Magnetic Stimulation of Living Tissue

Joshua Hatheway
Magnetic Stimulation of Living Tissue

An Honors Thesis (HONRS 499)

By

Joshua Hatheway

Thesis Advisor:
Dr. Ranjith Wijesinghe

Ball State University
Muncie, Indiana

December 2009
Date of Graduation: December 19, 2009
Abstract

Magnetic stimulation appears to be a very promising medical diagnostic tool that is completely non-invasive. Non-invasive diagnostic tools are critical to the medical field because of their ability to give diagnostic data on all sorts of patients in a wide variety of conditions. This paper will give an overview of research efforts being made to construct an algorithm that will accurately model the optimal coil geometry to produce the desired stimulation of living tissues. The new model proposed is based upon the cylindrical coordinate system and utilizes modified Bessel functions to solve this problem. A successful model will hopefully lead to breakthroughs in the realm of diagnostic medical physics.

Acknowledgements:

-I would like to thank Dr. Ranjith Wijesinghe for his patience and advice throughout this project. He was extremely helpful and always made time to answer my endless questions in addition to being a great professor and mentor.
Introduction:

Magnetic stimulation is a tool that has been developed in recent years to assist in medical diagnostics. This tool can be used to stimulate nerves non-invasively so that tissue functionality can be determined. To date, the overwhelming amount of research done on this topic has been conducted using a spherical coordinate system, and has been primarily centered upon the brain and the head. Roth, Momen, and Turner published an algorithm that addresses this issue in the spherical coordinate system. While extremely useful, the mainstream effort has somewhat neglected the cylindrical coordinate system and the body parts that more closely resemble the cylindrical shape. This tool appears to have much potential in medical diagnostics because of its non-invasive procedure.

Finding a non-invasive diagnostic method is absolutely critical in the field of medical physics. For instance, the well-known test, electroencephalography, or better known as the EEG, is a test that provides good data, but is extremely tedious to conduct. Some of these tests can take up to an hour to set up, even on the best patients. When also considering all those patients who are not as easy, and also those with illnesses that have difficulty being still, the process of connecting thirty-two electrodes becomes exponentially less appealing. One must also consider the purposes of the test. For example, one of the main purposes of electroencephalography is to determine whether organs are transferrable or not immediately before or after death. In situations where time is clearly of the essence, a non-invasive technique would be extremely valuable. When perfected, magnetic stimulation

---

will be one of the best non-invasive procedures available for nerve diagnostics and
treatment. Because of its simple setup, essentially just a magnetic coil held near the body,
nearly every patient will be more comfortable with the painless and non-invasive test
rather than the previous method of using electrodes.

Theory:

From the classical study of electrodynamics and magnetism, there are two principal
equations relating magnetic fields and currents. The first of these is the Biot-Savart Law,
which relates current and magnetic field as follows:

$$\mathbf{B} = \int \frac{\mu_0 I d\mathbf{l} \times \hat{r}}{4\pi r^2}, \quad (1)$$

and the second of these is Ampere's Law, which relates them the following way:

$$\oint_C \mathbf{B} \cdot d\mathbf{l} = \mu_0 I_{enc} \quad (2)$$

The primary conclusion drawn from this equation is that a current induces a magnetic field,
and this study is called magnetostatics. More information on the derivations and
applications of these laws and methods of finding magnetic fields can be found in Woosley,
Roth, and Wikswo. And while many great theories exist from this conclusion, it is also
important to deduce that a magnetic field can in fact induce a current as well as a potential.
It is this property that will be used in this paper to stimulate the desired axons and nerves

---

to selected potentials or currents. From the results of Basser, Wijesinghe, and Roth, a potential of 90-100 mV will be desired for the stimulating potential\textsuperscript{5}.

The basic concept of this stimulation is that a current is passed through a coil of an arbitrary geometry. Once this current is passed through the coil, a magnetic field is induced through this equation:

\begin{equation}
\hat{B} = \frac{\mu_0 N d\vec{I}}{dl}
\end{equation}

In this case, \(N\) stands for the number of turns, and \(dl\) is an infinitely small piece of the arbitrary coil shape in cylindrical coordinates. In order to produce desirable magnetic fields, careful consideration must be made towards the coil geometry. In previous papers, a “guess and check” method has been used to select coil geometry. This method has obvious limitations, as the perfect coil geometry needs to be “guessed,” and a good guess could take years. A better but significantly more tedious solution is to solve this problem inversely, as also done by Roth, Momen, and Turner\textsuperscript{6}. In other words, instead of first selecting a coil geometry and then experimentally determining the resulting field, this method uses the desired fields or stimulated potentials as a starting point, and then inversely solves for the coil geometry. When this coil geometry is found, it can then be constructed and implemented for diagnostic use. A model of this problem is shown below, with an infinitely long tissue cylinder assumed for calculations.


Next, the parameters of the tissue must be explored further. The following cartoon gives a basic understanding of nerve structure;

**Figure 1.** This drawing demonstrates the arbitrary coil shape of N turns and basic problem geometry of the cylindrical tissue.

**Figure 2.** Schematic drawing of a nerve bundle. Note that inside the nerve are many axon bundles, which individually contain many single axons. This model will account for the axon bundle as well as the axons inside.
While this cartoon is not to scale and contains many inaccuracies, it is effective in
demonstrating the model used in this paper. Esselle and Stuchly created a model that
approached this problem in cylindrical tissue, but it considered only the axon, and assumed
all else as homogeneous surrounding tissue. The model presented in this paper uses the
same method as Esselle and Stuchly, but is expanded to account for the axon bundle. Then
the nerve is assumed to be infinitely long, and all else outside the bundle sheath is assumed
to be homogenous tissue. While this model is not exact, it is a more accurate approximation
than has been used before. An even more accurate model could be achieved by accounting
for every bundle in the nerve, but currently the time lost in the magnified complexity of
those calculations is not worth the relatively trivial improvement in accuracy at this point
in time.

