING)
C
      DATA   TPCJX/0.,1.883698955D+9/
C
C     INITIALIZE D ARRAY, USED FOR TEMPORARY STORAGE OF LOADING
C     INFORMATION.
C
      WRITE (2,25)
      DO 1  I= N2, N
    1 ZARRAY(I)=(0.,0.)
C
C     CYCLE OVER LOADING CARDS
C
      IWARN=0
      ISTEP=0
    2 ISTEP= ISTEP+1
      IF(ISTEP.LE. NLOAD) GOTO 5
      IF(IWARN.EQ.1) WRITE (2,26)
      IF(N1+2* M1.GT.0) GOTO 4
      NOP= N/ NP
      IF(NOP.EQ.1) GOTO 4
      DO 3  I=1, NP
      ZT= ZARRAY( I)
      L1=I
      DO 3  L2=2, NOP
      L1=L1+ NP
    3 ZARRAY( L1)= ZT
    4 RETURN
    5 IF(LDTYP(ISTEP).LE.5) GOTO 6
      WRITE (2,27)  LDTYP( ISTEP)
      STOP
    6 LDTAGS= LDTAG( ISTEP)
      JUMP= LDTYP( ISTEP)+1
C
C     SEARCH SEGMENTS FOR PROPER ITAGS
```

```
C
      ICHK=0
      L1=N2
      L2=N
      IF(LDTAGS.NE.0) GOTO 7
      IF(LDTAGF(ISTEP).EQ.0.AND. LDTAGT( ISTEP).EQ.0) GOTO 7
      L1=LDTAGF(ISTEP)
      L2=LDTAGT(ISTEP)
      IF(L1.GT.N1) GOTO 7
      WRITE(2,29)
      STOP
    7 DO 17 I= L1, L2
      IF(LDTAGS.EQ.0) GOTO 8
      IF(LDTAGS.NE. ITAG( I)) GOTO 17
      IF(LDTAGF(ISTEP).EQ.0) GOTO 8
      ICHK=ICHK+1
      IF(ICHK.GE.LDTAGF(ISTEP).AND.ICHK.LE.LDTAGT(ISTEP)) GOTO 9
      GOTO 17
C
C     CALCULATION OF LAMDA*IMPED. PER UNIT LENGTH, JUMP TO APPROPRIATE
C     SECTION FOR LOADING TYPE
C
    8 ICHK=1
    9 GOTO(10,11,12,13,14,15), JUMP
   10 ZT= ZLR( ISTEP)/ SI( I)+ TPCJ* ZLI( ISTEP)/( SI( I)* WLAM)
      IF(ABS( ZLC( ISTEP)).GT.1.D-20) ZT= ZT+ WLAM/( TPCJ* SI( I)* ZLC
     *( ISTEP))
      GOTO 16
   11 ZT= TPCJ* SI( I)* ZLC( ISTEP)/ WLAM
      IF(ABS( ZLI( ISTEP)).GT.1.D-20) ZT= ZT+ SI( I)* WLAM/( TPCJ* ZLI
     *( ISTEP))
      IF(ABS( ZLR( ISTEP)).GT.1.D-20) ZT= ZT+ SI( I)/ ZLR( ISTEP)
      ZT=1./ ZT
      GOTO 16
   12 ZT= ZLR( ISTEP)* WLAM+ TPCJ* ZLI( ISTEP)
      IF(ABS( ZLC( ISTEP)).GT.1.D-20) ZT= ZT+1./( TPCJ* SI( I)* SI( I)
     ** ZLC( ISTEP))
      GOTO 16
   13 ZT= TPCJ* SI( I)* SI( I)* ZLC( ISTEP)
      IF(ABS( ZLI( ISTEP)).GT.1.D-20) ZT= ZT+1./( TPCJ* ZLI( ISTEP))
      IF(ABS( ZLR( ISTEP)).GT.1.D-20) ZT= ZT+1./( ZLR( ISTEP)* WLAM)
      ZT=1./ZT
      GOTO 16
   14 ZT=CMPLX( ZLR( ISTEP), ZLI( ISTEP))/ SI( I)
      GOTO 16
   15 ZT=ZINT( ZLR( ISTEP)* WLAM, BI( I))
   16 IF((ABS( REAL( ZARRAY( I)))+ ABS( AIMAG( ZARRAY( I)))).GT.1.D-20
     *) IWARN=1
      ZARRAY(I)=ZARRAY( I)+ ZT
```

```
   17 CONTINUE
      IF(ICHK.NE.0) GOTO 18
      WRITE(2,28)  LDTAGS
C
C     PRINTING THE SEGMENT LOADING DATA, JUMP TO PROPER PRINT
C
      STOP
   18 GOTO(19,20,21,22,23,24), JUMP
   19 CALL PRNT( LDTAGS, LDTAGF( ISTEP), LDTAGT( ISTEP), ZLR( ISTEP),
     *ZLI(ISTEP), ZLC( ISTEP),0.,0.,0.,' SERIES ',2)
      GOTO 2
   20 CALL PRNT( LDTAGS, LDTAGF( ISTEP), LDTAGT( ISTEP), ZLR( ISTEP),
     *ZLI( ISTEP), ZLC( ISTEP),0.,0.,0.,'PARALLEL',2)
      GOTO 2
   21 CALL PRNT( LDTAGS, LDTAGF( ISTEP), LDTAGT( ISTEP), ZLR( ISTEP),
     *ZLI( ISTEP), ZLC( ISTEP),0.,0.,0.,'SERIES (PER METER)',5)
      GOTO 2
   22 CALL PRNT( LDTAGS, LDTAGF( ISTEP), LDTAGT( ISTEP), ZLR( ISTEP),
     *ZLI( ISTEP), ZLC( ISTEP),0.,0.,0.,'PARALLEL (PER METER)',5)
      GOTO 2
   23 CALL PRNT( LDTAGS, LDTAGF( ISTEP), LDTAGT( ISTEP),0.,0.,0., ZLR(
     *ISTEP), ZLI( ISTEP),0.,'FIXED IMPEDANCE ',4)
      GOTO 2
   24 CALL PRNT( LDTAGS, LDTAGF( ISTEP), LDTAGT( ISTEP),0.,0.,0.,0.,0.,
     * ZLR( ISTEP),'  WIRE  ',2)
C
      GOTO 2
   25 FORMAT(//,7X,'LOCATION',10X,'RESISTANCE',3X,'INDUCTANCE',2X,
     *'CAPACITANCE',7X,'IMPEDANCE (OHMS)',5X,'CONDUCTIVITY',4X,'TYPE',/
     *,4X,'ITAG',' FROM THRU',10X,'OHMS',8X,'HENRYS',7X,'FARADS',8X,
     *'REAL',6X,'IMAGINARY',4X,'MHOS/METER')
   26 FORMAT(/,10X,'NOTE, SOME OF THE ABOVE SEGMENTS HAVE BEEN LOADED',
     *' TWICE - IMPEDANCES ADDED')
   27 FORMAT(/,10X,'IMPROPER LOAD TYPE CHOOSEN, REQUESTED TYPE IS ',I3)
     *
   28 FORMAT(/,10X,'LOADING DATA CARD ERROR, NO SEGMENT HAS AN ITAG =',
     *I5)
   29 FORMAT(' ERROR - LOADING MAY NOT BE ADDED TO SEGMENTS IN N.G.F.',
     *' SECTION')
      END
```

LTSOLV

PURPOSE

To solve the matrix equation $X^R LU = B^R$, where R denotes a row vector and L and U are the lower and upper triangular matrices stored as blocks on files.

METHOD

The L and U triangular matrices are written in a square array, where the 1's on the diagonal of the L matrix are suppressed. The array is stored by blocks of columns in ascending order on file IFL1 and descending order an file IFL2. The solution procedure is as follows. First solve the equation

$$Y^R U = B^R$$

then

$$X^R L = Y^R$$

since $X^R LU = B^R$. The solutions of equations (1) and (2) are straightforward, since both matrices are triangular. In particular for equation (1),

$$y_j^R = \frac{1}{u_{jj}} \left(b_j^R - \sum_{i=1}^{j-1} y_i^R u_{i,j} \right) \qquad j = 1, ..., n$$

and similarly for equation (2).

Several right-hand side vectors may be stored in the two dimensional array B. The forward and backward substitution is then done on each vector in the loops from LT 23 to LT 34 and LT 43 to LT 56. This can be much faster than calling LTSOLV for each vector since the files IFL1 and LFL2 are read only once. This feature is used in computing A−1B for the NGF solution. It is not used with the multiple excitations for a receiving pattern or to compute the driving point interaction matrix in NETWK but could reduce the out-of-core time in these cases.

Row interchanges were used to position elements for size in factoring the transposed structure matrix; therefore, the elements in the solution vector X^R are not in the original locations. Using the IX array (filled by LUNSCR), the vector can be put back into the original order. The integer contained in IX(J) is the index of the original location of the parameter now in the j-th location. The solution vector is overwritten on the input right-hand side vector B^R.

SYMBOL DICTIONARY

A	=	array for matrix blocks
B	=	B^R, right-hand side and solution
I2	=	number of words in a block
IFLI	=	file with blocks in normal order
IFL2	=	file with blocks in reversed order
IX	=	solution unscramble vector
IXBLK1	=	block number
J	=	row index
JST	=	initial value for J
K2	=	number of columns in a block
KP	=	column index
NEQ	=	total number of equations
NRR	=	number of right-hand side vectora in B
NROW	=	row dimension of A (number of equations in a symmetric section)
SUM	=	summation result

```
      SUBROUTINE LTSOLV(A,NROW,IX,B,NEQ,NRH,IFL1,IFL2)
C
C     LTSOLV SOLVES THE MATRIX EQ. Y(R)*LU(T)=B(R) WHERE (R) DENOTES ROW
C     VECTOR AND LU(T) DENOTES THE LU DECOMPOSITION OF THE TRANSPOSE OF
C     THE ORIGINAL COEFFICIENT MATRIX.  THE LU(T) DECOMPOSITION IS
C     STORED ON TAPE 5 IN BLOCKS IN ASCENDING ORDER AND ON FILE 3 IN
C     BLOCKS OF DESCENDING ORDER.
C
      COMPLEX  A, B, Y, SUM
      COMMON/MATPAR/ ICASE, NBLOKS, NPBLK, NLAST, NBLSYM, NPSYM,
     *NLSYM, IMAT, ICASX, NBBX, NPBX, NLBX, NBBL, NPBL, NLBL
      COMMON/SCRATM/ Y(N2M)
C
C     FORWARD SUBSTITUTION
C
      DIMENSION  A(NROW,NROW),B(NEQ,NRH),IX(NEQ)
      I2=2*NPSYM*NROW
      DO 4 IXBLK1=1,NBLSYM
      CALL BLCKIN(A,IFL1,1,I2,1,121)
      K2=NPSYM
      IF(IXBLK1.EQ.NBLSYM) K2=NLSYM
      JST=(IXBLK1-1)*NPSYM
      DO 4 IC=1,NRH
      J=JST
      DO 3 K=1,K2
      JM1=J
      J=J+1
      SUM=(0.,0.)
      IF(JM1.LT.1) GOTO 2
      DO 1 I=1,JM1
    1 SUM=SUM+A(I,K)*B(I,IC)
    2 B(J,IC)=(B(J,IC)-SUM)/A(J,K)
    3 CONTINUE
C
C     BACKWARD SUBSTITUTION
C
    4 CONTINUE
      JST=NROW+1
      DO 8  IXBLK1=1,NBLSYM
      CALL BLCKIN(A,IFL2,1,I2,1,122)
      K2=NPSYM
      IF(IXBLK1.EQ.1) K2=NLSYM
      DO 7  IC=1,NRH
      KP=K2+1
      J=JST
      DO 6 K=1,K2
      KP=KP-1
      JP1=J
      J=J-1
```

```
      SUM=(0.,0.)
      IF(NROW.LT.JP1) GOTO 6
      DO 5 I=JP1,NROW
    5 SUM=SUM+ A(I,KP)*B(I,IC)
      B(J,IC)=B(J,IC)- SUM
    6 CONTINUE
    7 CONTINUE
C
C     UNSCRAMBLE SOLUTION
C
    8 JST=JST-K2
      DO 10  IC=1,NRH
      DO 9  I=1,NROW
      IXI=IX(I)
    9 Y(IXI)=B(I,IC)
      DO 10 I=1, NROW
   10 B(I,IC)=Y(I)
      RETURN
      END
```

LUNSCR

PURPOSE

To unscramble the lower triangular matrix of the factored out-of-core matrix and to determine the appropriate ordering of the unknowns. The unscrambled factored matrix is written in blocks on file IU3 in ascending order and on file IU4 in descending order.

METHOD

During factorization by LFACTR, the elements in the lower triangular matrix L were not explicitly arranged in accordance with the row interchanges used in positioning for size during the calculations. Specifically, as the factorization proceeds by columns from left to right in the matrix, row rearrangements in the r-th column are not explicitly performed in the left r-1 columns; rather, positioning information is stored in the IP array. For the in-core calculations, these rearrangements are included during the final solution (subroutine SOLVE). For the out-of-core case, rearrangement during the solution (subroutine LTSOLV) is inconvenient, since the transposed system $x^r A^t = B^r$ is being solved, where r signifies a row vector.

The procedure for unscrambling the L matrix is as follows. p_k is the positioning information contained in IP(K). Then for the r-th column, let t be a temporary variable:

$t = \ell_{k,r}$

$\ell_{p_k,r}$ overwrites $\ell_{k,r}$

t overwrites $\ell_{p_k,r}$ for $k = r+1, ..., n-1$

Since row interchanges were used on the transposed matrix, the positions of the unknowns in the equations have changed. The final arrangement is determined by performing interchanges on a vector of integers. Specifically, let

$x_i = i, \quad i = 1, ...n$

then set

$t = x_k$

x_{p_k} overwrites x_k

t overwrites x_{p_k} for $k = 1, ..., n$

The integer now contained in x_i specifies the original placement of the i-th unknown.

SYMBOL DICTIONARY

A	=	array for matrix blocks
I1	=	first word of matrix block
I2	=	last word of matrix block
IP	=	array of pivot index data
IU2	=	input file
IU3	=	output file, blacks in normal order
IU4	=	output file, blocks in reversed order
IX	=	array x_i
IXBLK1	=	block number
KA	=	increment to locate the KK-th submatrix in case of symmetry
NOP	=	number of symmetric sections
NROW	=	row dimension of A

```
      SUBROUTINE LUNSCR( A, NROW, NOP, IX, IP, IU2, IU3, IU4)
C
C     S/R WHICH UNSCRAMBLES, SCRAMBLED FACTORED MATRIX
C
      COMPLEX  A, TEMP
      COMMON  /MATPAR/ ICASE, NBLOKS, NPBLK, NLAST, NBLSYM, NPSYM,
     *NLSYM, IMAT, ICASX, NBBX, NPBX, NLBX, NBBL, NPBL, NLBL
      DIMENSION  A( NROW,1), IP( NROW), IX( NROW)
      I1=1
      I2=2* NPSYM* NROW
      NM1= NROW-1
      REWIND IU2
      REWIND IU3
      REWIND IU4
      DO 9  KK=1, NOP
      KA=( KK-1)* NROW
      DO 4  IXBLK1=1, NBLSYM
      CALL BLCKIN( A, IU2, I1, I2,1,121)
      K1=( IXBLK1-1)* NPSYM+2
      IF(NM1.LT. K1) GOTO 3
      J2=0
      DO 2  K= K1, NM1
      IF(J2.LT. NPSYM) J2= J2+1
      IPK= IP( K+ KA)
      DO 1  J=1, J2
      TEMP= A( K, J)
      A( K, J)= A( IPK, J)
      A( IPK, J)= TEMP
    1 CONTINUE
    2 CONTINUE
    3 CONTINUE
      CALL BLCKOT( A, IU3, I1, I2,1,122)
    4 CONTINUE
      DO 5  IXBLK1=1, NBLSYM
      BACKSPACE IU3
      IF(IXBLK1.NE.1) BACKSPACE IU3
      CALL BLCKIN( A, IU3, I1, I2,1,123)
      CALL BLCKOT( A, IU4, I1, I2,1,124)
    5 CONTINUE
      DO 6  I=1, NROW
      IX( I+ KA)= I
    6 CONTINUE
      DO 7  I=1, NROW
      IPI= IP( I+ KA)
      IXT= IX( I+ KA)
      IX( I+ KA)= IX( IPI+ KA)
      IX( IPI+ KA)= IXT
    7 CONTINUE
      IF(NOP.EQ.1) GOTO 9
```

```
C     SKIP NB1 LOGICAL RECORDS FORWARD
      NB1= NBLSYM-1
      DO 8  IXBLK1=1, NB1
      CALL BLCKIN( A, IU3, I1, I2,1,125)
    8 CONTINUE
    9 CONTINUE
      REWIND IU2
      REWIND IU3
      REWIND IU4
      RETURN
      END
```

MOVE

PURPOSE

To rotate and translate a previously defined structure, either moving original segments and patches or leaving the original fixed and producing new segments and patches.

METHOD

The formal parameters ROX, ROY, RDZ are the angles of rotation about the x, y, and z axes, respectively, and XS, YS, ZS are the translation distances in the x, y, and z directions. Angles are in radians, and a positive angle represents a right-hand rotation. The structure is first rotated about the x axis by ROX, then about the y axis by ROY, then about the z axis by ROZ, and finally translated by XS, YS, ZS. These operations transform a point with coordinates x, y, z to x', y', z', where

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} x_s \\ y_s \\ z_s \end{pmatrix}$$

Where

$T_{ll} = \cos\Phi\cos\theta$
$T_{12} = \cos\Phi\sin\theta\sin\psi - \sin\Phi\cos\psi;$
$T_{13} = \cos\Phi\sin\theta\cos\psi + \sin\Phi\sin\psi$
$T_{21} = \sin\Phi\cos\theta$
$T_{22} = \sin\Phi\sin\theta\sin\psi + \cos\Phi\cos\psi$
$T_{23} = \sin\Phi\sin\theta\cos\psi - \cos\Phi\sin\psi;$
$T_{31} = -\sin\theta$
$T_{32} = \cos\theta\sin\psi$
$T_{33} = \cos\theta\cos\psi$

with

ψ = ROX
θ = ROY
Φ = ROZ
X_s = XS
Y_s = YS
Z_s = ZS

This transformation is applied to those wire segments from segment number i_s to the last defined segment in COMMON/DATA/. Thus, if i_s is greater than 1, the segments from 1 to i_s-1 are unaffected. All patches are transformed.

NRPT is the structure repetition factor. If NRPT is zero, the transformed segment and patch coordinates overwrite the original coordinates so that the structure is moved with nothing left in the original location. If NRPT is greater than zero, the transformed coordinates are written on the ends of the arrays in COMMON/DATA/ and the process repeated NRPT times so that NRPT new structures are formed, each shifted from the previous one by the specified transformation, while the original structure is unchanged.

CODING

M018	Adjust symmetry flag if structure is rotated about the x or y axis. If the ground plane flag is also set on the GE card, symmetry will not be used in the solution.
M019-M033	Compute transformation matrix.
M037-M061	Transform segment coordinates.
M063-M093	Transform patch coordinates.
M094-M097	Set parametere to no-symmetry condition if NRPT > 0 or IX > 1.

SYMBOL DICTIONARY

CPH	=	cos Φ
CPS	=	cos ψ
CTH	=	cos θ
IR	=	DO loop index, array index for original patch
ISEGNO	=	external routine (searches segment tag numbers)
ITGI	=	increment applied to segment tag numbers as segments are transformed
ITS	=	i_s is the first occurring segment Ln COMMON/DATA] with tag ITS
IX	=	i_s
I1	=	lower DO loop limit for I (initially I1 = i_s)
K	=	increment to segment number for transformed segment
KR	=	array index for new patch
LDI	=	LD + 1
NRP	=	upper DO loop limit for IR
NRPT	=	repetition factor
ROX	=	ψ (radians)
ROY	=	θ
ROZ	=	Φ
SPH	=	sin Φ
SPS	=	sin ψ
STH	=	sin θ
T1X,T1Y,T1Z	=	arrays containing components of $\hat{t}_1$ for patches
T2X,T2Y,T2Z	=	arrays containing components of $\hat{t}_2$ for patches
XI	=	old x coordinate
XS	=	x_s
XX	=	T_{11}
XY	=	T_{12}
XZ	=	T_{13}
X2(I),Y2(I),Z2(I)	=	x,y,z coordinates of end 2 of segment I
YI	=	old y coordinate
YS	=	y_s
YX	=	T_{21}
YY	=	T_{22}
YZ	=	T_{23}
ZI	=	old Z coordinate
ZS	=	Z_s
ZX	=	T_{31}
ZY	=	T_{32}
ZZ	=	T_{33}

```
      SUBROUTINE MOVE( ROX, ROY, ROZ, XS, YS, ZS, ITS, NRPT, ITGI)
C
C     SUBROUTINE MOVE MOVES THE STRUCTURE WITH RESPECT TO ITS
C     COORDINATE SYSTEM OR REPRODUCES STRUCTURE IN NEW POSITIONS.
C     STRUCTURE IS ROTATED ABOUT X,Y,Z AXES BY ROX,ROY,ROZ
C     RESPECTIVELY, THEN SHIFTED BY XS,YS,ZS
C
      COMMON  /DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X( NM), Y( NM),
     *Z( NM), SI( NM), BI( NM), ALP( NM), BET( NM), ICON1( N2M), ICON2(
     * N2M), ITAG( N2M), ICONX( NM), WLAM, IPSYM
      COMMON  /ANGL/ SALP( NM)
      DIMENSION  T1X(1), T1Y(1), T1Z(1), T2X(1), T2Y(1), T2Z(1), X2(1),
     * Y2(1), Z2(1)
      EQUIVALENCE(X2(1),SI(1)),(Y2(1),ALP(1)),(Z2(1),BET(1))
      EQUIVALENCE(T1X,SI),(T1Y,ALP),(T1Z,BET),(T2X,ICON1),(T2Y,ICON2),(
     *T2Z,ITAG)
      IF(ABS( ROX)+ ABS( ROY).GT.1.D-10) IPSYM= IPSYM*3
      SPS= SIN( ROX)
      CPS= COS( ROX)
      STH= SIN( ROY)
      CTH= COS( ROY)
      SPH= SIN( ROZ)
      CPH= COS( ROZ)
      XX= CPH* CTH
      XY= CPH* STH* SPS- SPH* CPS
      XZ= CPH* STH* CPS+ SPH* SPS
      YX= SPH* CTH
      YY= SPH* STH* SPS+ CPH* CPS
      YZ= SPH* STH* CPS- CPH* SPS
      ZX=- STH
      ZY= CTH* SPS
      ZZ= CTH* CPS
      NRP= NRPT
      IF(NRPT.EQ.0) NRP=1
      IX=1
      IF(N.LT. N2) GOTO 3
      I1= ISEGNO( ITS,1)
      IF(I1.LT. N2) I1= N2
      IX= I1
      K= N
      IF(NRPT.EQ.0) K= I1-1
      DO 2  IR=1, NRP
      DO 1  I= I1, N
      K= K+1
      XI= X( I)
      YI= Y( I)
      ZI= Z( I)
      X( K)= XI* XX+ YI* XY+ ZI* XZ+ XS
      Y( K)= XI* YX+ YI* YY+ ZI* YZ+ YS
```

```
      Z( K)= XI* ZX+ YI* ZY+ ZI* ZZ+ ZS
      XI= X2( I)
      YI= Y2( I)
      ZI= Z2( I)
      X2( K)= XI* XX+ YI* XY+ ZI* XZ+ XS
      Y2( K)= XI* YX+ YI* YY+ ZI* YZ+ YS
      Z2( K)= XI* ZX+ YI* ZY+ ZI* ZZ+ ZS
      BI( K)= BI( I)
      ITAG( K)= ITAG( I)
      IF(ITAG( I).NE.0) ITAG( K)= ITAG( I)+ ITGI
    1 CONTINUE
      I1= N+1
      N= K
    2 CONTINUE
    3 IF(M.LT. M2) GOTO 6
      I1= M2
      K= M
      LDI= LD+1
      IF(NRPT.EQ.0) K= M1
      DO 5  II=1, NRP
      DO 4  I= I1, M
      K= K+1
      IR= LDI- I
      KR= LDI- K
      XI= X( IR)
      YI= Y( IR)
      ZI= Z( IR)
      X( KR)= XI* XX+ YI* XY+ ZI* XZ+ XS
      Y( KR)= XI* YX+ YI* YY+ ZI* YZ+ YS
      Z( KR)= XI* ZX+ YI* ZY+ ZI* ZZ+ ZS
      XI= T1X( IR)
      YI= T1Y( IR)
      ZI= T1Z( IR)
      T1X( KR)= XI* XX+ YI* XY+ ZI* XZ
      T1Y( KR)= XI* YX+ YI* YY+ ZI* YZ
      T1Z( KR)= XI* ZX+ YI* ZY+ ZI* ZZ
      XI= T2X( IR)
      YI= T2Y( IR)
      ZI= T2Z( IR)
      T2X( KR)= XI* XX+ YI* XY+ ZI* XZ
      T2Y( KR)= XI* YX+ YI* YY+ ZI* YZ
      T2Z( KR)= XI* ZX+ YI* ZY+ ZI* ZZ
      SALP( KR)= SALP( IR)
    4 BI( KR)= BI( IR)
      I1= M+1
    5 M= K
    6 IF(( NRPT.EQ.0).AND.( IX.EQ.1)) RETURN
      NP= N
      MP= M
```

```
      IPSYM=0                                                           MO  99
      RETURN                                                            MO 100
      END                                                               MO 101
```

NEFLD

PURPOSE

To compute the near electric field due to currents induced on a structure.

CODING

NE30-NE93	Near E field due to currents on segments is computed.
NE30-NE41	Each segment is checked to determine whether the field observation point (XOB,YOB,ZOB) falls within the segment volume. If it does, AX is set to the radius of that segment. AX is then sent to routine EFLD as the radius of the observation segment. If (XOB,YOB,ZOB) is on the axis of a segment at its center, the field calculation with AX set to the segment radius is the same as that used in filling the matrix.
NE42-NE93	Loop computing the field contribution of each segment.
NE43-NE50	Parameters of source segment are stored in COMMON/DATAJ/.
NE51-NE85	When the extended thin wire approximation is used, INDI is set to 0 if end 1 of segment I is connected to a single parallel segment of the same radius, 1 if it is a free end, and 2 if it connects to a multiple junction, a bend, or a segment of different radius. IND2 is the same for end 2. If IND1 or IND2 is 2, the extended thin wire approximation will not be used for that end.
NE87	EFLD stores the electric fields due to constant, sin ks, and cos ks currents in COMMON/DATAJ/.
NE88-NE93	The field components are multiplied by the coefficients of the constant, sin ka, and cos ks components of the total segment current, and the field is summed.
NE95-NE117	Near field due to patch currents is computed.

SYMBOL DICTIONARY

ACX	=	constant component of segment current at NE88; $\hat{t}_1$ component of patch current at NE110
AX	=	segment radius when the field evaluation point falls within a segment volume
B	=	source segment radius
BCX	=	sin ks component of segment current at NE89; $\hat{t}_2$ component of patch current at NE111
CCX	=	cos ks component of segment current at NE90
EX	=	x-component of total electric field
EY	=	y-component of total electric field
EZ	=	z-component of total electric field
EXC	=	x-component E field due to a cos ks current on a segment
EYC	=	y-component E field due to a cos ks current on a segment
EZC	=	z-component E field due to a cos ks current on a segment

EXK,EYK,EZK	=	E field due to a constant current at NE87; E field due to the $\hat{t}_1$ component of patch current at NE114
EXS,EYS,EZS	=	E field due to a sin ks current at NE87; E field due to the $\hat{t}_1$ component of patch current at NE114
IP	=	loop index for direct and reflected field (1,2 respectively)
T1X,T1Y,T1Z	=	arrays for $\hat{t}_1$
T1XJ,T1YJ,T1ZJ	=	$\hat{t}_1$ for source patch
T2X,T2Y,T2Z	=	arrays for $\hat{t}_2$
T2XJ,T2YJ,T2ZJ	=	$\hat{t}_2$ for source patch
XI	=	cosine of the angle between segment I and the segment connected to its end
XOB,YOB,ZOB	=	field evaluation point
ZP	=	coordinates of the field evaluation point, z or ρ^2 in a cylindrical coordinate system centered on the source element
0.5001	=	fraction of segment length used to test whether the field evaluation point falls within a segment
0.9	=	fraction of segment radius used to test whether the field evaluation point falls within a segment
0.999999	=	minimum XI for extended thin wire kernel (maximum angle = 0.08 degree)

```
      SUBROUTINE NEFLD( XOB, YOB, ZOB, EX, EY, EZ)
C
C     NEFLD COMPUTES THE NEAR FIELD AT SPECIFIED POINTS IN SPACE AFTER
C     THE STRUCTURE CURRENTS HAVE BEEN COMPUTED.
C
      COMPLEX  EX, EY, EZ, CUR, ACX, BCX, CCX, EXK, EYK, EZK, EXS,
     *EYS, EZS, EXC, EYC, EZC, ZRATI, ZRATI2, T1, FRATI
      COMMON  /DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X( NM), Y( NM),
     *Z( NM), SI( NM), BI( NM), ALP( NM), BET( NM), ICON1( N2M), ICON2(
     * N2M), ITAG( N2M), ICONX( NM), WLAM, IPSYM
      COMMON  /ANGL/ SALP( NM)
      COMMON  /CRNT/ AIR( NM), AII( NM), BIR( NM), BII( NM), CIR( NM),
     *CII( NM), CUR( N3M)
      COMMON  /DATAJ/ S, B, XJ, YJ, ZJ, CABJ, SABJ, SALPJ, EXK, EYK,
     *EZK, EXS, EYS, EZS, EXC, EYC, EZC, RKH, IEXK, IND1, INDD1, IND2,
     *INDD2, IPGND
      COMMON  /GND/ ZRATI, ZRATI2, FRATI, CL, CH, SCRWL, SCRWR, NRADL,
     *KSYMP, IFAR, IPERF, T1, T2
      DIMENSION  CAB(1), SAB(1), T1X(1), T1Y(1), T1Z(1), T2X(1), T2Y(1)
     *, T2Z(1)
      EQUIVALENCE(CAB,ALP),(SAB,BET)
      EQUIVALENCE(T1X,SI),(T1Y,ALP),(T1Z,BET),(T2X,ICON1),(T2Y,ICON2),(
     *T2Z,ITAG)
      EQUIVALENCE(T1XJ,CABJ),(T1YJ,SABJ),(T1ZJ,SALPJ),(T2XJ,B),(T2YJ,
     *IND1),(T2ZJ,IND2)
      EX=(0.,0.)
      EY=(0.,0.)
      EZ=(0.,0.)
      AX=0.
      IF(N.EQ.0) GOTO 20
      DO 1  I=1, N
      XJ= XOB- X( I)
      YJ= YOB- Y( I)
      ZJ= ZOB- Z( I)
      ZP= CAB( I)* XJ+ SAB( I)* YJ+ SALP( I)* ZJ
      IF(ABS( ZP).GT.0.5001* SI( I)) GOTO 1
      ZP= XJ* XJ+ YJ* YJ+ ZJ* ZJ- ZP* ZP
      XJ= BI( I)
      IF(ZP.GT.0.9* XJ* XJ) GOTO 1
      AX= XJ
      GOTO 2
    1 CONTINUE
    2 DO 19  I=1, N
      S= SI( I)
      B= BI( I)
      XJ= X( I)
      YJ= Y( I)
      ZJ= Z( I)
      CABJ= CAB( I)
```

```
      SABJ= SAB( I)
      SALPJ= SALP( I)
      IF(IEXK.EQ.0) GOTO 18
      IPR= ICON1( I)
      IF(IPR) 3,8,4
    3 IPR=- IPR
      IF(- ICON1( IPR).NE. I) GOTO 9
      GOTO 6
    4 IF(IPR.NE. I) GOTO 5
      IF(CABJ* CABJ+ SABJ* SABJ.GT.1.D-8) GOTO 9
      GOTO 7
    5 IF(ICON2( IPR).NE. I) GOTO 9
    6 XI= ABS( CABJ* CAB( IPR)+ SABJ* SAB( IPR)+ SALPJ* SALP( IPR))
      IF(XI.LT.0.999999D+0) GOTO 9
      IF(ABS( BI( IPR)/ B-1.).GT.1.D-6) GOTO 9
    7 IND1=0
      GOTO 10
    8 IND1=1
      GOTO 10
    9 IND1=2
   10 IPR= ICON2( I)
      IF(IPR) 11,16,12
   11 IPR=- IPR
      IF(- ICON2( IPR).NE. I) GOTO 17
      GOTO 14
   12 IF(IPR.NE. I) GOTO 13
      IF(CABJ* CABJ+ SABJ* SABJ.GT.1.D-8) GOTO 17
      GOTO 15
   13 IF(ICON1( IPR).NE. I) GOTO 17
   14 XI= ABS( CABJ* CAB( IPR)+ SABJ* SAB( IPR)+ SALPJ* SALP( IPR))
      IF(XI.LT.0.999999D+0) GOTO 17
      IF(ABS( BI( IPR)/ B-1.).GT.1.D-6) GOTO 17
   15 IND2=0
      GOTO 18
   16 IND2=1
      GOTO 18
   17 IND2=2
   18 CONTINUE
      CALL EFLD( XOB, YOB, ZOB, AX,1)
      ACX= CMPLX( AIR( I), AII( I))
      BCX= CMPLX( BIR( I), BII( I))
      CCX= CMPLX( CIR( I), CII( I))
      EX= EX+ EXK* ACX+ EXS* BCX+ EXC* CCX
      EY= EY+ EYK* ACX+ EYS* BCX+ EYC* CCX
   19 EZ= EZ+ EZK* ACX+ EZS* BCX+ EZC* CCX
      IF(M.EQ.0) RETURN
   20 JC= N
      JL= LD+1
      DO 21  I=1, M
```