To begin the calculations, several more parameters must be set. In this case, the
bundle will be assumed to be uniform. Also, the transmembrane potential is assumed to be
known, and the coordinate system will be based at the center of the bundle for
mathematical simplification. Once these have been set, it is time to examine the electric
fields present. The electric field that is most prominent is the quasi-static electric field that
is induced by the coil, and it is,

\[ d\vec{E}_{\text{one}} = -\frac{\mu_o N (dI/dt) d\vec{l}}{4\pi R}. \]  


stimulation of neurons: effects of coil geometry," Transactions of Biomedical Engineering,
42, pp 934- 941.
The second electric field is due to the surface charge present on the bundle and is defined by,

\[ d\vec{E}^{\text{two}} = -\nabla \psi \quad (5). \]

By definition the Laplacian of the scalar potential is equal to zero as shown,

\[ \nabla^2 \psi = 0 \quad (6). \]

This has been assumed many times before and has yielded strong results\(^9\). Because of the quasi static condition, the normal of the electric field inside of the bundle is also zero and is indicated by

\[ (d\vec{E}^{\text{one}} + d\vec{E}^{\text{two}}) \cdot \hat{r} \big|_{r=a} = 0 \quad (7). \]

However, knowing these electric fields is quintessential to this calculation, and must be done using computer code. These quasi static fields were calculated using equations found in a different paper from Esselle and Stuchly, and are quite complex, and would have been even worse to calculate without their much needed help\(^{10}\).

From Wijesinghe, Gielen, and Wikswo, after applying a Fourier transformation, the potentials for the axon, the bundle, and surrounding tissue can be expressed as\(^{11}\),

\[ \text{References} \]


Hatheway 9

\[ \phi_{\text{axon}}(\rho', \theta', k) = A_o(k)I_o(k|\rho') + 2 \sum_{m=1}^{\infty} A_m(k)I_m(k|\rho') \cos m\theta', \quad (8) \]

\[ \phi_{\text{bundle}}(\rho^*, \theta', k) = B_o(k)I_o(k|\rho^*) + C_o(k)K_o(k|\rho^*) + 2 \sum_{m=1}^{\infty} [B_m(k)I_m(k|\rho^*) + C_m(k)K_m(k|\rho^*)] \cos m\theta', \quad (9) \]

\[ \phi_{\text{outside}}(\rho, \theta, k) = F_o(k)K_o(k|\rho) + 2 \sum_{m=1}^{\infty} F_m(k)K_m(k|\rho) \cos m\theta. \quad (10) \]

\[ I_m \text{ and } K_m \text{ are modified Bessel Functions, and } A_m, B_m, C_m, \text{ and } F_m \text{ are all Fourier expansion coefficients.} \]

To solve for the Fourier coefficients, boundary conditions must be used. The first boundary condition is that the transmembrane potential is equivalent to the difference in potential between the internal and external membrane surfaces. The second boundary condition is that the radial current density across the axon membrane is continuous. For the third boundary condition, the radial current density at the outer surface of the nerve is continuous, and the final condition is that the potential at the outer surface of the nerve bundle is continuous. Then the Gaussian elimination method was used to find the Fourier coefficients. Once all these are solved and found, the optimal coil geometry for the desired parameters can be determined.

**Results and Discussion:**

To solve for the Fourier coefficients and the modified Bessel Functions, a computer code was written in Fortran and simulated on Ball State University's CCN cluster. This code also calculated the coil geometry and simulated the resulting fields and potentials. A copy of this code and its subroutines are attached to the end of this document. The simulation showed strong results, and further simulations will be run in the future to continue the verification of this method with other parameters and desired potentials. Esselle and
Stuchly's results are shown in graphical form below, and are similar to the simulation's results, although with less precision.  

Figure 3. Contour plots for a square coil: (a) cross sectional plane; (b) cylindrical surface.  

Figure 4. Contour plots for a planar DS coil: (a) cross sectional plane; (b) cylindrical surface.

---

The results also indicate that the assumptions made in this model were reasonable. However, it must be noted that this model does have significant limitations. Primarily, it is based upon the assumption that all the external tissue surrounding the nerve bundle is homogenous. While small areas of heterogeneous tissue may prove negligible, this model will not be sufficient for areas that do not satisfy this condition. A more complex model may be needed for these special cases in the future. Nevertheless, this model produced strong results and is worth further investigation. Simulations and code are attached.
Bibliography


FILE: MAGFIELD.F

PROGRAM MAGFIELD

COMPLEX INPUT(1025),OUT(1025),LAMBDA
DOUBLE COMPLEX C(60,60),AA(60)
COMPLEX AC(3600)
COMPLEX AN(2049),B(60),WORK(60), COND
COMPLEX ANISOP,EFFCON,GBZ,GBR,ZMEM,XAL,XBL,XTL,ALPHA
DOUBLE PRECISION CR(60,60),CI(60,60),AAR(60),AAI(60)
DOUBLE PRECISION CX(20,4),CDX(20,4),CXC(20,4),CDXC(20,4),CDXTO(3,4)
REAL GMEN,GINT,TEMP2,CMEN,DTOR,TEMP4,TEMP3,GEXT,U0,CTOR
REAL DBUN,BBUN,G5,PI,GINTER,XDOR,DT,ST,XCTOR,TEMP1
REAL TEMP5,U,E2,E,P,GG,AAA
DOUBLE PRECISION CXB(20,4),CDXB(20,4),CXA(20,4),CDXA(20,4)
REAL X,R,XA,XT,T,XC,CBUN,XB
DOUBLE PRECISION CDXDT(3,4),CXCTO(3,4),CDXCT(3,4)
DOUBLE PRECISION CXBL(20,4),CDXBL(20,4),CXAL(20,4)
DOUBLE PRECISION CDXAL(20,4),CXTL(20,4),CDXTL(20,4)
REAL BUF1(5020), BUF2(5020), BUF3(5020),BBB(5020),CCC(5020)
REAL VALS(25),WTS(2),K,HDR(24),A,TLB(1000)
REAL DOW,DOWN,BETA,SHI,VGIVEN
REAL K1,K2,NFILT
REAL REAL1, REAL2
REAL RPARAM(5)
DOUBLE PRECISION SUM1,SUM2,SUM,X1,Y1,RSUM,ESUM,X2,Y2,X3
DOUBLE PRECISION Y3,X4,Y4,X11,Y11,RSUM1,ESUM1,RSUM2,ESUM2
LOGICAL TOROID,NOISE
INTEGER IPVT(60),TOKEN,IP(60)
INTEGER IF,IDCN,NUM,IDS1,N2,N4,N1,IDCI,L,NDIM,IFOUT,N3
INTEGER IER
INTEGER IDSN,II1,JJ,N,II1,I2,I,J,N5
INTEGER IPARAM(6),IPATH,IROW(3600),JCOL(3600)
character*1 ANSWER

COMMON /BUF1/ BUF1
COMMON /BUF2/ BUF2
COMMON /BUF3/ BUF3
COMMON /FDIF/ DZ, DK
COMMON /HDR/ HDR

NDIM=10
NDIM=4

OPEN DATA FILE
OPEN (UNIT=10,FILE='VMEM.DAT', STATUS='NEW')