```
      JL= JL-1
      S= BI( JL)
      XJ= X( JL)
      YJ= Y( JL)
      ZJ= Z( JL)
      T1XJ= T1X( JL)
      T1YJ= T1Y( JL)
      T1ZJ= T1Z( JL)
      T2XJ= T2X( JL)
      T2YJ= T2Y( JL)
      T2ZJ= T2Z( JL)
      JC= JC+3
      ACX= T1XJ* CUR( JC-2)+ T1YJ* CUR( JC-1)+ T1ZJ* CUR( JC)
      BCX= T2XJ* CUR( JC-2)+ T2YJ* CUR( JC-1)+ T2ZJ* CUR( JC)
      DO 21  IP=1, KSYMP
      IPGND= IP
      CALL UNERE( XOB, YOB, ZOB)
      EX= EX+ ACX* EXK+ BCX* EXS
      EY= EY+ ACX* EYK+ BCX* EYS
   21 EZ= EZ+ ACX* EZK+ BCX* EZS
      RETURN
      END
```

NETWK

PURPOSE

To solve for the voltages and currents at the ports of non-radiating networks that are part of the antenna. This routine also is involved in the solution for current when there are no non-radiating networks, and computes the relative driving point matrix asymmetry when this option is requested.

METHOD

Driving Point Matrix Asymmetry (NT32 to NTS4);

To satisfy physical reciprocity, the elements of the inverse of the interaction matrix should satisfy the condition

$$G_{ij}^{-1}/\Delta_i = G_{ji}^{-1}/Delta_i \qquad i,j = 1,...,n,$$

where Δ_i = length of segment i. This condition is not satisfied exactly, except on special structures, since the terms computed are not true reactions. The relative asymmetry of a matrix element is defined as

$$A = \left| \frac{\left(G_{ij}^{-1}/Delta_j - G_{ji}^{-1}/Delta_i \right)}{(G_{ij}^{-1}/Delta_j)} \right| .$$

The code from NT32 to NT84 computes the relative asymmetries of matrix elements for i and j of all driving point segments: either voltage source driving points or network connection points. The maximum relative asymmetry is located, and the rms relative asymmetry of all elements used is computed.

LOCAL CODING STRUCTURE

NT32-NT44	Determine numbers of segments that are network connection points.
NT46-NT54	Determine numbers of segments that are voltage source driving points. Indices of segments with network connections or voltage sources are stored in array IPNT with no duplication of numbers.
NT59-NT69	Compute $G_{k\ell}^{-1}/Delta_\ell$ for k,ℓ = all segment numbers in IPNT.
NT70-NT84	Compute relative asymmetries of elements computed above, search for maximum and compute rms asymmetry.

LOCAL SYMBOL DICTIONARY

ASA	=	sum of squares of relative asymmetries and rms value
ASM	=	Δ_{ISC1} before NT70; maximum relative asymmetry after NT69
CMN(J,I)	=	$G^{-1}_{k\ell}/Delta_\ell$; k = IPNT(J), ℓ = IPNT(I)
CUR	=	temporary storage of $G^{-1}_{\ell k}/\Delta_k$
IPNT	=	array of driving point segment indices
IROW1	=	number of entries in IPNT
ISC1	=	temporary storage of segment index
MASYM	=	flag; if non-zero, matrix asymmetry is computed
NTEQ	=	row index of element having maximum asymmetry
NTSC	=	column index of element having maximum asymmetry
PWR	=	relative matrix asymmetry
RHS	=	vector for matrix solution used in obtaining $G^{-1}_{k\ell}$

Non-radiating Network Solution (NT89 to NT262;

The solution method when non-radiating networks are present is discussed in Part I.

Data from non-radiating networks is passed through the COMMON/NETCX/ where

ISEG1(I)	=	number of the segment to which end 1 of i-th two-port network is connected
ISEG2(I)	=	number of segment to which end 2 of i-th two-port network is connected
NONET	=	number of two-port networks for which data is given

Network parameters are contained in the arrays X11R, X11I, X12R, X12I, X22R, and X22I, and the type of network is determined by NTYP:

If NTYP is 1 --- the network parameters are the short-oircuit admittance parameters of the network:

X11R, X11I = real and imaginary parts of Y_{11}
X12R, X12I = real and imaginary parts of $Y_{12} = Y_{21}$
X22R, X22I = real and imaginary parts of Y_{22}

If NTYP is 2 or 3 --- the network is a transmission line:

X11R = characteristic impedance of transmission line
X11I = length at transmission line in meters
Xl2R = real part oi shunt admittance on end 1 of line
X12I = imaginary part of shunt admittance on end 1 of line
X22R = real part of shunt admittance on end 2 of line
X22I = imaginary part of shunt admittance on end 2 of line

If NTYP is 2 -- the transmission line runs straight between the segments with respect to the segment reference directions.

If NTYP is 3 -- the transmission line is twisted as shown in figure 8.

The short circuit admittance parameters of the transmission line, Y_{11}, Y_{12}, and Y_{22}, are computed from NT110 to NT120 in the code. When NTYP is 3, the sign of Y_{12} is reversed.

The code from NT99 to NT194 forms a loop that for each network: computes the network parameters Y_{11}, Y_{12} and Y_{22}; sorts the segment indices involved; and adds the parameters Y_{11}, Y_{12}, and Y_{22} to the appropriate network equations. The sorting procedure for the connection of end 1 of the network is described in figure 9. Decision 1 is made in the code from NTl21 to NT126, decision 2 from NT128 to NTl33, and decision 3 from NT138 to NT143. Segments having network connections only are assigned equation rows in the array CMN starting from the top in the order that the segments are encountered. Segments with both network and voltage source connections are assigned equation rows in CMN starting at the bottom and proceeding up. The former are eventually solved for the unknown gap voltages, while the latter are used to obtain source input admittances after the structure currents have been computed. The code from NT148 to NT174 assigns equation numbers for the connection of end 2 of the networks and sets IROW2 and ISC2.

The network short circuit parameters are added to the network equations from NT182 to NT193. The coefficient matrix is transposed in filling the CMN array, since the matrix solution routines operate on a transposed system. Hence, the first index should be considered the column number and the second index the row number. If a segment NSEC1 does not have a voltage source connected, the parameters Y_{11} and Y_{12} are added to column IROW1 at rows IROW1 and IROW2, respectively. IROW2 may be either (1) in the upper rows as part of the equations far the unknown gap voltages, or (2) if a voltage source is connected to segment NSEG2, in the lower rows for later determination of the source current. If a voltage source is connected to segment NSEG1, the coefficients Y_{11} and Y_{12} are multiplied by the known source voltage and added to the right-hand side of the network equation in the rows IRoW1 and IROW2. The parameters Y_{12} and Y_{22} are added to the equations in a similar manner.

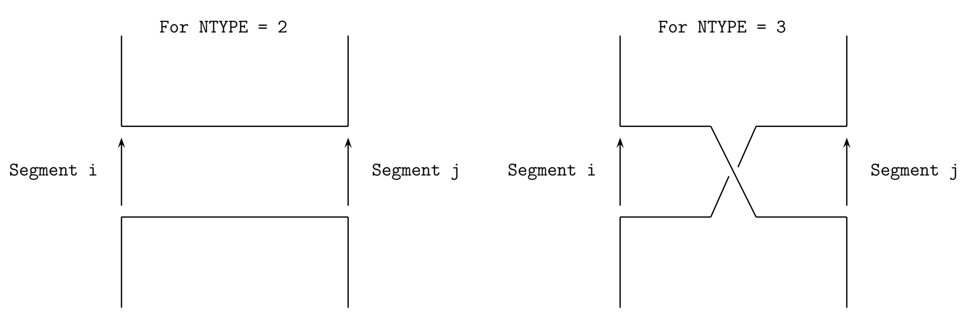


Figure 8. Options for Transmission Line Connection

The loop from NT199 to NT208 computes the elements of the inverse matrix G_{mn}^{-1} and adds them to the network equations. The network matrix is then factored at NT213, The code from NT218 to NT225 computes B_i = RHS(I), where

$$B_i = \sum_{j=1}^{N} G_{ij}^{-1} E_j' \qquad i = 1, ..., N \ ,$$

with ($-E_j$') being the known applied field on segment j, not including unknown voltage drops at network ports. Those elements B_i for segments in the network equations are then added to the right-hand side of the network equations. At NT229 the network equations are solved for the excitation fields due to voltage drops at the network ports. The negatives of these fields are added to the excitation vector at NT234 to NT236, completing the definition of the excitation vector E_j. The structure equations are then solved for the induced currents.

$$I_j = \sum_{j=1}^{N} G_{ij}^{-1} E_j \ .$$

From NT24l to NT26l, the voltage, current, admittance, and power seen looking into the structure at each network port are printed. This current does not include current through any voltage sources that are connected to the port.

The code from NT269 to NT294 computes and prints the voltage, current, admittance, and power seen by each voltage source looking into the structure and parallel connected network port, if a network is present.

After the network equations have once been set up, they can be solved for various incident fields by entering the code at NT218. If the location of voltage sources is changed, however, the equations must be recomputed.

If a structure has no non-radiating networks, the currents are computed at NT266.

SYMBOL DICTIONARY

ASA	=	sum of squares of relative matrix asymmetries and rms value
ASM	=	segment length and maximum relative matrix asymmetry
CABS	=	external routine (magnitude of complex number)
CM	=	array of matrix elements G_{ij}
CMN	=	array for network equation coefficients
CMPLX	=	external routine (forms complex number)
CONJG	=	external routine (conjugate)
COS	=	external routine (cosine)
CUR	=	current
EINC	=	excitation Vector
FACTR	=	external routine (Gauss-Doolittle matrix factoring)
FLOAT	=	external routine (integer to real conversion)
I	=	DO loop index
IP	=	array of positioning data from factoring of CM
IPNT	=	array of positioning data from factoring of CMN
IROW1	=	matrix element index
IROW2	=	matrix element index
ISANT	=	array of segment numbers for voltage source connection
ISC1	=	segment location in array ISANT
ISC2	=	segment location in array ISANT
ISEG1	=	number of segment to which port 1 of network is connected
ISEG2	=	number of segment co which port 2 is connected
IX	=	array of positioning data from factoring of GM
J	=	DO loop index
MASYM	=	flag to request matrix asymmetry calculation
NCOL	=	number of columns in CM
NDIMN	=	array dimension of CMN
NDIMNP	=	NDIMN + 1
NONET	=	number of networks
NOP	=	N/NP
NPRINT	=	flag to control printing
NROW	=	number of rows in CM
NSANT	=	number of voltage sources
NSEG1	=	array of segments to which port 1 of a network connects
NSEG2	=	array of segments to which port 2 of a network connects
NTEQA(I)	=	segment number associated with i-th network equation
NTSC	=	number of network-voltage source equations
NTSCA(I)	=	segment number associated with i-th network-voltage source equation
NTSOL	=	flag to indicate network equations do not need to be recomputed
NTYP(I)	=	type of i-th network
PIN	=	total input power from sources
PNLS	=	power lost in networks

PWR	=	power
REAL	=	external routine (real part of complex number)
RHNT	=	vector for right-hand side of network equations
RHNX	=	component of RHNT due to Y_{11}, Y_{12}, Y_{22} terms
RHS	=	vector for right-hand side of structure interaction equation
SIN	=	external routine (sine)
SOLVE	=	external routine (Gauss-Doolittle solution)
SOLVES	=	external routine (Geuss-Doolittle solution of CM matrix)
SQRT	=	external routine (square root)
TP	=	2π
VLT	=	voltage
VSANT(I)	=	voltage of source on segment NSANT(I)
VSRC(I)	=	voltage of source on i-th segment in network-voltage source equations
X11I,X11R		
X12I,X12R	=	network or transmission line specification parameters
X22I,X22R		
YMIT	=	admittance
Y1lI	=	imaginary part of Y_{ll}
Y1lR	=	real part of Y_{11}
Y12I	=	imaginary part of Y_{12}
Y12R	=	real part of Y_{12}
Y22I	=	imaginary part of Y_{22}
Y22R	=	real part of Y_{22}
ZPED	=	impedance
6.283185308	=	2π
30	=	row and column dimensions of CMN
31	=	(row and column dimensions of CMN) + 1

```
      SUBROUTINE NETWK( CM, CMB, CMC, CMD, IP, EINC)
C
C     SUBROUTINE NETWK SOLVES FOR STRUCTURE CURRENTS FOR A GIVEN
C     EXCITATION INCLUDING THE EFFECT OF NON-RADIATING NETWORKS IF
C     PRESENT.
C
      COMPLEX  CMN, RHNT, YMIT, RHS, ZPED, EINC, VSANT, VLT, CUR,
     *VSRC, RHNX, VQD, VQDS, CUX, CM, CMB, CMC, CMD
      COMMON  /DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X( NM), Y( NM),
     *Z( NM), SI( NM), BI( NM), ALP( NM), BET( NM), ICON1( N2M), ICON2(
     * N2M), ITAG( N2M), ICONX( NM), WLAM, IPSYM
      COMMON  /CRNT/ AIR( NM), AII( NM), BIR( NM), BII( NM), CIR( NM),
     *CII( NM), CUR( N3M)
      COMMON  /VSORC/ VQD(30), VSANT(30), VQDS(30), IVQD(30), ISANT(30)
     *, IQDS(30), NVQD, NSANT, NQDS
      COMMON  /NETCX/ ZPED, PIN, PNLS, NEQ, NPEQ, NEQ2, NONET, NTSOL,
     *NPRINT, MASYM, ISEG1(150), ISEG2(150), X11R(150), X11I(150),
     *X12R(150), X12I(150), X22R(150), X22I(150), NTYP(150)
      DIMENSION  EINC(1), IP(1), CM(1), CMB(1), CMC(1), CMD(1)
      DIMENSION  CMN(150,150), RHNT(150), IPNT(150), NTEQA(150),
     *NTSCA(150), RHS( N3M), VSRC(10), RHNX(150)
      DATA   NDIMN, NDIMNP/150,151/, TP/6.283185308D+0/
      NEQZ2= NEQ2
      IF(NEQZ2.EQ.0) NEQZ2=1
      PIN=0.
      PNLS=0.
      NEQT= NEQ+ NEQ2
      IF(NTSOL.NE.0) GOTO 42
      NOP= NEQ/ NPEQ
C
C     COMPUTE RELATIVE MATRIX ASYMMETRY
C
      IF(MASYM.EQ.0) GOTO 14
      IROW1=0
      IF(NONET.EQ.0) GOTO 5
      DO 4  I=1, NONET
      NSEG1= ISEG1( I)
      DO 3  ISC1=1,2
      IF(IROW1.EQ.0) GOTO 2
      DO 1  J=1, IROW1
      IF(NSEG1.EQ. IPNT( J)) GOTO 3
    1 CONTINUE
    2 IROW1= IROW1+1
      IPNT( IROW1)= NSEG1
    3 NSEG1= ISEG2( I)
    4 CONTINUE
    5 IF(NSANT.EQ.0) GOTO 9
      DO 8  I=1, NSANT
      NSEG1= ISANT( I)
```

```
      IF(IROW1.EQ.0) GOTO 7
      DO 6  J=1, IROW1
      IF(NSEG1.EQ. IPNT( J)) GOTO 8
    6 CONTINUE
    7 IROW1= IROW1+1
      IPNT( IROW1)= NSEG1
    8 CONTINUE
    9 IF(IROW1.LT. NDIMNP) GOTO 10
      WRITE (2,59)
      STOP
   10 IF(IROW1.LT.2) GOTO 14
      DO 12  I=1, IROW1
      ISC1= IPNT( I)
      ASM= SI( ISC1)
      DO 11  J=1, NEQT
   11 RHS( J)=(0.,0.)
      RHS( ISC1)=(1.,0.)
      CALL SOLGF( CM, CMB, CMC, CMD, RHS, IP, NP, N1, N, MP, M1, M, NEQ
     *, NEQ2, NEQZ2)
      CALL CABC( RHS)
      DO 12  J=1, IROW1
      ISC1= IPNT( J)
   12 CMN( J, I)= RHS( ISC1)/ ASM
      ASM=0.
      ASA=0.
      DO 13  I=2, IROW1
      ISC1= I-1
      DO 13  J=1, ISC1
      CUX= CMN( I, J)
      PWR= ABS(( CUX- CMN( J, I))/ CUX)
      ASA= ASA+ PWR* PWR
      IF(PWR.LT. ASM) GOTO 13
      ASM= PWR
      NTEQ= IPNT( I)
      NTSC= IPNT( J)
   13 CONTINUE
      ASA= SQRT( ASA*2./ DFLOAT( IROW1*( IROW1-1)))
      WRITE (2,58)  ASM, NTEQ, NTSC, ASA
C
C     SOLUTION OF NETWORK EQUATIONS
C
   14 IF(NONET.EQ.0) GOTO 48
      DO 15  I=1, NDIMN
      RHNX( I)=(0.,0.)
      DO 15  J=1, NDIMN
   15 CMN( I, J)=(0.,0.)
      NTEQ=0
C
C     SORT NETWORK AND SOURCE DATA AND ASSIGN EQUATION NUMBERS TO
```

```
C     SEGMENTS.
C
      NTSC=0
      DO 38  J=1, NONET
      NSEG1= ISEG1( J)
      NSEG2= ISEG2( J)
      IF(NTYP( J).GT.1) GOTO 16
      Y11R= X11R( J)
      Y11I= X11I( J)
      Y12R= X12R( J)
      Y12I= X12I( J)
      Y22R= X22R( J)
      Y22I= X22I( J)
      GOTO 17
   16 Y22R= TP* X11I( J)/ WLAM
      Y12R=0.
      Y12I=1./( X11R( J)* SIN( Y22R))
      Y11R= X12R( J)
      Y11I=- Y12I* COS( Y22R)
      Y22R= X22R( J)
      Y22I= Y11I+ X22I( J)
      Y11I= Y11I+ X12I( J)
      IF(NTYP( J).EQ.2) GOTO 17
      Y12R=- Y12R
      Y12I=- Y12I
   17 IF(NSANT.EQ.0) GOTO 19
      DO 18  I=1, NSANT
      IF(NSEG1.NE. ISANT( I)) GOTO 18
      ISC1= I
      GOTO 22
   18 CONTINUE
   19 ISC1=0
      IF(NTEQ.EQ.0) GOTO 21
      DO 20  I=1, NTEQ
      IF(NSEG1.NE. NTEQA( I)) GOTO 20
      IROW1= I
      GOTO 25
   20 CONTINUE
   21 NTEQ= NTEQ+1
      IROW1= NTEQ
      NTEQA( NTEQ)= NSEG1
      GOTO 25
   22 IF(NTSC.EQ.0) GOTO 24
      DO 23  I=1, NTSC
      IF(NSEG1.NE. NTSCA( I)) GOTO 23
      IROW1= NDIMNP- I
      GOTO 25
   23 CONTINUE
   24 NTSC= NTSC+1
```

```
      IROW1= NDIMNP- NTSC                                               NT 148
      NTSCA( NTSC)= NSEG1                                               NT 149
      VSRC( NTSC)= VSANT( ISC1)                                         NT 150
   25 IF(NSANT.EQ.0) GOTO 27                                            NT 151
      DO 26  I=1, NSANT                                                 NT 152
      IF(NSEG2.NE. ISANT( I)) GOTO 26                                   NT 153
      ISC2= I                                                           NT 154
      GOTO 30                                                           NT 155
   26 CONTINUE                                                          NT 156
   27 ISC2=0                                                            NT 157
      IF(NTEQ.EQ.0) GOTO 29                                             NT 158
      DO 28  I=1, NTEQ                                                  NT 159
      IF(NSEG2.NE. NTEQA( I)) GOTO 28                                   NT 160
      IROW2= I                                                          NT 161
      GOTO 33                                                           NT 162
   28 CONTINUE                                                          NT 163
   29 NTEQ= NTEQ+1                                                      NT 164
      IROW2= NTEQ                                                       NT 165
      NTEQA( NTEQ)= NSEG2                                               NT 166
      GOTO 33                                                           NT 167
   30 IF(NTSC.EQ.0) GOTO 32                                             NT 168
      DO 31  I=1, NTSC                                                  NT 169
      IF(NSEG2.NE. NTSCA( I)) GOTO 31                                   NT 170
      IROW2= NDIMNP- I                                                  NT 171
      GOTO 33                                                           NT 172
   31 CONTINUE                                                          NT 173
   32 NTSC= NTSC+1                                                      NT 174
      IROW2= NDIMNP- NTSC                                               NT 175
      NTSCA( NTSC)= NSEG2                                               NT 176
      VSRC( NTSC)= VSANT( ISC2)                                         NT 177
   33 IF(NTSC+ NTEQ.LT. NDIMNP) GOTO 34                                 NT 178
      WRITE (2,59)                                                      NT 179
C                                                                       NT 180
C     FILL NETWORK EQUATION MATRIX AND RIGHT HAND SIDE VECTOR WITH      NT 181
C     NETWORK SHORT-CIRCUIT ADMITTANCE MATRIX COEFFICIENTS.             NT 182
C                                                                       NT 183
      STOP                                                              NT 184
   34 IF(ISC1.NE.0) GOTO 35                                             NT 185
      CMN( IROW1, IROW1)= CMN( IROW1, IROW1)- CMPLX( Y11R, Y11I)* SI(   NT 186
     *NSEG1)                                                            NT 187
      CMN( IROW1, IROW2)= CMN( IROW1, IROW2)- CMPLX( Y12R, Y12I)* SI(   NT 188
     *NSEG1)                                                            NT 189
      GOTO 36                                                           NT 190
   35 RHNX( IROW1)= RHNX( IROW1)+ CMPLX( Y11R, Y11I)* VSANT( ISC1)/     NT 191
     *WLAM                                                              NT 192
      RHNX( IROW2)= RHNX( IROW2)+ CMPLX( Y12R, Y12I)* VSANT( ISC1)/     NT 193
     *WLAM                                                              NT 194
   36 IF(ISC2.NE.0) GOTO 37                                             NT 195
      CMN( IROW2, IROW2)= CMN( IROW2, IROW2)- CMPLX( Y22R, Y22I)* SI(   NT 196
```

```
     *NSEG2)
      CMN( IROW2, IROW1)= CMN( IROW2, IROW1)- CMPLX( Y12R, Y12I)* SI(
     *NSEG2)
      GOTO 38
   37 RHNX( IROW1)= RHNX( IROW1)+ CMPLX( Y12R, Y12I)* VSANT( ISC2)/
     *WLAM
      RHNX( IROW2)= RHNX( IROW2)+ CMPLX( Y22R, Y22I)* VSANT( ISC2)/
     *WLAM
C
C     ADD INTERACTION MATRIX ADMITTANCE ELEMENTS TO NETWORK EQUATION
C     MATRIX
C
   38 CONTINUE
      DO 41  I=1, NTEQ
      DO 39  J=1, NEQT
   39 RHS( J)=(0.,0.)
      IROW1= NTEQA( I)
      RHS( IROW1)=(1.,0.)
      CALL SOLGF( CM, CMB, CMC, CMD, RHS, IP, NP, N1, N, MP, M1, M, NEQ
     *, NEQ2, NEQZ2)
      CALL CABC( RHS)
      DO 40  J=1, NTEQ
      IROW1= NTEQA( J)
   40 CMN( I, J)= CMN( I, J)+ RHS( IROW1)
C
C     FACTOR NETWORK EQUATION MATRIX
C
   41 CONTINUE
C
C     ADD TO NETWORK EQUATION RIGHT HAND SIDE THE TERMS DUE TO ELEMENT
C     INTERACTIONS
C
      CALL FACTR( NTEQ, CMN, IPNT, NDIMN)
   42 IF(NONET.EQ.0) GOTO 48
      DO 43  I=1, NEQT
   43 RHS( I)= EINC( I)
      CALL SOLGF( CM, CMB, CMC, CMD, RHS, IP, NP, N1, N, MP, M1, M, NEQ
     *, NEQ2, NEQZ2)
      CALL CABC( RHS)
      DO 44  I=1, NTEQ
      IROW1= NTEQA( I)
C
C     SOLVE NETWORK EQUATIONS
C
   44 RHNT( I)= RHNX( I)+ RHS( IROW1)
C
C     ADD FIELDS DUE TO NETWORK VOLTAGES TO ELECTRIC FIELDS APPLIED TO
C     STRUCTURE AND SOLVE FOR INDUCED CURRENT
C
```

```
      CALL SOLVE( NTEQ, CMN, IPNT, RHNT, NDIMN)
      DO 45  I=1, NTEQ
      IROW1= NTEQA( I)
   45 EINC( IROW1)= EINC( IROW1)- RHNT( I)
      CALL SOLGF( CM, CMB, CMC, CMD, EINC, IP, NP, N1, N, MP, M1, M,
     *NEQ, NEQ2, NEQZ2)
      CALL CABC( EINC)
      IF(NPRINT.EQ.0) WRITE (2,61)
      IF(NPRINT.EQ.0) WRITE (2,60)
      DO 46  I=1, NTEQ
      IROW1= NTEQA( I)
      VLT= RHNT( I)* SI( IROW1)* WLAM
      CUX= EINC( IROW1)* WLAM
      YMIT= CUX/ VLT
      ZPED= VLT/ CUX
      IROW2= ITAG( IROW1)
      PWR=.5* REAL( VLT* CONJG( CUX))
      PNLS= PNLS- PWR
   46 IF(NPRINT.EQ.0) WRITE (2,62)  IROW2, IROW1, VLT, CUX, ZPED, YMIT
     *, PWR
      IF(NTSC.EQ.0) GOTO 49
      DO 47  I=1, NTSC
      IROW1= NTSCA( I)
      VLT= VSRC( I)
      CUX= EINC( IROW1)* WLAM
      YMIT= CUX/ VLT
      ZPED= VLT/ CUX
      IROW2= ITAG( IROW1)
      PWR=.5* REAL( VLT* CONJG( CUX))
      PNLS= PNLS- PWR
   47 IF(NPRINT.EQ.0) WRITE (2,62)  IROW2, IROW1, VLT, CUX, ZPED, YMIT
     *, PWR
C
C     SOLVE FOR CURRENTS WHEN NO NETWORKS ARE PRESENT
C
      GOTO 49
   48 CALL SOLGF( CM, CMB, CMC, CMD, EINC, IP, NP, N1, N, MP, M1, M,
     *NEQ, NEQ2, NEQZ2)
      CALL CABC( EINC)
      NTSC=0
   49 IF(NSANT+ NVQD.EQ.0) RETURN
      WRITE (2,63)
      WRITE (2,60)
      IF(NSANT.EQ.0) GOTO 56
      DO 55  I=1, NSANT
      ISC1= ISANT( I)
      VLT= VSANT( I)
      IF(NTSC.EQ.0) GOTO 51
      DO 50  J=1, NTSC
```

```
      IF(NTSCA( J).EQ. ISC1) GOTO 52
   50 CONTINUE
   51 CUX= EINC( ISC1)* WLAM
      IROW1=0
      GOTO 54
   52 IROW1= NDIMNP- J
      CUX= RHNX( IROW1)
      DO 53  J=1, NTEQ
   53 CUX= CUX- CMN( J, IROW1)* RHNT( J)
      CUX=( EINC( ISC1)+ CUX)* WLAM
   54 YMIT= CUX/ VLT
      ZPED= VLT/ CUX
      PWR=.5* REAL( VLT* CONJG( CUX))
      PIN= PIN+ PWR
      IF(IROW1.NE.0) PNLS= PNLS+ PWR
      IROW2= ITAG( ISC1)
   55 WRITE (2,62)  IROW2, ISC1, VLT, CUX, ZPED, YMIT, PWR
   56 IF(NVQD.EQ.0) RETURN
      DO 57  I=1, NVQD
      ISC1= IVQD( I)
      VLT= VQD( I)
      CUX= CMPLX( AIR( ISC1), AII( ISC1))
      YMIT= CMPLX( BIR( ISC1), BII( ISC1))
      ZPED= CMPLX( CIR( ISC1), CII( ISC1))
      PWR= SI( ISC1)* TP*.5
      CUX=( CUX- YMIT* SIN( PWR)+ ZPED* COS( PWR))* WLAM
      YMIT= CUX/ VLT
      ZPED= VLT/ CUX
      PWR=.5* REAL( VLT* CONJG( CUX))
      PIN= PIN+ PWR
      IROW2= ITAG( ISC1)
   57 WRITE (2,64)  IROW2, ISC1, VLT, CUX, ZPED, YMIT, PWR
C
      RETURN
   58 FORMAT(///,3X,'MAXIMUM RELATIVE ASYMMETRY OF THE DRIVING POINT',
     *' ADMITTANCE MATRIX IS',1P,E10.3,' FOR SEGMENTS',I5,4H AND,I5,/,3
     *X,'RMS RELATIVE ASYMMETRY IS',E10.3)
   59 FORMAT(1X,'ERROR - - NETWORK ARRAY DIMENSIONS TOO SMALL')
   60 FORMAT(/,3X,'TAG',3X,'SEG.',5X,'VOLTAGE (VOLTS)',11X,'CURRENT (',
     *'AMPS)',11X,'IMPEDANCE (OHMS)',10X,'ADMITTANCE (MHOS)',8X,'POWER',
     */,3X,'NO.',3X,'NO.',5X,'REAL',9X,'IMAG.',3(8X,'REAL',9X,'IMAG.'),6
     *X,'(WATTS)')
   61 FORMAT(///,27X,'- - - STRUCTURE EXCITATION DATA AT NETWORK CONN',
     *'ECTION POINTS - - -')
   62 FORMAT(2(1X,I5),1P,9E13.5)
   63 FORMAT(///,42X,'- - - ANTENNA INPUT PARAMETERS - - -')
   64 FORMAT(1X,I5,' *',I4,1P,9E13.5)
      END
```

<u>NFPAT</u>

PURPOSE

To compute and print the near E or H field over a range of points.

METHOD

The range of points in rectangular or spherical coordinates is obtained from parameters in COMMON/FPAT/. Subroutine NEFLD is called for near E field and NHFLD is called for near H field.

SYMBOL DICTIONARY

CPH	=	cos Φ
CTH	=	cos θ
DXNR	=	increment for x in rectangular coordinates or R in spherical coordinates
DYNR	=	increment for y in rectangular coordinates or Φ in spherical coordinates
DZNR	=	increment for z in rectangular coordinates or θ in spherical coordinates
EX,EY,EZ	=	x,y and z components of E or H
NEAR	=	0 for rectangular coordinates 1 for spherical coordinates
NFEH	=	0 for near E field 1 for near H field
NRX,NRY,NRZ	=	number of values for x,y and z or R, Φ, θ
SPH	=	sin Φ
STH	=	sin θ
TA	=	$\pi/180$
XNR	=	initial x or R
XNRT	=	x or R
XOB	=	x
YNR	=	initial y or Φ
YNRT	=	y or Φ
YOB	=	y
ZNR	=	initial z or θ
ZNRT	=	z or θ
ZOB	=	z

```
      SUBROUTINE NFPAT
C     COMPUTE NEAR E OR H FIELDS OVER A RANGE OF POINTS
      COMPLEX  EX, EY, EZ
      COMMON  /DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X( NM), Y( NM),
     *Z( NM), SI( NM), BI( NM), ALP( NM), BET( NM), ICON1( N2M), ICON2(
     * N2M), ITAG( N2M), ICONX( NM), WLAM, IPSYM

      COMMON  /FPAT/ NTH, NPH, IPD, IAVP, INOR, IAX, THETS, PHIS, DTH,
     *DPH, RFLD, GNOR, CLT, CHT, EPSR2, SIG2, IXTYP, XPR6, PINR, PNLR,
     *PLOSS, NEAR, NFEH, NRX, NRY, NRZ, XNR, YNR, ZNR, DXNR, DYNR, DZNR
     *

      COMMON  /PLOT/ IPLP1, IPLP2, IPLP3, IPLP4
      DATA   TA/1.745329252D-02/
      IF(NFEH.EQ.1) GOTO 1
      WRITE (2,10)
      GOTO 2
    1 WRITE (2,12)
    2 ZNRT= ZNR- DZNR
      DO 9  I=1, NRZ
      ZNRT= ZNRT+ DZNR
      IF(NEAR.EQ.0) GOTO 3
      CTH= COS( TA* ZNRT)
      STH= SIN( TA* ZNRT)
    3 YNRT= YNR- DYNR
      DO 9  J=1, NRY
      YNRT= YNRT+ DYNR
      IF(NEAR.EQ.0) GOTO 4
      CPH= COS( TA* YNRT)
      SPH= SIN( TA* YNRT)
    4 XNRT= XNR- DXNR
      DO 9  KK=1, NRX
      XNRT= XNRT+ DXNR
      IF(NEAR.EQ.0) GOTO 5
      XOB= XNRT* STH* CPH
      YOB= XNRT* STH* SPH
      ZOB= XNRT* CTH
      GOTO 6
    5 XOB= XNRT
      YOB= YNRT
      ZOB= ZNRT
    6 TMP1= XOB/ WLAM
      TMP2= YOB/ WLAM
      TMP3= ZOB/ WLAM
      IF(NFEH.EQ.1) GOTO 7
      CALL NEFLD( TMP1, TMP2, TMP3, EX, EY, EZ)
      GOTO 8
    7 CALL NHFLD( TMP1, TMP2, TMP3, EX, EY, EZ)
    8 TMP1= ABS( EX)
```

```
      TMP2= CANG( EX)
      TMP3= ABS( EY)
      TMP4= CANG( EY)
      TMP5= ABS( EZ)
      TMP6= CANG( EZ)

      WRITE (2,11)  XOB, YOB, ZOB, TMP1, TMP2, TMP3, TMP4, TMP5, TMP6
      IF(IPLP1.NE.2) GOTO 9
      GOTO (14,15,16), IPLP4
   14 XXX= XOB
      GOTO 17
   15 XXX= YOB
      GOTO 17
   16 XXX= ZOB
   17 CONTINUE
      IF(IPLP2.NE.2) GOTO 13
      IF(IPLP3.EQ.1) WRITE( 8,*)  XXX, TMP1, TMP2
      IF(IPLP3.EQ.2) WRITE( 8,*)  XXX, TMP3, TMP4
      IF(IPLP3.EQ.3) WRITE( 8,*)  XXX, TMP5, TMP6
      IF(IPLP3.EQ.4) WRITE( 8,*)  XXX, TMP1, TMP2, TMP3, TMP4, TMP5,
     *TMP6
      GOTO 9
   13 IF(IPLP2.NE.1) GOTO 9
      IF(IPLP3.EQ.1) WRITE( 8,*)  XXX, EX
      IF(IPLP3.EQ.2) WRITE( 8,*)  XXX, EY
      IF(IPLP3.EQ.3) WRITE( 8,*)  XXX, EZ

      IF(IPLP3.EQ.4) WRITE( 8,*)  XXX, EX, EY, EZ
    9 CONTINUE
C
      RETURN
   10 FORMAT(///,35X,'- - - NEAR ELECTRIC FIELDS - - -',//,12X,'-  L',
     *'OCATION  -',21X,'-  EX  -',15X,'-  EY  -',15X,'-  EZ  -',/,8X,
     *'X',10X,'Y',10X,'Z',10X,'MAGNITUDE',3X,'PHASE',6X,'MAGNITUDE',3X,
     *'PHASE',6X,'MAGNITUDE',3X,'PHASE',/,6X,'METERS',5X,'METERS',5X,
     *'METERS',8X,'VOLTS/M',3X,'DEGREES',6X,'VOLTS/M',3X,'DEGREES',6X
     *,'VOLTS/M',3X,'DEGREES')
   11 FORMAT(2X,3(2X,F9.4),1X,3(3X,1P,E11.4,2X,0P,F7.2))
   12 FORMAT(///,35X,'- - - NEAR MAGNETIC FIELDS - - -',//,12X,'-  L',
     *'OCATION  -',21X,'-  HX  -',15X,'-  HY  -',15X,'-  HZ  -',/,8X,
     *'X',10X,'Y',10X,'Z',10X,'MAGNITUDE',3X,'PHASE',6X,'MAGNITUDE',3X,
     *'PHASE',6X,'MAGNITUDE',3X,'PHASE',/,6X,'METERS',5X,'METERS',5X,
     *'METERS',9X,'AMPS/M',3X,'DEGREES',7X,'AMPS/M',3X,'DEGREES',7X,
     *'AMPS/M',3X,'DEGREES')
      END
```

NHFLD

PURPOSE

To compute the near magnetic field due to currents induced on a structure.

CODING

NH28-NH56	Near H field due to currents on segments is computed.
NH29-NH40	Each segment is checked to determine whether the field observation point (XOB,YOB,ZOB) falls within the segment volume. If it does, AX is set to the radius of that segment. AX is then sent to routine HSFLD as the radius of the observation segment to avoid a singularity in the field.
NH41-NH56	Loop computing the field contribution of each segment.
NH42-NH49	Parameters of source segment are stored in COMMON/DATAJ/.
NH50	HSFLH stores the magnetic field due to constant, sin ks, and cos ks currents in COMMON/DATAJ/.
NH54-NH56	The field components are multiplied by the coefficients of the constant, sin ks, and cos ks components of the total segment current, and the field is summed.
NH58-NH78	Near H fields due to patch currents are computed.
NH62-NH71	Parameters of source patch are set in COMMON/DATAJ/.
NH72	H field is computed by HINTC.
NH76-NH78	H fields due to $\hat{t}_1$ and $\hat{t}_2$ current components are nunapuea by the current strengths and summed.

SYMBOL DICTIONARY

ACX	=	constant component of the segment current at NH51; $\hat{t}_1$ component of patch current at NH74
AX	=	segment radius when the field evaluation point falls within a segment volume
BCX	=	sin ks component of segment current at NH52; $\hat{t}_2$ component of patch current at NH75
CCX	=	cos ks component of segment current at NH53
HX,HY,HZ	=	total H field
T1X,T1Y,T1Z	=	arrays for $\hat{t}_1$
T1XJ,T1YJ,T1ZJ	=	$\hat{t}_1$ for patch I
T2X,T2Y,T2Z	=	arrays for $\hat{t}_2$
T2XJ,T2YJ,T2ZJ	=	$\hat{t}_2$ for patch I
XOB,YOB,ZOB	=	field evalution point
ZP	=	coordinates of the field evaluation point, z or ρ^2, in a cylindrical coordinate system centered on the source element.
0.5001	=	fraction of segment length used to test whether the field evaluation point falls within a segment
0.9	=	fraction of segment radius used to test whether the field evaluation point falls within a segment

```
      SUBROUTINE NHFLD(XOB,YOB,ZOB,HX,HY,HZ)
C
C     NHFLD COMPUTES THE NEAR FIELD AT SPECIFIED POINTS IN SPACE AFTER
C     THE STRUCTURE CURRENTS HAVE BEEN COMPUTED.
C
      COMPLEX HX,HY,HZ,CUR,ACX,BCX,CCX,EXK,EYK,EZK,EXS,EYS,
     *EZS, EXC, EYC, EZC
      COMMON/DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X( NM), Y( NM),
     *Z( NM), SI( NM), BI( NM), ALP( NM), BET( NM), ICON1( N2M), ICON2(
     * N2M), ITAG( N2M), ICONX( NM), WLAM, IPSYM
      COMMON/ANGL/ SALP( NM)
      COMMON/CRNT/ AIR( NM), AII( NM), BIR( NM), BII( NM), CIR( NM),
     *CII( NM), CUR( N3M)
      COMMON/DATAJ/ S, B, XJ, YJ, ZJ, CABJ, SABJ, SALPJ, EXK, EYK,
     *EZK, EXS, EYS, EZS, EXC, EYC, EZC, RKH, IEXK, IND1, INDD1, IND2,
     *INDD2,IPGND
      DIMENSION  CAB(1), SAB(1)
      DIMENSION  T1X(1), T1Y(1), T1Z(1), T2X(1), T2Y(1), T2Z(1), XS(1),
     * YS(1), ZS(1)
      EQUIVALENCE(T1X,SI),(T1Y,ALP),(T1Z,BET),(T2X,ICON1),(T2Y,ICON2),(
     *T2Z,ITAG),(XS,X),(YS,Y),(ZS,Z)
      EQUIVALENCE(T1XJ,CABJ),(T1YJ,SABJ),(T1ZJ,SALPJ),(T2XJ,B),(T2YJ,
     *IND1),(T2ZJ,IND2)
      EQUIVALENCE(CAB,ALP),(SAB,BET)
      HX=(0.,0.)
      HY=(0.,0.)
      HZ=(0.,0.)
      AX=0.
      IF(N.EQ.0) GOTO 4
      DO 1  I=1, N
      XJ= XOB- X( I)
      YJ= YOB- Y( I)
      ZJ= ZOB- Z( I)
      ZP= CAB( I)* XJ+ SAB( I)* YJ+ SALP( I)* ZJ
      IF(ABS( ZP).GT.0.5001* SI( I)) GOTO 1
      ZP= XJ* XJ+ YJ* YJ+ ZJ* ZJ- ZP* ZP
      XJ= BI( I)
      IF(ZP.GT.0.9* XJ* XJ) GOTO 1
      AX= XJ
      GOTO 2
    1 CONTINUE
    2 DO 3  I=1, N
      S= SI( I)
      B= BI( I)
      XJ= X( I)
      YJ= Y( I)
      ZJ= Z( I)
      CABJ= CAB( I)
      SABJ= SAB( I)
```