PI=ACOS(-1.00)
U0=4.0*PI*1.0E-7

print *, U0

INPUT CALCULATION PARAMETERS
I stopped here at 12:47 07/30/08

N=256
M=0
CALL NTEST(N,M)
A=0.01
A=A/1000.0
R=2.1
R=R/1000.0
BBUN=2.0
BBUN=BBUN/1000.0
DBUN=5.0
DBUN=DBUN/100000.0

T=2.*BBUN/3.
CBUN=BBUN+DBUN
CMEM=0.0027
CMEM=CMEM*0.1
READ *,GMEM
GMEM=GMEM*10000.0

P=0.89
GS=GEXT

U=VALS(10)
U=16.5
IF(A.EQ.0.0) A=U/(4.1282*1000000.0)
AAA=0.0
AAA=AAA/1000000.0
IF(AAA.EQ.0.0) AAA=A
ST=0.482*(AAA)**2
GINTER=0.882

CC
CC NOISE FILTER
CC WE NEED TO ADD THIS IN THE FUTURE
K1=5.0
K2=100.
K1=K1*1000.0
K2=K2*1000.0

2600 CONTINUE
C
C

600 CONTINUE
C
C
C PUTTING DATA INTO USEFUL FORM
C
C
DO 700 I=1,N
700 INPUT(I)=CMPLX(BUF1(I),0.0)
C
DT=1.0/70.0 ! this is the conversion factor from SFAS.DAT to
DZ=U*DT/1000.0
DK=(2.*PI)/(DZ*FLOAT(N))
C
C SELECT OUTPUT FILE AND DATA SET
C
C
TEMP1=AAA*(1./(GINTER*(1.-P)*ST)+1./(GINT*P*ST))
TEMP2=(1.-P)/(1.+P)*GINTER
TEMP3=AAA*SQRT(3.0)/2.
TEMP4=GINTER*(1.-P)+GINT*P
TEMP5=GINT*P/(GINTER*(1-P))
C
C CALCULATE FFT
C
CALL FTCALL(INPUT,M,0)
C
*** MAIN LOOP ***
C
C FILTER CALCULATIONS IN TRANSFORM SPACE
DO 800 L=1,N
IF(L.EQ.(N/2+1)) GO TO 900
K=(L-N/2-1)*DK
print *, 'K=', K
ZMEM=1./CMPLX(GMEM,K*U*CMEM)
LAMBDA=CSQRT(CMPLX(TEMP1)/ZMEM)
GBR=TEMP2+TEMP3/ZMEM
GBZ=TEMP4/(1.0+TEMP5*(1.0/((LAMBDA/K)**Z+1.0)))
ANISOP=CSQRT(GBZ/GBR)
PRINT *, 'ANISOP=', ANISOP, 'LAMBDA=', LAMBDA, 'ZMEM=', ZMEM
EFFCON=CSQRT(GBZ*GBR)
CALCULATE ARGUMENTS OF BESSEL FUNCTIONS
X=ABS(K)*R
XA=ABS(K)*A
XT=ABS(K)*T
XCTOR=ABS(K)*CTOR !IF(TOROID) XCTOR=ABS(K)*CTOR
XDTOR=ABS(K)*DTOR !IF(TOROID) XDTOR=ABS(K)*DTOR
XC=ABS(K)*CBUN
XB=ABS(K)*BBUN
XAL=ABS(K)*A*ANISOP
XTL=ABS(K)*T*ANISOP
XBL=ABS(K)*BBUN*ANISOP
CHECK FOR TOO LARGE AN ARGUMENT
IF(ABS(X-XA).GT.70.0) GO TO 900
IF(ABS(XT).GT.70.) GO TO 900
IF(ABS(XC).GT.70.) GO TO 900
IF(ABS(XB).GT.70.) GO TO 900
IF(CABS(XAL).GT.70.) GO TO 900
IF(CABS(XTL).GT.70.) GO TO 900
IF(CABS(XBL).GT.70.) GO TO 900
IF(TOROID.AND.(XCTOR.GT.70.)) GO TO 900
IF(TOROID.AND.(XDTOR.GT.70.)) GO TO 900
PREPARE NOISE FILTER
NFILT=0.5*(1.0+COS((ABS(K)-ABS(K1))*3.14151/(ABS(K2)-ABS(K1))))
IF(ABS(K).LT.ABS(K1)) NFilt=1.0
IF(ABS(K).GT.ABS(K2)) NFilt=0.0
IF(ABS(K).GT.ABS(K2)) GO TO 900
1000 CONTINUE

C
C C SETUP COEFFICIENT MATRIX
C
C C GET BESSEL FUNCTION VALUES

DO 1100 I=1,20
DO 1100 J=1,4
CX(I,J)=0.0D0
CDX(I,J)=0.0D0
CXC(I,J)=0.0D0
CDXC(I,J)=0.0D0
CXB(I,J)=0.0D0
CDXB(I,J)=0.0D0
CXA(I,J)=0.0D0
CDXA(I,J)=0.0D0
CXBL(I,J)=0.0D0
CDXBL(I,J)=0.0D0
CXAL(I,J)=0.0D0
CDXAL(I,J)=0.0D0
CXTL(I,J)=0.0D0
CDXTL(I,J)=0.0D0
1100 CONTINUE

C C
C PRINT *, 'XBL=', XBL, 'XAL=', XAL, 'XTL=', XTL
CALL CKN(X, 0., CX, CDX, NUM+1)
CALL CKN(XC, 0., CXC, CDXC, NUM+1)
CALL CKN(XB, 0., CXB, CDXB, NUM+1)
CALL CKN(XA, 0., CXA, CDXA, NUM+1)
C CALL CKN(REAL(XCL), AIMAG(XCL), CXCL, CDXCL, 10)
CALL CKN(REAL(XBL), AIMAG(XBL), CXBL, CDXBL, NUM+1)
CALL CKN(REAL(XAL), AIMAG(XAL), CXAL, CDXAL, NUM+1)
CALL CKN(REAL(XTL), AIMAG(XTL), CXTL, CDXTL, 10)