```
      SALPJ= SALP( I)
      CALL HSFLD( XOB, YOB, ZOB, AX)
      ACX= CMPLX( AIR( I), AII( I))
      BCX= CMPLX( BIR( I), BII( I))
      CCX= CMPLX( CIR( I), CII( I))
      HX= HX+ EXK* ACX+ EXS* BCX+ EXC* CCX
      HY= HY+ EYK* ACX+ EYS* BCX+ EYC* CCX
    3 HZ= HZ+ EZK* ACX+ EZS* BCX+ EZC* CCX
      IF(M.EQ.0) RETURN
    4 JC= N
      JL= LD+1
      DO 5  I=1, M
      JL= JL-1
      S= BI( JL)
      XJ= X( JL)
      YJ= Y( JL)
      ZJ= Z( JL)
      T1XJ= T1X( JL)
      T1YJ= T1Y( JL)
      T1ZJ= T1Z( JL)
      T2XJ= T2X( JL)
      T2YJ= T2Y( JL)
      T2ZJ= T2Z( JL)
      CALL HINTG( XOB, YOB, ZOB)
      JC= JC+3
      ACX= T1XJ* CUR( JC-2)+ T1YJ* CUR( JC-1)+ T1ZJ* CUR( JC)
      BCX= T2XJ* CUR( JC-2)+ T2YJ* CUR( JC-1)+ T2ZJ* CUR( JC)
      HX= HX+ ACX* EXK+ BCX* EXS
      HY= HY+ ACX* EYK+ BCX* EYS
    5 HZ= HZ+ ACX* EZK+ BCX* EZS
      RETURN
      END
```

PATCH (entry SUBPH)

PURPOSE

To generate patch data for surfaces.

METHOD

The code from PA14 to PA129 generates data for a single new patch or multiple patches. There are four options for defining a single patch, as illustrated in Figure 5 of Part III. For a single patch, NX is zero and NY is NS+1 where NS is the parameter from the SP input card and is shown on Figure 5. Rectangular, triangular or quadrilateral patches are defined by the coordinates of three or four corners in the parameters X1 through Z4. In the arbitrary shape option (Figure 5A in Part III) the center of the patch is X1,Yl,Zl; α is X2; β is Y2; and the area is Z2. The patch data is stored in COMMON/DATA/ from the top of the arrays downward (see Section III).

The code from PA131 to PA190 divides s patch into four patches and is used when a wire connect: to a patch. If NY is equal to zero the patch NX is divided into four patches that become patches NX through NX+3. Patches following NX are shifted in the arrays in COMMON/DATA/ to leave space for the three additional patches. If NY is greater than zero, patch NX is left in the arrays but four new patches to replace it are added to the end of the arrays. The z coordinate of patch NK is then changed to 10,000 at PA189.

SYMBOL DICTIONARY

MI	=	array index for patch data
MIA	=	array index for patch data
NTP	=	patch type (NY for s single patch)
NX	=	zero for a single patch. For multiple patches NX is defined in Figure 6 of Part III. After ENTRY SUBPH, NX is the number of the patch to be divided
S1X,S1Y,S1Z	=	vector from corner 1 to corner 2
S2X,S2Y,S2Z	=	vector from corner 2 to corner 3
SALN	=	± 1 from array SALP
SALPN	=	factor in computing center of mass of quadrilateral
XA	=	$\|\vec{S}_1 \times \vec{S}_2\|$ = area of rectangle or twice area of triangle (PA53)
XN2,YN2,ZN2	=	$\vec{S}_3 \times \vec{S}_4$ at PA79 to RASL. Line use eneeke that the four corners are coplanar by the test $(\vec{S}_1 \times \vec{S}_2) \cdot (\vec{S}_3 \times \vec{S}4)/\|\vec{S}_1 \times \vec{S}_2\|\|\vec{S}_3 \times \vec{S}_4\| > 0.998$
XNV,YNV,ZNV	=	unit vector normal ta the patch at PA54 to PASS
XS,YS,ZS	=	patch center at PA151 to PA153
XST	=	$\|\vec{S}_1 \times \vec{S}_2\|$ at PA57
0.9998	$\approx$	cos(1.0^o) in test for planar patch

```
      SUBROUTINE PATCH(NX,NY,X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4)
C     PATCH GENERATES AND MODIFIES PATCH GEOMETRY DATA
      COMMON  /DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X( NM), Y( NM),
     *Z( NM), SI( NM), BI( NM), ALP( NM), BET( NM), ICON1( N2M), ICON2(
     * N2M), ITAG( N2M), ICONX( NM), WLAM, IPSYM
      COMMON  /ANGL/ SALP( NM)
      DIMENSION  T1X(1), T1Y(1), T1Z(1), T2X(1), T2Y(1), T2Z(1)
C     NEW PATCHES.  FOR NX=0, NY=1,2,3,4 PATCH IS (RESPECTIVELY)
C     ARBITRARY, RECTAGULAR, TRIANGULAR, OR QUADRILATERAL.
C     FOR NX AND NY .GT. 0 A RECTANGULAR SURFACE IS PRODUCED WITH
C     NX BY NY RECTANGULAR PATCHES.
      EQUIVALENCE(T1X,SI),(T1Y,ALP),(T1Z,BET),(T2X,ICON1),(T2Y,ICON2),(
     *T2Z,ITAG)
      M= M+1
      MI= LD+1- M
      NTP= NY
      IF(NX.GT.0) NTP=2
      IF(NTP.GT.1) GOTO 2
      X( MI)= X1
      Y( MI)= Y1
      Z( MI)= Z1
      BI( MI)= Z2
      ZNV= COS( X2)
      XNV= ZNV* COS( Y2)
      YNV= ZNV* SIN( Y2)
      ZNV= SIN( X2)
      XA= SQRT( XNV* XNV+ YNV* YNV)
      IF(XA.LT.1.D-6) GOTO 1
      T1X( MI)=- YNV/ XA
      T1Y( MI)= XNV/ XA
      T1Z( MI)=0.
      GOTO 6
    1 T1X( MI)=1.
      T1Y( MI)=0.
      T1Z( MI)=0.
      GOTO 6
    2 S1X= X2- X1
      S1Y= Y2- Y1
      S1Z= Z2- Z1
      S2X= X3- X2
      S2Y= Y3- Y2
      S2Z= Z3- Z2
      IF(NX.EQ.0) GOTO 3
      S1X= S1X/ NX
      S1Y= S1Y/ NX
      S1Z= S1Z/ NX
      S2X= S2X/ NY
      S2Y= S2Y/ NY
      S2Z= S2Z/ NY
```

```
3 XNV= S1Y* S2Z- S1Z* S2Y
  YNV= S1Z* S2X- S1X* S2Z
  ZNV= S1X* S2Y- S1Y* S2X
  XA= SQRT( XNV* XNV+ YNV* YNV+ ZNV* ZNV)
  XNV= XNV/ XA
  YNV= YNV/ XA
  ZNV= ZNV/ XA
  XST= SQRT( S1X* S1X+ S1Y* S1Y+ S1Z* S1Z)
  T1X( MI)= S1X/ XST
  T1Y( MI)= S1Y/ XST
  T1Z( MI)= S1Z/ XST
  IF(NTP.GT.2) GOTO 4
  X( MI)= X1+.5*( S1X+ S2X)
  Y( MI)= Y1+.5*( S1Y+ S2Y)
  Z( MI)= Z1+.5*( S1Z+ S2Z)
  BI( MI)= XA
  GOTO 6
4 IF(NTP.EQ.4) GOTO 5
  X( MI)=( X1+ X2+ X3)/3.
  Y( MI)=( Y1+ Y2+ Y3)/3.
  Z( MI)=( Z1+ Z2+ Z3)/3.
  BI( MI)=.5* XA
  GOTO 6
5 S1X= X3- X1
  S1Y= Y3- Y1
  S1Z= Z3- Z1
  S2X= X4- X1
  S2Y= Y4- Y1
  S2Z= Z4- Z1
  XN2= S1Y* S2Z- S1Z* S2Y
  YN2= S1Z* S2X- S1X* S2Z
  ZN2= S1X* S2Y- S1Y* S2X
  XST= SQRT( XN2* XN2+ YN2* YN2+ ZN2* ZN2)
  SALPN=1./(3.*( XA+ XST))
  X( MI)=( XA*( X1+ X2+ X3)+ XST*( X1+ X3+ X4))* SALPN
  Y( MI)=( XA*( Y1+ Y2+ Y3)+ XST*( Y1+ Y3+ Y4))* SALPN
  Z( MI)=( XA*( Z1+ Z2+ Z3)+ XST*( Z1+ Z3+ Z4))* SALPN
  BI( MI)=.5*( XA+ XST)
  S1X=( XNV* XN2+ YNV* YN2+ ZNV* ZN2)/ XST
  IF(S1X.GT.0.9998) GOTO 6
  WRITE (2,14)
  STOP
6 T2X( MI)= YNV* T1Z( MI)- ZNV* T1Y( MI)
  T2Y( MI)= ZNV* T1X( MI)- XNV* T1Z( MI)
  T2Z( MI)= XNV* T1Y( MI)- YNV* T1X( MI)
  SALP( MI)=1.
  IF(NX.EQ.0) GOTO 8
  M= M+ NX* NY-1
  XN2= X( MI)- S1X- S2X
```

```
      YN2= Y( MI)- S1Y- S2Y
      ZN2= Z( MI)- S1Z- S2Z
      XS= T1X( MI)
      YS= T1Y( MI)
      ZS= T1Z( MI)
      XT= T2X( MI)
      YT= T2Y( MI)
      ZT= T2Z( MI)
      MI= MI+1
      DO 7  IY=1, NY
      XN2= XN2+ S2X
      YN2= YN2+ S2Y
      ZN2= ZN2+ S2Z
      DO 7  IX=1, NX
      XST= IX
      MI= MI-1
      X( MI)= XN2+ XST* S1X
      Y( MI)= YN2+ XST* S1Y
      Z( MI)= ZN2+ XST* S1Z
      BI( MI)= XA
      SALP( MI)=1.
      T1X( MI)= XS
      T1Y( MI)= YS
      T1Z( MI)= ZS
      T2X( MI)= XT
      T2Y( MI)= YT
    7 T2Z( MI)= ZT
    8 IPSYM=0
      NP= N
      MP= M
C     DIVIDE PATCH FOR WIRE CONNECTION
      RETURN
      ENTRY SUBPH( NX, NY, X1, Y1, Z1, X2, Y2, Z2, X3, Y3, Z3, X4, Y4,
     *Z4)
      IF(NY.GT.0) GOTO 10
      IF(NX.EQ. M) GOTO 10
      NXP= NX+1
      IX= LD- M
      DO 9  IY= NXP, M
      IX= IX+1
      NYP= IX-3
      X( NYP)= X( IX)
      Y( NYP)= Y( IX)
      Z( NYP)= Z( IX)
      BI( NYP)= BI( IX)
      SALP( NYP)= SALP( IX)
      T1X( NYP)= T1X( IX)
      T1Y( NYP)= T1Y( IX)
      T1Z( NYP)= T1Z( IX)
```

```
      T2X( NYP)= T2X( IX)
      T2Y( NYP)= T2Y( IX)
    9 T2Z( NYP)= T2Z( IX)
   10 MI= LD+1- NX
      XS= X( MI)
      YS= Y( MI)
      ZS= Z( MI)
      XA= BI( MI)*.25
      XST= SQRT( XA)*.5
      S1X= T1X( MI)
      S1Y= T1Y( MI)
      S1Z= T1Z( MI)
      S2X= T2X( MI)
      S2Y= T2Y( MI)
      S2Z= T2Z( MI)
      SALN= SALP( MI)
      XT= XST
      YT= XST
      IF(NY.GT.0) GOTO 11
      MIA= MI
      GOTO 12
   11 M= M+1
      MP= MP+1
      MIA= LD+1- M
   12 DO 13  IX=1,4
      X( MIA)= XS+ XT* S1X+ YT* S2X
      Y( MIA)= YS+ XT* S1Y+ YT* S2Y
      Z( MIA)= ZS+ XT* S1Z+ YT* S2Z
      BI( MIA)= XA
      T1X( MIA)= S1X
      T1Y( MIA)= S1Y
      T1Z( MIA)= S1Z
      T2X( MIA)= S2X
      T2Y( MIA)= S2Y
      T2Z( MIA)= S2Z
      SALP( MIA)= SALN
      IF(IX.EQ.2) YT=- YT
      IF(IX.EQ.1.OR. IX.EQ.3) XT=- XT
      MIA= MIA-1
   13 CONTINUE
      M= M+3
      IF(NX.LE. MP) MP= MP+3
      IF(NY.GT.0) Z( MI)=10000.
C
      RETURN
   14 FORMAT(' ERROR -- CORNERS OF QUADRILATERAL PATCH DO NOT LIE IN ',
     *'A PLANE')
      END
```

PCINT

PURPOSE

To compute the interacrion matrix elements representing the electric field, tangent to a segment connected to a surface, due to the current on the four patches around the connection point.

METHOD

The four patches at the base of a connected wire are located as shown in figure 10 with respect to the vectors $\hat{t}_1$ and $\hat{t}_2$, where patch numbers indicate the order of the patches in the data arrays. The position of a point on the surface is defined by $\vec{\rho}(S_1, S_2) = \vec{\rho}_0 + S_1\hat{t}_1 + S_2\hat{t}_2$, where $\vec{\rho}_0$ is the position of the center of the four patches where the wire connects, and S_1 and S_2 are coordinates measured from the center. The current over the surface is represented by $\vec{J}(S_1, S_2)$, the currents at the centers of the four patches are

$\vec{J}_1 = \vec{J}(d, d)$
$\vec{J}_2 = \vec{J}(-d, d)$
$\vec{J}_3 = \vec{J}(-d, -d)$
$\vec{J}_4 = \vec{J}(d, -d)$

and the current at the base of the segment, flowing onto the surface, is I_0. The current interpolation function is then

$$\vec{J}(S_1, S_2) = \left[\vec{f}(S_1, S_2) - \sum_{i=1}^{4} g_i(S_1, S_2)\vec{f}_i\right] I_0 + \sum_{i=1}^{4} g_i(S_1, S_2)\vec{J}_i \ ,$$

where

$$\vec{f}(S_1, S_2) = \frac{S_1\hat{t}_1 + S_2\hat{t}_2}{2\pi(S_1^2 + S_2^2)}$$

$\vec{f}_1 = \vec{f}(d, d) = (\hat{t}_1 + \hat{t}_2)/(4\pi d)$
$\vec{f}_2 = \vec{f}(-d, d) = (-\hat{t}_1 + \hat{t}_2)/(4\pi d)$
$\vec{f}_3 = \vec{f}(-d, -d) = (-\hat{t}_1 + -\hat{t}_2)/(4\pi d)$
$\vec{f}_4 = \vec{f}(d, -d) = (\hat{t}_1 + -\hat{t}_2)/(4\pi d)$

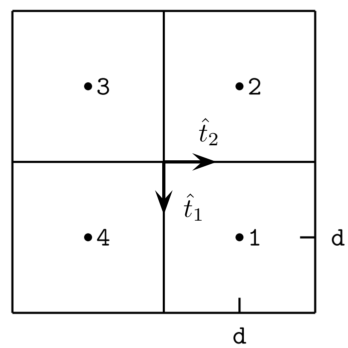


Figure 10. Patches at a Wire Connection Point.

$g_1(S_1, S_2) = (d + S_1)(d + S_2)/(4d^2)$
$g_2(S_1, S_2) = (d - S_1)(d + S_2)/(4d^2)$
$g_3(S_1, S_2) = (d - S_1)(d - S_2)/(4d^2)$
$g_4(S_1, S_2) = (d + S_1)(d - S_2)/(4d^2)$

If $\vec{\Gamma}_1(\vec{\rho})dA$ and $\vec{\Gamma}_2(\vec{\rho})dA$ are the electric fields at the center of the connected segment due to unit currents at $\vec{\rho}$ on the surface dA, flowing in the directions $\hat{t}_1$ and $\hat{t}_2$ respectively, the nine matrix elements to be computed are

$$E_1 = \int_S g_1(S_1, S_2)\ \ \hat{i} \cdot \vec{\Gamma}_1(\vec{\rho})\, dA$$

$$E_2 = \int_S g_2(S_1, S_2)\ \ \hat{i} \cdot \vec{\Gamma}_1(\vec{\rho})\, dA$$

$$E_3 = \int_S g_3(S_1, S_2)\ \ \hat{i} \cdot \vec{\Gamma}_1(\vec{\rho})\, dA$$

$$E_4 = \int_S g_4(S_1, S_2)\ \ \hat{i} \cdot \vec{\Gamma}_1(\vec{\rho})\, dA$$

$$E_5 = \int_S g_1(S_1, S_2)\ \ \hat{i} \cdot \vec{\Gamma}_2(\vec{\rho})\, dA$$

$$E_6 = \int_S g_2(S_1, S_2)\ \ \hat{i} \cdot \vec{\Gamma}_2(\vec{\rho})\, dA$$

$$E_7 = \int_S g_3(S_1, S_2)\ \ \hat{i} \cdot \vec{\Gamma}_2(\vec{\rho})\, dA$$

$$E_8 = \int_S g_4(S_1, S_2)\ \ \hat{i} \cdot \vec{\Gamma}_2(\vec{\rho})\, dA$$

$$E_9 = \int_S \left\{ \left[\vec{h}(S_1, S_2) \cdot \hat{t}_1\right] \left[\hat{i} \cdot \vec{\Gamma}_1(\vec{\rho})\right] + \left[\vec{h}(S_1, S_2) \cdot \hat{t}_2\right] \left[\hat{i} \cdot \vec{\Gamma}_2(\vec{\rho})\right] \right\}\ dA$$

where

$$\vec{h}(S_1, S_2) = \vec{\Gamma}(S_1, S_2) - \sum_{i=1}^{4} g_i(S_1, S_2)\vec{f}_i \ ,$$

and where $\hat{i}$ = the unit vector in the direction of the connected segment.

The integration is over the total area of the four patches and is performed by numerical quadrature. The number of increments in S_1 and S_2 used in integration is set by the variable NINT. When PCINT is called, the parameters in COMMON/DATAJ/ have the values for the first connected patch. During integration, these parameters are set for each integration patch. At the end of PCINT, they are reset to their original values.

SYMBOL DICTIONARY

CABI	=	x component of $\hat{i}$
D	=	d
DA	=	area of the surface element used in integration
DS	=	width of the surface element of area DA
E	=	array used to return the values E_1, E_2, ..., E_9
EXK,EYK,EZK	=	x, y, and z components of $\vec{\Gamma}_1(\vec{\rho})$DA at PC30; at PC51, EXK is set to $\hat{i} \cdot \vec{\Gamma}_1(\vec{\rho})DA$
EXS,EYS,EZS	=	x, y, and z components of $\vec{\Gamma}_2(\vec{\rho})$DA at PC30; at PC51, EXS is set to $\hat{i} \cdot \vec{\Gamma}_2(\vec{\rho})DA$
E1	=	E_1
E2	=	E_2
E3	=	E_3
E4	=	E_4
E5	=	E_5
E6	=	E_6
E7	=	E_7
E8	=	E_8
E9	=	E_9
FCON	=	$1/(4\pi d)$ factor in $\vec{f}_1$, $\vec{f}_2$, ...
F1	=	$\vec{h}(S_1, S_2) \cdot \hat{t}_1$
F2	=	$\vec{h}(S_1, S_2) \cdot \hat{t}_2$
GCON	=	$1/(4d^2)$ factor in $g_1(S_1, S_2)$, ...
G1	=	$g_1(S_1, S_2)$
G2	=	$g_2(S_1, S_2)$
G3	=	$g_3(S_1, S_2)$
G4	=	$g_4(S_1, S_2)$
I1	=	DO loop index
I2	=	DO loop index
NINT	=	number of steps in S_1 and S_3 used in approximating the integrals for E_1, E_2, ..., E_9
S	=	area of each of the four patches at PC11; area of the surface element used in integration at PC20
SABI	=	y component of $\hat{i}$
SALPI	=	z compenent of $\hat{i}$

S1	=	S_1
S2	=	S_2
S2X	=	initial value of S_2
TPI	=	2π
T1XJ,T1YJ,T1ZJ	=	x, y, and z components of $\hat{t}_1$
T2XJ,T2YJ,T2ZJ	=	x, y, and x uanmrmcnts of $\hat{t}_2$
X1	=	x coordinate of the center of the connected segment
XJ,YJ,ZJ	=	center of first patch abave PC41; center of integration element below PC4l
XS	=	x component of $\vec{\rho}(S_l, S_2)$
XSS	=	initial x coordinate of $\vec{\rho}(S_l, S_2)$
XXJ,XYJ,XZJ	=	initial value of XJ, YJ, ZJ saved
Xl	=	x component of $\vec{\rho}$(d,d) used as reference for computing $\vec{\rho}(S_1, S_2)$
YI	=	y coordinate of the center of the connected segment
YS	=	y component of $\vec{\rho}(S_1, S_2)$
YSS	=	initial y component of $\vec{\rho}(S_1, S_2)$
Y1	=	y component of $\vec{\rho}$(d,d)
ZI	=	z coordinate of the center at the connected segment
ZS	=	z component of $\vec{\rho}(S_1, S_2)$
ZSS	=	initial z component of $\vec{\rho}(S_1, S_2)$
Zl	=	z component of $\vec{\rho}$(d,d)

```
      SUBROUTINE PCINT( XI, YI, ZI, CABI, SABI, SALPI, E)
C     INTEGRATE OVER PATCHES AT WIRE CONNECTION POINT
      COMPLEX  EXK, EYK, EZK, EXS, EYS, EZS, EXC, EYC, EZC, E, E1,
     *E2, E3, E4, E5, E6, E7, E8, E9
      COMMON  /DATAJ/ S, B, XJ, YJ, ZJ, CABJ, SABJ, SALPJ, EXK, EYK,
     *EZK, EXS, EYS, EZS, EXC, EYC, EZC, RKH, IEXK, IND1, INDD1, IND2,
     *INDD2, PGND
      DIMENSION  E(9)
      EQUIVALENCE(T1XJ,CABJ),(T1YJ,SABJ),(T1ZJ,SALPJ),(T2XJ,B),(T2YJ,
     *IND1),(T2ZJ,IND2)
      DATA   TPI/6.283185308D+0/, NINT/10/
      D= SQRT( S)*.5
      DS=4.* D/ DFLOAT( NINT)
      DA= DS* DS
      GCON=1./ S
      FCON=1./(2.* TPI* D)
      XXJ= XJ
      XYJ= YJ
      XZJ= ZJ
      XS= S
      S= DA
      S1= D+ DS*.5
      XSS= XJ+ S1*( T1XJ+ T2XJ)
      YSS= YJ+ S1*( T1YJ+ T2YJ)
      ZSS= ZJ+ S1*( T1ZJ+ T2ZJ)
      S1= S1+ D
      S2X= S1
      E1=(0.,0.)
      E2=(0.,0.)
      E3=(0.,0.)
      E4=(0.,0.)
      E5=(0.,0.)
      E6=(0.,0.)
      E7=(0.,0.)
      E8=(0.,0.)
      E9=(0.,0.)
      DO 1  I1=1, NINT
      S1= S1- DS
      S2= S2X
      XSS= XSS- DS* T1XJ
      YSS= YSS- DS* T1YJ
      ZSS= ZSS- DS* T1ZJ
      XJ= XSS
      YJ= YSS
      ZJ= ZSS
      DO 1  I2=1, NINT
      S2= S2- DS
      XJ= XJ- DS* T2XJ
      YJ= YJ- DS* T2YJ
```

```
      ZJ= ZJ- DS* T2ZJ
      CALL UNERE( XI, YI, ZI)
      EXK= EXK* CABI+ EYK* SABI+ EZK* SALPI
      EXS= EXS* CABI+ EYS* SABI+ EZS* SALPI
      G1=( D+ S1)*( D+ S2)* GCON
      G2=( D- S1)*( D+ S2)* GCON
      G3=( D- S1)*( D- S2)* GCON
      G4=( D+ S1)*( D- S2)* GCON
      F2=( S1* S1+ S2* S2)* TPI
      F1= S1/ F2-( G1- G2- G3+ G4)* FCON
      F2= S2/ F2-( G1+ G2- G3- G4)* FCON
      E1= E1+ EXK* G1
      E2= E2+ EXK* G2
      E3= E3+ EXK* G3
      E4= E4+ EXK* G4
      E5= E5+ EXS* G1
      E6= E6+ EXS* G2
      E7= E7+ EXS* G3
      E8= E8+ EXS* G4
    1 E9= E9+ EXK* F1+ EXS* F2
      E(1)= E1
      E(2)= E2
      E(3)= E3
      E(4)= E4
      E(5)= E5
      E(6)= E6
      E(7)= E7
      E(8)= E8
      E(9)= E9
      XJ= XXJ
      YJ= XYJ
      ZJ= XZJ
      S= XS
      RETURN
      END
```

PRNT

PURPOSE

To set up the formats for printing a record of three integers, six floating point numbers, and a Hollerith string, where the variables equal to zero are replaced by blanks. This routine is used by LOAD in printing the impedance data table.

METHOD

A variable format is used to generate the record with arbitrary blank fill. Elements of the format are picked from the array IFORM in the DATA statement. Through IF statements operating on the subroutine input quantities, this routine chooses the desired format elements and builds the format in the array IVAR. The program is divided into two sections: the first builds the integer part of the format and the second the floating point part.

SYMBOL DICTIONARY

ABS	=	external routine (absolute value)
FL	=	elements of this array are set equal to the floating point input quantities FL1 - FL6
FLT	=	array of non-zero floating point input quantities to be printed
FL1		
FL2		
FL3	=	input floating point quantities
FL4		
FL5		
FL6		
HALL	=	4H ALL (Hollerith ALL)
I	=	DO loop index
IA	=	input Hollerith string (array)
ICHAR	=	number of characters in the input Hollerith string
IFORM	=	array containing format elements
IN	=	array set equal to input integer quantities (IN1 - IN3)
INT	=	non-zero integer quantities to be printed
IN1		
IN2	=	I input integer quantities
IN3		
IVAR	=	variable format array
I1	=	DO loop limit
J	=	implied DO loop index
K	=	index parameter
L	=	implied DO loop index
NCPW	=	number of Hollerith characters per computer word
NFLT	=	floating point print index, number of non-zero reals
NINT	=	integer print index; number of non-zero integers
NWORDS	=	number of computer words in the input Hollerith string

```
      SUBROUTINE PRNT(IN1,IN2,IN3,FL1,FL2,FL3,FL4,FL5,FL6,IA,ICHAR)
C
C     PRNT SETS UP THE PRINT FORMATS FOR IMPEDANCE LOADING
C
      CHARACTER*6 IFORM, IVAR
      CHARACTER *(*) IA
      DIMENSION  IVAR(13), IA(1), IFORM(8), IN(3), INT(3), FL(6), FLT(6
     *)
      INTEGER  HALL
C
C     NUMBER OF CHARACTERS PER COMPUTER WORD IS NCPW
C
      DATA   IFORM/5H(/3X,,3HI5,,3H5X,,3HA5,,6HE13.4,,4H13X,,3H3X,,
     *4H5A4)/
      DATA   HALL/4H ALL/
      IN(1)= IN1
      IN(2)= IN2
      IN(3)= IN3
      FL(1)= FL1
      FL(2)= FL2
      FL(3)= FL3
      FL(4)= FL4
      FL(5)= FL5
C
C     INTEGER FORMAT
C
      FL(6)= FL6
      NINT=0
      IVAR(1)= IFORM(1)
      K=1
      I1=1
      IF(.NOT.( IN1.EQ.0.AND. IN2.EQ.0.AND. IN3.EQ.0)) GOTO 1
      INT(1)= HALL
      NINT=1
      I1=2
      K= K+1
      IVAR( K)= IFORM(4)
    1 DO 3  I= I1,3
      K= K+1
      IF(IN( I).EQ.0) GOTO 2
      NINT= NINT+1
      INT( NINT)= IN( I)
      IVAR( K)= IFORM(2)
      GOTO 3
    2 IVAR( K)= IFORM(3)
    3 CONTINUE
      K= K+1
C
C     DFLOATING POINT FORMAT
```

```
C
      IVAR( K)= IFORM(7)
      NFLT=0
      DO 5  I=1,6
      K= K+1
      IF(ABS( FL( I)).LT.1.D-20) GOTO 4
      NFLT= NFLT+1
      FLT( NFLT)= FL( I)
      IVAR( K)= IFORM(5)
      GOTO 5
    4 IVAR( K)= IFORM(6)
    5 CONTINUE
      K= K+1
      IVAR( K)= IFORM(7)
      K= K+1
      IVAR( K)= IFORM(8)
      WRITE (2,IVAR) ( INT( I), I=1, NINT),( FLT( J), J=1, NFLT),
     *      ( IA( L), L=1, ICHAR)
      RETURN
      END
```

QDSRC

PURPOSE

To fill the excitation array for a current slope discontinuity voltage source.

METHOD

The current slope discontinuity voltage source is described in section IV-1 of Part I.

CODING

QD22-QD25	The connection number for end 1 of segment IS is temporarily set to 0, and TBF is called to generate the function $f_\ell^*(s)$ for ℓ - IS. The zero in the second argument of TBF causes f_ℓ^* to go to zero at the first end of segment IS rather than the usual non-zero value that allows for current flowing onto the wire end cap.
QD26-QD31	β_ℓ is computed and other quantities set.
QD32-QD119	This loop computes the fields due to each segment on which f_ℓ^* is non-zero.
QD33-QD77	Parameters of the source segment are stored in COMMON/DATAJ/. Flags for the extended thin wire approximation are set as in routine CMSET.
QD75-QD91	This loop evaluates the electric field on each segment.
QD95-QD116	This loop evaluates the magnetic field at each patch.

SYMBOL DICTIONARY

AI	=	radius of segment on which field is evaluated.
CABI	=	x component of unit vector in the direction of segment I
CCJ	=	CCJX = -j/60
CURD	=	β_ℓ
E	=	array of segment and patch excitation fields
ETC ETK ETS	=	E field tangent to a segment or H field components on a patch due to cosine, constant, and sine current components, respectively, on a segment
I1	=	array index for patch excitation
IJ	=	flag which, if zero, indicates that the Eield is being evaluated an the source segment.
IPR	=	temporary storage of connection number
IS	=	segment which has the source location on end 1
J	=	source segment number
SABI	=	y component of unit vector in the direction of segment I
T1X,T1Y,T1Z	=	arrays of components of $\hat{t}_1$ for patches
T2X,T2Y,T2Z	=	arrays of components of $\hat{t}_2$ for patches
TP	=	2π
TX,TY,TZ	=	components of $\hat{t}_1$ or $\hat{t}_2$ for patches
V	=	source voltage
XI YI ZI	=	coordinates of point: where field is evaluated; XI is also used in the test for the extended thin wire approximation for the electric field

```
      SUBROUTINE QDSRC( IS, V, E)
C     FILL INCIDENT FIELD ARRAY FOR CHARGE DISCONTINUITY VOLTAGE SOURCE
      COMPLEX  VQDS, CURD, CCJ, V, EXK, EYK, EZK, EXS, EYS, EZS, EXC
     *, EYC, EZC, ETK, ETS, ETC, VSANT, VQD, E, ZARRAY
      COMMON  /DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X( NM), Y( NM),
     *Z( NM), SI( NM), BI( NM), ALP( NM), BET( NM), ICON1( N2M), ICON2(
     * N2M), ITAG( N2M), ICONX( NM), WLAM, IPSYM
      COMMON  /VSORC/ VQD(30), VSANT(30), VQDS(30), IVQD(30), ISANT(30)
     *, IQDS(30), NVQD, NSANT, NQDS
      COMMON  /SEGJ/ AX(30), BX(30), CX(30), JCO(30), JSNO, ISCON(50),
     *NSCON, IPCON(10), NPCON
      COMMON  /DATAJ/ S, B, XJ, YJ, ZJ, CABJ, SABJ, SALPJ, EXK, EYK,
     *EZK, EXS, EYS, EZS, EXC, EYC, EZC, RKH, IEXK, IND1, INDD1, IND2,
     *INDD2, IPGND
      COMMON  /ANGL/ SALP( NM)
      COMMON  /ZLOAD/ ZARRAY( NM), NLOAD, NLODF
      DIMENSION  CCJX(2), E(1), CAB(1), SAB(1)
      DIMENSION  T1X(1), T1Y(1), T1Z(1), T2X(1), T2Y(1), T2Z(1)
      EQUIVALENCE(CCJ,CCJX),(CAB,ALP),(SAB,BET)
      EQUIVALENCE(T1X,SI),(T1Y,ALP),(T1Z,BET),(T2X,ICON1),(T2Y,ICON2),(
     *T2Z,ITAG)
      DATA   TP/6.283185308D+0/, CCJX/0.,-.01666666667D+0/
      I= ICON1( IS)
      ICON1( IS)=0
      CALL TBF( IS,0)
      ICON1( IS)= I
      S= SI( IS)*.5
      CURD= CCJ* V/(( LOG(2.* S/ BI( IS))-1.)*( BX( JSNO)* COS( TP* S)+
     * CX( JSNO)* SIN( TP* S))* WLAM)
      NQDS= NQDS+1
      VQDS( NQDS)= V
      IQDS( NQDS)= IS
      DO 20  JX=1, JSNO
      J= JCO( JX)
      S= SI( J)
      B= BI( J)
      XJ= X( J)
      YJ= Y( J)
      ZJ= Z( J)
      CABJ= CAB( J)
      SABJ= SAB( J)
      SALPJ= SALP( J)
      IF(IEXK.EQ.0) GOTO 16
      IPR= ICON1( J)
      IF(IPR) 1,6,2
    1 IPR=- IPR
      IF(- ICON1( IPR).NE. J) GOTO 7
      GOTO 4
    2 IF(IPR.NE. J) GOTO 3
```

```
      IF(CABJ* CABJ+ SABJ* SABJ.GT.1.D-8) GOTO 7
      GOTO 5
    3 IF(ICON2( IPR).NE. J) GOTO 7
    4 XI= ABS( CABJ* CAB( IPR)+ SABJ* SAB( IPR)+ SALPJ* SALP( IPR))
      IF(XI.LT.0.999999D+0) GOTO 7
      IF(ABS( BI( IPR)/ B-1.).GT.1.D-6) GOTO 7
    5 IND1=0
      GOTO 8
    6 IND1=1
      GOTO 8
    7 IND1=2
    8 IPR= ICON2( J)
      IF(IPR) 9,14,10
    9 IPR=- IPR
      IF(- ICON2( IPR).NE. J) GOTO 15
      GOTO 12
   10 IF(IPR.NE. J) GOTO 11
      IF(CABJ* CABJ+ SABJ* SABJ.GT.1.D-8) GOTO 15
      GOTO 13
   11 IF(ICON1( IPR).NE. J) GOTO 15
   12 XI= ABS( CABJ* CAB( IPR)+ SABJ* SAB( IPR)+ SALPJ* SALP( IPR))
      IF(XI.LT.0.999999D+0) GOTO 15
      IF(ABS( BI( IPR)/ B-1.).GT.1.D-6) GOTO 15
   13 IND2=0
      GOTO 16
   14 IND2=1
      GOTO 16
   15 IND2=2
   16 CONTINUE
      DO 17  I=1, N
      IJ= I- J
      XI= X( I)
      YI= Y( I)
      ZI= Z( I)
      AI= BI( I)
      CALL EFLD( XI, YI, ZI, AI, IJ)
      CABI= CAB( I)
      SABI= SAB( I)
      SALPI= SALP( I)
      ETK= EXK* CABI+ EYK* SABI+ EZK* SALPI
      ETS= EXS* CABI+ EYS* SABI+ EZS* SALPI
      ETC= EXC* CABI+ EYC* SABI+ EZC* SALPI
   17 E( I)= E( I)-( ETK* AX( JX)+ ETS* BX( JX)+ ETC* CX( JX))* CURD
      IF(M.EQ.0) GOTO 19
      IJ= LD+1
      I1= N
      DO 18  I=1, M
      IJ= IJ-1
      XI= X( IJ)
```

```
      YI= Y( IJ)
      ZI= Z( IJ)
      CALL HSFLD( XI, YI, ZI,0.)
      I1= I1+1
      TX= T2X( IJ)
      TY= T2Y( IJ)
      TZ= T2Z( IJ)
      ETK= EXK* TX+ EYK* TY+ EZK* TZ
      ETS= EXS* TX+ EYS* TY+ EZS* TZ
      ETC= EXC* TX+ EYC* TY+ EZC* TZ
      E( I1)= E( I1)+( ETK* AX( JX)+ ETS* BX( JX)+ ETC* CX( JX))* CURD*
     * SALP( IJ)
      I1= I1+1
      TX= T1X( IJ)
      TY= T1Y( IJ)
      TZ= T1Z( IJ)
      ETK= EXK* TX+ EYK* TY+ EZK* TZ
      ETS= EXS* TX+ EYS* TY+ EZS* TZ
      ETC= EXC* TX+ EYC* TY+ EZC* TZ
   18 E( I1)= E( I1)+( ETK* AX( JX)+ ETS* BX( JX)+ ETC* CX( JX))* CURD*
     * SALP( IJ)
   19 IF(NLOAD.GT.0.OR. NLODF.GT.0) E( J)= E( J)+ ZARRAY( J)* CURD*(
     *AX( JX)+ CX( JX))
   20 CONTINUE
      RETURN
      END
```

RDPAT

PURPOSE

To compute and print radiated field quantities.

METHOD

The quantities computed and the output formats depend on the options selected by the first integer (IFAR) and fourth integer (IPD, IAVP, INOR, IAX) on the RP card (see Part III). These quantities are defined as follows:

(1) Power Gain

In the direction (θ, Φ)

$$G_p(\theta,\Phi) = 4\pi \frac{P_\Omega(\theta,\Phi)}{P_{in}} \quad ,$$

where $P_\Omega(\theta,\Phi)$ is the power radiated per unit solid angle in the given direction, and P_{in} is the total power accepted by the antenna. Therefore, $P_{in} = (1/2)Re(VI*)$, where V is the applied source voltage, and

$$P_\Omega(\theta,\Phi) = (1/2)R^2 Re(\vec{E}\times\vec{H}^*) = \frac{R^2}{2\eta}\vec{E}\cdot\vec{E}^* \quad ,$$

where R is Lhe observation sphere radius. Since the electric field calculated by FFLD (call it $\vec{E}'$) does not include exp=(-jkR)/(R/λ),

$$\vec{E} = \frac{\exp(-jkR)}{R/\lambda}\,\vec{E}'$$

and

$$P_\Omega = \frac{\lambda^2}{2\eta}(\vec{E}'\cdot\vec{E}'*) \quad .$$

Thus,

$$G_P(\theta,\Phi) = \frac{2\pi\lambda^2}{\eta P_{in}}(\vec{E}'\cdot\vec{E}'^*)$$

in terms of the program variables.

(2) Directive Gain

In the direction (θ, Φ),

$$G_d(\theta, \Phi) = 4\pi \frac{P_\Omega(\theta, \Phi)}{P_{rad}}$$

where P_{rad} is the total power radiated by the antenna. The only difference from power gain is that P_{in} is replaced by P_{rad}, and $P_{rad} = P_{in} - P_{loss}$ where P_{loss} is calculated as the power lost in distributed and lumped loads on the structure and in the networks loads.

(3) Component Gain

The gains are also calculated for separate, orthogonal field components (u,v). In this case, $\vec{E}' \cdot \vec{E}'^*$ is replaced by $E'_u E'^*_u$ or $E'_v E'^*_v$ and the total gain is the sum of the two components.

(4) Average Gain

The user specifies a range and number of points in theta and phi that in turn specify the total solid angle covered, Ω, and the sampling density for the integral in the expression for average gain;

$$G_{av} = \frac{\int_\Omega G_p d\Omega}{\Omega}$$

The trapezoidal rule is used in evaluating the integral.

(5) Normalized Gain

Normalized gain is simply the gain divided by its maximum value or some value specified by the user.

The discussion of gains applies only to the case of a structure used as a radiating antenna. For the case of an incident plane wave, the program constants are defined such that the value of σ/λ^2 is printed under the heading "GAIN." The calculation is

$$\frac{\sigma}{\lambda^2} = \frac{4\pi R^2}{\lambda^2} \frac{W_{scat}}{W_{inc}} = \frac{4\pi}{\vec{E}_{inc} \cdot \vec{E}^*_{inc}} (\vec{E}'_{scat} \cdot \vec{E}'^*_{scat}) \ ,$$

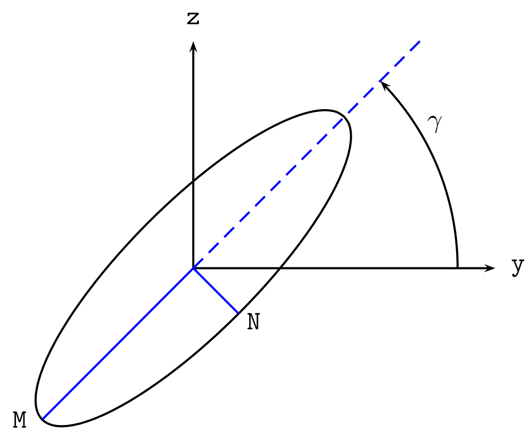
where W_{scat} is the scattered power per unit area at distance R in a given direction, W_{inc} is the power per unit area of the incident plane wave, and the primes on the electric fields specify the fields used in the program as defined above. For the case of a Hertzian dipole used as a source, the gain equations are used; however, P_{in} is equal to the total power radiated by the Hertzian source. That is

$$P_{in} = \frac{\pi\eta}{3} \left| \frac{I\ell}{\lambda} \right|^2 \ ,$$

where the quantity $I\ell$ is an input quantity.

(6) elliptic Polarization

Elliptic polarization parameters are calculated as follows:



$$M = [(E_{ym}\cos\gamma + E_{zm}\cos\xi\sin\gamma)^2 + E_{zm}^2\sin^2\xi\sin^2\gamma]^{1/2} \ ,$$

$$N = [(E_{ym}\sin\gamma - E_{zm}\cos\xi\cos\gamma)^2 + E_{zm}^2\sin^2\xi\cos^2\gamma]^{1/2} \ ,$$

where

$$E_y = E_{ym}\exp[j(\omega t - kx)] \ ,$$

$$E_z = E_{zm}\exp[j(\omega t - kx + \xi)] \ ,$$

and γ is given by

$$\tan 2y = \frac{2E_{ym}E_{zm}\cos\xi}{E_{ym}^2 - E_{zm}^2}$$

In this routine, the coordinates y and z above are replaced by θ and Φ, respectively.

The field is computed by FFLD at RD74 for space wave or by GFLD at RD76 for space and ground wave. Elliptic polarization parameters are computed from RD87 to RD118. RD127 to RD137 stores gain in the array GAIN for normalization. The integral of radiated power for the average gain calculation is summed at RD140 to RD147. Fields and gain are printed at RD162 for space wave or RD165 for ground wave. Average gain is computed and printed from RD168 to RD173. Normalized gain is printed from RD174 to RD208.

SYMBOL DICTIONARY

AXRAT	=	N/M (elliptic axial ratio)
CHT	=	height of cliff in meters
CLT	=	distance in meters of cliff edge from origin
DA	=	element of solid angle for average gain summation
DFAZ	=	phase difference between E_θ and E_Φ, for elliptic polarization
DPH	=	increment for Φ
DTH	=	increment for θ
EMAJR2	=	M^2 (M = major axis)
EMINR2	=	N^2
EPH	=	E_Φ (phi component of electric field, with or without the term exp(-jkR)/(R/λ) depending an return from GFLD or FFLD)
EPHA	=	phase angle of EPH
EPHM	=	|EPH|
EPHM2	=	$\|EPH\|^2$
EPSR	=	relative dielectric constant
EPSR2	=	relative dielectric constant of second medium
ERD	=	radial electric field for ground wave
ERDA	=	phase of ERD
ERDM	=	|ERD|
ETH	=	E_θ
ETHA	=	phase of E_θ
ETHM	=	$\|E_\theta\|$
ETHM2	=	$\|E_\theta\|^2$
EXRA	=	phase of exp(-jkR)
EXRM	=	1/R
GCON	=	factor multiplying $\|E^2\|$ to yield gain or σ/λ^2
GCOP	=	GCON except when GCON yields directive gain; then GCUP remains power gain
GMAX	=	value used for normalized gain
GNH	=	horizontal gain in decibels, Φ component
GNMJ	=	major axis gain in decibels
GNMN	=	minor axis gain in decibels
GNOR	=	if non-zero, equals input gain quantity
GNV	=	vertical gain (θ)
GTOT	=	total gain
IAVP	=	flag for average gain
IAX	=	flag for gain type
IFAR	=	first integer from RP card
INOR	=	integer to select normalized gain
IPD	=	flag to select power or directive gain
IXTYP	=	excitation type
NORMAX	=	dimension of FNORM (maximum number of gain values that will be stored for normalization)
NPH	=	number of Φ values

NTH	=	number of θ values
PHA	=	Φ in radians
PHI	=	Φ in degrees
PHIS	=	initial Φ
PI	=	π
PINR	=	input power for current element source
PINT	=	summation variable for average gain
PLOSS	=	power dissipated in structure loads
PNLR	=	power dissipated in networks and transmission lines
PRAD	=	power radiated by the antenna
RFLD	=	if non-zero, equal to the observation distance in meters
SIG	=	conductivity of ground (mhos/m)
SIG2	=	conductivity of second medium (mhos/m)
STILTA	=	sin γ, γ is tilt angle of the palarization ellipse
TA	=	$\pi/180$
TD	=	$180/\pi$
THA	=	θ in radians
THET	=	θ in degrees
TRETS	=	initial θ
TILTA	=	γ (tilt angle of ellipse)
XPR6	=	minor axis of polarization ellipse or strength of current element source

1.745329252E 2	=	$\pi/180$
1.E-20	=	small value test
1.E-5	=	small value test
3.141592654	=	π
376.73	=	$\eta_0 = \sqrt{\mu_0/\epsilon_0}$
394.51	=	$\pi\eta_0/3$
57.2957795	=	$180/\pi$
59.96	=	$\eta_0/(2\pi)$
90.01	=	test value for angle exceeding 90 degrees

```
      SUBROUTINE RDPAT
C     COMPUTE RADIATION PATTERN, GAIN, NORMALIZED GAIN
C     INTEGER HBLK,HCIR,HCLIF
      CHARACTER*6 IGNTP, IGAX, IGTP, HPOL, HCIR, HCLIF, HBLK
      CHARACTER*6 ISENS
      INTEGER*4 COM
      COMPLEX  ETH, EPH, ERD, ZRATI, ZRATI2, T1, FRATI
      COMMON  /DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X( NM), Y( NM),
     *Z( NM), SI( NM), BI( NM), ALP( NM), BET( NM), ICON1( N2M), ICON2(
     * N2M), ITAG( N2M), ICONX( NM), WLAM, IPSYM
      COMMON  /SAVE/ IP( N2M), KCOM, COM(20,5), EPSR, SIG, SCRWLT,
     *SCRWRT, FMHZ
      COMMON  /GND/ ZRATI, ZRATI2, FRATI, CL, CH, SCRWL, SCRWR, NRADL,
     *KSYMP, IFAR, IPERF, T1, T2
      COMMON  /FPAT/ NTH, NPH, IPD, IAVP, INOR, IAX, THETS, PHIS, DTH,
     *DPH, RFLD, GNOR, CLT, CHT, EPSR2, SIG2, IXTYP, XPR6, PINR, PNLR,
     *PLOSS, NEAR, NFEH, NRX, NRY, NRZ, XNR, YNR, ZNR, DXNR, DYNR, DZNR
     *

      COMMON  /SCRATM/ GAIN(2*N2M)

      COMMON  /PLOT/ IPLP1, IPLP2, IPLP3, IPLP4
      DIMENSION  IGTP(4), IGAX(4), IGNTP(10), HPOL(3)
      DATA   HPOL/6HLINEAR,5HRIGHT,4HLEFT/, HBLK, HCIR/1H ,6HCIRCLE/
      DATA   IGTP/6H    - ,6HPOWER ,6H- DIRE,6HCTIVE /
      DATA   IGAX/6H MAJOR,6H MINOR,6H VERT.,6H HOR. /
      DATA   IGNTP/6H MAJOR,6H AXIS ,6H MINOR,6H AXIS ,6H   VER,
     *6HTICAL ,6H HORIZ,6HONTAL ,6H      ,6HTOTAL /
      DATA   PI, TA, TD/3.141592654D+0,1.745329252D-02,57.29577951D+0/
      DATA   NORMAX/1200/
      IF(IFAR.LT.2) GOTO 2
      WRITE (2,35)
      IF(IFAR.LE.3) GOTO 1
      WRITE (2,36)  NRADL, SCRWLT, SCRWRT
      IF(IFAR.EQ.4) GOTO 2
    1 IF(IFAR.EQ.2.OR. IFAR.EQ.5) HCLIF= HPOL(1)
      IF(IFAR.EQ.3.OR. IFAR.EQ.6) HCLIF= HCIR
      CL= CLT/ WLAM
      CH= CHT/ WLAM
      ZRATI2= SQRT(1./ CMPLX( EPSR2,- SIG2* WLAM*59.96))
      WRITE (2,37)  HCLIF, CLT, CHT, EPSR2, SIG2
    2 IF(IFAR.NE.1) GOTO 3
      WRITE (2,41)
      GOTO 5
    3 I=2* IPD+1
      J= I+1
      ITMP1=2* IAX+1
      ITMP2= ITMP1+1
      WRITE (2,38)
```

```
      IF(RFLD.LT.1.D-20) GOTO 4
      EXRM=1./ RFLD
      EXRA= RFLD/ WLAM
      EXRA=-360.*( EXRA- AINT( EXRA))
      WRITE (2,39)  RFLD, EXRM, EXRA
    4 WRITE (2,40)  IGTP( I), IGTP( J), IGAX( ITMP1), IGAX( ITMP2)
    5 IF(IXTYP.EQ.0.OR. IXTYP.EQ.5) GOTO 7
      IF(IXTYP.EQ.4) GOTO 6
      PRAD=0.
      GCON=4.* PI/(1.+ XPR6* XPR6)
      GCOP= GCON
      GOTO 8
    6 PINR=394.51* XPR6* XPR6* WLAM* WLAM
    7 GCOP= WLAM* WLAM*2.* PI/(376.73* PINR)
      PRAD= PINR- PLOSS- PNLR
      GCON= GCOP
      IF(IPD.NE.0) GCON= GCON* PINR/ PRAD
    8 I=0
      GMAX=-1.E10
      PINT=0.
      TMP1= DPH* TA
      TMP2=.5* DTH* TA
      PHI= PHIS- DPH
      DO 29  KPH=1, NPH
      PHI= PHI+ DPH
      PHA= PHI* TA
      THET= THETS- DTH
      DO 29  KTH=1, NTH
      THET= THET+ DTH
      IF(KSYMP.EQ.2.AND. THET.GT.90.01.AND. IFAR.NE.1) GOTO 29
      THA= THET* TA
      IF(IFAR.EQ.1) GOTO 9
      CALL FFLD( THA, PHA, ETH, EPH)
      GOTO 10
    9 CALL GFLD( RFLD/ WLAM, PHA, THET/ WLAM, ETH, EPH, ERD, ZRATI,
     *KSYMP)
      ERDM= ABS( ERD)
      ERDA= CANG( ERD)
   10 ETHM2= REAL( ETH* CONJG( ETH))
      ETHM= SQRT( ETHM2)
      ETHA= CANG( ETH)
      EPHM2= REAL( EPH* CONJG( EPH))
      EPHM= SQRT( EPHM2)
      EPHA= CANG( EPH)
C     ELLIPTICAL POLARIZATION CALC.
      IF(IFAR.EQ.1) GOTO 28
      IF(ETHM2.GT.1.D-20.OR. EPHM2.GT.1.D-20) GOTO 11
      TILTA=0.
      EMAJR2=0.
```

```
      EMINR2=0.                                                         RD  99
      AXRAT=0.                                                          RD 100
      ISENS= HBLK                                                       RD 101
      GOTO 16                                                           RD 102
   11 DFAZ= EPHA- ETHA                                                  RD 103
      IF(EPHA.LT.0.) GOTO 12                                            RD 104
      DFAZ2= DFAZ-360.                                                  RD 105
      GOTO 13                                                           RD 106
   12 DFAZ2= DFAZ+360.                                                  RD 107
   13 IF(ABS( DFAZ).GT. ABS( DFAZ2)) DFAZ= DFAZ2                        RD 108
      CDFAZ= COS( DFAZ* TA)                                             RD 109
      TSTOR1= ETHM2- EPHM2                                              RD 110
      TSTOR2=2.* EPHM* ETHM* CDFAZ                                      RD 111
      TILTA=.5* ATGN2( TSTOR2, TSTOR1)                                  RD 112
      STILTA= SIN( TILTA)                                               RD 113
      TSTOR1= TSTOR1* STILTA* STILTA                                    RD 114
      TSTOR2= TSTOR2* STILTA* COS( TILTA)                               RD 115
      EMAJR2=- TSTOR1+ TSTOR2+ ETHM2                                    RD 116
      EMINR2= TSTOR1- TSTOR2+ EPHM2                                     RD 117
      IF(EMINR2.LT.0.) EMINR2=0.                                        RD 118
      AXRAT= SQRT( EMINR2/ EMAJR2)                                      RD 119
      TILTA= TILTA* TD                                                  RD 120
      IF(AXRAT.GT.1.D-5) GOTO 14                                        RD 121
      ISENS= HPOL(1)                                                    RD 122
      GOTO 16                                                           RD 123
   14 IF(DFAZ.GT.0.) GOTO 15                                            RD 124
      ISENS= HPOL(2)                                                    RD 125
      GOTO 16                                                           RD 126
   15 ISENS= HPOL(3)                                                    RD 127
   16 GNMJ= DB10( GCON* EMAJR2)                                         RD 128
      GNMN= DB10( GCON* EMINR2)                                         RD 129
      GNV= DB10( GCON* ETHM2)                                           RD 130
      GNH= DB10( GCON* EPHM2)                                           RD 131
      GTOT= DB10( GCON*( ETHM2+ EPHM2))                                 RD 132
      IF(INOR.LT.1) GOTO 23                                             RD 133
      I= I+1                                                            RD 134
      IF(I.GT. NORMAX) GOTO 23                                          RD 135
      GOTO (17,18,19,20,21), INOR                                       RD 136
   17 TSTOR1= GNMJ                                                      RD 137
      GOTO 22                                                           RD 138
   18 TSTOR1= GNMN                                                      RD 139
      GOTO 22                                                           RD 140
   19 TSTOR1= GNV                                                       RD 141
      GOTO 22                                                           RD 142
   20 TSTOR1= GNH                                                       RD 143
      GOTO 22                                                           RD 144
   21 TSTOR1= GTOT                                                      RD 145
   22 GAIN( I)= TSTOR1                                                  RD 146
      IF(TSTOR1.GT. GMAX) GMAX= TSTOR1                                  RD 147
```

```
 23 IF(IAVP.EQ.0) GOTO 24
    TSTOR1= GCOP*( ETHM2+ EPHM2)
    TMP3= THA- TMP2
    TMP4= THA+ TMP2
    IF(KTH.EQ.1) TMP3= THA
    IF(KTH.EQ. NTH) TMP4= THA
    DA= ABS( TMP1*( COS( TMP3)- COS( TMP4)))
    IF(KPH.EQ.1.OR. KPH.EQ. NPH) DA=.5* DA
    PINT= PINT+ TSTOR1* DA
    IF(IAVP.EQ.2) GOTO 29
 24 IF(IAX.EQ.1) GOTO 25
    TMP5= GNMJ
    TMP6= GNMN
    GOTO 26
 25 TMP5= GNV
    TMP6= GNH
 26 ETHM= ETHM* WLAM
    EPHM= EPHM* WLAM
    IF(RFLD.LT.1.D-20) GOTO 27
    ETHM= ETHM* EXRM
    ETHA= ETHA+ EXRA
    EPHM= EPHM* EXRM
    EPHA= EPHA+ EXRA

 27 WRITE (2,42)  THET, PHI, TMP5, TMP6, GTOT, AXRAT, TILTA, ISENS,
   *              ETHM, ETHA, EPHM, EPHA
    IF(IPLP1.NE.3) GOTO 299
    IF(IPLP3.EQ.0) GOTO 290
    IF(IPLP2.EQ.1.AND. IPLP3.EQ.1) WRITE( 8,*)  THET, ETHM, ETHA
    IF(IPLP2.EQ.1.AND. IPLP3.EQ.2) WRITE( 8,*)  THET, EPHM, EPHA
    IF(IPLP2.EQ.2.AND. IPLP3.EQ.1) WRITE( 8,*)  PHI, ETHM, ETHA
    IF(IPLP2.EQ.2.AND. IPLP3.EQ.2) WRITE( 8,*)  PHI, EPHM, EPHA
    IF(IPLP4.EQ.0) GOTO 299
290 IF(IPLP2.EQ.1.AND. IPLP4.EQ.1) WRITE( 8,*)  THET, TMP5
    IF(IPLP2.EQ.1.AND. IPLP4.EQ.2) WRITE( 8,*)  THET, TMP6
    IF(IPLP2.EQ.1.AND. IPLP4.EQ.3) WRITE( 8,*)  THET, GTOT
    IF(IPLP2.EQ.2.AND. IPLP4.EQ.1) WRITE( 8,*)  PHI, TMP5
    IF(IPLP2.EQ.2.AND. IPLP4.EQ.2) WRITE( 8,*)  PHI, TMP6
    IF(IPLP2.EQ.2.AND. IPLP4.EQ.3) WRITE( 8,*)  PHI, GTOT
    GOTO 299
 28 WRITE (2,43)  RFLD, PHI, THET, ETHM, ETHA, EPHM, EPHA, ERDM, ERDA
   *

299 CONTINUE
 29 CONTINUE
    IF(IAVP.EQ.0) GOTO 30
    TMP3= THETS* TA
    TMP4= TMP3+ DTH* TA* DFLOAT( NTH-1)
    TMP3= ABS( DPH* TA* DFLOAT( NPH-1)*( COS( TMP3)- COS( TMP4)))
```

```
      PINT= PINT/ TMP3
      TMP3= TMP3/ PI
      WRITE (2,44)  PINT, TMP3
   30 IF(INOR.EQ.0) GOTO 34
      IF(ABS( GNOR).GT.1.D-20) GMAX= GNOR
      ITMP1=( INOR-1)*2+1
      ITMP2= ITMP1+1
      WRITE (2,45)  IGNTP( ITMP1), IGNTP( ITMP2), GMAX
      ITMP2= NPH* NTH
      IF(ITMP2.GT. NORMAX) ITMP2= NORMAX
      ITMP1=( ITMP2+2)/3
      ITMP2= ITMP1*3- ITMP2
      ITMP3= ITMP1
      ITMP4=2* ITMP1
      IF(ITMP2.EQ.2) ITMP4= ITMP4-1
      DO 31  I=1, ITMP1
      ITMP3= ITMP3+1
      ITMP4= ITMP4+1
      J=( I-1)/ NTH
      TMP1= THETS+ DFLOAT( I- J* NTH-1)* DTH
      TMP2= PHIS+ DFLOAT( J)* DPH
      J=( ITMP3-1)/ NTH
      TMP3= THETS+ DFLOAT( ITMP3- J* NTH-1)* DTH
      TMP4= PHIS+ DFLOAT( J)* DPH
      J=( ITMP4-1)/ NTH
      TMP5= THETS+ DFLOAT( ITMP4- J* NTH-1)* DTH
      TMP6= PHIS+ DFLOAT( J)* DPH
      TSTOR1= GAIN( I)- GMAX
      IF(I.EQ. ITMP1.AND. ITMP2.NE.0) GOTO 32
      TSTOR2= GAIN( ITMP3)- GMAX
      PINT= GAIN( ITMP4)- GMAX
   31 WRITE (2,46)  TMP1, TMP2, TSTOR1, TMP3, TMP4, TSTOR2, TMP5, TMP6,
     * PINT
      GOTO 34
   32 IF(ITMP2.EQ.2) GOTO 33
      TSTOR2= GAIN( ITMP3)- GMAX
      WRITE (2,46)  TMP1, TMP2, TSTOR1, TMP3, TMP4, TSTOR2
      GOTO 34
   33 WRITE (2,46)  TMP1, TMP2, TSTOR1
C
   34 RETURN
   35 FORMAT(///,31X,'- - - FAR FIELD GROUND PARAMETERS - - -',//)
   36 FORMAT(40X,'RADIAL WIRE GROUND SCREEN',/,40X,I5,' WIRES',/,40X,
     *'WIRE LENGTH=',F8.2,' METERS',/,40X,'WIRE RADIUS=',1P,E10.3,
     *' METERS')
   37 FORMAT(40X,A6,' CLIFF',/,40X,'EDGE DISTANCE=',F9.2,' METERS',/,40
     *X,'HEIGHT=',F8.2,' METERS',/,40X,'SECOND MEDIUM -',/,40X,'RELA',
     *'TIVE DIELECTRIC CONST.=',F7.3,/,40X,'CONDUCTIVITY=',1P,E10.3,
     *' MHOS')
```

```
38 FORMAT(///,48X,'- - - RADIATION PATTERNS - - -')                       RD 246
39 FORMAT(54X,'RANGE=',1P,E13.6,' METERS',/,54X,'EXP(-JKR)/R=',E12.5  RD 247
  *,' AT PHASE',0P,F7.2,' DEGREES',/)                                  RD 248
40 FORMAT(/,2X,'- - ANGLES - -',7X,2A6,'GAINS -',7X,'- - - POLARI',   RD 249
  *'ZATION - - -',4X,'- - - E(THETA) - - -',4X,'- - - E(PHI) - -',    RD 250
  *' -',/,2X,'THETA',5X,'PHI',7X,A6,2X,A6,3X,'TOTAL',6X,'AXIAL',5X,   RD 251
  *'TILT',3X,'SENSE',2(5X,'MAGNITUDE',4X,'PHASE'),/,2(1X,'DEGREES',1  RD 252
  *X),3(6X,'DB'),8X,'RATIO',5X,'DEG.',8X,2(6X,'VOLTS/M',4X,'DEGRE',   RD 253
  *'ES'))                                                              RD 254
41 FORMAT(///,28X,' - - - RADIATED FIELDS NEAR GROUND - - -',//,8X,   RD 255
  *'- - - LOCATION - - -',10X,'- - E(THETA) - -',8X,'- - E(PHI) -',   RD 256
  *' -',8X,'- - E(RADIAL) - -',/,7X,'RHO',6X,'PHI',9X,'Z',12X,'MAG',  RD 257
  *6X,'PHASE',9X,'MAG',6X,'PHASE',9X,'MAG',6X,'PHASE',/,5X,'METERS',  RD 258
  *3X,'DEGREES',4X,'METERS',8X,'VOLTS/M',3X,'DEGREES',6X,'VOLTS/M',3  RD 259
  *X,'DEGREES',6X,'VOLTS/M',3X,'DEGREES',/)                           RD 260
42 FORMAT(1X,F7.2,F9.2,3X,3F8.2,F11.5,F9.2,2X,A6,2(1P,E15.5,0P,F9.2)) RD 261
  *)                                                                   RD 262
43 FORMAT(3X,F9.2,2X,F7.2,2X,F9.2,1X,3(3X,1P,E11.4,2X,0P,F7.2))       RD 263
44 FORMAT(//,3X,'AVERAGE POWER GAIN=',1P,E12.5,7X,'SOLID ANGLE U',    RD 264
  *'SED IN AVERAGING=(',0P,F7.4,')*PI STERADIANS.',//)                 RD 265
45 FORMAT(//,37X,'- - - - NORMALIZED GAIN - - - -',//,37X,2A6,'GAI',  RD 266
  *'N',/,38X,'NORMALIZATION FACTOR =',F9.2,' DB',//,3(4X,             RD 267
  *'- - ANGLES'' - -',6X,'GAIN',7X),/,3(4X,'THETA',5X,'PHI',8X,'DB',  RD 268
  *8X),/,3(3X,'DEGREES',2X,'DEGREES',16X))                             RD 269
46 FORMAT(3(1X,2F9.2,1X,F9.2,6X))                                      RD 270
   END                                                                 RD 271
```

```
      SUBROUTINE READGM( GM, I1, I2, X1, Y1, Z1, X2, Y2, Z2, RAD)
      INTEGER*4 NTOT
      INTEGER*4 NINT
      INTEGER*4 NFLT
      PARAMETER (NTOT=9, NINT=2, NFLT=7)
      INTEGER  IARR( NINT), BP( NTOT), EP( NTOT)
      DIMENSION  RARR( NFLT)
      CHARACTER   LINE*133, GM*2, BUFFER*132, BUFFER1*132
      READ (1, 10)  LINE
   10 FORMAT(A)
      NLIN= LEN(LINE)
      CALL STROPC( LINE(1: NLIN), LINE(1: NLIN))
      IF(NLIN.LT.2) GOTO 110
      IF(NLIN.LE.132) GOTO 20
      NLIN=132
      LINE(133:133)=' '
   20 GM= LINE(1:2)
      NLIN= NLIN+1
      DO 30  I=1, NINT
   30 IARR( I)=0
      DO 40  I=1, NFLT
   40 RARR( I)=0.0
      IC=2
      IFOUND=0
      DO 70  I=1, NTOT
   50 IC= IC+1
      IF(IC.GE. NLIN) GOTO 80
      IF(LINE( IC: IC).EQ.' '.OR. LINE( IC: IC).EQ.',') GOTO 50
C BEGINNING OF I-TH NUMERICAL FIELD
      BP( I)= IC
   60 IC= IC+1
      IF(IC.GT. NLIN) GOTO 80
      IF(LINE( IC: IC).NE.' '.AND. LINE( IC: IC).NE.',') GOTO 60
C END OF I-TH NUMERICAL FIELD
      EP( I)= IC-1
      IFOUND= I
   70 CONTINUE
   80 CONTINUE
      DO 90  I=1, MIN( IFOUND, NINT)
      NLEN= EP( I)- BP( I)+1
      BUFFER= LINE( BP( I): EP( I))
      IND= INDEX( BUFFER(1: NLEN),'.')
      IF(IND.GT.0.AND. IND.LT. NLEN) GOTO 110
C USER PUT DECIMAL POINT FOR INTEGER
      IF(IND.EQ. NLEN) NLEN= NLEN-1
      READ(BUFFER(1: NLEN),111,ERR=110)  IARR( I)
```

```
111   FORMAT(I3)
   90 CONTINUE
      DO 100  I= NINT+1, IFOUND
      NLEN= EP( I)- BP( I)+1
      BUFFER= LINE( BP( I): EP( I))
      IND= INDEX( BUFFER(1: NLEN),'.')
C USER FORGOT DECIMAL POINT FOR REAL
      IF(IND.EQ.0) THEN
      IF(NLEN.GE.15) GOTO 110
      INDE= INDEX( BUFFER(1: NLEN),'E')
      NLEN= NLEN+1
      IF(INDE.EQ.0) THEN
      BUFFER( NLEN: NLEN)='.'
      ELSE
      BUFFER1= BUFFER(1: INDE-1)//'.'// BUFFER( INDE: NLEN-1)
      BUFFER= BUFFER1
      ENDIF
      ENDIF
      READ(BUFFER(1: NLEN),112,ERR=110)  RARR( I- NINT)
  112 FORMAT (F15.7)
  100 CONTINUE
      I1= IARR(1)
      I2= IARR(2)
      X1= RARR(1)
      Y1= RARR(2)
      Z1= RARR(3)
      X2= RARR(4)
      Y2= RARR(5)
      Z2= RARR(6)
      RAD= RARR(7)
      RETURN
  110 WRITE (2,*) ' GEOMETRY DATA CARD ERROR'
      WRITE (2,*)  LINE(1: MAX(1, NLIN-1))
      STOP
      END
```

```
      SUBROUTINE READMN( GM, I1, I2, I3, I4, F1, F2, F3, F4, F5, F6)
      INTEGER*4 NTOT
      INTEGER*4 NINT
      INTEGER*4 NFLT
      PARAMETER (NTOT=10, NINT=4, NFLT=6)
      INTEGER  IARR( NINT), BP( NTOT), EP( NTOT)
      DIMENSION  RARR( NFLT)
      CHARACTER   LINE*133, GM*2, BUFFER*132, BUFFER1*132
      READ (1,10)  LINE
   10 FORMAT(A)
      NLIN= LEN(LINE)
      CALL STROPC( LINE(1: NLIN), LINE(1: NLIN))
      IF(NLIN.LT.2) GOTO 110
      IF(NLIN.LE.132) GOTO 20
      NLIN=132
      LINE(133:133)=' '
   20 GM= LINE(1:2)
      NLIN= NLIN+1
      DO 30  I=1, NINT
   30 IARR( I)=0
      DO 40  I=1, NFLT
   40 RARR( I)=0.0
      IC=2
      IFOUND=0
      DO 70  I=1, NTOT
   50 IC= IC+1
      IF(IC.GE. NLIN) GOTO 80
      IF(LINE( IC: IC).EQ.' '.OR. LINE( IC: IC).EQ.',') GOTO 50
C BEGINNING OF I-TH NUMERICAL FIELD
      BP( I)= IC
   60 IC= IC+1
      IF(IC.GT. NLIN) GOTO 80
      IF(LINE( IC: IC).NE.' '.AND. LINE( IC: IC).NE.',') GOTO 60
C END OF I-TH NUMERICAL FIELD
      EP( I)= IC-1
      IFOUND= I
   70 CONTINUE
   80 CONTINUE
      DO 90  I=1, MIN( IFOUND, NINT)
      NLEN= EP( I)- BP( I)+1
      BUFFER= LINE( BP( I): EP( I))
      IND= INDEX( BUFFER(1: NLEN),'.')
      IF(IND.GT.0.AND. IND.LT. NLEN) GOTO 110
C USER PUT DECIMAL POINT FOR INTEGER
      IF(IND.EQ. NLEN) NLEN= NLEN-1
      READ(BUFFER(1: NLEN),111,ERR=110)  IARR( I)
  111 FORMAT(I5)
   90 CONTINUE
      DO 100  I= NINT+1, IFOUND
```

```
      NLEN= EP( I)- BP( I)+1
      BUFFER= LINE( BP( I): EP( I))
      IND= INDEX( BUFFER(1: NLEN),'.')
C USER FORGOT DECIMAL POINT FOR REAL
      IF(IND.EQ.0) THEN
      IF(NLEN.GE.15) GOTO 110
      INDE= INDEX( BUFFER(1: NLEN),'E')
      NLEN= NLEN+1
      IF(INDE.EQ.0) THEN
      BUFFER( NLEN: NLEN)='.'
      ELSE
      BUFFER1= BUFFER(1: INDE-1)//'.'// BUFFER( INDE: NLEN-1)
      BUFFER= BUFFER1
      ENDIF
      ENDIF
      READ(BUFFER(1: NLEN),112,ERR=110)  RARR( I- NINT)
  112 FORMAT(F15.7)
  100 CONTINUE
      I1= IARR(1)
      I2= IARR(2)
      I3= IARR(3)
      I4= IARR(4)
      F1= RARR(1)
      F2= RARR(2)
      F3= RARR(3)
      F4= RARR(4)
      F5= RARR(5)
      F6= RARR(6)
      RETURN
  110 WRITE (2,*) '          FAULTY DATA CARD AFTER GEOMETRY SECTION'
      WRITE (2,*)  LINE(1: MAX(1, NLIN-1))
      STOP
      END
```

REBLK

PURPOSE

To read the matrix B by blocks of rows and write it by blocks of columns.

METHOD

When ICASX is 3 or 4 subroutine CMNGF writes as to file 14 by blocks of rows. Filling B by rows is convenient since the field of a single segment may contribute to several columns. However, blocks of columns are needed when $A^{-1}B$ is computed. Hence the format is converted.

NBBX is the number of block of B stared by rows and NBBL is the number of blocks stored by columns. The loop from RB16 to RB23 reads file 14 and stores the elements for block NPB of columns. This process is repeated for each of the NBBL blocks of columns.

SYMBOL DICTIONARY

B	=	array for blocks of columns of B
AX	=	array for blocks of rows of B
NZC	=	number of columns in B
NB	=	number of rows in B
NBX	=	number of rows in blocks of rows of B (NPBX)
NPB	=	number of columns in blocks of columns (NPBL or NLBL for last block)
NPX	=	NPBK or NLBX for last block of rows