C C
C DO 1200 II=1,60
DO 1200 JJ=1,60
1200 C(II, JJ)=DCMPLX(0.0D0, 0.0D0)
DO 1300 II=1,NUM+1
C(II,II)=DCMPLX(CXA(II,1),CXA(II,2))
C(II,II+(NUM+1))=-DCMPLX(CXAL(II,1),CXAL(II,2))
C(II,II+2*(NUM+1))=-DCMPLX(CXAL(II,3),CXAL(II,4))
C(II+(NUM+1),II)=GINT/EFFCON*(DCMPLX(CDXA(II,1),
CDXA(II,2)))
C(II+(NUM+1),II+(NUM+1))=-DCMPLX(CDXAL(II,1),
CDXAL(II,2))
C(II+(NUM+1),II+2*(NUM+1))=-DCMPLX(CDXAL(II,3),
CDXAL(II,4))
C(II+2*(NUM+1),II+3*(NUM+1))=-DCMPLX(CXB(II,1),
CXB(II,2))
C(II+2*(NUM+1),II+4*(NUM+1))=-DCMPLX(CXB(II,3),
CXB(II,4))
C(II+3*(NUM+1),II+3*(NUM+1))=-GS/EFFCON*(DCMPLX
(CDXB(II,1),CDXB(II,2)))
C(II+3*(NUM+1),II+4*(NUM+1))=-GS/EFFCON*(DCMPLX
(CDXB(II,3),CDXB(II,4)))
C(II+4*(NUM+1),II+3*(NUM+1))=-DCMPLX(CXC(II,1),
CXC(II,2))
C(II+4*(NUM+1),II+4*(NUM+1))=-DCMPLX(CXC(II,3),
CXC(II,4))
C(II+4*(NUM+1),II+5*(NUM+1))=-DCMPLX(CXC(II,4),
CXC(II,5))
C(II+5*(NUM+1),II+3*(NUM+1))=-DCMPLX(CDXC(II,1),
CDXC(II,2))
C(II+5*(NUM+1),II+4*(NUM+1))=-DCMPLX(CDXC(II,3),
CDXC(II,4))
C(II+5*(NUM+1),II+5*(NUM+1))=-GEXT/GS*(DCMPLX
(CDXC(II,3),CDXC(II,4)))
C(II+2*(NUM+1),1+(NUM+1))=(-1.0)**(II-1)*
DCMPLX(CXBL(II,1),CXBL(II,2))*DCMPLX(CXTL(II,1),
2
CXTL(II,2))
C(II+2*(NUM+1),1+2*(NUM+1))=CMPLX(CXBL(II,3),
CXBL(II,4))*DCMPLX(CXTL(II,1),CXTL(II,2))
C(II+3*(NUM+1),1+(NUM+1))=(-1.0)**(II-1)
*DCMPLX(CDXB(II,1),CDXBL(II,2))
2 *DCMPLX(CXTL(II,1),CXTL(II,2))
C(II+3*(NUM+1),1+2*(NUM+1))=DCMPLX(CDXB(II,3),
1 CDBL(II,4))*DCMPLX(CXTL(II,1),CXTL(II,2))
1300 CONTINUE
C
C
IF(NUM.LE.0) GO TO 1400
C
DO 1500 II=1,NUM+1
  DO 1500 JJ=2,NUM+1
  C(II+2*(NUM+1),JJ+(NUM+1))=(-1)**(II-1)*DCMPLX(CXTL((II+JJ-1),1),CXTL((II+JJ-1),Z))+DCMPLX(CXTL((IABS(II-JJ)+1),1),CXTL((IABS(II-JJ)+1),Z))*DCMPLX(CXBL(II,1),CXBL(II,Z))
  C(II+Z*(NUM+1),JJ+Z*(NUM+1))=(DCMPLX(CXTL((II+JJ-1),1),CXTL((II+JJ-1),Z))+DCMPLX(CXTL((IABS(II-JJ)+1),1),CXTL((IABS(II-JJ)+1),Z)))*DCMPLX(CXBL(II,3),CXBL(II,4))
  C(II+3*(NUM+1),JJ+(NUM+1))=(-1)**(II-1)*(DCMPLX(CXTL((II+JJ-1),1),CXTL((II+JJ-1),Z))+CMPLX(CXTL((IABS(II-JJ)+1),1),CXTL((IABS(II-JJ)+1),Z)))*DCMPLX(CDXBL(II,1),CDXBL(II,Z))
  C(II+3*(NUM+1),JJ+Z*(NUM+1))=(DCMPLX(CXTL((II+JJ-1),1),CXTL((II+JJ-1),Z))+DCMPLX(CXTL((IABS(II-JJ)+1),1),CXTL((IABS(II-JJ)+1),Z)))*DCMPLX(CDXBL(II,3),CDXBL(II,4))
  CONTINUE
C SETUP VECTOR
C
1400  AA(1)=DCMPLX(1.000,0.0D0)
  DO 1600 II=Z,6*(NUM+1)
       AA(II)=CMPLX(0.000,0.000)
  C
1600  DO 1601 II=1,6*(NUM+1)
       AAR(II)=REAL(AA(II))
       AAI(II)=AIMAG(AA(II))
       PRINT *,II,'AAR(',II,',AAR(II),'AAI(II)=',AAI(II)
  CONTINUE
C
1601  DO 1602 II=1,6*NDIM
       DO 1602 III=1,6*NDIM
       CR(II,III)=0.000
       CI(II,III)=0.000
  CONTINUE
C
1602  DO 1603 II=1,6*(NUM+1)
       DO 1603 III=1,6*(NUM+1)
       CR(II,III)=REAL(C(II,III))
       CI(II,III)=AIMAG(C(II,III))
       PRINT *,II,III,C(II,III),'CR=',CR(II,III),'CI=',CI(II,III)
  CONTINUE
OPEN (UNIT=15, FILE='CMATRIX.DAT', STATUS='NEW')
DO 1604 II=1,6*(NUM+1)
DO 1604 III=1,6*(NUM+1)
WRITE(15,*),II,III,C(II,III), CR=', CR(II,III), CR=', CI(II,III)
C1604 CONTINUE
CLOSE (15)
print *, aar(1), aai(1), cr(4,4), ci(4,4)
STOP AT 3:01 ON 7/30/08
DO GAUSSIAN ELIMINATION.
 CALL DECC(6*(NUM+1), 6*NDIM, CR, CI, IP, IER)
 CALL SOLC(6*(NUM+1), 6*NDIM, CR, CI, AAR, AAI, IP)
1800 CONTINUE
.AXON TERM
 RSUM=DBLE(GINT)*DBLE(A)*(DBLE(REAL(AA(1)))*DBLE(CXA(2,1))
 1-DBLE(AIMAG(AA(1)))*DBLE(CXA(2,2)))
 RSUM=DBLE(GINT)*DBLE(A)*(AAR(1)*CXA(2,1)
 1-AAI(1)*CXA(2,2))
 ESUM=0.0D0
print *, 'aar=', aar(1), 'cxa=', cxa(2,1), 'aai=', aai(1), 'cxa=', cxa(2,2)
print *, 'RUM=', RSUM, 'ESUM=', ESUM
GO TO 2201
SHEATH TERM
X1=0.0D0
Y1=0.0D0
X2=0.0D0
Y2=0.0D0
X3=0.0D0
Y3=0.0D0
X4=0.0D0
Y4=0.0D0
X1=DBLE(CBUN)*CX(2,1)-DBLE(BBUN)*CXB(2,1)
X2=DBLE(BBUN)*CXB(2,3)-DBLE(CBUN)*CXC(2,3)

C

X3=AAR(1+3*(NUM+1))
Y3=AAI(1+3*(NUM+1))
X4=AAR(1+4*(NUM+1))
Y4=AAI(1+4*(NUM+1))

C

RSUM=RSUM+DBLE(GS)*(X3*X1+X4*XZ)
ESUM=ESUM+DBLE(GS)*(Y3*X1+Y4*XZ)

C

BUNDLE TERM

C

X1=0.0D0
Y1=0.0D0
X1=AAR(1+Z*(NUM+1))
Y1=AAI(1+Z*(NUM+1))
RSUM1=X1*CXTL(1,3)-Y1*CXTL(1,4)
ESUM1=Y1*CXTL(1,3)+X1*CXTL(1,4)

C

DO 1900 II=Z,NUM+1
X2=0.0D0
Y2=0.0D0
X2=AAR(II+Z*(NUM+1))
Y2=AAI(II+Z*(NUM+1))
RSUM1=RSUM1+Z.0D0*(X2*CXTL(II,3)-Y2*CXTL(II,4))
ESUM1=ESUM1+Z.0D0*(Y2*CXTL(II,3)+X2*CXTL(II,4))
CONTINUE

1900

C

RSUM=RSUM+DBLE(REAL(EFFCON))*DBLE(T)*(RSUM1*CXTL(2,1)
1 -ESUM1*CXTL(2,2))-DBLE(AIMAG(EFFCON))*DBLE(T)*(ESUM1*CXTL(2,1)
2 +RSUM1*CXTL(2,2))

C

ESUM=ESUM+DBLE(REAL(EFFCON))*DBLE(T)*(ESUM1*CXTL(2,1)
1 +RSUM1*CXTL(2,2))+DBLE(AIMAG(EFFCON))*DBLE(T)*(RSUM1*CXTL(2,1)
2 -ESUM1*CXTL(2,2))
BUNDLE, SECOND TERM

X1=0.0D0
Y1=0.0D0
X2=0.0D0
Y2=0.0D0
X1=AAR(1+(NUM+1))
Y1=AAI(1+(NUM+1))
X2=AAR(1+Z*(NUM+1))
Y2=AAI(1+Z*(NUM+1))
RSUM1=0.0D0
ESUM1=0.0D0
RSUM2=0.0D0
ESUM2=0.0D0

DO 2000 II=Z,NUM+1
X1=0.0D0
Y1=0.0D0
XZ=0.0D0
Y2=0.0D0
X1=AAR(II+(NUM+1))
Y1=AAI(II+(NUM+1))
XZ=AAR(II+Z*(NUM+1))
Y2=AAI(II+Z*(NUM+1))
RSUM1=RSUM1+Z.0D0*(X1*CXTL(II,1)-Y1*CXTL(II,2))
ESUM1=ESUM1+Z.0D0*(X1*CXTL(II,2)+Y1*CXTL(II,1))
RSUM2=RSUM2+Z.0D0*(XZ*CXTL(II,1)-Y2*CXTL(II,2))
ESUM2=ESUM2+2.0D0*(X2*CXTL(II,2)+Y2*CXTL(II,1))

2000 CONTINUE
2100 CONTINUE

X1=0.0D0
Y1=0.0D0
X2=0.0D0
Y2=0.0D0
X1=DBLE(T)*CXTL(2,3)-DBLE(BBUN)*CXBL(2,3)
Y1=DBLE(T)*CXTL(Z,4)-DBLE(BBUN)*CXBL(Z,4)
X2=RSUM2*X1-ESUM2*Y1
Y2=RSUM2*Y1+ESUMZ*X1
Y1=0.0D0
X1=0.0D0
X1=DBLE(BBUN)*(RSUM1*CXBL(Z,1)-ESUM1*CXBL(2,2))+X2
Y1=DBLE(BBUN)*(ESUM1*CXBL(2,1)+RSUM1*CXBL(2,2))+Y2
RSUM=RSUM+DBLE(REAL(EFFCON))*X1-DBLE(IMAG(EFFCON))*Y1
ESUM=ESUM+DBLE(REAL(EFFCON))*Y1+DBLE(IMAG(EFFCON))*X1

C BUNDLE, THIRD TERM

X1=0.0D0
Y1=0.0D0
X2=0.0D0
Y2=0.0D0
X11=DBLE(REAL(ANISOP))
Y11=DBLE(IMAG(ANISOP))

X1=DBLE(REAL(ANISOP))/DBLE(ABS(K))/(X11**2+Y11**2)
1 -DBLE(A)*CXAL(Z,3)
Y1=-DBLE(IMAG(ANISOP))/DBLE(ABS(K))/(X11**2+Y11**2)
1 -DBLE(A)*CXAL(Z,4)
X2=0.0D0
Y2=0.0D0
X2=AAR(1+2*(NUM+1))*X1-AAI(1+2*(NUM+1))*X1
1
Y2=AAR(1+2*(NUM+1))*Y1+AAI(1+2*(NUM+1))*Y1
1

C

Y1=0.0D0
X1=0.0D0
X11=0.0D0
Y11=0.0D0
X11=AAR(1+(NUM+1))
Y11=AAI(1+(NUM+1))
X1=DBLE(A)*(X11*CXAL(Z,1)-Y11*CXAL(Z,2))
Y1=DBLE(A)*(X11*CXAL(Z,2)+Y11*CXAL(Z,1))
X2=X2+X1
Y2=Y2+Y1
RSUM=RSUM-(DBLE(REAL(EFFCON))*X2-DBLE(IMAG(EFFCON))*Y2)
ESUM=ESUM-(DBLE(REAL(EFFCON))*Y2+DBLE(IMAG(EFFCON))*X2)