```
      SUBROUTINE REBLK( B, BX, NB, NBX, N2C)
C     REBLOCK ARRAY B IN N.G.F. SOLUTION FROM BLOCKS OF ROWS ON TAPE14
C     TO BLOCKS OF COLUMNS ON TAPE16
      COMPLEX  B, BX
      COMMON  /MATPAR/ ICASE, NBLOKS, NPBLK, NLAST, NBLSYM, NPSYM,
     *NLSYM, IMAT, ICASX, NBBX, NPBX, NLBX, NBBL, NPBL, NLBL
      DIMENSION  B( NB,1), BX( NBX,1)
      REWIND 16
      NIB=0
      NPB= NPBL
      DO 3  IB=1, NBBL
      IF(IB.EQ. NBBL) NPB= NLBL
      REWIND 14
      NIX=0
      NPX= NPBX
      DO 2  IBX=1, NBBX
      IF(IBX.EQ. NBBX) NPX= NLBX
      READ(14) (( BX( I, J), I=1, NPX), J=1, N2C)
      DO 1  I=1, NPX
      IX= I+ NIX
      DO 1  J=1, NPB
    1 B( IX, J)= BX( I, J+ NIB)
    2 NIX= NIX+ NPBX
      WRITE( 16) (( B( I, J), I=1, NB), J=1, NPB)
    3 NIB= NIB+ NPBL
      REWIND 14
      REWIND 16
      RETURN
      END
```

REFLC

PURPOSE

To generate geometry data for structures having plane or cylindrical symmetry by forming symmetric images of a previously defined structure unit.

METHOD

The first pact of the code, from statement RE20 to RE153, forms plane symmetric structures by reflecting segments and patches in the coordinate planes. The reflection planes are selected by the formal patameters IX, IY, and IZ. If IZ is greater than zero, an image of the existing segments and patches is formed by reflection in the x-y plane, which will be called reflection along the z axis. Next, if IY is greater than zero, an image of the existing segments and patches, including those generated in the previous step by reflection along the z axis, is formed by reflection along the y axis. Finally, if IX is greater than zero, an image of all segments and patches, including any previously formed by reflection along the z and y axes, is formed by reflection along the x axis. Any combination of zero and non-zero values of IX, IY, and IZ may be used to generate structures with one, two, or three planes of symmetry. Tag numbers of image segments are incremented by ITX from tags of the original segments, except that tags of zero are not incremented. After each reflection in a coordinate plane, ITX is doubled. Thus, if ITX is initially greater than the largest tag of the existing segments, no duplicate tags will be formed by reflection in one, two, at three planes.

The code from RE157 to RE204 forms cylindrically symmetric structures by forming images of previously defined segments and patches rotated about the z axis. The number of images, including the original structure, is selected by NOP in the formal parameters. The angle by which each image is rotated about the z axis from the previous image is computed as 2π/NOP, so that the images are uniformly distributed about the z axis. Tag numbers af segments are incremented by ITX, except that tags of zero are not incremented.

When REFLC is used to form structures with either plane or cylindrical symmetry, the data in COMMON/DATA/ is set so that the program will take advantage of symmetry in filling and factoring the matrix. This is done by setting N equal to the total number of segments but leaving NP equal to the number of segments in the original structure unit that was reflected or rotated. The symmetry flag IPSYM is also set to indicate the type of symmetry: positive values indicating plane symmetry and negative values cylindrical symmetry. These symmetry conditions may later be changed if the structure is modified in such a way that symmetry is destroyed.

SYMBOL DICTIONARY

ABS	=	external routine (absolute value)
COS	=	external routine (cosine)
CS	=	cos (2π/NOP)
E1	=	segment coordinate (temporary storage)
E2	=	segment coordinate (temporary storage)
FNOP	=	NOP
I	=	DO loop index
ITAGI	=	segment tag (temporary storage)
IT1	=	segment tag increment
ITX	=	segment tag increment
IX	=	flag for reflection along x axis
IY	=	flag for reflection along y axis
IZ	=	flag for reflection along z axis
J	=	array location for new patch data
K	=	segment index and array location for old patch data
NOP	=	number of sections in cylindrically symmetric structure
NX	=	segment index and array location for new patch data
NNX	=	array location for old patch
SAM	=	2π/NOP
SIN	=	external routine (sine)
SS	=	sin (2π/NOP)
T1X,TlY,T1Z	=	x,y,z components of $\hat{t}_1$
T2X,T2Y,T2Z	=	x,y,z components of $\hat{t}_2$
XK	=	x coordinate of segment
X2(I)	=	x coordinate of end two of segment I
YK	=	y coordinate of segment
Y2(I)	=	y coordinate of end two of segment I
Z2(I)	=	z coordinate of end two of segment I
l.E-6	=	tolerance in test for zero
1.E-5	=	tolerance in test for zero
6.283185308	=	2π

```
      SUBROUTINE REFLC(IX,IY,IZ,ITX,NOP)
C
C     REFLC REFLECTS PARTIAL STRUCTURE ALONG X,Y, OR Z AXES OR ROTATES
C     STRUCTURE TO COMPLETE A SYMMETRIC STRUCTURE.
C
      COMMON/DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X( NM), Y( NM),
     *Z(NM), SI( NM), BI( NM), ALP( NM), BET( NM), ICON1( N2M), ICON2(
     * N2M), ITAG( N2M), ICONX( NM), WLAM, IPSYM
      COMMON/ANGL/ SALP( NM)
      DIMENSION  T1X(1), T1Y(1), T1Z(1), T2X(1), T2Y(1), T2Z(1), X2(1),
     * Y2(1), Z2(1)
      EQUIVALENCE(T1X,SI),(T1Y,ALP),(T1Z,BET),(T2X,ICON1),(T2Y,ICON2),(
     *T2Z,ITAG),(X2,SI),(Y2,ALP),(Z2,BET)
      NP=N
      MP=M
      IPSYM=0
      ITI=ITX
      IF(IX.LT.0) GOTO 19
      IF(NOP.EQ.0) RETURN
      IPSYM=1
C
C     REFLECT ALONG Z AXIS
C
      IF(IZ.EQ.0) GOTO 6
      IPSYM=2
      IF(N.LT. N2) GOTO 3
      DO 2 I= N2, N
      NX=I+ N- N1
      E1=Z( I)
      E2=Z2( I)
      IF(ABS(E1)+ ABS( E2).GT.1.D-5.AND. E1* E2.GE.-1.D-6) GOTO 1
      WRITE(2,24)  I
      STOP
    1 X(NX)=X( I)
      Y(NX)=Y( I)
      Z(NX)=-E1
      X2(NX)=X2( I)
      Y2(NX)=Y2( I)
      Z2(NX)=-E2
      ITAGI=ITAG( I)
      IF(ITAGI.EQ.0) ITAG( NX)=0
      IF(ITAGI.NE.0) ITAG( NX)= ITAGI+ ITI
    2 BI(NX)= BI( I)
      N=N*2- N1
      ITI=ITI*2
    3 IF(M.LT. M2) GOTO 6
      NXX=LD+1- M1
      DO 5 I= M2, M
      NXX=NXX-1
```

```
      NX=NXX- M+ M1
      IF(ABS(Z( NXX)).GT.1.D-10) GOTO 4
      WRITE(2,25)  I
      STOP
    4 X(NX)= X( NXX)
      Y(NX)= Y( NXX)
      Z(NX)=- Z( NXX)
      T1X(NX)= T1X( NXX)
      T1Y(NX)= T1Y( NXX)
      T1Z(NX)=- T1Z( NXX)
      T2X(NX)= T2X( NXX)
      T2Y(NX)= T2Y( NXX)
      T2Z(NX)=- T2Z( NXX)
      SALP(NX)=- SALP( NXX)
    5 BI(NX)= BI( NXX)
      M=M*2- M1
C
C     REFLECT ALONG Y AXIS
C
    6 IF(IY.EQ.0) GOTO 12
      IF(N.LT. N2) GOTO 9
      DO 8  I= N2, N
      NX=I+ N- N1
      E1=Y( I)
      E2=Y2( I)
      IF(ABS(E1)+ ABS( E2).GT.1.D-5.AND. E1* E2.GE.-1.D-6) GOTO 7
      WRITE(2,24)  I
      STOP
    7 X(NX)= X( I)
      Y(NX)=- E1
      Z(NX)= Z( I)
      X2(NX)= X2( I)
      Y2(NX)=- E2
      Z2(NX)= Z2( I)
      ITAGI=ITAG( I)
      IF(ITAGI.EQ.0) ITAG( NX)=0
      IF(ITAGI.NE.0) ITAG( NX)= ITAGI+ ITI
    8 BI(NX)= BI( I)
      N=N*2- N1
      ITI=ITI*2
    9 IF(M.LT. M2) GOTO 12
      NXX=LD+1- M1
      DO 11 I= M2, M
      NXX=NXX-1
      NX=NXX- M+ M1
      IF(ABS(Y( NXX)).GT.1.D-10) GOTO 10
      WRITE(2,25)  I
      STOP
   10 X(NX)= X( NXX)
```

```
      Y(NX)=- Y( NXX)
      Z(NX)= Z( NXX)
      T1X(NX)= T1X( NXX)
      T1Y(NX)=- T1Y( NXX)
      T1Z(NX)= T1Z( NXX)
      T2X(NX)= T2X( NXX)
      T2Y(NX)=- T2Y( NXX)
      T2Z(NX)= T2Z( NXX)
      SALP(NX)=- SALP( NXX)
   11 BI(NX)= BI( NXX)
      M=M*2- M1
C
C     REFLECT ALONG X AXIS
C
   12 IF(IX.EQ.0) GOTO 18
      IF(N.LT. N2) GOTO 15
      DO 14  I= N2, N
      NX=I+ N- N1
      E1=X( I)
      E2=X2( I)
      IF(ABS( E1)+ ABS( E2).GT.1.D-5.AND. E1* E2.GE.-1.D-6) GOTO 13
      WRITE (2,24)  I
      STOP
   13 X(NX)=- E1
      Y(NX)= Y( I)
      Z(NX)= Z( I)
      X2(NX)=- E2
      Y2(NX)= Y2( I)
      Z2(NX)= Z2( I)
      ITAGI=ITAG( I)
      IF(ITAGI.EQ.0) ITAG( NX)=0
      IF(ITAGI.NE.0) ITAG( NX)= ITAGI+ ITI
   14 BI(NX)= BI( I)
      N=N*2- N1
   15 IF(M.LT. M2) GOTO 18
      NXX=LD+1- M1
      DO 17  I= M2, M
      NXX=NXX-1
      NX=NXX- M+ M1
      IF(ABS(X( NXX)).GT.1.D-10) GOTO 16
      WRITE(2,25)  I
      STOP
   16 X(NX)=- X( NXX)
      Y(NX)= Y( NXX)
      Z(NX)= Z( NXX)
      T1X(NX)=- T1X( NXX)
      T1Y(NX)= T1Y( NXX)
      T1Z(NX)= T1Z( NXX)
      T2X(NX)=- T2X( NXX)
```

```
      T2Y(NX)= T2Y( NXX)
      T2Z(NX)= T2Z( NXX)
      SALP(NX)=- SALP( NXX)
   17 BI(NX)= BI( NXX)
      M=M*2- M1
C
C     REPRODUCE STRUCTURE WITH ROTATION TO FORM CYLINDRICAL STRUCTURE
C
   18 RETURN
   19 FNOP=NOP
      IPSYM=-1
      SAM=6.283185308D+0/ FNOP
      CS=COS(SAM)
      SS=SIN(SAM)
      IF(N.LT.N2) GOTO 21
      N=N1+( N- N1)* NOP
      NX=NP+1
      DO 20 I= NX, N
      K=I-NP+ N1
      XK=X( K)
      YK=Y( K)
      X(I)= XK* CS- YK* SS
      Y(I)= XK* SS+ YK* CS
      Z(I)= Z( K)
      XK=X2( K)
      YK=Y2( K)
      X2(I)= XK* CS- YK* SS
      Y2(I)= XK* SS+ YK* CS
      Z2(I)= Z2( K)
      ITAGI=ITAG( K)
      IF(ITAGI.EQ.0) ITAG( I)=0
      IF(ITAGI.NE.0) ITAG( I)= ITAGI+ ITI
   20 BI(I)= BI( K)
   21 IF(M.LT. M2) GOTO 23
      M=M1+( M- M1)* NOP
      NX=MP+1
      K=LD+1- M1
      DO 22  I= NX, M
      K=K-1
      J=K- MP+ M1
      XK=X( K)
      YK=Y( K)
      X(J)= XK* CS- YK* SS
      Y(J)= XK* SS+ YK* CS
      Z(J)= Z( K)
      XK=T1X( K)
      YK=T1Y( K)
      T1X(J)= XK* CS- YK* SS
      T1Y(J)= XK* SS+ YK* CS
```

```
      T1Z(J)= T1Z( K)                                                   RE 197
      XK=T2X( K)                                                        RE 198
      YK= T2Y( K)                                                       RE 199
      T2X(J)= XK* CS- YK* SS                                            RE 200
      T2Y(J)= XK* SS+ YK* CS                                            RE 201
      T2Z(J)= T2Z( K)                                                   RE 202
      SALP(J)= SALP( K)                                                 RE 203
   22 BI(J)= BI( K)                                                     RE 204
C                                                                       RE 205
   23 RETURN                                                            RE 206
   24 FORMAT(' GEOMETRY DATA ERROR--SEGMENT,I5,26H LIES IN PLANE OF S',  RE 207
     *'YMMETRY')                                                        RE 208
   25 FORMAT(' GEOMETRY DATA ERROR--PATCH,I4,26H LIES IN PLANE OF SYM',  RE 209
     *'METRY')                                                          RE 210
      END                                                               RE 211
```

ROM2

PURPOSE

To numerically integrate over the current distribution on a segment to obtain the field due to the Sommerfeld integral term.

METHOD

ROM2 integrates the product of $\vec{E}_s(\vec{r})$ (see discussion of EFLD) and the current over a segment. Separate integrals are evaluated for current distributions of constant, sin k(s - s_0) and cos k(s - s_0). With three vector components of the field, there are nine integrals evaluated simultaneously and stored in the array SUM. The integration method is the same as that described for subroutine INTX, but loops from one through nine are used at each step.

The parameter DMIN is set in EFLD to

$$DMIN = 0.01\left[|E_x^{'}|^2 + |E_y^{'}|^2 + |E_z^{'}|^2\right]^{1/2}$$

where $\vec{E}' = \int_{segment}[\vec{E}_D(\vec{r}) + \frac{k_1^2 - k_2^2}{k_1^2 + k_2^2}\vec{E}_I(\vec{r})ds.$

DMIN is passed to TEST as the lower limit for the denominator in the relative error evaluation to avoid trying to maintain relative accuracy in integrating the Sommerfeld integral when it is much smaller than the other terms.

SYMBOL DICTIONARY

A	=	lower limit of integral
B	=	upper limit of integral
DMIN	=	minimum for denominator in relative error test
DZ	=	subinterval size
DZOT	=	0.5 DZ
EP	=	tolerance for hitting upper limit
Gl,G2,...G5	=	integrand values at points within the subinterval
N	=	number of functions (9)
NM	=	minimum subinterval size is (B - A)/NM
NS	=	present subinterval size is (B - A)/NS
NT	=	counter to control increasing subinterval size
NTS	=	larger values retard increasing subinterval size
NX	=	maximum subinterval size is (B - A)/NX
RX	=	relative error limit
S	=	B - A
SUM	=	array for integral values
T00,T01,T02	=	(see subroutine INTX)
T10,T11,T20	=	(see subroutine INTX)
TMAG1,TMAG2	=	sum of the magnitudes of the integral contributions for the constant current distribution
Z	=	integration variable at left side at subinterval
ZE	=	B
ZEND	=	upper limit
65536	=	limit for cutting subinterval size

```
      SUBROUTINE ROM2(A,B,SUM,DMIN)
C
C     FOR THE SOMMERFELD GROUND OPTION, ROM2 INTEGRATES OVER THE SOURCE
C     SEGMENT TO OBTAIN THE TOTAL FIELD DUE TO GROUND.  THE METHOD OF
C     VARIABLE INTERVAL WIDTH ROMBERG INTEGRATION IS USED.  THERE ARE 9
C     FIELD COMPONENTS - THE X, Y, AND Z COMPONENTS DUE TO CONSTANT,
C     SINE, AND COSINE CURRENT DISTRIBUTIONS.
C
      COMPLEX  SUM,G1,G2,G3,G4,G5,T00,T01,T10,T02,T11,T20
      DIMENSION  SUM(9),G1(9),G2(9),G3(9),G4(9),G5(9),T01(9),T10
     *(9),T20(9)
      DATA NM,NTS,NX,N/65536,4,1,9/,RX/1.D-4/
      Z=A
      ZE=B
      S=B- A
      IF(S.GE.0.) GOTO 1
      WRITE (2,18)
      STOP
    1 EP=S/(1.E4*NM)
      ZEND=ZE-EP
      DO 2 I=1,N
    2 SUM(I)=(0.,0.)
      NS=NX
      NT=0
      CALL SFLDS(Z,G1)
    3 DZ=S/NS
      IF(Z+DZ.LE.ZE) GOTO 4
      DZ=ZE-Z
      IF(DZ.LE.EP) GOTO 17
    4 DZOT=DZ*.5
      CALL SFLDS(Z+DZOT,G3)
      CALL SFLDS(Z+DZ,G5)
    5 TMAG1=0.
C
C     EVALUATE 3 POINT ROMBERG RESULT AND TEST CONVERGENCE.
C
      TMAG2=0.
      DO 6  I=1,N
      T00=(G1(I)+G5(I))* DZOT
      T01(I)=(T00+DZ*G3(I))*.5
      T10(I)=(4.*T01(I)-T00)/3.
      IF(I.GT.3) GOTO 6
      TR=REAL(T01( I))
      TI=AIMAG(T01( I))
      TMAG1=TMAG1+ TR* TR+ TI* TI
      TR=REAL(T10(I))
      TI=AIMAG(T10(I))
      TMAG2=TMAG2+TR*TR+TI*TI
    6 CONTINUE
```

```
      TMAG1=SQRT(TMAG1)
      TMAG2=SQRT(TMAG2)
      CALL TEST(TMAG1,TMAG2,TR,0.0,0.0,TI,DMIN)
      IF(TR.GT. RX) GOTO 8
      DO 7  I=1,N
    7 SUM(I)=SUM(I)+T10(I)
      NT=NT+2
      GOTO 12
    8 CALL SFLDS(Z+DZ*.25,G2)
      CALL SFLDS(Z+DZ*.75,G4)
      TMAG1=0.
C
C     EVALUATE 5 POINT ROMBERG RESULT AND TEST CONVERGENCE.
C
      TMAG2=0.
      DO 9  I=1,N
      T02=(T01(I)+ DZOT*( G2( I)+ G4( I)))*.5
      T11=(4.0*T02-T01(I))/3.
      T20(I)=(16.*T11-T10( I))/15.
      IF(I.GT.3) GOTO 9
      TR=REAL( T11)
      TI=AIMAG( T11)
      TMAG1=TMAG1+ TR* TR+ TI* TI
      TR=REAL(T20( I))
      TI=AIMAG(T20( I))
      TMAG2=TMAG2+TR*TR+TI*TI
    9 CONTINUE
      TMAG1=SQRT(TMAG1)
      TMAG2=SQRT(TMAG2)
      CALL TEST(TMAG1, TMAG2, TR,0.,0., TI, DMIN)
      IF(TR.GT. RX) GOTO 14
   10 DO 11  I=1, N
   11 SUM( I)= SUM( I)+ T20( I)
      NT= NT+1
   12 Z= Z+ DZ
      IF(Z.GT. ZEND) GOTO 17
      DO 13  I=1, N
   13 G1( I)= G5( I)
      IF(NT.LT. NTS.OR. NS.LE. NX) GOTO 3
      NS= NS/2
      NT=1
      GOTO 3
   14 NT=0
      IF(NS.LT. NM) GOTO 15
      WRITE (2,19)  Z
      GOTO 10
   15 NS= NS*2
      DZ= S/ NS
      DZOT= DZ*.5
```

```
      DO 16  I=1, N
      G5( I)= G3( I)
   16 G3( I)= G2( I)
      GOTO 5
   17 CONTINUE
C
      RETURN
   18 FORMAT(' ERROR - B LESS THAN A IN ROM2')
   19 FORMAT(' ROM2 -- STEP SIZE LIMITED AT Z =',1P,E12.5)
      END
```

SBF

PURPOSE

To evaluate the current expansion function associated with a given segment, returning only that portion on a particular segment.

METHOD

SBF is very similar to routine TBF. Both routines evaluate the current expansion functions. However, while TBF stores the coefficients for each segment on which a given expansion function is non-zero, SBF returns the coefficients for only a single specified segment.

In the call to SBF, I is the segment on which the expansion function is centered. IS is the segment far which the function coefficients A_j, B_j and C_j are requested. These coefficients are returned in AA, BB, CC, respectively.

Refer to TBF for a discussion of the coding and variables. One additional variable in SBF -- JUNE -- is set to -1 or +1 if segment IS is found connected to end 1 or end 2, respectively, of segment I. If I = IS and segment I is not connected to a surface or ground plane, then JUNE is set to 0.

```
      SUBROUTINE SBF( I, IS, AA, BB, CC)
C     COMPUTE COMPONENT OF BASIS FUNCTION I ON SEGMENT IS.
      COMMON  /DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X( NM), Y( NM),
     *Z( NM), SI( NM), BI( NM), ALP( NM), BET( NM), ICON1( N2M), ICON2(
     * N2M), ITAG( N2M), ICONX( NM), WLAM, IPSYM
      DATA   PI/3.141592654D+0/, JMAX/30/
      AA=0.
      BB=0.
      CC=0.
      JUNE=0
      JSNO=0
      PP=0.
      JCOX= ICON1( I)
      IF(JCOX.GT.10000) JCOX= I
      JEND=-1
      IEND=-1
      SIG=-1.
      IF(JCOX) 1,11,2
    1 JCOX=- JCOX
      GOTO 3
    2 SIG=- SIG
      JEND=- JEND
    3 JSNO= JSNO+1
      IF(JSNO.GE. JMAX) GOTO 24
      D= PI* SI( JCOX)
      SDH= SIN( D)
      CDH= COS( D)
      SD=2.* SDH* CDH
      IF(D.GT.0.015) GOTO 4
      OMC=4.* D* D
      OMC=((1.3888889D-3* OMC-4.1666666667D-2)* OMC+.5)* OMC
      GOTO 5
    4 OMC=1.- CDH* CDH+ SDH* SDH
    5 AJ=1./( LOG(1./( PI* BI( JCOX)))-.577215664D+0)
      PP= PP- OMC/ SD* AJ
      IF(JCOX.NE. IS) GOTO 6
      AA= AJ/ SD* SIG
      BB= AJ/(2.* CDH)
      CC=- AJ/(2.* SDH)* SIG
      JUNE= IEND
    6 IF(JCOX.EQ. I) GOTO 9
      IF(JEND.EQ.1) GOTO 7
      JCOX= ICON1( JCOX)
      GOTO 8
    7 JCOX= ICON2( JCOX)
    8 IF(IABS( JCOX).EQ. I) GOTO 10
      IF(JCOX) 1,24,2
    9 IF(JCOX.EQ. IS) BB=- BB
   10 IF(IEND.EQ.1) GOTO 12
```

```
11 PM=- PP
   PP=0.
   NJUN1= JSNO
   JCOX= ICON2( I)
   IF(JCOX.GT.10000) JCOX= I
   JEND=1
   IEND=1
   SIG=-1.
   IF(JCOX) 1,12,2
12 NJUN2= JSNO- NJUN1
   D= PI* SI( I)
   SDH= SIN( D)
   CDH= COS( D)
   SD=2.* SDH* CDH
   CD= CDH* CDH- SDH* SDH
   IF(D.GT.0.015) GOTO 13
   OMC=4.* D* D
   OMC=((1.3888889D-3* OMC-4.1666666667D-2)* OMC+.5)* OMC
   GOTO 14
13 OMC=1.- CD
14 AP=1./( LOG(1./( PI* BI( I)))-.577215664D+0)
   AJ= AP
   IF(NJUN1.EQ.0) GOTO 19
   IF(NJUN2.EQ.0) GOTO 21
   QP= SD*( PM* PP+ AJ* AP)+ CD*( PM* AP- PP* AJ)
   QM=( AP* OMC- PP* SD)/ QP
   QP=-( AJ* OMC+ PM* SD)/ QP
   IF(JUNE) 15,18,16
15 AA= AA* QM
   BB= BB* QM
   CC= CC* QM
   GOTO 17
16 AA=- AA* QP
   BB= BB* QP
   CC=- CC* QP
17 IF(I.NE. IS) RETURN
18 AA= AA-1.
   BB= BB+( AJ* QM+ AP* QP)* SDH/ SD
   CC= CC+( AJ* QM- AP* QP)* CDH/ SD
   RETURN
19 IF(NJUN2.EQ.0) GOTO 23
   QP= PI* BI( I)
   XXI= QP* QP
   XXI= QP*(1.-.5* XXI)/(1.- XXI)
   QP=-( OMC+ XXI* SD)/( SD*( AP+ XXI* PP)+ CD*( XXI* AP- PP))
   IF(JUNE.NE.1) GOTO 20
   AA=- AA* QP
   BB= BB* QP
   CC=- CC* QP
```

```
      IF(I.NE. IS) RETURN
   20 AA= AA-1.
      D= CD- XXI* SD
      BB= BB+( SDH+ AP* QP*( CDH- XXI* SDH))/ D
      CC= CC+( CDH+ AP* QP*( SDH+ XXI* CDH))/ D
      RETURN
   21 QM= PI* BI( I)
      XXI= QM* QM
      XXI= QM*(1.-.5* XXI)/(1.- XXI)
      QM=( OMC+ XXI* SD)/( SD*( AJ- XXI* PM)+ CD*( PM+ XXI* AJ))
      IF(JUNE.NE.-1) GOTO 22
      AA= AA* QM
      BB= BB* QM
      CC= CC* QM
      IF(I.NE. IS) RETURN
   22 AA= AA-1.
      D= CD- XXI* SD
      BB= BB+( AJ* QM*( CDH- XXI* SDH)- SDH)/ D
      CC= CC+( CDH- AJ* QM*( SDH+ XXI* CDH))/ D
      RETURN
   23 AA=-1.
      QP= PI* BI( I)
      XXI= QP* QP
      XXI= QP*(1.-.5* XXI)/(1.- XXI)
      CC=1./( CDH- XXI* SDH)
      RETURN
   24 WRITE (2,25)  I
C
      STOP
   25 FORMAT(' SBF - SEGMENT CONNECTION ERROR FOR SEGMENT',I5)
      END
```

SECOND

PURPOSE

To obtain the time in seconds

METHOD

This subroutine acts as an interface of the computer system's time function and the NEC program. The system time function is called, the number is converted to seconds, and returned to the NEC program through the argument of subroutine SECOND. On CDC 6000 series computers, the system time function is SECOND and is called by the NEC program. This subroutine is, therefore, omitted on CDC 6000 computers.

```
      SUBROUTINE SECONDS(X)
C     CHUCK ADAMS, K7QO
C     LINUX AND UNIX ETIME USED TO CALCULATE ELAPSED TIMES
      REAL ETIME,TIME(2)
      EXTERNAL ETIME
      X=ETIME(TIME)
      X=TIME(1)
      RETURN
      END
```

SFLDS

PURPOSE

To evaluate the Sommerfeld-integral field components due to an infinitesimal current element an a segment.

METHOD

The coordinates of the segment are stored in COMMON/DATAJ/. The current element, at a distance T from the center of the segment, is located at (XT,YT,ZT). From SL16 to SL42 the ρ, Φ and z coordinates of the field evaluation point (X0,Y0,Z0) are computed in a coordinate system with the z axis passing through the current element and Φ=0 in the direction of the segment reference direction projected on the x,y plane. R2 is as shown in Figure 6 (page 160) and is the same as R1 in Section IV of Part I.

The Sommerfeld-integral field is computed from SL85 to SL111 by giving R_2 and θ', with

$$\theta' = \tan^{-1}\left(\frac{z+z'}{\rho}\right),$$

to subroutine INTRP. INTRP returns the quantities in equations 156 through 159 of Part I as

ERV = I_ρ^V

ERV = I_z^V

ERV = I_ρ^H

ERV = I_Φ^H

these quantities are then multiplied by exp(-jkR_2)/R_2. The components for a horizontal current element are multiplied by the appropriate factors of sin Φ or cos Φ and combined with the components for a vertical current element according to the elevation angle of the segment. Thus lines SL94 to SL96 are the ρ, z and Φ components of the field of the current element. These are converted to x, y and z components and stored in E(1), E(2) and E(3). They are also multiplied by sin(kT) and cos(kT) for the sine and cosine current distributions and stored in other elements of E.

When the separation of the source segment and observation point is large enough that the Norton approximation is used for the field, the code from SL49 to SL80 is executed. In this case SFLDS is called directly by EFLD, with T equal to zero, and returns an approximation to the field of the whole segment. The current is lumped at the center for a point source approximation.

GWAVE computes the total field including direct field and the asymptotic approximation of the field due to ground. Since EFLD has already computed

$$\vec{E}_D(\vec{r}) + \frac{k_1^2 - k_2^2}{k_1^2 + k_2^2}\vec{E}_I(\vec{r})$$

these terms must be removed from the field computed by GWAVE. The direct field $\vec{E}_D$ is set to zero by setting XX1 to zero before calling GWAVE. The second term is substracted from the field returned by GWAVE from SL59 to SL63. The field components of a vertical (V) and horizontal (H) current element in the direction Φ = 0 at the image point are

$$E_\rho^V = (E_R + E_T)\sin\theta\cos\theta$$

$$E_Z^V = E_R\cos^2\theta - E_T\sin^2\theta$$

$$E_\rho^H = (E_R\sin^2\theta - E_T\cos^2\theta)\cos\Phi$$

$$E_Z^H = (E_R + E_T)\sin\theta\cos\theta\cos\Phi$$

$$E_\Phi^H = E_T\sin\Phi$$

where

$$E_R = \frac{-j\eta}{4\pi^2}\ \frac{\exp(-jkR_2)}{(R_2/\lambda)^3}(1 + jkR_2)$$

$$E_T = \frac{-j\eta}{8\pi^2}\ \frac{\exp(-jkR_2)}{(R_2/\lambda)^3}(1 - k^2R_2^2 + jkR_2)$$

$$\cos\theta = (z + z')/R_2$$

$$\sin\theta = \rho/R_2$$

and current moment, $I\ell/\lambda^2$ = 1.

The sin Φ and cos Φ factors are omitted to match the quantities returned by GWAVE. Also, the fields of the horizontal current are reversed since the image of the source is in the direction Φ = 180 degrees. These quantities are multiplied by FRATI and subtracted from the fields returned by GWAVE.

The total field, in x, y and z components, is stored from SL70 to SL72. S is the length of the segment in wavelengths. Hence it is $I\ell/\lambda^2$ when I/λ = 1. The current moment for a sine distribution is zero and for a cosine distribution is $\sin(\pi S)/\pi$.

SYMBOL DICTIONARY

CPH	=	cos Φ
E	=	array for returning field components
EPH	=	E_Φ^H or I_Φ^H
ER	=	E_R
ERH	=	E_ρ^H or I_ρ^H
ERV	=	E_ρ^V or I_ρ^V
ET	=	E_T
EZH	=	E_Z^H or I_Z^H
EZV	=	E_Z^V or I_Z^V
FRATI	=	$(k_1^2 - k_2^2)/(k_1^2 + k_2^2)$
HRH	=	H_ρ^H for image of source current element
HRV	=	H_ρ^V
HZH	=	H_Z^H

PHX	=	x component of $\hat{\Phi}$
PHY	=	y component of $\hat{\Phi}$
PI	=	π
POT	=	$\pi/2$
R1	=	direct distance to source (set to arbitrary value)
R2	=	distance to image
R2S	=	$(R2)^2$
RHS	=	ρ
RRX	=	ρ^2
RHX	=	x component af ρ
RHY	=	y component of ρ
RK	=	kR_2
SFAC	=	value of current or current moment
SPH	=	sin Φ
T	=	distance from center of segment to current element
THET	=	θ'
TP	=	2π
XT,YT,ZT	=	coordinates of current element
ZPHS	=	$(z + z')^2$

1.570796327	=	$\pi/2$
3.141592654	=	π
6.283185308	=	2π

```
      SUBROUTINE SFLDS( T, E)
C
C     SFLDX RETURNS THE FIELD DUE TO GROUND FOR A CURRENT ELEMENT ON
C     THE SOURCE SEGMENT AT T RELATIVE TO THE SEGMENT CENTER.
C
      COMPLEX  E, ERV, EZV, ERH, EZH, EPH, T1, EXK, EYK, EZK, EXS,
     *EYS, EZS, EXC, EYC, EZC, XX1, XX2, U, U2, ZRATI, ZRATI2, FRATI,
     *ER, ET, HRV, HZV, HRH
      COMMON  /DATAJ/ S, B, XJ, YJ, ZJ, CABJ, SABJ, SALPJ, EXK, EYK,
     *EZK, EXS, EYS, EZS, EXC, EYC, EZC, RKH, IEXK, IND1, INDD1, IND2,
     *INDD2, IPGND
      COMMON  /INCOM/ XO, YO, ZO, SN, XSN, YSN, ISNOR
      COMMON  /GWAV/ U, U2, XX1, XX2, R1, R2, ZMH, ZPH
      COMMON  /GND/ ZRATI, ZRATI2, FRATI, CL, CH, SCRWL, SCRWR, NRADL,
     *KSYMP, IFAR, IPERF, T1, T2
      DIMENSION  E(9)
      DATA   PI/3.141592654D+0/, TP/6.283185308D+0/, POT/1.570796327D+0
     */
      XT= XJ+ T* CABJ
      YT= YJ+ T* SABJ
      ZT= ZJ+ T* SALPJ
      RHX= XO- XT
      RHY= YO- YT
      RHS= RHX* RHX+ RHY* RHY
      RHO= SQRT( RHS)
      IF(RHO.GT.0.) GOTO 1
      RHX=1.
      RHY=0.
      PHX=0.
      PHY=1.
      GOTO 2
    1 RHX= RHX/ RHO
      RHY= RHY/ RHO
      PHX=- RHY
      PHY= RHX
    2 CPH= RHX* XSN+ RHY* YSN
      SPH= RHY* XSN- RHX* YSN
      IF(ABS( CPH).LT.1.D-10) CPH=0.
      IF(ABS( SPH).LT.1.D-10) SPH=0.
      ZPH= ZO+ ZT
      ZPHS= ZPH* ZPH
      R2S= RHS+ ZPHS
      R2= SQRT( R2S)
      RK= R2* TP
      XX2= CMPLX( COS( RK),- SIN( RK))
C
C     USE NORTON APPROXIMATION FOR FIELD DUE TO GROUND.  CURRENT IS
C     LUMPED AT SEGMENT CENTER WITH CURRENT MOMENT FOR CONSTANT, SINE,
C     OR COSINE DISTRIBUTION.
```

```
C
      IF(ISNOR.EQ.1) GOTO 3
      ZMH=1.
      R1=1.
      XX1=0.
      CALL GWAVE( ERV, EZV, ERH, EZH, EPH)
      ET=-(0.,4.77134)* FRATI* XX2/( R2S* R2)
      ER=2.* ET* CMPLX(1.0, RK)
      ET= ET* CMPLX(1.0 - RK* RK, RK)
      HRV=( ER+ ET)* RHO* ZPH/ R2S
      HZV=( ZPHS* ER- RHS* ET)/ R2S
      HRH=( RHS* ER- ZPHS* ET)/ R2S
      ERV= ERV- HRV
      EZV= EZV- HZV
      ERH= ERH+ HRH
      EZH= EZH+ HRV
      EPH= EPH+ ET
      ERV= ERV* SALPJ
      EZV= EZV* SALPJ
      ERH= ERH* SN* CPH
      EZH= EZH* SN* CPH
      EPH= EPH* SN* SPH
      ERH= ERV+ ERH
      E(1)=( ERH* RHX+ EPH* PHX)* S
      E(2)=( ERH* RHY+ EPH* PHY)* S
      E(3)=( EZV+ EZH)* S
      E(4)=0.
      E(5)=0.
      E(6)=0.
      SFAC= PI* S
      SFAC= SIN( SFAC)/ SFAC
      E(7)= E(1)* SFAC
      E(8)= E(2)* SFAC
      E(9)= E(3)* SFAC
C
C     INTERPOLATE IN SOMMERFELD FIELD TABLES
C
      RETURN
    3 IF(RHO.LT.1.D-12) GOTO 4
      THET= ATAN( ZPH/ RHO)
      GOTO 5
    4 THET= POT
C     COMBINE VERTICAL AND HORIZONTAL COMPONENTS AND CONVERT TO X,Y,Z
C     COMPONENTS.  MULTIPLY BY EXP(-JKR)/R.
    5 CALL INTRP( R2, THET, ERV, EZV, ERH, EPH)
      XX2= XX2/ R2
      SFAC= SN* CPH
      ERH= XX2*( SALPJ* ERV+ SFAC* ERH)
      EZH= XX2*( SALPJ* EZV- SFAC* ERV)
```