DO 2200 I=1,3
DO 2200 J=1,4
CXCTO(I,J)=0.0D0
CXDTO(I,J)=0.0D0
2200 CONTINUE
RSUM=RSUM+DBLE(GEXT)*AAR(1+5*(NUM+1))*
1       DBLE(R)*(DBLE(CBUN)*CXC(2,3)/DLOG(DBLE(DTOR
2    /CTOR))+((CXDTO(1,3)))-(CXCTO(1,3))
3    /DBLE(ABS(K)))/((DBLE(DTOR)-DBLE(CTOR))))

C
C
C

ESUM=ESUM+DBLE(GEXT)*AAI(1+5*(NUM+1))*
1       DBLE(R)*(DBLE(CBUN)*CXC(2,3)/DLOG(DBLE(DTOR
2    /CTOR))+((CXDTO(1,3)))-(CXCTO(1,3))
3    /DBLE(ABS(K)))/((DBLE(DTOR)-DBLE(CTOR))))

C
C
C

2201 continue
AN(L)=CMPLX(SNGL(RSUM),SNGL(ESUM))*CMPLX(0.0,U0/R)*K/ABS(K)
C       IF(TOROID) AN(L)=AN(L)*SIN(K*E2)/(K*E2)
C       IF(NOISE)  AN(L)=AN(L)*NFILT

C
C
C
801       CONTINUE
       GO TO 800

C
C
900       AN(L)=CMPLX(0.0,0.0)

C
800       CONTINUE
C
C
**** END OF MAIN LOOP ****
C
C
C
C
C
CALCULATE INVERSE TRANSFORMS
C
C BRANCH ACCORDING TO TOKEN VALUE
C
C
GO TO (2300,2400),TOKEN
C
C
POTENTIAL
C
C
2300    CONTINUE
C
C
FOR EACH N, DO INVERSE FOURIER TRANSFORM
C
DO 2500 L=1,N
   OUT(L)=(AN(L)*INPUT(L))
C
print *,L,'AN(L)=' ,AN(L),'INPUT(L)=' ,INPUT(L)
2500 CONTINUE
C
CALL FTCALL(OUT,M,1)
C
N5=N1+N3
C
DO 500 L=1,N
   BUF1(L)=REAL(OUT(L))*1.0E9 ! Use 1000.0 for potential
C
PRINT *,L,BUF1(L)
IF(N5.EQ.0) GO TO 500
IF(L.GT.N5) BUF1(L)=0.0
500 CONTINUE
OPEN(UNIT=20,FILE='POTENTIAL.DAT',STATUS='NEW')
DO 510 I=1,N
WRITE(20,*)float(I)*DT,BUF1(I)*32000/1000.0 ! Now answer is in nT
510 CONTINUE
C
CLOSE(UNIT=20)
C
C MAGNETIC FIELD
C
2400 CONTINUE
C
DO 2900 L=1,N
   OUT(L)=(AN(L)*INPUT(L)*1.0E9)
2900 CONTINUE
C
CALL FTCALL(OUT,M,1)
C
C OUTPUT
C
DO 3000 L=1,N
   BUF2(L)=REAL(OUT(L))
IF(N5.EQ.0) GO TO 3000
IF(L.GT.N5) BUF2(L)=0.0
3000 CONTINUE
C
OPEN(UNIT=21,FILE='BFIELD.DAT',STATUS='NEW')
DO 520 I=1,N
WRITE(22,*)I,BUF2(I)
520 CONTINUE
C
2800 CONTINUE
C 3100 CLOSE(33)
STOP
END
SUBROUTINE SOLC (N, NDIM, AR, AI, BR, BI, IP)
C VERSION COMPLEX DOUBLE PRECISION
C I CHANGED THE IMPLICIT REAL*8 TO DOUBLE PRECISION
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
INTEGER N,NDIM,IP,NM1,K,KP1,M,I,KB,KM1
DIMENSION AR(NDIM,N), AI(NDIM,N), BR(N), BI(N), IP(N)

C-----------------------------------------------------------------------
C SOLUTION OF LINEAR SYSTEM, A*X = B .
C INPUT ..
C N = ORDER OF MATRIX.
C NDIM = DECLARED DIMENSION OF ARRAYS AR AND AI.
C (AR, AI) = TRIANGULARIZED MATRIX OBTAINED FROM DEC.
C (BR, BI) = RIGHT HAND SIDE VECTOR.
C IP = PIVOT VECTOR OBTAINED FROM DEC.
C DO NOT USE IF DEC HAS SET IER .NE. 0.
C OUTPUT..
C (BR, BI) = SOLUTION VECTOR, X .
C-----------------------------------------------------------------------
IF (N .EQ. 1) GO TO 50
NM1 = N - 1
DO 20 K = 1,NM1
   KP1 = K + 1
   M = IP(K)
   TR = BR(M)
   TI = BI(M)
   BR(M) = BR(K)
   BI(M) = BI(K)
   BR(K) = TR
   BI(K) = TI
   DO 10 I = KP1,N
      PRODR=AR(I,K)*TR-AI(I,K)*TI
      PRODI=AI(I,K)*TR+AR(I,K)*TI
      BR(I) = BR(I) + PRODR
      BI(I) = BI(I) + PRODI
  10   CONTINUE
  20 CONTINUE
DO 40 KB = 1,NM1
   KM1 = N - KB
   K = KM1 + 1
   DEN=AR(K,K)*AR(K,K)+AI(K,K)*AI(K,K)
   PRODR=BR(K)*AR(K,K)+BI(K)*AI(K,K)
   PRODI=BI(K)*AR(K,K)-BR(K)*AI(K,K)
   BR(K)=PRODR/DEN
   BI(K)=PRODI/DEN
   TR = -BR(K)
\[ TI = -BI(K) \]

\[ \text{DO } 30 \text{ I }= \text{1,KM1} \]
\[ \quad \text{PRODR} = \text{AR(I,K)TR} - \text{AI(I,K)TI} \]
\[ \quad \text{PRODI} = \text{AI(I,K)TR} + \text{AR(I,K)TI} \]
\[ \quad \text{BR}(I) = \text{BR}(I) + \text{PRODR} \]
\[ \quad \text{BI}(I) = \text{BI}(I) + \text{PRODI} \]

\[ \text{30 CONTINUE} \]
\[ \text{40 CONTINUE} \]
\[ \text{50 CONTINUE} \]
\[ \text{DEN} = \text{AR(1,1)TR} + \text{AI(1,1)TI} \]
\[ \quad \text{PRODR} = \text{BR(1)AR(1,1)} + \text{BI(1)AI(1,1)} \]
\[ \quad \text{PRODI} = \text{BI(1)AR(1,1)} - \text{BR(1)AI(1,1)} \]
\[ \quad \text{BR}(1) = \text{PRODR/DEN} \]
\[ \quad \text{BI}(1) = \text{PRODI/DEN} \]

\[ \text{RETURN} \]

C----------------------- END OF SUBROUTINE SOLC -----------------------

END
SUBROUTINE FFT(X,M,INVFLG)
COMPLEX X(2048),U,W,TMP
NPTS=2**M
NM1=NPTS-1
M1=M-1
PI=3.1415926
C TO DO INVERSE FFT COMPLEX CONJUGATE THE INPUT
IF(INVFLG.EQ.0) GO TO 20
DO 10 I=1,NPTS
10 X(I)=CONJG(X(I))
C NUM IS THE NUMBER OF INPUTS TO THIS STAGE
C N2 IS THE NUMBER OF BUTTERFLIES
C
20 continue
DO 40 L=1,M1
NUM=2**(M+1-L)
N2=NUM/2
W=(1.,0.0)
C U IS THE INCREMENTAL KERNAL
U=CEXP(CMPLX(0.,PI/N2))
DO 40 J=1,N2
C PERFORM THE BUTTERFLY
DO 30 I=J,NPTS,NUM
INDEX=I+N2
TMP=X(I)+X(INDEX)
X(INDEX)=(X(I)-X(INDEX))*W
30 X(I)=TMP
C NOW INCREMENT W FOR THE NEXT BUTTERFLY
40 W=W*U
C SPECIAL LOOP FOR LAST STAGE( WHERE W=1. FOR ALL)
DO 50 J=1,NPTS,2
INDEX=J+1
TMP=X(J)+X(INDEX)
X(INDEX)=X(J)-X(INDEX)
50 X(J)=TMP
C FOR INVERSE FFT COMPLEX CONJUGATE THE RESULT AND DIVE BY NPTS
IF(INVFLG.EQ.0) GO TO 70
RN=1./NPTS
DO 60 I=1,NPTS
60 X(I)=CONJG(X(I))*RN
70 CONTINUE
C COULD EXIT HERE FOR CONVOLUTIONS. ETC
C********************************************
C NOW FOR THE BIT-REVERSAL UNSHUFFLING
N2=NPTS/2
J=1
DO 90 I=1,NM1
IF(I.GE.J) GO TO 75
TMP=X(J)
X(J)=X(I)
X(I)=TMP
75 K=N2
80 IF(K.GE.J) GO TO 90
J=J-K
K=K/2
GO TO 80
90 J=J+K
RETURN
END
SUBROUTINE DECC (N, NDIM, AR, AI, IP, IER)
VERSION COMPLEX DOUBLE PRECISION
I CHANGED IMPLICIT REAL*8 TO IMPLICIT DOUBLE PRECISION

IMPLICIT DOUBLE PRECISION (A-H,O-Z)
INTEGER N,NDIM,IP,IERR1,NM1,K,KP1,M,I,J
DIMENSION AR(NDIM,N), AI(NDIM,N), IP(N)

C-----------------------------------------------------------------------
C MATRIX TRIANGULARIZATION BY GAUSSIAN ELIMINATION
C ------- MODIFICATION FOR COMPLEX MATRICES -------
C INPUT..  
C N = ORDER OF MATRIX.
C NDIM = DECLARED DIMENSION OF ARRAYS AR AND AI .
C (AR, AI) = MATRIX TO BE TRIANGULARIZED.
C OUTPUT..  
C AR(I,J), I.LE.J = UPPER TRIANGULAR FACTOR, U ; REAL PART.
C AI(I,J), I.LE.J = UPPER TRIANGULAR FACTOR, U ; IMAGINARY PART.
C AR(I,J), I.GT.J = MULTIPLIERS = LOWER TRIANGULAR FACTOR, I - L.
C REAL PART.
C AI(I,J), I.GT.J = MULTIPLIERS = LOWER TRIANGULAR FACTOR, I - L.
C IMAGINARY PART.
C IP(K), K.LT.N = INDEX OF K-TH PIVOT ROW.
C IP(N) = (-1)**(NUMBER OF INTERCHANGES) OR 0 .
C IER = 0 IF MATRIX A IS NONSINGULAR, OR K IF FOUND TO BE
C SINGULAR AT STAGE K.
C USE SOL TO OBTAIN SOLUTION OF LINEAR SYSTEM.
C IF IP(N)=0, A IS SINGULAR, SOL WILL DIVIDE BY ZERO.
C
C REFERENCE..
C C. B. MOLER, ALGORITHM 423, LINEAR EQUATION SOLVER,
C C.A.C.M. 15 (1972), P. 274.
C-----------------------------------------------------------------------
IER = 0
IP(N) = 1
IF (N .EQ. 1) GO TO 70
NM1 = N - 1
DO 60 K = 1,NM1
KP1 = K + 1
M = K
DO 10 I = KP1,N
    IF (DABS(AR(I,K))+DABS(AI(I,K)) .GT.
        & DABS(AR(M,K))+DABS(AI(M,K))) M = I
 10 CONTINUE
IP(K) = M
TR = AR(M,K)
TI = AI(M,K)
IF (M .EQ. K) GO TO 20
IP(N) = -IP(N)
AR(M,K) = AR(K,K)
AI(M,K) = AI(K,K)
AR(K,K) = TR
AI(K,K) = TI
20 CONTINUE
IF (DABS(AR)+DABS(TI) .EQ. 0.00) GO TO 80
DEN=TR*TR+TI*TI
TR=TR/DEN
TI=-TI/DEN
DO 30 I = KP1,N
   PRODR=AR(I,K)*TR-AI(I,K)*TI
   PRODI=AI(I,K)*TR+AR(I,K)*TI
   AR(I,K)=-PRODR
   AI(I,K)=-PRODI
30 CONTINUE
DO 50 J = KP1,N
   TR = AR(M,J)
   TI = AI(M,J)
   AR(M,J) = AR(K,J)
   AI(M,J) = AI(K,J)
   AR(K,J) = TR
   AI(K,J) = TI
IF (DABS(AR)+DABS(TI) .EQ. 0.00) GO TO 48
IF (TI .EQ. 0.00) THEN
   DO 40 I = KP1,N
      PRODR=-AI(I,K)*TI
      PRODI=AR(I,K)*TI
      AR(I,J) = AR(I,J) + PRODR
      AI(I,J) = AI(I,J) + PRODI
40 CONTINUE
GO TO 48
END IF
IF (TR .EQ. 0.00) THEN
   DO 45 I = KP1,N
      PRODR=AR(I,K)*TR
      PRODI=AI(I,K)*TR
      AR(I,J) = AR(I,J) + PRODR
      AI(I,J) = AI(I,J) + PRODI
45 CONTINUE
GO TO 48
END IF
DO 47 I = KP1,N
   PRODR=AR(I,K)*TR-AI(I,K)*TI
47 CONTINUE
\[ \text{PRODI} = \text{AI}(I,K) \times TR + \text{AR}(I,K) \times TI \]
\[ \text{AR}(I,J) = \text{AR}(I,J) + \text{PRODR} \]
\[ \text{AI}(I,J) = \text{AI}(I,J) + \text{PRODI} \]