```
C     X,Y,Z FIELDS FOR CONSTANT CURRENT
      EPH= SN* SPH* XX2* EPH
      E(1)= ERH* RHX+ EPH* PHX
      E(2)= ERH* RHY+ EPH* PHY
      E(3)= EZH
C     X,Y,Z FIELDS FOR SINE CURRENT
      RK= TP* T
      SFAC= SIN( RK)
      E(4)= E(1)* SFAC
      E(5)= E(2)* SFAC
C     X,Y,Z FIELDS FOR COSINE CURRENT
      E(6)= E(3)* SFAC
      SFAC= COS( RK)
      E(7)= E(1)* SFAC
      E(8)= E(2)* SFAC
      E(9)= E(3)* SFAC
      RETURN
      END
```

SOLGF

PURPOSE

To solve for the basis function amplitudes in the NGF procedure.

METHOD

The operations performed here are described in the NGF overview in Section VI. SOLGF is called for either a NGF solution or a normal solution. For the normal solution, or for a NGF solution when no new segments or patches have been added, the solution is obtained by calling SOLVES at SF14. Otherwise, the rest of the code is executed.

The excitation vector XY is filled in the subroutine ETMNS in the order

1. E on NGF segments (N1 elements)
2. E on new segments (N - N1 elements)
3. H on NGF patches (2M1 elements)
4. H on new patches (2M - 2M1 elements)

From SF18 to SF29 this vector is put in the order

1. E on NGF segments for E_1
2. H on NGF patches for E_1

3. E on new segments for E_2
4. H on new patches for E_2

to conform to the matrix structure. From SF30 to SF36, zeros are stored in XY in the locations opposite the rows of the C' matrix. Line SF37 then computes $A^{-1}E_l$ storing it in place of El.

SF41 to SF52 computes E_2 - C $A^{-1}E_l$ and stores it in place of E_2. Matrix C is read from file 15 if necessary to form the product with $A^{-l}E_l$. From SF55 to SF80

$$I_2 = [D - CA^{-1}B]^{-1}[E_2 - CA^{-1}E_1]$$

is computed in the original location of E2. If ICASX is in the block parameters for the primary matrix are temporarily changed to those of D - $CA^{-1}B$ so that LTSOLV, which uses the primary block parameters, can perform the solution procedure. From SF84 to SF95

$$I - 1 = A^{-1}E_1 - (A^{-1}B)I_2$$

is computed. The reordering step at the beginning of SOLGF is then reversed from SF98 to SF107 to put the solution vector in the order

1. amplitudes of NGF basis functions
2. amplitudes of new basis functions
3. NGF patch currents
4. new patch currents

5. amplitudes of modified basis functions for NGF segments that connect to new segments
6. meaningless values associated with B'_{ss}

Finally, from SF109 to SF113 the amplitudes of the modified basis functions are stored in place of the NGF basis functions that were set to zero.

SYMBOL DICTIONARY

A	=	array for matrix A_F
B	=	array starting just after A in CM (used for factoring D - $CA^{-1}B$ for ICASX = 2,3 or 4)
C	=	array for matrix C
D	=	array used for factoring D - $CA^{-1}B$ when ICASX = 1
ICASS	=	saved value of ICASE
IFL	=	file in which blocks of AF are stored in descending order (ascending order is always on 13)
IP	=	array of pivot element indices
M	=	number of patches
M1	=	number of patches in NGF
MP	=	number of patches in one symmetric section of the NGF structure
N	=	number of segments
N1	=	number of segments in NGF
N1C	=	number of unknowns in NGF (N1 + 2M1)
N2	=	N1 + 1
N2C	=	number of new unknowns (order of D)
NBLSYS	=	saved value of NBLSYM
NEQ	=	wan number of unknowns (NGF and new)
NEQS	=	number of columns in B'_{sw} and B'_{ss}
NLSYS	=	saved value of NLSYM
NP	=	number of segments in a symmetric section of the NGF structure
NFSYS	=	saved value of NPSYM
SUM	=	summation variable far matrix products
XY	=	excitation and solution vector

```
      SUBROUTINE SOLGF( A, B, C, D, XY, IP, NP, N1, N, MP, M1, M, N1C,
     *N2C, N2CZ)
C     SOLVE FOR CURRENT IN N.G.F. PROCEDURE
      COMPLEX  A, B, C, D, SUM, XY, Y
      COMMON  /SCRATM/ Y( N2M)
      COMMON  /SEGJ/ AX(30), BX(30), CX(30), JCO(30), JSNO, ISCON(50),
     *NSCON, IPCON(10), NPCON
      COMMON  /MATPAR/ ICASE, NBLOKS, NPBLK, NLAST, NBLSYM, NPSYM,
     *NLSYM, IMAT, ICASX, NBBX, NPBX, NLBX, NBBL, NPBL, NLBL
      DIMENSION  A(1), B( N1C,1), C( N1C,1), D( N2CZ,1), IP(1), XY(1)
      IFL=14
      IF(ICASX.GT.0) IFL=13
C     NORMAL SOLUTION.  NOT N.G.F.
      IF(N2C.GT.0) GOTO 1
      CALL SOLVES( A, IP, XY, N1C,1, NP, N, MP, M,13, IFL)
      GOTO 22
C     REORDER EXCITATION ARRAY
    1 IF(N1.EQ. N.OR. M1.EQ.0) GOTO 5
      N2= N1+1
      JJ= N+1
      NPM= N+2* M1
      DO 2  I= N2, NPM
    2 Y( I)= XY( I)
      J= N1
      DO 3  I= JJ, NPM
      J= J+1
    3 XY( J)= Y( I)
      DO 4  I= N2, N
      J= J+1
    4 XY( J)= Y( I)
    5 NEQS= NSCON+2* NPCON
      IF(NEQS.EQ.0) GOTO 7
      NEQ= N1C+ N2C
C     COMPUTE INV(A)E1
      NEQS= NEQ- NEQS+1
      DO 6  I= NEQS, NEQ
    6 XY( I)=(0.,0.)
    7 CALL SOLVES( A, IP, XY, N1C,1, NP, N1, MP, M1,13, IFL)
      NI=0
C     COMPUTE E2-C(INV(A)E1)
      NPB= NPBL
      DO 10  JJ=1, NBBL
      IF(JJ.EQ. NBBL) NPB= NLBL
      IF(ICASX.GT.1) READ(15) (( C( I, J), I=1, N1C), J=1, NPB)
      II= N1C+ NI
      DO 9  I=1, NPB
      SUM=(0.,0.)
      DO 8  J=1, N1C
    8 SUM= SUM+ C( J, I)* XY( J)
```

```
      J= II+ I
    9 XY( J)= XY( J)- SUM
   10 NI= NI+ NPBL
      REWIND 15
C     COMPUTE INV(D)(E2-C(INV(A)E1)) = I2
      JJ= N1C+1
      IF(ICASX.GT.1) GOTO 11
      CALL SOLVE( N2C, D, IP( JJ), XY( JJ), N2C)
      GOTO 13
   11 IF(ICASX.EQ.4) GOTO 12
      NI= N2C* N2C
      READ(11) ( B( J,1), J=1, NI)
      REWIND 11
      CALL SOLVE( N2C, B, IP( JJ), XY( JJ), N2C)
      GOTO 13
   12 NBLSYS= NBLSYM
      NPSYS= NPSYM
      NLSYS= NLSYM
      ICASS= ICASE
      NBLSYM= NBBL
      NPSYM= NPBL
      NLSYM= NLBL
      ICASE=3
      REWIND 11
      REWIND 16
      CALL LTSOLV( B, N2C, IP( JJ), XY( JJ), N2C,1,11,16)
      REWIND 11
      REWIND 16
      NBLSYM= NBLSYS
      NPSYM= NPSYS
      NLSYM= NLSYS
      ICASE= ICASS
   13 NI=0
C     COMPUTE INV(A)E1-(INV(A)B)I2 = I1
      NPB= NPBL
      DO 16  JJ=1, NBBL
      IF(JJ.EQ. NBBL) NPB= NLBL
      IF(ICASX.GT.1) READ(14) (( B( I, J), I=1, N1C), J=1, NPB)
      II= N1C+ NI
      DO 15  I=1, N1C
      SUM=(0.,0.)
      DO 14  J=1, NPB
      JP= II+ J
   14 SUM= SUM+ B( I, J)* XY( JP)
   15 XY( I)= XY( I)- SUM
   16 NI= NI+ NPBL
      REWIND 14
C     REORDER CURRENT ARRAY
      IF(N1.EQ. N.OR. M1.EQ.0) GOTO 20
```

```
      DO 17  I= N2, NPM
   17 Y( I)= XY( I)
      JJ= N1C+1
      J= N1
      DO 18  I= JJ, NPM
      J= J+1
   18 XY( J)= Y( I)
      DO 19  I= N2, N1C
      J= J+1
   19 XY( J)= Y( I)
   20 IF(NSCON.EQ.0) GOTO 22
      J= NEQS-1
      DO 21  I=1, NSCON
      J= J+1
      JJ= ISCON( I)
   21 XY( JJ)= XY( J)
   22 RETURN
      END
```

SOLVE

PURPOSE

To solve the system LUx = B, where L is a lower triangular matrix with ones on the diagonal, U is an upper triangular matrix, and B is the right-hand side vector (RHS).

METHOD

The algorithm used is described on pages 409-415 of ref. 1. The solution of the matrix equation LUx = B is found by first solving

$$Ly = B \ ,$$

and then

$$Ux = y \ .$$

since

$$LUx = Ly = B \ .$$

The solution of equations Ly = B and Ux = y is straightforward since the matrices are both triangular. The solution of equation Ly = B can be written

$$y_i = \frac{1}{\ell_{ii}}\left(b_i - \sum_{j=1}^{i-1} \ell_{ij} y_j\right) \qquad i = 1, ..., n \ .$$

Ux = y can be written similarly.

The L and U matrices are both supplied by the subroutine FACTR and are stored in the matrix A; the 1's on the diagonal of L are suppressed. Care must be exercised in the solution, since rows were interchanged during factorization, and this necessitates rearranging the RHS vector; furthermore, the L matrix itself is not completely rearranged. The information pertinent to the row rearrangements has been stared by FACTR in an integer array (IP), and it is used in the computations. The final solution of the equations is overwritten on the input RHS vector E.

The only differences between the coding in SOLVE and the coding suggested in ref. 1 are: (1) double precision variables are not used for the accumulation of sums, since, for the size of matrices anticipated in core, the computer word length is sufficient, and (2) the transposes of the L and U matrices are supplied in A by FACTR. Thus, the row and column indices used in the routine are reversed to account for this transposition.

CODING

SO15-SO25	The solution for y in Ly = B.
SO29-SO39	The solution for x in equation Ux = y and the storage of the solution in B.

SYMBOL DICTIONARY

A	=	array contains the input L and U matrices
B	=	array contains the input RHS and is overwritten with the solution
I	=	DO loop index
IP	=	array contains row positioning information
IP1	=	I + 1
J	=	DO luop index
K	=	DO loop index
N	=	order of the matrix being solved
NDIM	=	dimension of the array where the matrix is stored NDIM $\geq$ N
PI	=	intermediate integer
SUM	=	intermediate variable
Y	=	scratch vector

```
      SUBROUTINE SOLVE( N, A, IP, B, NDIM)
C
C     SUBROUTINE TO SOLVE THE MATRIX EQUATION LU*X=B WHERE L IS A UNIT
C     LOWER TRIANGULAR MATRIX AND U IS AN UPPER TRIANGULAR MATRIX BOTH
C     OF WHICH ARE STORED IN A.  THE RHS VECTOR B IS INPUT AND THE
C     SOLUTION IS RETURNED THROUGH VECTOR B.    (MATRIX TRANSPOSED.
C
      COMPLEX  A, B, Y, SUM
      INTEGER  PI
      COMMON  /SCRATM/ Y( N2M)
C
C     FORWARD SUBSTITUTION
C
      DIMENSION  A( NDIM, NDIM), IP( NDIM), B( NDIM)
      DO 3  I=1, N
      PI= IP( I)
      Y( I)= B( PI)
      B( PI)= B( I)
      IP1= I+1
      IF(IP1.GT. N) GOTO 2
      DO 1  J= IP1, N
      B( J)= B( J)- A( I, J)* Y( I)
    1 CONTINUE
    2 CONTINUE
C
C     BACKWARD SUBSTITUTION
C
    3 CONTINUE
      DO 6  K=1, N
      I= N- K+1
      SUM=(0.,0.)
      IP1= I+1
      IF(IP1.GT. N) GOTO 5
      DO 4  J= IP1, N
      SUM= SUM+ A( J, I)* B( J)
    4 CONTINUE
    5 CONTINUE
      B( I)=( Y( I)- SUM)/ A( I, I)
    6 CONTINUE
      RETURN
      END
```

<u>SOLVES</u>

PURPOSE

To control solution of the matrix equation, including transforming and reordering the solution vector.

METHOD

When SOLVES is called, the array B contains the excitation computed by subroutines ETMNS or NETWK. The exciting electric field on all segments is stored first in B, followed by the magnetic fields on all patches. In the case of a symmetric structure, however, the matrix is filled with the coefficients of all segment and patch equations in the first symmetric sector occurring first. These are followed by the coefficients for successive sectors in the same order. This order is required for the solution procedure for symmetric structures described in section III-S of Part I. For the case of a symmetric structure with both segments and patches, SOLVES first rearranges the excitation coefficients in array B to correspond to the order of the matrix coefficients.

For symmetric structures, SOLVES then computes the transforms of the subvectors in B according to equation (88) of Part I. Subroutine SOLVE or LTSOLV is then called to compute the solution or solution subvectors. The procedure is selected by the parameter ICASE as follows.

1 No symmetry, matrix in core. SOLVE is called for the solution.
2 Symmetry, matrix in care. SOLVE is called for each subvector.
3 No symmetry, matrix out of core. LTSOLV is called for the solution.
4 Symmetry, complete matrix does not fit in core but submatrices do. SOLVE is called for each subvector after first reading the appropriate submatrix from file IFL1.
5 Symmetry, submatrlces do not fit in core. LTSOLV is called for each subvector.

SOLVES then computes the total current by inverse transforming the subvectors by equation (115) of Part I. For a symmetric structure with segments and patches, SOLVES then rearranges the solution in array B to put all segment currents First, followed by all patch currents, which is the order of the original excitation coefficients.

Multiple right-hand-side vectors (NKH) may be processed simultaneously at each step in SOLVES. This reduces the time spent reading files when LTSOLV is called, and is used in computing A-LB in the NGF procedure.

CODING

SS22-SS39	Rearrange excitation ccefficients.
SS43-SS56	Transform subvectors.
SS63-SS75	Solve for each subvector.
SS8l-SS94	Inverse transform subvetturs.
SS96-SS1l3	Rearrange solution coefficients.

SYMBOL DICTIONARY

A	=	array set aside for in-core matrix storage, i.e., factored matrices
B	=	right-hand side; the solution is overwritten on this array also
FNOP	=	decimal form of NOP
FNORM	=	1/FNOP
IFL1	=	file with matrix blocks in normal order
IFL2	=	file with matrix blocks in reversed erder
IP	=	array containing positioning data used in SOLVE
M	=	number of patches
MP	=	number of patches in a symmetric sector
N	=	number of segments
NCOL	=	number of columns in array A
NEQ	=	order of complete matrix
NOP	=	number of symmetric sectors
NP	=	number of segments in a symmetric sector
NPEQ	=	order of a submatrix
NRH	=	number of right-hand-side vectors in B
NROW	=	number of rows in A
SSX	=	array containing the coefficients S_{ik} in equation (89) of Part I
SUM	=	summation variable
Y	=	scratch vector

```
      SUBROUTINE SOLVES( A, IP, B, NEQ, NRH, NP, N, MP, M, IFL1, IFL2)
C
C     SUBROUTINE SOLVES, FOR SYMMETRIC STRUCTURES, HANDLES THE
C     TRANSFORMATION OF THE RIGHT HAND SIDE VECTOR AND SOLUTION OF THE
C     MATRIX EQ.
C
      COMPLEX  A, B, Y, SUM, SSX
      COMMON  /SMAT/ SSX(16,16)
      COMMON  /SCRATM/ Y( N2M)
      COMMON  /MATPAR/ ICASE, NBLOKS, NPBLK, NLAST, NBLSYM, NPSYM,
     *NLSYM, IMAT, ICASX, NBBX, NPBX, NLBX, NBBL, NPBL, NLBL
      DIMENSION  A(1), IP(1), B( NEQ, NRH)
      NPEQ= NP+2* MP
      NOP= NEQ/ NPEQ
      FNOP= NOP
      FNORM=1./ FNOP
      NROW= NEQ
      IF(ICASE.GT.3) NROW= NPEQ
      IF(NOP.EQ.1) GOTO 11
      DO 10  IC=1, NRH
      IF(N.EQ.0.OR. M.EQ.0) GOTO 6
      DO 1  I=1, NEQ
    1 Y( I)= B( I, IC)
      KK=2* MP
      IA= NP
      IB= N
      J= NP
      DO 5  K=1, NOP
      IF(K.EQ.1) GOTO 3
      DO 2  I=1, NP
      IA= IA+1
      J= J+1
    2 B( J, IC)= Y( IA)
      IF(K.EQ. NOP) GOTO 5
    3 DO 4  I=1, KK
      IB= IB+1
      J= J+1
    4 B( J, IC)= Y( IB)
C
C     TRANSFORM MATRIX EQ. RHS VECTOR ACCORDING TO SYMMETRY MODES
C
    5 CONTINUE
    6 DO 10  I=1, NPEQ
      DO 7  K=1, NOP
      IA= I+( K-1)* NPEQ
    7 Y( K)= B( IA, IC)
      SUM= Y(1)
      DO 8  K=2, NOP
    8 SUM= SUM+ Y( K)
```

```
      B( I, IC)= SUM* FNORM
      DO 10  K=2, NOP
      IA= I+( K-1)* NPEQ
      SUM= Y(1)
      DO 9  J=2, NOP
    9 SUM= SUM+ Y( J)* CONJG( SSX( K, J))
   10 B( IA, IC)= SUM* FNORM
   11 IF(ICASE.LT.3) GOTO 12
      REWIND IFL1
C
C     SOLVE EACH MODE EQUATION
C
      REWIND IFL2
   12 DO 16  KK=1, NOP
      IA=( KK-1)* NPEQ+1
      IB= IA
      IF(ICASE.NE.4) GOTO 13
      I= NPEQ* NPEQ
      READ(IFL1) ( A( J), J=1, I)
      IB=1
   13 IF(ICASE.EQ.3.OR. ICASE.EQ.5) GOTO 15
      DO 14  IC=1, NRH
   14 CALL SOLVE( NPEQ, A( IB), IP( IA), B( IA, IC), NROW)
      GOTO 16
   15 CALL LTSOLV( A, NPEQ, IP( IA), B( IA,1), NEQ, NRH, IFL1, IFL2)
   16 CONTINUE
C
C     INVERSE TRANSFORM THE MODE SOLUTIONS
C
      IF(NOP.EQ.1) RETURN
      DO 26  IC=1, NRH
      DO 20  I=1, NPEQ
      DO 17  K=1, NOP
      IA= I+( K-1)* NPEQ
   17 Y( K)= B( IA, IC)
      SUM=Y(1)
      DO 18 K=2,NOP
   18 SUM=SUM+ Y( K)
      B(I,IC)=SUM
      DO 20  K=2, NOP
      IA=I+(K-1)*NPEQ
      SUM=Y(1)
      DO 19  J=2,NOP
   19 SUM=SUM+ Y( J)* SSX( K, J)
   20 B(IA, IC)= SUM
      IF(N.EQ.0.OR. M.EQ.0) GOTO 26
      DO 21  I=1, NEQ
   21 Y(I)= B( I, IC)
      KK=2* MP
```

```
      IA=NP
      IB=N
      J=NP
      DO 25 K=1, NOP
      IF(K.EQ.1) GOTO 23
      DO 22 I=1, NP
      IA=IA+1
      J=J+1
   22 B(IA,IC)=Y( J)
      IF(K.EQ.NOP) GOTO 25
   23 DO 24  I=1, KK
      IB=IB+1
      J=J+1
   24 B(IB,IC)=Y(J)
   25 CONTINUE
   26 CONTINUE
      RETURN
      END
```

TBF

PURPOSE

To evaluate the current expansion function associated with a given segment.

METHOD

The current expansion function is described in section III-1 of Part I. The parameter I is the number of the segment on which the function is centered. On segment I and on all segments connected to either end of segment I, the function has the form

$$f_j(s) = A_j + B_j \sin[k(s - s_j)] + C_j \cos[k(s - s_j)] \ ,$$

where j is the segment number. TBF locates all connected segments and stores the segment numbers, j, in JCO in COMMON/SEGJ/. It computes A_j, B_j, and C_j and stores them in AX,BX, and CX, respectively, in the same location as was used in JCO. A_j, B_j, and C_j for j = I are stored last in the arrays.

If ICAP = 0, the function goes to zero at an end of segment I to which no other segment or surface is connected. If ICAP $\neq$ 0, the function has a non-zero value at a free end, allowing for the current onto the wire end cap.

CODING

Equations and symbols refer to Part I.

TB9-TB55	This code forms a loop that locates all segments connected to the ends of segment I, first for end 1 (IEND = -1) and then for end 2 (IEND = 1).
TB9-TB16	Parameters are initialized to start search for segments connected to end 1 of segment I.
TB34	PP = P_i^- for end 1 of segment I or P_i^+ for end 2 of segment I.
TB35-TB37	Equations (43) to (48) of Part I evaluated except for $Q_i^\pm$: AX(JSNO) = $A_j^\pm/Q_i^\pm$ BX(JSNO) = $B_j^\pm/Q_i^\pm$ CX(JSNO) = $C_j^\pm/Q_i^\pm$ JCO(JSNO) = j
TB38	Exit from loop if segment I is connected to s surface or ground plane. Segment I win occur in COMMON/SEGJ/ twice in this case, once for the center of the expansion function on segment I and once for the part of the function extending onto the image of segment I in the surface. Line TB45 changes the sign of $B_j^\pm$ for the image term. The sum of the two parts of the function on segment I then has zero derivative at the end connected to the surface.

TB39-TB42	Check appropriate end of segment j to determine whether it shows a connection to segment I (end of search) or connection to another segment (multiple junction).
TB44	Continue search for connected segments (multiple junction).
TB46	Exit from loop after finishing search for both ends of segment I.
TB47-TB55	Store values for end l of segment I and initialize for end 2. Then return to previous loop.
TB59-TB70	Evaluate functions of segment length and radius for segment I. For kΔ < 0.03, a series is used for l - cos kΔ, where Δ = segment length.
TB73-TB86	Final calculations if neither end of segment I is a free end.
TB89-TB102	Final calculations for free end on end 1 of segment I.
TB104-TB117	Final calculations for free end on end 2 of segment I.
TB119-TB126	Final calculations for free ends on both ends of segment I.
TB128	A_j = -l for j = I in all cases.

SYMBOL DICTIONARY

AJ	=	a_j^-
AP	=	a_j^+
CD	=	cos kΔ_j
CDH	=	cos (kΔ_j/2)
D	=	kΔ_j/2 or cos $k\Delta_i - X_i \sin k\Delta_i$
ICAP	=	flag to determine whether the function goes to zero at a free end
IEND	=	-1 during calculations for end 1 of segment I and +1 for end 2.
JCOX	=	connection index
JEND	=	-7 if end 1 of a segment is connected to segment I, +l if end 2 is connected to segment I.
JMAX	=	maximum number of segments allowed in the expansion function. This includes segment 1 and all segments connected to either end.
JSNOP	=	JSN + 1
NJUN1	=	N^-
NJUN2	=	N^+
OMC	=	1 - cos kΔ_j
PI	=	π
PM	=	P_i^-
PP	=	P_i^+
QM	=	Q_i^-
QP	=	Q_i^+
SD	=	sin kΔ_j
SDH	=	sin (kΔ_j/2)
SIG	=	sign for calculation of A_j and C_j
XX1	=	$J_1(ka_i)/J_0(ka_i)$ (small argument series used for Bessel functions)
0.577215664	=	Eulers constant
0.015	=	0.03/2
1.388B889E-3	=	1/720
3.141592654	=	π
4.1666666667E-2	=	1/24

```
      SUBROUTINE TBF(I,ICAP)
C     COMPUTE BASIS FUNCTION I
      COMMON/DATA/ LD,N1,N2,N,NP,M1,M2,M,MP,X(NM),Y(NM),
     *Z(NM),SI(NM),BI(NM),ALP(NM),BET(NM),ICON1(N2M),ICON2(
     * N2M),ITAG(N2M),ICONX(NM),WLAM,IPSYM
      COMMON/SEGJ/ AX(30), BX(30), CX(30), JCO(30), JSNO, ISCON(50),
     *NSCON,IPCON(10), NPCON
      DATA PI/3.141592654D+0/, JMAX/30/
      JSNO=0
      PP=0.
      JCOX=ICON1( I)
      IF(JCOX.GT.10000) JCOX= I
      JEND=-1
      IEND=-1
      SIG=-1.
      IF(JCOX) 1,10,2
    1 JCOX=-JCOX
      GOTO 3
    2 SIG=-SIG
      JEND=-JEND
    3 JSNO=JSNO+1
      IF(JSNO.GE.JMAX) GOTO 28
      JCO(JSNO)=JCOX
      D=PI*SI(JCOX)
      SDH=SIN(D)
      CDH=COS(D)
      SD=2.*SDH*CDH
      IF(D.GT.0.015) GOTO 4
      OMC=4.*D*D
      OMC=((1.3888889D-3*OMC-4.1666666667D-2)*OMC+.5)*OMC
      GOTO 5
    4 OMC=1.- CDH*CDH+SDH*SDH
    5 AJ=1./(LOG(1./(PI*BI( JCOX)))-.577215664D+0)
      PP=PP-OMC/ SD* AJ
      AX(JSNO)= AJ/ SD* SIG
      BX(JSNO)= AJ/(2.* CDH)
      CX(JSNO)=- AJ/(2.* SDH)* SIG
      IF(JCOX.EQ. I) GOTO 8
      IF(JEND.EQ.1) GOTO 6
      JCOX=ICON1( JCOX)
      GOTO 7
    6 JCOX=ICON2( JCOX)
    7 IF(IABS(JCOX).EQ. I) GOTO 9
      IF(JCOX) 1,28,2
    8 BX(JSNO)=- BX( JSNO)
    9 IF(IEND.EQ.1) GOTO 11
   10 PM=-PP
      PP=0.
      NJUN1=JSNO
```

```
      JCOX=ICON2( I)
      IF(JCOX.GT.10000) JCOX= I
      JEND=1
      IEND=1
      SIG=-1.
      IF(JCOX) 1,11,2
   11 NJUN2=JSNO- NJUN1
      JSNOP=JSNO+1
      JCO(JSNOP)= I
      D=PI* SI( I)
      SDH=SIN( D)
      CDH=COS( D)
      SD=2.*SDH* CDH
      CD=CDH* CDH- SDH* SDH
      IF(D.GT.0.015) GOTO 12
      OMC=4.* D* D
      OMC=((1.3888889D-3* OMC-4.1666666667D-2)* OMC+.5)* OMC
      GOTO 13
   12 OMC=1.- CD
   13 AP=1./( LOG(1./( PI* BI( I)))-.577215664D+0)
      AJ=AP
      IF(NJUN1.EQ.0) GOTO 16
      IF(NJUN2.EQ.0) GOTO 20
      QP=SD*( PM* PP+ AJ* AP)+ CD*( PM* AP- PP* AJ)
      QM=(AP* OMC- PP* SD)/ QP
      QP=-(AJ* OMC+ PM* SD)/ QP
      BX(JSNOP)=( AJ* QM+ AP* QP)* SDH/ SD
      CX(JSNOP)=( AJ* QM- AP* QP)* CDH/ SD
      DO 14  IEND=1, NJUN1
      AX(IEND)= AX( IEND)* QM
      BX(IEND)= BX( IEND)* QM
   14 CX(IEND)= CX( IEND)* QM
      JEND= NJUN1+1
      DO 15  IEND= JEND, JSNO
      AX(IEND)=- AX( IEND)* QP
      BX(IEND)= BX( IEND)* QP
   15 CX(IEND)=- CX( IEND)* QP
      GOTO 27
   16 IF(NJUN2.EQ.0) GOTO 24
      IF(ICAP.NE.0) GOTO 17
      XXI=0.
      GOTO 18
   17 QP=PI* BI( I)
      XXI=QP* QP
      XXI=QP*(1.-.5* XXI)/(1.- XXI)
   18 QP=-(OMC+ XXI* SD)/( SD*( AP+ XXI* PP)+ CD*( XXI* AP- PP))
      D=CD-XXI* SD
      BX(JSNOP)=( SDH+ AP* QP*( CDH- XXI* SDH))/ D
      CX(JSNOP)=( CDH+ AP* QP*( SDH+ XXI* CDH))/ D
```

```
      DO 19  IEND=1, NJUN2                                              TB  99
      AX(IEND)=- AX( IEND)* QP                                          TB 100
      BX(IEND)= BX( IEND)* QP                                           TB 101
   19 CX(IEND)=- CX( IEND)* QP                                          TB 102
      GOTO 27                                                           TB 103
   20 IF(ICAP.NE.0) GOTO 21                                             TB 104
      XXI=0.                                                            TB 105
      GOTO 22                                                           TB 106
   21 QM=PI* BI( I)                                                     TB 107
      XXI=QM* QM                                                        TB 108
      XXI=QM*(1.-.5* XXI)/(1.- XXI)                                     TB 109
   22 QM=(OMC+ XXI* SD)/( SD*( AJ- XXI* PM)+ CD*( PM+ XXI* AJ))         TB 110
      D=CD- XXI* SD                                                     TB 111
      BX(JSNOP)=( AJ* QM*( CDH- XXI* SDH)- SDH)/ D                      TB 112
      CX(JSNOP)=( CDH- AJ* QM*( SDH+ XXI* CDH))/ D                      TB 113
      DO 23  IEND=1, NJUN1                                              TB 114
      AX(IEND)= AX( IEND)* QM                                           TB 115
      BX(IEND)= BX( IEND)* QM                                           TB 116
   23 CX(IEND)= CX( IEND)* QM                                           TB 117
      GOTO 27                                                           TB 118
   24 BX(JSNOP)=0.                                                      TB 119
      IF(ICAP.NE.0) GOTO 25                                             TB 120
      XXI=0.                                                            TB 121
      GOTO 26                                                           TB 122
   25 QP=PI*BI( I)                                                      TB 123
      XXI=QP*QP                                                         TB 124
      XXI=QP*(1.-.5* XXI)/(1.- XXI)                                     TB 125
   26 CX(JSNOP)=1./( CDH- XXI* SDH)                                     TB 126
   27 JSNO=JSNOP                                                        TB 127
      AX(JSNO)=-1.                                                      TB 128
      RETURN                                                            TB 129
   28 WRITE(2,29)  I                                                    TB 130
C                                                                       TB 131
      STOP                                                              TB 132
   29 FORMAT(' TBF - SEGMENT CONNECTION ERROR FOR SEGMENT',I5)          TB 133
      END                                                               TB 134
```

TEST

PURPOSE

To compute the relative difference of two numerical integration results for the Romluerg variable-interval-width integration routines.

METHOD

The first numerical integration result is the complex number (F1R, F1I) and the second is (F2R, F2I). The real and imaginary parts of the two results are subtracted and the differences are divided by the largest of F2R, F2I, DMIN or 10^{-34}. The denominator is chosen to avoid trying to maintain a small relative error for a quantity that is insignificantly small.

SYMBOL DICTIONARY

ABS	=	external routine (absolute value)
DEN	=	largest of |F2R| and |F2I|
DMIN	=	minimum denominator
F1I	=	imaginary part of first integration result
F1R	=	real part of first integration result
F2I	=	imaginary part of second integration result
F2R	=	real part of second integration result
TI	=	relative difference of imaginary parts
TR	=	relative difference of real parts
1.E-37	=	tolerance in test for zero

```
      SUBROUTINE TEST(F1R,F2R,TR,F1I,F2I,TI,DMIN)
C
C     TEST FOR CONVERGENCE IN NUMERICAL INTEGRATION
C
      DEN=ABS(F2R)
      TR=ABS(F2I)
      IF(DEN.LT.TR) DEN= TR
      IF(DEN.LT.DMIN) DEN= DMIN
      IF(DEN.LT.1.D-37) GOTO 1
      TR=ABS((F1R-F2R)/ DEN)
      TI=ABS((F1I-F2I)/ DEN)
      RETURN
    1 TR=0.
      TI=0.
      RETURN
      END
```

TRIO

PURPOSE

To evaluate each of the parts of current expansion functions on a single segment due to each of the segments connected to the given segment.

METHOD

TRIO consists of a loop that uses the connection data in arrays ICON1 and ICON2 to locate all segments connected to segment J. Subroutine SBF is called to evaluate the current expansion function centered on each connected segment and on segment J. Only the function coefficients for that part of each expansion function on segment J are returned and are stored in arrays AX, BX, and CX. The number of the segment with which each expansion function part is associated is stored in array JCO and the total number of expansion functions involved is stored as JSNO.

SYMBOL DICTIONARY

IEND	=	-1 during calculations for end 1 of segment J, and +1 for end 2
JCOX	=	number of a segment connected to segment J
JEND	=	-1 if end 1 of segment JCOX is connected to segment J +1 if end 2 of segment JCOX is connected to segment J
JMAX	=	dimension of the arrays in COMMON/SEGJ/

```
      SUBROUTINE TRIO(J)
C     COMPUTE THE COMPONENTS OF ALL BASIS FUNCTIONS ON SEGMENT J
      COMMON/DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X( NM), Y( NM),
     *Z( NM), SI( NM), BI( NM), ALP( NM), BET( NM), ICON1( N2M), ICON2(
     * N2M), ITAG( N2M), ICONX( NM), WLAM, IPSYM
      COMMON/SEGJ/ AX(30), BX(30), CX(30), JCO(30), JSNO, ISCON(50),
     *NSCON,IPCON(10), NPCON
      DATA  JMAX/30/
      JSNO=0
      JCOX=ICON1( J)
      IF(JCOX.GT.10000) GOTO 7
      JEND=-1
      IEND=-1
      IF(JCOX) 1,7,2
    1 JCOX=-JCOX
      GOTO 3
    2 JEND=-JEND
    3 IF(JCOX.EQ. J) GOTO 6
      JSNO=JSNO+1
      IF(JSNO.GE. JMAX) GOTO 9
      CALL SBF( JCOX, J, AX( JSNO), BX( JSNO), CX( JSNO))
      JCO(JSNO)= JCOX
      IF(JEND.EQ.1) GOTO 4
      JCOX=ICON1( JCOX)
      GOTO 5
    4 JCOX=ICON2( JCOX)
    5 IF(JCOX) 1,9,2
    6 IF(IEND.EQ.1) GOTO 8
    7 JCOX=ICON2( J)
      IF(JCOX.GT.10000) GOTO 8
      JEND=1
      IEND=1
      IF(JCOX) 1,8,2
    8 JSNO=JSNO+1
      CALL SBF( J, J, AX( JSNO), BX( JSNO), CX( JSNO))
      JCO(JSNO)= J
      RETURN
    9 WRITE(2,10)  J
C
      STOP
   10 FORMAT(' TRIO - SEGMENT CONNENTION ERROR FOR SEGMENT',I5)
      END
```

UNERE

PURPOSE

To calcuLate the electric Field due to unit currents in the $\hat{t}_1$ and $\hat{t}_2$ directions on a surface patch.

METHOD

The electric field due to at patch j is calculated by the expression

$$\vec{E}(\vec{r}_0) = \frac{\eta_0}{18\pi^2}\left[\left(\frac{-1-12\pi R/\lambda+4\pi^2(R/\lambda)^2}{(R/\lambda)^3}\right)\vec{J}_j\right.$$

$$\left.+\left(\frac{3+16\pi R/\lambda-4\pi^2(R/\lambda)^2}{(R/\lambda)^5}\right)\vec{J}_j\cdot(\vec{R}/\lambda)(\vec{R}/\lambda)\right]\exp(-i2\pi R/\lambda)\frac{\Delta A_j}{\lambda^2}\ ,$$

where i = $\sqrt{-1}$, $\vec{J}_j = J_{1j}\hat{t}_{1j}+J_{2j}\hat{t}_{2j}$, $\vec{R}$ is the vector from the source to the observation point, and ΔA_j is the area of the patch. For UNERE, J_{1j} , and J_{2j} are unity. The expression above for a single patch is obtained from the surface integral in equation (3) in Part I where constant current and one step integration are used for the patch.

CODING

UE14-UE20	z components of patch parameters are adjusted for direct or reflected fields.
UE25-UE32	For R < 10^{-10}, the fields are set to zero.
UE34-UE47	Expression for $\vec{E}$ is evaluated for J_j equal to $\hat{t}_1$ and $\hat{t}_2$.
UE50-UE55	For reflection in a perfect ground, $\vec{E}$ is reversed in sign.
UE57-UE79	For reflection in an imperfect ground, $\vec{E}$ is multiplied by the reflection coefficients.

SYMBOL DICTIONARY

CONST	=	$\eta_0/(8\pi^2)$
CTH	=	cos θ; θ is the angle between the reflected ray and the normal to the surface
EDP	=	$(\vec{E}\cdot\hat{p})(R_H-R_V)$
ER	=	$\eta_0/(18\pi^2)\ \exp(-i\,2\pi\,R/\lambda)\ \Delta A_j/\lambda^2$ at UE37
	=	Q2 $(\hat{t}_{1j}\cdot\vec{R}/\lambda)$ at UE40
	=	Q2 $(\hat{t}_{2j}\cdot\vec{R}/\lambda)$ at UE44
EXK,EYK,EZK	=	$\vec{E}$ due to current $\hat{t}_{1j}$
EXS,EYS,EZS	=	$\vec{E}$ due to current $\hat{t}_{2j}$
IPGND	=	flag to cause computation of reflected field when equal to 2
PX,PY	=	= $\hat{p}$; unit vector normal to the plane of incident of the reflected ray

Q1	=	$\left([(-1 - i2\pi R/\lambda + 4\pi^2(R/\lambda)^2]/[(R/\lambda)^3]\right)(ER)$
Q2	=	$\left([(-1 - i6\pi R/\lambda - 4\pi^2(R/\lambda)^2]/[(R/\lambda)^5]\right)(ER)$
R	=	R/λ
RRH	=	R$_H$
RRV	=	RV$_V$
RT	=	$(R/\lambda)^3$
RX,RY,RZ	=	$\vec{R}/\lambda$
R2	=	$(R/\lambda)^2$
S	=	$\Delta A_j/\lambda^2$
T1XJ,T1YJ,T1ZJ	=	$\hat{t}_{1j}$
T2XJ,T2YJ,T2ZJ	=	$\hat{t}_{2j}$
TPI	=	2π
TT1	=	$-2\pi R/\lambda$
TT2	=	$4\pi 2(R/\lambda)^2$
XOB,YOB,ZOB	=	field evaluation point
XYMAG	=	magnitude of the projection of $\vec{R}/\lambda$ onto the x-y plane
ZR	=	z component of $\vec{R}/\lambda$ after reflection
4.771341188	=	$\eta_0/8\pi^2$
6.283185308	=	2π

```
      SUBROUTINE UNERE(XOB,YOB,ZOB)
C     CALCULATES THE ELECTRIC FIELD DUE TO UNIT CURRENT IN THE T1 AND T2
C     DIRECTIONS ON A PATCH
      COMPLEX  EXK, EYK, EZK, EXS, EYS, EZS, EXC, EYC, EZC, ZRATI,
     *ZRATI2, T1, ER, Q1, Q2, RRV, RRH, EDP, FRATI
      COMMON/DATAJ/ S, B, XJ, YJ, ZJ, CABJ, SABJ, SALPJ, EXK, EYK,
     *EZK, EXS, EYS, EZS, EXC, EYC, EZC, RKH, IEXK, IND1, INDD1, IND2,
     *INDD2,IPGND
      COMMON /GND/ ZRATI, ZRATI2, FRATI, CL, CH, SCRWL, SCRWR, NRADL,
     *KSYMP,IFAR, IPERF, T1, T2
      EQUIVALENCE(T1XJ,CABJ),(T1YJ,SABJ),(T1ZJ,SALPJ),(T2XJ,B),(T2YJ,
     *IND1),(T2ZJ,IND2)
C     CONST=ETA/(8.*PI**2)
      DATA  TPI, CONST/6.283185308D+0,4.771341188D+0/
      ZR=ZJ
      T1ZR=T1ZJ
      T2ZR=T2ZJ
      IF(IPGND.NE.2) GOTO 1
      ZR=- ZR
      T1ZR=- T1ZR
      T2ZR=- T2ZR
    1 RX=XOB- XJ
      RY=YOB- YJ
      RZ=ZOB- ZR
      R2=RX* RX+ RY* RY+ RZ* RZ
      IF(R2.GT.1.D-20) GOTO 2
      EXK=(0.,0.)
      EYK=(0.,0.)
      EZK=(0.,0.)
      EXS=(0.,0.)
      EYS=(0.,0.)
      EZS=(0.,0.)
      RETURN
    2 R=SQRT( R2)
      TT1=- TPI* R
      TT2=TT1* TT1
      RT=R2* R
      ER=CMPLX( SIN( TT1),- COS( TT1))*( CONST* S)
      Q1=CMPLX( TT2-1., TT1)* ER/ RT
      Q2=CMPLX(3.- TT2,-3.* TT1)* ER/( RT* R2)
      ER=Q2*( T1XJ* RX+ T1YJ* RY+ T1ZR* RZ)
      EXK=Q1* T1XJ+ ER* RX
      EYK=Q1* T1YJ+ ER* RY
      EZK=Q1* T1ZR+ ER* RZ
      ER=Q2*( T2XJ* RX+ T2YJ* RY+ T2ZR* RZ)
      EXS=Q1* T2XJ+ ER* RX
      EYS=Q1* T2YJ+ ER* RY
      EZS=Q1* T2ZR+ ER* RZ
      IF(IPGND.EQ.1) GOTO 6
```

```
      IF(IPERF.NE.1) GOTO 3
      EXK=- EXK
      EYK=- EYK
      EZK=- EZK
      EXS=- EXS
      EYS=- EYS
      EZS=- EZS
      GOTO 6
    3 XYMAG=SQRT( RX* RX+ RY* RY)
      IF(XYMAG.GT.1.D-6) GOTO 4
      PX=0.
      PY=0.
      CTH=1.
      RRV=(1.,0.)
      GOTO 5
    4 PX=- RY/ XYMAG
      PY=RX/ XYMAG
      CTH=RZ/ SQRT( XYMAG* XYMAG+ RZ* RZ)
      RRV=SQRT(1.- ZRATI* ZRATI*(1.- CTH* CTH))
    5 RRH=ZRATI* CTH
      RRH=( RRH- RRV)/( RRH+ RRV)
      RRV=ZRATI* RRV
      RRV=-( CTH- RRV)/( CTH+ RRV)
      EDP=( EXK* PX+ EYK* PY)*( RRH- RRV)
      EXK=EXK* RRV+ EDP* PX
      EYK=EYK* RRV+ EDP* PY
      EZK=EZK* RRV
      EDP=( EXS* PX+ EYS* PY)*( RRH- RRV)
      EXS=EXS* RRV+ EDP* PX
      EYS=EYS* RRV+ EDP* PY
      EZS=EZS* RRV
    6 RETURN
      END
```

WIRE

PURPOSE

To compute segment coordinates to fill COMMON/DATA/ for a straight line of Segments.

METHOD

The formal parameters specify the beginning and ending points of the line and the number of segments into which it is to be divided. The code computes the coordinates of the end points of each segment. The lengths of successive segments are scaled by the factor RDEL if this factor is not one. For NS segments, the length of the first segment is

$$S_1 = \frac{L(1-RDEL)}{1-(RDEL)^{NS}}$$

or

$$S_l = L/NS \text{ if } RDEL = l$$

where L is the total length of wire.

The radius is RAD for the first segment and is scaled by RRAD.

SYMBOL DICTIONARY

DELZ	=	segment length
FNS	=	real number equivalent of NS
IST	=	initial segment number
ITG	=	tag number assigned to all segments of the line
NS	=	number of segments into which line is divided
RAD	=	radius of first segment
RADZ	=	segment radius
RD,RDEL	=	scaling factor for segment length
RRAD	=	scaling factor for segment radius
XD	=	increment to x-coordinates
XS1	=	x-coordinate of first end of segment
XS2	=	x-coordinate of second end of segment
XW1	=	x-coordinate of first end of line
XW2	=	x-coordinate of second end of line
X2(1)	=	x-coordinate of end 2 of segment I
YD	=	increment to y coordinates
YS1	=	y-coordinate of first end of segment
YS2	=	y-coordinate of second end of segment
YW1	=	y-coordinate of first end of wire
YW2	=	y-coordinate of second end of wire
Y2(I)	=	y-coordinate of end 2 of segment I
ZD	=	increment to z-coordinates
ZS1	=	z-coordinate of first end of segment
ZS2	=	z-coordinate of second end of segment
ZW1	=	z-coordinate of first end of line
ZW2	=	z-coordinate of second end of line
Z2(I)	=	z-coordinate of second end of segment I

```
      SUBROUTINE WIRE(XW1,YW1,ZW1,XW2,YW2,ZW2,RAD,RDEL,RRAD,NS,ITG)
C
C     SUBROUTINE WIRE GENERATES SEGMENT GEOMETRY DATA FOR A STRAIGHT
C     WIRE OF NS SEGMENTS.
C
      COMMON  /DATA/ LD, N1, N2, N, NP, M1, M2, M, MP, X(NM), Y(NM),
     *Z(NM), SI(NM), BI(NM), ALP(NM), BET(NM), ICON1(N2M), ICON2(
     * N2M), ITAG(N2M), ICONX(NM), WLAM, IPSYM
      DIMENSION  X2(1), Y2(1), Z2(1)
      EQUIVALENCE(X2(1),SI(1)),(Y2(1),ALP(1)),(Z2(1),BET(1))
      IST=N+1
      N=N+ NS
      NP=N
      MP=M
      IPSYM=0
      IF(NS.LT.1) RETURN
      XD=XW2-XW1
      YD=YW2-YW1
      ZD=ZW2-ZW1
      IF(ABS(RDEL-1.).LT.1.D-6) GOTO 1
      DELZ=SQRT(XD* XD+ YD* YD+ ZD* ZD)
      XD=XD/DELZ
      YD=YD/DELZ
      ZD=ZD/DELZ
      DELZ=DELZ*(1.- RDEL)/(1.- RDEL** NS)
      RD=RDEL
      GOTO 2
    1 FNS=NS
      XD=XD/FNS
      YD=YD/FNS
      ZD=ZD/FNS
      DELZ=1.
      RD=1.
    2 RADZ=RAD
      XS1=XW1
      YS1=YW1
      ZS1=ZW1
      DO 3 I=IST, N
      ITAG(I)=ITG
      XS2=XS1+ XD* DELZ
      YS2=YS1+ YD* DELZ
      ZS2=ZS1+ ZD* DELZ
      X(I)=XS1
      Y(I)=YS1
      Z(I)=ZS1
      X2(I)=XS2
      Y2(I)=YS2
      Z2(I)=ZS2
      BI(I)=RADZ
```

```
      DELZ=DELZ* RD                                                     WI  50
      RADZ=RADZ* RRAD                                                   WI  51
      XS1=XS2                                                           WI  52
      YS1=YS2                                                           WI  53
    3 ZS1=ZS2                                                           WI  54
      X2(N)=XW2                                                         WI  55
      Y2(N)=YW2                                                         WI  56
      Z2(N)=ZW2                                                         WI  57
      RETURN                                                            WI  58
      END                                                               WI  59
```

ZINT

PURPOSE

To compute the internal impedance of a circular wire with finite conductivity.

METHOD

The internal impedance per unit length of a circular wire is given by

$$Z = \frac{i}{j}\sqrt{\frac{fp}{2\pi\sigma}}\left[\frac{Ber(q) + jBei(q)}{Ber'(q) + jBei'(q)}\right] \ ,$$

where

q	=	b$\sqrt{2\pi f \mu \sigma}$
σ	=	wire conductivity
μ	=	permeability of free space
b	=	wire radius
f	=	frequency
Ber	=	Kelvin function
Bei	=	Kelvin function

The term that modifies the diagonal matrix element G_{ii} in the interaction matrix is the total impedance of segment i divided by Δ_i/λ, where Δ_i = segment length. Thus, if G_{ii} is the diagonal matrix element without loading, the new element is

$$G_{ii} - Z\Delta_i/(\Delta/\lambda) = G_{ii} - Z\lambda \ .$$

Normalized to wavelength, this term is

$$Z_i = Z\lambda = \frac{j}{(b/\lambda)}\sqrt{\frac{c\mu}{2\pi(\sigma\lambda)}}\left[\frac{Ber(q) + jBei(q)}{Ber'(q) + jBei'(q)}\right] \ ,$$

where

q	=	(b/λ) $\sqrt{2\pi c \mu(\sigma\lambda)}$
u	=	velocity of light

The Kelvin functions and derivatlves of Kelvln functions are computed from their polynomial approximations.

CODING

ZI8-ZI15	Functions θ, Φ, f, and g for large argument polynomial approximations (see ref. 5).
ZI19-ZI26	Compute Ber(q) + jBei(q) for q $\leq$ 8.
ZI27-ZI31	Compute Ber'(q) + jBei'(q) for q $\leq$ 8.
ZI32	[Ber(q) + jBei<q)]/[Ber'(q) + jBei'(q)].
ZI34	Ber(q) + jBei(q) for 8 < q $\leq$ 110.
ZI35	Ber'(q) + jBei'(q) for 8 < q $\leq$ 110.
ZI36	[Ber(q) + jBei(q)]/[Ber'(q) + jBei'(q)].
ZI38	[Ber(q) + jBei(q)]/[Ber'(q) + jBei'(q)] for 110 < q < ∞.
ZI39	Computation of Z_i.

SYMBOL DICTIONARY

BEI	=	Bei(q) or Bei'(q)
BER	=	Ber(q) or Ber'(q)
BR1	=	Ber(q) + jBei(q) or [Ber(q) + jBei(q)]/[Ber'(q) + Bei'(q)]
BR2	=	Ber'(q) + jBei'(q)
CEXP	=	external routine [exp(comp1ex argument)]
CMOTP	=	$c\mu/(2\pi)$
CMPLX	=	external routine (forms complex number)
CN	=	(1 + j)/$\sqrt{2}$
D	=	function argument
F	=	f(D) (see ref. 5)
FJ	=	j
G(D)	=	g(D) (see ref. 5)
PH(D)	=	Φ(X), D = 8/X (see ref. 5)
PI	=	π
POT	=	π/2
ROLAM	=	b/λ
S	=	$(X/8)^4$
SIGL	=	$\sigma\lambda$
SQRT	=	external routine (square root)
TH(D)	=	θ(X), D = 8/X (see ref. 5)
TP	=	2π
TPCMU	=	$2\pi c\mu$; c = velocity of light
X	=	q
Y	=	$(X/8)^2$
ZINT	=	Z_i
1.5707963	=	π/2
3.141592654	=	π
6.283185308	=	2π
60.	=	$c\mu/2\pi$
2.368705E+3	=	$2\pi c\mu$
(0,1)	=	j
(0.70710678,0.70710678)	=	(1 + j)/$\sqrt{2}$
(0.70710678,-0.70710678)	=	limit for q $\rightarrow\infty$ of [Ber(q)+jBei(q)]/[Ber'(q)+jBei'(q)]

Other constants are factors in the polynomial approximations.

```
      FUNCTION ZINT(SIGL,ROLAM)
C
C     ZINT COMPUTES THE INTERNAL IMPEDANCE OF A CIRCULAR WIRE
C
C
      COMPLEX  TH, PH, F, G, FJ, CN, BR1, BR2, ZINT
      COMPLEX  CC1, CC2, CC3, CC4, CC5, CC6, CC7, CC8, CC9, CC10,
     *CC11, CC12, CC13, CC14
      DIMENSION  FJX(2),CNX(2), CCN(28)
      EQUIVALENCE(FJ,FJX),(CN,CNX),(CC1,CCN(1)),(CC2,CCN(3)),(CC3,CCN(5
     *)),(CC4,CCN(7)),(CC5,CCN(9)),(CC6,CCN(11)),(CC7,CCN(13)),(CC8,CCN
     *(15)),(CC9,CCN(17)),(CC10,CCN(19)),(CC11,CCN(21)),(CC12,CCN(23)),
     *(CC13,CCN(25)),(CC14,CCN(27))
      DATA   PI, POT, TP, TPCMU/3.1415926D+0,1.5707963D+0,6.2831853D+0,
     *2.368705D+3/
      DATA CMOTP/60.00/, FJX/0.,1./, CNX/.70710678D+0,.70710678D+0/
      DATA CCN/6.D-7,1.9D-6,-3.4D-6,5.1D-6,-2.52D-5,0.,-9.06D-5,-
     *9.01D-5,0.,-9.765D-4,.0110486D+0,-.0110485D+0,0.,-.3926991D+0,
     *1.6D-6,-3.2D-6,1.17D-5,-2.4D-6,3.46D-5,3.38D-5,5.D-7,2.452D-4,-
     *1.3813D-3,1.3811D-3,-6.25001D-2,-1.D-7,.7071068D+0,.7071068D+0/
      TH(D)=((((( CC1* D+ CC2)* D+ CC3)* D+ CC4)* D+ CC5)* D+ CC6)* D+
     * CC7
      PH(D)=((((( CC8* D+ CC9)* D+ CC10)* D+ CC11)* D+ CC12)* D+ CC13)
     * *D+CC14
      F(D)= SQRT( POT/ D)* EXP(- CN* D+ TH(-8./ X))
      G(D)= EXP( CN* D+ TH(8./ X))/ SQRT( TP* D)
      X=SQRT( TPCMU* SIGL)* ROLAM
      IF(X.GT.110.) GOTO 2
      IF(X.GT.8.) GOTO 1
      Y=X/8.
      Y=Y* Y
      S=Y* Y
      BER=((((((-9.01D-6* S+1.22552D-3)* S-.08349609D+0)* S+
     *2.6419140D+0)* S-32.363456D+0)* S+113.77778D+0)* S-64.)* S+1.
      BEI=((((((1.1346D-4* S-.01103667D+0)* S+.52185615D+0)* S-
     *10.567658D+0)* S+72.817777D+0)* S-113.77778D+0)* S+16.)* Y
      BR1= CMPLX( BER, BEI)
      BER=(((((((-3.94D-6* S+4.5957D-4)* S-.02609253D+0)* S+
     *.66047849D+0)* S-6.0681481D+0)* S+14.222222D+0)* S-4.)* Y)* X
      BEI=((((((4.609D-5* S-3.79386D-3)* S+.14677204D+0)* S-
     *2.3116751D+0)* S+11.377778D+0)* S-10.666667D+0)* S+.5)* X
      BR2=CMPLX(BER,BEI)
      BR1=BR1/BR2
      GOTO 3
    1 BR2=FJ*F(X)/PI
      BR1=G(X)+BR2
      BR2=G(X)*PH(8./X)-BR2*PH(-8./X)
      BR1=BR1/BR2
      GOTO 3
```

```
2 BR1=CMPLX(.70710678D+0,-.70710678D+0)                               ZI  50
3 ZINT=FJ*SQRT(CMOTP/SIGL)*BR1/ROLAM                                  ZI  51
  RETURN                                                              ZI  52
  END                                                                 ZI  53
```

```
      SUBROUTINE STROPC( STRING, STRING1)
      CHARACTER *(*)  STRING, STRING1
      INTEGER*4  I, J, IC
      INTEGER IS_PC

      IS_PC = 0

      DO 150, I=1, LEN( STRING)
      IC= ICHAR( STRING( I: I))

      IF(IS_PC .NE. 0) THEN
         IF(IC.GE.97.AND. IC.LE.122) IC= IC-32
      ENDIF

      STRING1( I: I)= CHAR( IC)
  150 CONTINUE

      RETURN
      END
```

Section III

Common Blocks

This section discusses each labeled common block which is used in the NEC 2 code. For each common block, a list of the routines in which it is used is given along with a definition of the variables used in conjunction with the common block. The common blacks are presented in alphabetical order.

COMMON/ANGL/ SALP(300)

Routines Using /ANGL/

CABC, CMSS, CMSW, CMWS, CMWW, DATAGN, ETMNS, FFLD, GFIL, GFLD, GFOUT, MOVE, NEFLD, NHFLD, PATCH, QDSRC, KEFLC

/ANGL/ Parameters for Wire Segments

SALP(I) = sin (α), where α = elevation angle of segment I (see figure 11)

/ANGL/ Parameters for Surface Patches

SALP(LD-I+1) = +1 if $\hat{t}_1 \times \hat{t}_2 = \hat{n}$ for patch I, or -1 if $\hat{t}_1 \times \hat{t}_2 = -\hat{n}$ for patch I

The second case occurs when the patch has been produced by reflection of a patch originally input.

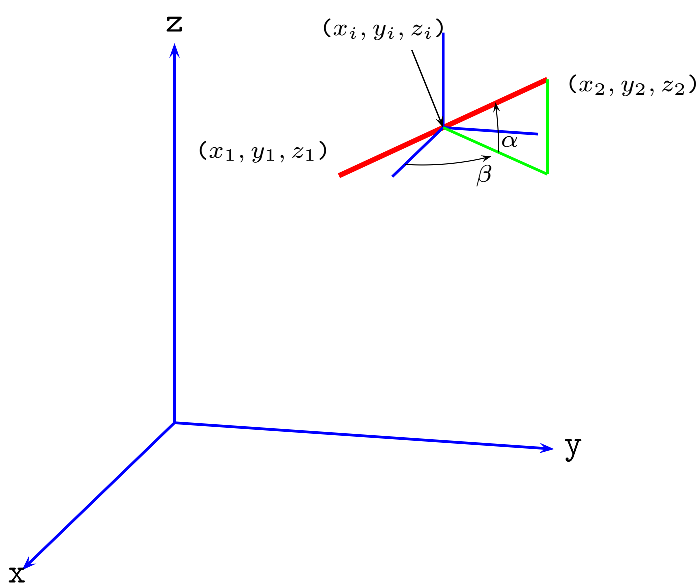


Figure 11. Coordinates of Segment i.

COMMON/CMB/ CM(4000)

Routines Using /CMB/

MAIN, GFIL, GFQHT

The interaction matrix is stored in array CM. If the matrix is too large to fit in CM, then pairs of blocks of the matrix are stored in GM as they are needed.

COMMON/CRNT/ AIR(300),AII(300),BIR(300),CIR(300),CII(300),CUR(900)

Routines using /CRNT/

MAIN, CABC, FFLD, GFLD, NEFLD, NETWK, NHFLD

/CRNT/ Parameters for Wire Segments

Subroutine CABC fills the first six arrays in /CRNT/ with the real and imaginary parts of the constants in the current expansion of each segment,

$$I_i(s) = A_i + B_i \sin[k(s - s_i)] + C_i \cos[k(s - s_i)] \ ,$$

where $s = s_i$ at the center of segment i. Except during intermediate calculations for non-radiating networks, the current basis-function amplitudes are computed and stored in array CUR. CABC replaces the basis function amplitudes in CUR by the current at the center of each segment, $(A_i + C_i)$. For i = I,

AIR(I),AII(I)	=	A_i/λ (real,imaginary)
BIR(I),BII(I)	=	B_i/λ (real,imaginary)
CIR(I),CII(I)	=	C_i/λ (real,imaginary)
CUR(I)	=	amplitude of x basis function going into CABC or $(A_i + C_i)/\lambda$ at end of CABC

/CRNT/ Parameters for Surface Patches

Surface current components are stored in CUR. Before CABC is called, the surface current strengths in directions $\hat{t}_1$ and $\hat{t}_2$ on patch i are stored in CUR(N + 2I - 1) and CUR(N + 2I), respectively where N is the number of segments. After CABC, the x, y and z components of surface current are stored in CUR(N + 3I - 2), CUR(N + 3I - 1) and CUR(N + 3I), respectively.

```
COMMON/DATA/ LD,N1,N2,N,NP,M1,M2,M,MP,X(300),Y(300),Z(300),SI(300),
BI(300),ALP(300),BET(300),ICON1(300),ICON2(300),ITAG(300),ICONX(300),WLAM,IPSM
```

Routines Using /DATA/

MAIN, ARC, CABC, CMNGF, CMSET, CMSS, CMSW, CMWS, CMWW, CONECT, DATAGN, ETMNS, FFLD, FFLDS, GFIL, GFLD, GFOUT, ISEGNO, LOAD, MOVE, NEFLD, NETWK, NFPAT, NHFLD, PATCH, QDSRC, RDPAT, REFLC, SBF, TBF, TRIO, WIRE

/DATA/ Parameters for Wire Segments

The arrays in /DATA/ are used to store the parameters defining the segments. Two forms of the segment parameters are used.

During geometry input in routines ARC, CONECT, DATAGN, MOVE, REFLEC and WIRE, the coordinates of the segment ends are stored. The symbol meanings in the geometry routines are:

X(I)	=	X_1
Y(I)	=	Y_1
Z(I)	=	Z_1
SI(I)	=	X_2 [equivalencesd to X2(I)]
ALP(I)	=	Y_2 [equivalenced to Y2(I)]
BET(I)	=	Z_2 [equivalenced to Z2(I)]

where X_1, Y_1, Z_1 are the coordinates of the first end of the segment, and X_2, Y_2, Z_2 are the coordinates of the second end, as illustrated in figure 11. Coordinates may have any units but must be scaled to meters before data input is ended, since the main program requires meters.

In the main program, the segment data is converted to: the coordinates of the segment center, components of the unit vector in the direction of the segment, and the segment length. The symbol meanings after the geometry section are:

X(I),Y(I),Z(I)	=	X_i, Y_i, Z_i (see figure 11.)
SI(I)	=	segment length
ALP(I)	=	cos α cos β [equivalenced to CAB(I)]
BET(I)	=	cos α sin β [equivalenced to SAB(I)]

The z component of the unit vector in the direction of the segment, sin α, is stored in /ANGL/.

The other symbol meanings in /DATA/ for segments are:

BI(I) = radius of segment I

ICON1(I) = connection number for end 1 of segment I. If k is a positive integer less than 10,000, the meaning of ICON1 is as follows.

0: no connection.

$\pm$k: end 1 connects to segment k. If more than one segment connects to end 1 of segment I, then k is the number of the next connected segment encountered by starting at I and going through the list of segments in cyclic order.

+k: parallel reference directions with end 2 of the other segment connecting to end 1 of segment I.

-k: opposed reference directions.

1: end 1 of segment I connects to a ground plane.

10,000+k: end 1 of segment I connects to a surface with the 4 patches around the connection point numbered k, k+1, k+2 and k+3.

ICON2(I) = connection number for end 2 of segment I.

ITAG(I) = tag number of segment I. This number is assigned during structure input to permit later reference to the segment without knowing the segment index I in the data arrays.

ICONX(I) = equation number for the new basis function when segment I is in a numerical Green's function file and a new segment connects to segment I modifying the old basis function.

/DATA/ Parameters for Surface Patches

Patch parameters are set in subroutine PATCH. The input parameters for a patch are the coordinates of the patch center, patch area, and orientation of the outward, normal unit vector, $\hat{n}$. The parametere stored in /DATA/ are the center point coordinates, area, and the components of the two surface unit vectors, $\hat{t}_1$ and $\hat{t}_2$. The vector $\hat{t}_1$ is parallel to a side of the triangular, rectangular, or quadrilateral patch. For a patch of arbitrary shape, it is chosen by the following rules:

For a horizontal patch, $\hat{t}_1 = \hat{x}$;
For a nonhorizontal patch, $\hat{t}_1 = (\hat{z} \times \hat{n})/|\hat{z} \times \hat{n}|$;
$\hat{t}_2$ is then chosen as $\hat{t}_2 = \hat{n} \times \hat{t}_1$

with J = LD + 1 - I, the parameters for patch I are stored as follows.

```
X(J),Y(J),Z(J)              =  x, y, and z coordinates of the patch center
SI(J),ALP(J),BET(J)         =  x, y, z components of t̂1 (equivalences to T1X,T1Y,T1Z)
ICON1(J),ICON2(J),ITAC(J)   =  x, y, and z components of t̂2 (equivalenced to T2X,T2Y,T2Z)
BI(J)                       =  patch area
```

Scalar variables in /DATA/ are:

```
IPSYM  =  symmmetry flag.  The meanings of IPSYM are:
          0:  no symmetry
          >0:  plane symmetry
          <0:  cylindrical symmetry
          2:  plane symmetry about Z = 0
          >2:  structure has been rotated about x or y axis.  If
          ground plane is indicated by IGND≠0 in the call
          to subroutine CONECT and IPSYM = 2, symmetry about
          horizontal plane is removed by multiplying NP by 2.
          If |IPSYM|>2 and IGND≠0, all symmetry is
          removed by setting NP = N and IPSYM = 0 in CONECT.
LD     =  length of arrays in /DATA/
N1     =  number of segments in NGF. If NGF is not used NI=0
N2     =  N1 + 1
N      =  total number of segments
NP     =  number of segments in a symmetric cell
M1     =  number of patches in NGF. If NCF is not used M1=0
M2     =  M1 + 1
M      =  total number of patches
MP     =  number of patches in a symmetric cell
WLAM   =  wavelength in meters
```

```
COMMON/DATAJ/ S,B,XJ,YJ,ZJ,CABJ,SABJ,SALPJ,EXK,EYK,EZK,EXS,EYS,
   EZS,EXC,EYC,EZC,KKH,IEXK,IND1,IND2,IPGND
```

Routines Using /DATAJ/

CMNGF,CMSET,CMSS,CMSW,CMWS,CMWW,EFLD,HINTS,HSFLD,NEFLD,NHFLD,
PCINT,QDSRC,SFLDS,UNERE

/DATAJ/ is used to pass the parameters of the source segment or patch to the routines that compute the E or H field and to return the field components.

/DATAJ/ Parameters for Wire Segments

S	=	segment length
B	=	segment radius
XJ,YJ,ZJ	=	coordinates of segment center
CABJ,SABJ,SALPJ	=	x, y, and z, respectively, of the unit vector in the direction of the segment
EXK,EYK,EZK	=	x, y, and z components of the E or H field due to a constant current
EXS,EYS,EZS	=	x, y, and z components of the E or H field due to a sin ks current
EXC,EYC,EZC	=	x, y, and z components of the s or H field due to cos ks current
RKH	=	minimum distance for use of the Hertzian dipole approximation for computing the E field of a segment
IEXK	=	flag to select thin wire approximation or extended thin wire approximation for S field (IEXK=l for extended thin wire approximation)
IND1	=	flag to inhibit use of the extended thin wire approximation on end 1 of the source segment. This is used when there is a bend or change in radius at end 1. IND1=2 inhibits the extended thin wire approximation.
IND2	=	flag to inhibit use of the extended thin wire approximation on end 2 of the source segment
IPGND	=	not used

/DATAJ/ Parameters for Surface Patches

S	=	patch area in units of wavelength squared
B	=	x component of $\hat{t}_2$ for the patch
XJ,YJ,ZJ	=	x, y, and z components of the position of the patch center
CABJ,SABJ,SALPJ	=	x, y, and z components of $\hat{t}_1$
EXK,EYK,EZK	=	x, y, and z components of $\vec{E}$ or $\vec{H}$ due to a current with unit magnitude in the direction $\hat{t}_1$ on the patch
EXS,EYS,EZS	=	$\vec{E}$ or $\vec{H}$ due to a current $\hat{t}_2$2 on the patch
EXC,EYC,EZC	=	not used; may serve as intermediate variables in some routines
IND1	=	y component of $\hat{t}_2$
IND2	=	z component of $\hat{t}_2$
IPGND	=	flag to request calculation of the direct field or field reflected from the ground (two for ground)

```
COMMON/FPAT/ NTH,NPH,IPD,IAVP,INOR,IAX,THETS,PHIS,DTH,DPH,RFLD,GNOR,CLT,
    CHT,EPSR2,SIG2,IXTYP,XPR6,PINR,PNLR,PLOSS,NEAR,NFEH,NRX,
    NRY,NKZ,XNR,YNR,ZNR,DXNR,DYNR,DZNR
```

Routines Using /FPAT/

MAIN,NFPAT,RDPAT
Variables are defined in subroutine descriptions.

```
COMMON/GGRID/ AR1(11,10,4),AR2(17,5,4),AR3(9,8,4),EPSCF,DXA(3),DYA(3),
    XSA(3),YSA(3),NXA(3),NYA(3)
```

Routines Using /GGRID/

MAIN,GFIL,GFOLIT,INTRP
Variables are defined under subroutine INTKP.

```
COMMON/GND/ ZRATI,ZRATI2,FRATI,CL,CH,SCRWL,SCRWR,NRADL,KSYMP,IFAR,IPERF,T1,T2
```

Routines Using /GND/

MAIN,CMSN,EFLD,ETMNS,FFLD,GFIL,GFOUT,HINTS,HSFLD,NEFLD,RDPAT,SFLDS,UNERE

/GND/ contains parameters of the ground including the two-medium ground and radial-wire ground-screen cases. The symbol definitions are as follows.

ZRATI	=	$\lvert\epsilon_r - j\sigma/\omega\epsilon_0\rvert^{-1/2}$ σ is ground conductivity (mhos/meter) ϵ_r is the relative dielectric constant ϵ_0 is the permittivity of free space (farads/meter) $\omega = 2\pi f$.
ZRATI2	=	same as ZRATI, but for a second ground medium
FRATI	=	$(k_1^2 - k_2^2)/(k_1^2 + k_2^2)$ where $k_2 = \omega\sqrt{\mu_0\epsilon_0}$ and $k_1 = k_2$/ZRATI
CL	=	distance in wavelengths of cliff edge from origin
CH	=	cliff height in wavelengths
SCRAWL	=	length of wires in radial-wire ground screen (normalized to wavelength)
SCRWR	=	radius of wires in screen in wavelengths
NRADL	=	number of radials in ground screen; zero implies no screen (input quantity, GN card)
KSYMP	=	ground flag (*1, no ground; =2, ground present)
IFAR	=	input integer flag on RE card; specifies type of field computation or type of ground system for far fields
IPERF	=	flag to select type of ground (see GN card)
Tl,T2	=	constants for the radial-wire ground-screen impedance

---

COMMON/GWAVE/ U,U2,XX1,XX2,R1,R2,ZMH,ZPH

Routines Using /GWAV/

MAIN,GFLD,GWAVE,SFLDS

Symbol Definitions

U	=	$\lvert\epsilon_r - j\sigma/\omega\epsilon_0\rvert^{-1/2}$ σ is ground conductivity (mhos/meter) ϵ_r is the relative dielectric constant ϵ_0 is the permittivity of free space (farads/meter) $\omega = 2\pi f$.
U2	=	U^2
XX1,XX2	=	defined in GFLD and SFLDS
R1	=	distance from current element to point at which field is evaluated
R2	=	distance from image of current element to point at which field is evaluated
ZMH	=	Z - Z'
ZPH	=	Z + Z' where Z is height af the field evaluation point and Z' is the height of the current element

COMMON/INCOM/ XO,YO,ZO,SN,XSN,YSN,ISNOR

Routines Using /INCOM/

EFLD,SFLDS

Symbol Definitions:

XO,YO,ZO	=	point at which field due to ground will be evaluated
SN	=	cos α (see Figure 11)
XSN	=	cas β
YSN	=	sin β
ISNOR	=	1 to evaluate field due to ground by interpolation 0 to use Norton's approximation

```
COMMON/MATPAR/ ICASE,NBLOKS,NPBLK,NLAST,NBLSYM,NPSYM,NLSYM,NMAT,ICASX,
   NBBX,NPBX,NLBX,NBBL,NPBL,NLBL
```

Routines Using /MATPAR/

MAIN,CMNGF,CMSET,FACGF,FACIO,FACTR5,FBLOCK,FBNGF,CFIL,GFOUT,
LFACTR,LTSOLV,LUNSCK,REBLK,SOLCF,SOLVES

/MATPAR/ contains matrix blacking parameters for cases requiring file storage of the matrix. Symbol definitions in /MATPAR/ are as follows.

ICASE = storage made for primary matrix, defined as follows.
1 - unsymmetric matrix fits in core
2 - symmetric matrix fits in core
3 - unsymmetric matrix out of care
4 - symmetric matrix out of cure, but submatrices fit in care
5 - symmetric matrix out of core, submatrices also out of care

Section IX — SOMNEC

I. SOMNEC CODE DESCRIPTION

SOMNEC is an independent code that generates the interpolation tables for the Sommerfeld/Norton ground option for NEC. The tables are written on file TAPE21 which becomes an input file to NEC. Coding of the routines in SOMNEC is described in this section.

SOMNEC

PURPOSE

To generate interpolation tables for the Sommerfeld/Norton ground option and write them on file TAPE21.

METHOD

The code from SN17 to SN51 reads the input data and sets parameters in COMMON/EVLCOM/. Since all equations are scaled to a free-space wavelength of one meter the results depend only on the complex dielectric constant

$$\epsilon_c = \epsilon_1 - j\sigma_1/(\omega\epsilon_0)\ .$$

In the routines that evaluate the Sommerfeld integrals the time dependence is exp(-jωt) rather than exp(+jωt) which is used in the remainder of NEC. Hence the conjugate of ϵ_c (EPSCF) is taken before computing the parameters in COMMON/EVLCOM/. The conjugate of the results is taken at the end of EVLUA, so the results returned to SOMNEC and written on TAPE21 are for exp(+jωt).

Three interpolation tables, as shown in Figure 12 of Part I, are generated in the code from SN55 to SN123. For each R_1, θ pair in the tables the values of ρ and z + z' are computed and stored in COMMON/EVLCOM/. Subroutine EVLUA is then called and returns the quantities

$$ERV = \frac{\partial^2}{\partial\rho\partial z} k_1^2 V'_{22}$$

$$EZV = (\frac{\partial^2}{\partial z^2} + k_2^2) k_1^2 V'_{22}$$

$$ERH = (\frac{\partial^2}{\partial\rho^2} k_2^2 V'_{22} + k_2^2 U'_{22})$$

$$EPH = -(\frac{1}{\rho}\frac{\partial}{\partial\rho} k_2^2 k_2^2 V'_{22} + k_2^2 U'_{22})$$

These are multiplied by C_1 R_1 exp(jkR_1) to form the quantities in equation (156) through (159) in Part I. When R_1 is zero the limiting forms in equations (169) through (172) of Part I are used. The expressions from SN116 to SN118 are obtained by letting θ go to zero in the expreesions for R_1 = 0.

The data are stored in COMMON/GGRID/ which is identical to the common block in NEC. File 21 is written at SN127 and includes coordinates of the grid boundaries, number of points, and increments for R_1 and θ. Hence those grid parameters can be changed in SOMNEC without changing NEC. If the number of grid points is increased, however, the arrays in COMMON/GGRID/ must be increased in both SOMNEC and NEC. Also, the parameters NDA and NDPA in subroutine INTRP must be changed.

SYMBOL DICTIONARY

AR1	=	array for grid 1
AR2	=	array for grid 2
AR3	=	array for grid 3
CK1	=	k_1
CK1R	=	real part of k_1
CKISQ	=	k_1^2
CK2	=	k_2 (= 2π since λ = 1)
CK2SQ	=	k_2^2
CKSM	=	$k_2^2/(k_1^2+k_2^2)$
CL1	=	$k_2^2 C_l C_3$ (see Part I for C_1, C_2, and C_3)
CL2	=	$k_2^2 C_l C_2$
CON	=	$C_1 R_1 \exp(jkR_1)$
CT1	=	$(k_1^2-k_2^2)/2$
CT2	=	$(k_1^4-k_2^4)/8$
CT3	=	$(k_1^6-k_2^6)/16$
DR	=	ΔR_1
DTH	=	$\Delta\theta$
DXA	=	ΔR_1 for each grid
DYA	=	$\Delta\theta$ for each grid (radians)
EPR	=	ϵ_1
EPSCF	=	ϵ_c
EPH,ERH,ERV,EZV	=	EPH,ERH,ERV,EZV
FMHZ	=	frequency in MHz
IPT	=	flag to control printing of grid
IR	=	index for R_1 values
IRS	=	starting value for IR
ITH	=	index for θ values
LCOMP	=	labels for output
NR	=	number of R_1 values
NTH	=	number of θ values
NXA	=	number of R_1 values for each grid
NYA	=	number of θ values for each grid
R	=	R_1
RHO	=	ρ
RK	=	$k_2 R$
SIG	=	σ_1
TFACI	=	(1 - sin θ)/cos θ
TFAC2	=	(1 - sin θ)/cos$^2\theta$
THET	=	θ
TIM	=	time to fill arrays
TKMAG	=	100$\cdot\|k_1\|$
TSMAG	=	100$\cdot\|k_1\|^2$
TST	=	starting time
WLAM	=	wavelength in free space
XSA	=	starting value of R_1 in each grid
YSA	=	starting value of θ in each grid
ZPH	=	Z + Z’
59.96	=	1/$(2\pi c\epsilon_0)$, c = velocity of light

```
c     program somnec(input,output,tape21)
c
c     program to generate nec interpolation grids for fields due to
c     ground.  field components are computed by numerical evaluation
c     of modified sommerfeld integrals.
c
      program somnec
c

      implicit real*8 (a-h,o-z)
      real secnds,tst
      complex*16 ck1,ck1sq,erv,ezv,erh,eph,ar1,ar2,ar3,epscf,cksm,ct1,
     *        ct2,ct3,cl1,cl2,con
      common/evlcom/ cksm,ct1,ct2,ct3,ck1,ck1sq,ck2,ck2sq,tkmag,
     *               tsmag,ck1r,zph,rho,jh
      common/ggrid/ ar1(11,10,4),ar2(17,5,4),ar3(9,8,4),epscf,
     *              dxa(3),dya(3),xsa(3),ysa(3),nxa(3),nya(3)
      dimension lcomp(4)
      character*32 otfile
      data nxa/11,17,9/,nya/10,5,8/,xsa/0.,.2,.2/,ysa/0.,0.,.3490658504/
      data dxa/.02,.05,.1/,dya/.1745329252,.0872654626,.1745329252/
      data lcomp/3herv,3hezv,3herh,3heph/
c
c     read ground parameters - epr = relative dielectric constant
c                              sig = conductivity (mhos/m)
c                              fmhz = frequency (mhz)
c                              ipt = 1 to print grids.  =0 otherwise.
c     if sig .lt. 0. then complex dielectric constant = epr + j*sig
c     and fmhz is not used
c
c     read 15, epr,sig,fmhz,ipt

      print 100
100   format(' program to calculate ground interpolation grid')
      print 101
101   format(' for nec2 using sommerfeld-norton method')
      print 102
102   format(' ')
      print 103
103   format(' enter relative dielectric constant:')
      read *, epr
      print 104
104   format(' enter conductivity (mhos/meter):')
      read *, sig
      print 105
105   format(' enter frequency (mhz):')
      read *, fmhz
      print 106
106   format(' enter 1 to print grids, 0 to suppress printing:')
```

```
      read *, ipt
      print 107
107   format(' enter data output filename:')
      read 24, otfile
      print *, ' relative dielectric constant = ', epr
      print *, ' conductivity (mhos/meter) = ', sig
      print *, ' frequency, mhz = ', fmhz
      print *, ' printing flag = ', ipt
      print *, ' data output file name = ', otfile
      if (sig.lt.0) go to 1
      wlam=299.8/fmhz
      epscf=cmplx(epr,-sig*wlam*59.96)
      go to 2
1     epscf=cmplx(epr,sig)
2     tst=secnds(0.0)
      ck2=6.283185308
      ck2sq=ck2*ck2
c
c     sommerfeld integral evaluation uses exp(-jwt), nec uses exp(+jwt),
c     hence need dconjg(epscf).  conjugate of fields occurs in subroutine
c     evalua.
c
      ck1sq=ck2sq*dconjg(epscf)
      ck1=cdsqrt(ck1sq)
      ck1r=dreal(ck1)
      tkmag=100.*cdabs(ck1)
      tsmag=100.*ck1*dconjg(ck1)
      cksm=ck2sq/(ck1sq+ck2sq)
      ct1=.5*(ck1sq-ck2sq)
      erv=ck1sq*ck1sq
      ezv=ck2sq*ck2sq
      ct2=.125*(erv-ezv)
      erv=erv*ck1sq
      ezv=ezv*ck2sq
      ct3=.0625*(erv-ezv)
c
c     loop over 3 grid regions
c
      do 6 k=1,3
      nr=nxa(k)
      nth=nya(k)
      dr=dxa(k)
      dth=dya(k)
      r=xsa(k)-dr
      irs=1
      if (k.eq.1) r=xsa(k)
      if (k.eq.1) irs=2
c
c     loop over r.  (r=sqrt(rho**2 + (z+h)**2))
```

```
c
      do 6 ir=irs,nr
      r=r+dr
      thet=ysa(k)-dth
c
c     loop over theta.  (theta=atan((z+h)/rho))
c
      do 6 ith=1,nth
      thet=thet+dth
      rho=r*cos(thet)
      zph=r*sin(thet)
      if (rho.lt.1.e-7) rho=1.e-8
      if (zph.lt.1.e-7) zph=0.
      call evlua (erv,ezv,erh,eph)
      rk=ck2*r
      con=-(0.,4.77147)*r/cmplx(cos(rk),-sin(rk))
      go to (3,4,5), k
3     ar1(ir,ith,1)=erv*con
      ar1(ir,ith,2)=ezv*con
      ar1(ir,ith,3)=erh*con
      ar1(ir,ith,4)=eph*con
      go to 6
4     ar2(ir,ith,1)=erv*con
      ar2(ir,ith,2)=ezv*con
      ar2(ir,ith,3)=erh*con
      ar2(ir,ith,4)=eph*con
      go to 6
5     ar3(ir,ith,1)=erv*con
      ar3(ir,ith,2)=ezv*con
      ar3(ir,ith,3)=erh*con
      ar3(ir,ith,4)=eph*con
6     continue
c
c     fill grid 1 for r equal to zero.
c
      cl2=-(0.,188.370)*(epscf-1.)/(epscf+1.)
      cl1=cl2/(epscf+1.)
      ezv=epscf*cl1
      thet=-dth
      nth=nya(1)
      do 9 ith=1,nth
      thet=thet+dth
      if (ith.eq.nth) go to 7
      tfac2=cos(thet)
      tfac1=(1.-sin(thet))/tfac2
      tfac2=tfac1/tfac2
      erv=epscf*cl1*tfac1
      erh=cl1*(tfac2-1.)+cl2
      eph=cl1*tfac2-cl2
```

```
      go to 8
7     erv=0.
      erh=cl2-.5*cl1
      eph=-erh
8     ar1(1,ith,1)=erv
      ar1(1,ith,2)=ezv
      ar1(1,ith,3)=erh
9     ar1(1,ith,4)=eph
      tim=secnds(tst)
c
c     write grid on tape21
c
      open(unit=21,file=otfile,form='unformatted',status='new',err=21)

      write(21) ar1,ar2,ar3,epscf,dxa,dya,xsa,ysa,nxa,nya
      close (unit=21)
      if (ipt.eq.0) go to 14
c
c     print grid
c
      print 17, epscf
      do 13 k=1,3
      nr=nxa(k)
      nth=nya(k)
      print 18, k,xsa(k),dxa(k),nr,ysa(k),dya(k),nth
      do 13 l=1,4
      print 19, lcomp(l)
      do 13 ir=1,nr
      go to (10,11,12), k
10    print 20, ir,(ar1(ir,ith,l),ith=1,nth)
      go to 13
11    print 20, ir,(ar2(ir,ith,l),ith=1,nth)
      go to 13
12    print 20, ir,(ar3(ir,ith,l),ith=1,nth)
13    continue
14    continue
      print 16, tim
      go to 23
21    print 22, otfile
23    stop
c
15    format (3e10.3,i5)
16    format (6h time=,e12.3,8h seconds)
17    format (30h nec ground interpolation grid,/,21h dielectric constan
     1t=,2e12.5)
18    format (///,5h grid,i2,/,4x,5hr(1)=,f7.4,4x,3hdr=,f7.4,4x,3hnr=,i3
     1,/,9h thet(1)=,f7.4,3x,4hdth=,f7.4,3x,4hnth=,i3,//)
19    format (///1x,a3)
20    format (4h ir=,i3,/1x,(10(1pe12.5)))
```

```
22    format ('error creating output file = ',a)
24    format (a)
      end
```

```
c ***
c
c
      subroutine bessel (z,j0,j0p)
c
c     bessel evaluates the zero-order bessel function and its derivative
c     for complex argument z.
c
      implicit real*8 (a-h,o-z)
      complex*16 j0,j0p,p0z,p1z,q0z,q1z,z,zi,zi2,zk,fj,cz,sz,j0x,j0px
      dimension m(101), a1(25), a2(25), fjx(2)
      equivalence (fj,fjx)
      data c3,p10,p20,q10,q20/.7978845608,.0703125,.1121520996,
     1.125,.0732421875/
      data p11,p21,q11,q21/.1171875,.1441955566,.375,.1025390625/
      data pof,init/.7853981635,0/,fjx/0.,1./
      if (init.eq.0) go to 5
1     zms=z*dconjg(z)
      if (zms.gt.1.e-12) go to 2
      j0=(1.,0.)
      j0p=-.5*z
      return
2     ib=0
      if (zms.gt.37.21) go to 4
      if (zms.gt.36.) ib=1
c     series expansion
      iz=1.+zms
      miz=m(iz)
      j0=(1.,0.)
      j0p=j0
      zk=j0
      zi=z*z
      do 3 k=1,miz
      zk=zk*a1(k)*zi
      j0=j0+zk
3     j0p=j0p+a2(k)*zk
      j0p=-.5*z*j0p
      if (ib.eq.0) return
      j0x=j0
      j0px=j0p
c     asymptotic expansion
4     zi=1./z
      zi2=zi*zi
      p0z=1.+(p20*zi2-p10)*zi2
      p1z=1.+(p11-p21*zi2)*zi2
      q0z=(q20*zi2-q10)*zi
      q1z=(q11-q21*zi2)*zi
      zk=cdexp(fj*(z-pof))
      zi2=1./zk
```

```
      cz=.5*(zk+zi2)
      sz=fj*.5*(zi2-zk)
      zk=c3*cdsqrt(zi)
      j0=zk*(p0z*cz-q0z*sz)
      j0p=-zk*(p1z*sz+q1z*cz)
      if (ib.eq.0) return
      zms=cos((sqrt(zms)-6.)*31.41592654)
      j0=.5*(j0x*(1.+zms)+j0*(1.-zms))
      j0p=.5*(j0px*(1.+zms)+j0p*(1.-zms))
      return
c     initialization of constants
5     do 6 k=1,25
      a1(k)=-.25/(k*k)
6     a2(k)=1./(k+1.)
      do 8 i=1,101
      test=1.
      do 7 k=1,24
      init=k
      test=-test*i*a1(k)
      if (test.lt.1.e-6) go to 8
7     continue
8     m(i)=init
      go to 1
      end
```

```
c ***
c
c
      subroutine evlua (erv,ezv,erh,eph)
c
c     evlua controls the integration contour in the complex lambda
c     plane for evaluation of the sommerfeld integrals.
c
      implicit real*8 (a-h,o-z)
      complex*16 erv,ezv,erh,eph,a,b,ck1,ck1sq,bk,sum,delta,ans,
     *           delta2,cp1,cp2,cp3,cksm,ct1,ct2,ct3
      common /cntour/ a,b
      common /evlcom/ cksm,ct1,ct2,ct3,ck1,ck1sq,ck2,ck2sq,tkmag,tsmag,c
     1k1r,zph,rho,jh
      dimension sum(6), ans(6)
      data ptp/.6283185308/
      del=zph
      if (rho.gt.del) del=rho
      if (zph.lt.2.*rho) go to 4
c
c     bessel function form of sommerfeld integrals
c
      jh=0
      a=(0.,0.)
      del=1./del
      if (del.le.tkmag) go to 2
      b=dcmplx(.1*tkmag,-.1*tkmag)
      call rom1 (6,sum,2)
      a=b
      b=cmplx(del,-del)
      call rom1 (6,ans,2)
      do 1 i=1,6
1     sum(i)=sum(i)+ans(i)
      go to 3
2     b=cmplx(del,-del)
      call rom1 (6,sum,2)
3     delta=ptp*del
      call gshank (b,delta,ans,6,sum,0,b,b)
      go to 10
c
c     hankel function form of sommerfeld integrals
c
4     jh=1
      cp1=cmplx(0.,.4*ck2)
      cp2=cmplx(.6*ck2,-.2*ck2)
      cp3=cmplx(1.02*ck2,-.2*ck2)
      a=cp1
      b=cp2
      call rom1 (6,sum,2)
```

```
      a=cp2
      b=cp3
      call rom1 (6,ans,2)
      do 5 i=1,6
5     sum(i)=-(sum(i)+ans(i))
c     path from imaginary axis to -infinity
      slope=1000.
      if (zph.gt..001*rho) slope=rho/zph
      del=ptp/del
      delta=cmplx(-1.,slope)*del/sqrt(1.+slope*slope)
      delta2=-dconjg(delta)
      call gshank (cp1,delta,ans,6,sum,0,bk,bk)
      rmis=rho*(dreal(ck1)-ck2)
      if (rmis.lt.2.*ck2) go to 8
      if (rho.lt.1.e-10) go to 8
      if (zph.lt.1.e-10) go to 6
      bk=cmplx(-zph,rho)*(ck1-cp3)
      rmis=-dreal(bk)/dabs(dimag(bk))
      if(rmis.gt.4.*rho/zph)go to 8
c     integrate up between branch cuts, then to + infinity
6     cp1=ck1-(.1,.2)
      cp2=cp1+.2
      bk=cmplx(0.,del)
      call gshank (cp1,bk,sum,6,ans,0,bk,bk)
      a=cp1
      b=cp2
      call rom1 (6,ans,1)
      do 7 i=1,6
7     ans(i)=ans(i)-sum(i)
      call gshank (cp3,bk,sum,6,ans,0,bk,bk)
      call gshank (cp2,delta2,ans,6,sum,0,bk,bk)
      go to 10
c     integrate below branch points, then to + infinity
8     do 9 i=1,6
9     sum(i)=-ans(i)
      rmis=dreal(ck1)*1.01
      if (ck2+1..gt.rmis) rmis=ck2+1.
      bk=cmplx(rmis,.99*dimag(ck1))
      delta=bk-cp3
      delta=delta*del/cdabs(delta)
      call gshank (cp3,delta,ans,6,sum,1,bk,delta2)
10    ans(6)=ans(6)*ck1
c     conjugate since nec uses exp(+jwt)
      erv=dconjg(ck1sq*ans(3))
      ezv=dconjg(ck1sq*(ans(2)+ck2sq*ans(5)))
      erh=dconjg(ck2sq*(ans(1)+ans(6)))
      eph=-dconjg(ck2sq*(ans(4)+ans(6)))
      return
      end
```

```
c ***
c
c
      subroutine gshank (start,dela,sum,nans,seed,ibk,bk,delb)
c
c     gshank integrates the 6 sommerfeld integrals from start to
c     infinity (until convergence) in lambda.  at the break point, bk,
c     the step increment may be changed from dela to delb.  shank's
c     algorithm to accelerate convergence of a slowly converging series
c     is used
c
      implicit real*8 (a-h,o-z)
      complex*16 start,dela,sum,seed,bk,delb,a,b,q1,q2,ans1,ans2,
     *           a1,a2,as1,as2,del,aa
      common /cntour/ a,b
      dimension q1(6,20), q2(6,20), ans1(6), ans2(6), sum(6), seed(6)
      data crit/1.d-4/,maxh/20/
      rbk=dreal(bk)
      del=dela
      ibx=0
      if (ibk.eq.0) ibx=1
      do 1 i=1,nans
1     ans2(i)=seed(i)
      b=start
2     do 20 int=1,maxh
      inx=int
      a=b
      b=b+del
      if (ibx.eq.0.and.dreal(b).ge.rbk) go to 5
      call rom1 (nans,sum,2)
      do 3 i=1,nans
3     ans1(i)=ans2(i)+sum(i)
      a=b
      b=b+del
      if (ibx.eq.0.and.dreal(b).ge.rbk) go to 6
      call rom1 (nans,sum,2)
      do 4 i=1,nans
4     ans2(i)=ans1(i)+sum(i)
      go to 11
c     hit break point.  reset seed and start over.
5     ibx=1
      go to 7
6     ibx=2
7     b=bk
      del=delb
      call rom1 (nans,sum,2)
      if (ibx.eq.2) go to 9
      do 8 i=1,nans
8     ans2(i)=ans2(i)+sum(i)
```

```
      go to 2
9     do 10 i=1,nans
10    ans2(i)=ans1(i)+sum(i)
      go to 2
11    den=0.
      do 18 i=1,nans
      as1=ans1(i)
      as2=ans2(i)
      if (int.lt.2) go to 17
      do 16 j=2,int
      jm=j-1
      aa=q2(i,jm)
      a1=q1(i,jm)+as1-2.*aa
      if (dreal(a1).eq.0..and.dimag(a1).eq.0.) go to 12
      a2=aa-q1(i,jm)
      a1=q1(i,jm)-a2*a2/a1
      go to 13
12    a1=q1(i,jm)
13    a2=aa+as2-2.*as1
      if (dreal(a2).eq.0..and.dimag(a2).eq.0.) go to 14
      a2=aa-(as1-aa)*(as1-aa)/a2
      go to 15
14    a2=aa
15    q1(i,jm)=as1
      q2(i,jm)=as2
      as1=a1
16    as2=a2
17    q1(i,int)=as1
      q2(i,int)=as2
      amg=dabs(dreal(as2))+dabs(dimag(as2))
      if (amg.gt.den) den=amg
18    continue
      denm=1.e-3*den*crit
      jm=int-3
      if (jm.lt.1) jm=1
      do 19 j=jm,int
      do 19 i=1,nans
      a1=q2(i,j)
      den=(dabs(dreal(a1))+dabs(dimag(a1)))*crit
      if (den.lt.denm) den=denm
      a1=q1(i,j)-a1
      amg=dabs(dreal(a1))+dabs(dimag(a1))
      if (amg.gt.den) go to 20
19    continue
      go to 22
20    continue
      print 24
      do 21 i=1,nans
21    print 25, q1(i,inx),q2(i,inx)
```

```
22    do 23 i=1,nans
23    sum(i)=.5*(q1(i,inx)+q2(i,inx))
      return
c
24    format (46h **** no convergence in subroutine gshank ****)
25    format (10e12.5)
      end
```

```
c ***
c
c
      subroutine hankel (z,h0,h0p)
c
c     hankel evaluates hankel function of the first kind, order zero,
c     and its derivative for complex argument z.
c
      implicit real*8 (a-h,o-z)
      complex*16 clogz,h0,h0p,j0,j0p,p0z,p1z,q0z,q1z,y0,y0p,
     *           z,zi,zi2,zk,fj
      dimension m(101), a1(25), a2(25), a3(25), a4(25), fjx(2)
      equivalence (fj,fjx)
      data pi,gamma,c1,c2,c3,p10,p20/3.141592654,.5772156649,-.024578509
     15,.3674669052,.7978845608,.0703125,.1121520996/
      data q10,q20,p11,p21,q11,q21/.125,.0732421875,.1171875,.1441955566
     1,.375,.1025390625/
      data p0f,init/.7853981635,0/,fjx/0.,1./
      if (init.eq.0) go to 5
1     zms=z*dconjg(z)
      if (zms.ne.0.) go to 2
      print 9
      stop
2     ib=0
      if (zms.gt.16.81) go to 4
      if (zms.gt.16.) ib=1
c     series expansion
      iz=1.+zms
      miz=m(iz)
      j0=(1.,0.)
      j0p=j0
      y0=(0.,0.)
      y0p=y0
      zk=j0
      zi=z*z
      do 3 k=1,miz
      zk=zk*a1(k)*zi
      j0=j0+zk
      j0p=j0p+a2(k)*zk
      y0=y0+a3(k)*zk
3     y0p=y0p+a4(k)*zk
      j0p=-.5*z*j0p
      clogz=cdlog(.5*z)
      y0=(2.*j0*clogz-y0)/pi+c2
      y0p=(2./z+2.*j0p*clogz+.5*y0p*z)/pi+c1*z
      h0=j0+fj*y0
      h0p=j0p+fj*y0p
      if (ib.eq.0) return
      y0=h0
```

```
      y0p=h0p
c     asymptotic expansion
4     zi=1./z
      zi2=zi*zi
      p0z=1.+(p20*zi2-p10)*zi2
      p1z=1.+(p11-p21*zi2)*zi2
      q0z=(q20*zi2-q10)*zi
      q1z=(q11-q21*zi2)*zi
      zk=cdexp(fj*(z-p0f))*cdsqrt(zi)*c3
      h0=zk*(p0z+fj*q0z)
      h0p=fj*zk*(p1z+fj*q1z)
      if (ib.eq.0) return
      zms=cos((sqrt(zms)-4.)*31.41592654)
      h0=.5*(y0*(1.+zms)+h0*(1.-zms))
      h0p=.5*(y0p*(1.+zms)+h0p*(1.-zms))
      return
c     initialization of constants
5     psi=-gamma
      do 6 k=1,25
      a1(k)=-.25/(k*k)
      a2(k)=1./(k+1.)
      psi=psi+1./k
      a3(k)=psi+psi
6     a4(k)=(psi+psi+1./(k+1.))/(k+1.)
      do 8 i=1,101
      test=1.
      do 7 k=1,24
      init=k
      test=-test*i*a1(k)
      if (test*a3(k).lt.1.e-6) go to 8
7     continue
8     m(i)=init
      go to 1
c
9     format (34h error - hankel not valid for z=0.)
      end
```

LAMBDA

PURPOSE

To compute the complex value of λ from the real integration parameter in ROM1.

METHOD

For integration along a straight path between the points a and b in the λ plane, λ and dλ are

$$\lambda = a + (b - a)t$$

$$d\lambda = (b - a)dt$$

SYMBOL DICTIONARY

A	=	a
B	=	b
DXLAM	=	b - a
T	=	t
XLAM	=	λ

```
c ***
c
c
      subroutine lambda (t,xlam,dxlam)
c
c     compute integration parameter xlam=lambda from parameter t.
c
      implicit real*8 (a-h,o-z)
      complex*16 a,b,xlam,dxlam
      common /cntour/ a,b
      dxlam=b-a
      xlam=a+dxlam*t
      return
      end
```

ROM1

PURPOSE

To integrate the Sommerfeld integrands between two points in λ by the method of variable interval-width Romberg integration.

METHOD

A and B in common block /CNTOUR/ are the ends of the integration path and are set before ROM1 is called. The integration parameter Z in ROM1 starts at zero and ends at one. The corresponding value of λ is determined by subroutine LAMBDA as

$$\lambda = A + (B - A)Z$$

Subroutine SAOA returns six integrand values which are handled simultaneously in loops throughout the code. The Romberg variable interval-width integration method will not be described in detail since it is the same as that used in subroutine INTX in the main NEC program. The convergence test in ROMl requires that all six components satisfy the relative error tests simultaneously.

```
c ***
c
c
      subroutine rom1 (n,sum,nx)
c
c     rom1 integrates the 6 sommerfeld integrals from a to b in lambda.
c     the method of variable interval width romberg integration is used.
c
      implicit real*8 (a-h,o-z)
      complex*16 a,b,sum,g1,g2,g3,g4,g5,t00,t01,t10,t02,t11,t20
      common /cntour/ a,b
      dimension sum(6), g1(6), g2(6), g3(6), g4(6), g5(6), t01(6), t10(6
     1), t20(6)
      data nm,nts,rx/131072,4,1.e-4/
      lstep=0
      z=0.
      ze=1.
      s=1.
      ep=s/(1.e4*nm)
      zend=ze-ep
      do 1 i=1,n
1     sum(i)=(0.,0.)
      ns=nx
      nt=0
      call saoa (z,g1)
2     dz=s/ns
      if (z+dz.le.ze) go to 3
      dz=ze-z
      if (dz.le.ep) go to 17
3     dzot=dz*.5
      call saoa (z+dzot,g3)
      call saoa (z+dz,g5)
4     nogo=0
      do 5 i=1,n
      t00=(g1(i)+g5(i))*dzot
      t01(i)=(t00+dz*g3(i))*.5
      t10(i)=(4.*t01(i)-t00)/3.
c     test convergence of 3 point romberg result
      call test (dreal(t01(i)),dreal(t10(i)),tr,dimag(t01(i)),
     *          dimag(t10(i)),ti,0.0d0)
      if (tr.gt.rx.or.ti.gt.rx) nogo=1
5     continue
      if (nogo.ne.0) go to 7
      do 6 i=1,n
6     sum(i)=sum(i)+t10(i)
      nt=nt+2
      go to 11
7     call saoa (z+dz*.25,g2)
      call saoa (z+dz*.75,g4)
```

```
      nogo=0
      do 8 i=1,n
      t02=(t01(i)+dzot*(g2(i)+g4(i)))*.5
      t11=(4.*t02-t01(i))/3.
      t20(i)=(16.*t11-t10(i))/15.
c     test convergence of 5 point romberg result
      call test (dreal(t11),dreal(t20(i)),tr,dimag(t11),dimag(t20(i)),
     *           ti,0.0d0)
      if (tr.gt.rx.or.ti.gt.rx) nogo=1
8     continue
      if (nogo.ne.0) go to 13
9     do 10 i=1,n
10    sum(i)=sum(i)+t20(i)
      nt=nt+1
11    z=z+dz
      if (z.gt.zend) go to 17
      do 12 i=1,n
12    g1(i)=g5(i)
      if (nt.lt.nts.or.ns.le.nx) go to 2
      ns=ns/2
      nt=1
      go to 2
13    nt=0
      if (ns.lt.nm) go to 15
      if (lstep.eq.1) go to 9
      lstep=1
      call lambda (z,t00,t11)
      print 18, t00
      print 19, z,dz,a,b
      do 14 i=1,n
14    print 19, g1(i),g2(i),g3(i),g4(i),g5(i)
      go to 9
15    ns=ns*2
      dz=s/ns
      dzot=dz*.5
      do 16 i=1,n
      g5(i)=g3(i)
16    g3(i)=g2(i)
      go to 4
17    continue
      return
c
18    format (38h rom1 -- step size limited at lambda =,2e12.5)
19    format (10e12.5)
      end
```

```
c ***
c
c
      subroutine saoa (t,ans)
c
c     saoa computes the integrand for each of the 6
c     sommerfeld integrals for source and observer above ground
c
      implicit real*8 (a-h,o-z)
      complex*16 ans,xl,dxl,cgam1,cgam2,b0,b0p,com,ck1,ck1sq,
     *           cksm,ct1,ct2,ct3,dgam,den1,den2
      common /evlcom/ cksm,ct1,ct2,ct3,ck1,ck1sq,ck2,ck2sq,tkmag,tsmag,c
     1k1r,zph,rho,jh
      dimension ans(6)
      call lambda (t,xl,dxl)
      if (jh.gt.0) go to 1
c     bessel function form
      call bessel (xl*rho,b0,b0p)
      b0=2.*b0
      b0p=2.*b0p
      cgam1=cdsqrt(xl*xl-ck1sq)
      cgam2=cdsqrt(xl*xl-ck2sq)
      if (dreal(cgam1).eq.0.) cgam1=cmplx(0.,-dabs(dimag(cgam1)))
      if (dreal(cgam2).eq.0.) cgam2=cmplx(0.,-dabs(dimag(cgam2)))
      go to 2
c     hankel function form
1     call hankel (xl*rho,b0,b0p)
      com=xl-ck1
      cgam1=cdsqrt(xl+ck1)*cdsqrt(com)
      if (dreal(com).lt.0..and.dimag(com).ge.0.) cgam1=-cgam1
      com=xl-ck2
      cgam2=cdsqrt(xl+ck2)*cdsqrt(com)
      if (dreal(com).lt.0..and.dimag(com).ge.0.) cgam2=-cgam2
2     xlr=xl*dconjg(xl)
      if (xlr.lt.tsmag) go to 3
      if (dimag(xl).lt.0.) go to 4
      xlr=dreal(xl)
      if (xlr.lt.ck2) go to 5
      if (xlr.gt.ck1r) go to 4
3     dgam=cgam2-cgam1
      go to 7
4     sign=1.
      go to 6
5     sign=-1.
6     dgam=1./(xl*xl)
      dgam=sign*((ct3*dgam+ct2)*dgam+ct1)/xl
7     den2=cksm*dgam/(cgam2*(ck1sq*cgam2+ck2sq*cgam1))
      den1=1./(cgam1+cgam2)-cksm/cgam2
      com=dxl*xl*cdexp(-cgam2*zph)
```

```
      ans(6)=com*b0*den1/ck1
      com=com*den2
      if (rho.eq.0.) go to 8
      b0p=b0p/rho
      ans(1)=-com*xl*(b0p+b0*xl)
      ans(4)=com*xl*b0p
      go to 9
8     ans(1)=-com*xl*xl*.5
      ans(4)=ans(1)
9     ans(2)=com*cgam2*cgam2*b0
      ans(3)=-ans(4)*cgam2*rho
      ans(5)=com*b0
      return
      end
```

```
c ***
c
c
      subroutine test (f1r,f2r,tr,f1i,f2i,ti,dmin)
c
c     test for convergence in numerical integration
c
      implicit real*8 (a-h,o-z)
      den=dabs(f2r)
      tr=dabs(f2i)
      if (den.lt.tr) den=tr
      if (den.lt.dmin) den=dmin
      if (den.lt.1.e-37) go to 1
      tr=dabs((f1r-f2r)/den)
      ti=dabs((f1i-f2i)/den)
      return
1     tr=0.
      ti=0.
      return
      end
```

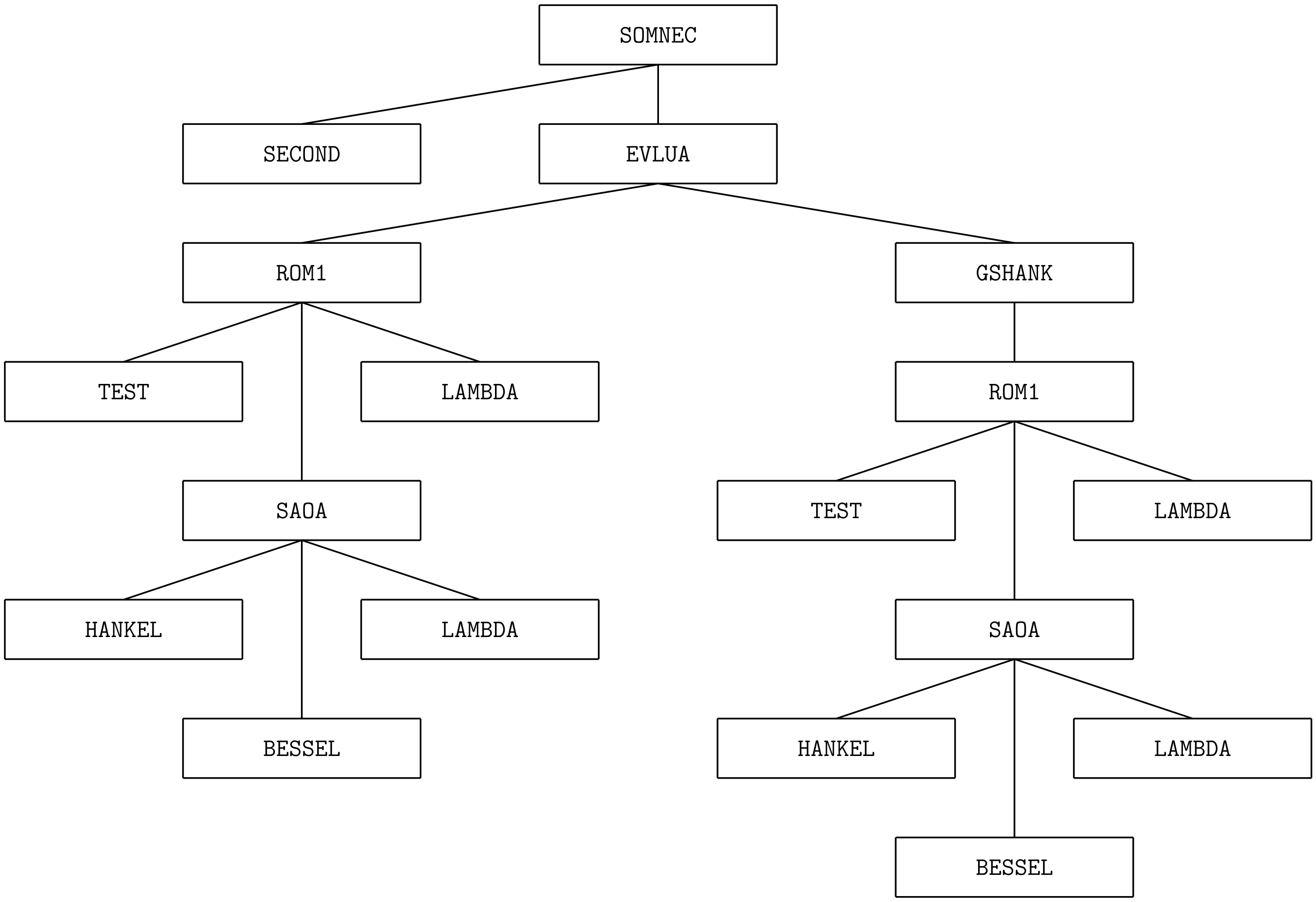


Figure 17. SOMNEC Subroutine Linkage Chart

1. Ralston. A,. A First Course in Numerical Analysis, McGraw-Hill, New York, 1965.

2. Norton, K. A., The Propagation of Radio Waves Over the Surface of the Earth and in the Upper Atmosphere, Proceedings of the Institute of Radio Engineers, Vol. 25, No. 9, Sept, 1937.

3. Miller, E. K., A Variable Interval Width Quadrature Technique Based on Romberg's Method, Journal of Computational Physics, Vol. 5, No. 2, April 1970.

4. Miller, E. K., and G. J. Burke, Numerical Integration Methods, IEEE Transactions, Vol. Ap-17, No. 5, Sept. 1969.

5. Handbook of Mathematical Functions, edited by M. Abramowitz, National Bureau of Standards (U.S.), Applied Mathematics Series 55, 1964.

6. Shanks, D., Non-Linear Transformations of Divergent and Slowly Convergent Sequences, J. Math, Phys. 24, 1, 1955.