47 CONTINUE
48 CONTINUE
50 CONTINUE
60 CONTINUE
70 K = N
    IF (DABS(AR(N,N)) + DABS(AI(N,N)) .EQ. 0.0) GO TO 80
    RETURN
80 IER = K
    IP(N) = 0
    RETURN
C----------------------- END OF SUBROUTINE DECC ------------------------
END
SUBROUTINE CKN(X, Y, SCK, SCI, NUM)
C
C CKN(#) HAS THE COMPLEX BESSEL FUNCTIONS
C CI(#) HAS THE FIRST DERIVATIVES OF THE CKN(#)
C
DOUBLE PRECISION RI0, EIO, RI1, EIl, RK0, EKO, RK1, EK1, XD, YD
DOUBLE PRECISION AA, CK(20,4), CI(20,4)
INTEGER NUM, K
REAL X, Y, FAC, SS
DOUBLE PRECISION SCI(Z0,4), SCK(20,4)
COMPLEX SUM

CALL XXX(X, Y, RI0, EIO, RI1, EIl, RK0, EKO, RK1, EK1)

C print *,x,y
C
IF(Y.NE.0.0) print *,x,y,RI0, EIO, RI1, EIl, RK0, EKO, RK1, EK1
XO=ONBLE(X)
C
C XO=DOUBLE(Y)
CK(1,1)=RI0
CK(1,2)=EIO
CK(1,3)=RK0
CK(1,4)=EKO
CK(2,1)=RI1
CK(2,2)=EIl
CK(2,3)=RK1
CK(2,4)=EK1
C
CI(1,1)=RI1
CI(1,2)=EIl
CI(1,3)=RK1
CI(1,4)=EK1
DO 80 J=1,4
SCI(1,J)=SNGL(CI(I,J))
SCK(1,J)=SNGL(CK(1,J))
80 
CONTINUE
IF(NUM.GE.1) GO TO 10
RETURN
10 CONTINUE
O0=2.0D0/(XO**2+Y0**2)
AA=(XO**2+YO**2)
C
C WRITE(6, *)AA
IF(NUM.EQ.1) GO TO 40
DO 20 I=3,NUM+2
IF((XO**2+YO**2).LE.0.1D0) GO TO 500
CKCI,-1)-CK(I-2)-XO*CKCI-1)+YO*CK(I-1, 2))
1 *DFLOAT(I-2)
CK(I,2)=CK(I-2)-XO*CK(I-1,2)+YO*CK(I-1,1))
1 *DFLOAT(I-2)
GO TO 510
500 CONTINUE
SUM=CMPLX(0.0,0.0)
SS=1.
DO 550 K=1,10
FAC=0.
SS=SS*FLOAT(K-1)
IF(XE.1) SS=1.
CALL FACTOR(K-1,FAC)
SUM=SUM+CMPLX(X, Y)/2.**(I-3+2*K))/FAC/SS
550 CONTINUE
CK(I,1)=OUBLE(REAL(SUM))
CK(I,2)=OUBLE(AIMAG(SUM))
C
510 CONTINUE
CKI,3)=CK(I-2,3)*DD*(XO*CK(I-1,3)+YO*CK(I-1,4))
1 *DFLOAT(I-2)
CKI,4)=CK(I-2,4)*DD*(XO*CK(I-1,4)+YO*CK(I-1,3))
1 *DFLOAT(I-2)
20 CONTINUE
C
40 CONTINUE
DO 30 I=2,NUM+1
CI(I,1)=CK(I-1,1)/AA*CK(I-1,2)+YO*CK(I-1,2))
CI(I,3)=500*CK((I-1,1)+CK(I-1,1))
C
C CI(I,2)=CK(I-1,2)/AA*CK(I-1,2)-YO*CK(I-1,2))
CI(I,2)=.5D0*(CK(I-1,2)+CK(I+1,2))
C CI(I,3)=CK(I-1,3)-DFLOAT(I-1)/AA*CK(I,3)+YD*
   C CI(I,3)=-0.5D0*(CK(I-1,3)+CK(I+1,3))
C 1 CK(I,3)
C CI(I,4)=CK(I-1,4)-DFLOAT(I-1)/AA*CK(I,4)-YD*
C CI(I,4)=-0.5D0*(CK(I-1,4)+CK(I+1,4))
C 1 CK(I,3)
30 CONTINUE
C
DO 100 I=1,NUM+1
DO 100 J=1,4
SCCI(J)=CI(I,J)
SCCK(J)=CK(I,J)
C print *,I,J,SCCI(J),SCCK(J)
100 CONTINUE
RETURN
END
FILE: BESSEL.F
PROGRAMMER:

LIBRARY: CAPLIB

PURPOSE: CALCULATE THE ZEROTH AND FIRST ORDER COMPLEX MODIFIED BESSEL FUNCTIONS OF THE FIRST AND SECOND KIND

SUBROUTINE XXX(X,Y,RIO,EIO,RI1,EI1,RKO,EKO,RK1,EK1)
DOUBLE PRECISION COEF(8,8),ARGU,ARGUT,CC,PI
DOUBLE PRECISION TETA,A,B,AA,BB,RIO,EIO,RI1,EI1,RKO,EKO
DOUBLE PRECISION RK1,EK1,R,AK,AL,XD,YD
REAL X,Y

THE FOLLOWING DATA WERE TAKEN FROM THE PAGES 378 AND 379 OF
"HANDBOOK OF MATHEMATICAL FUNCTIONS WITH FORMULAS, GRAPHS, AND MATHEMATICAL TABLES"
EDITED BY
M. ABROMOWITZ
AND
I. A. STEGAN

THE FORMULAS IN THE SAME PAGES WERE TRANSFERRED FROM REAL VARIABLES TO COMPLEX VARIABLES AND REFORMULATE ALL THE EQUATIONS.

DATA((COEF(I,J),J=1,8),I=1,8)
1 3.5156229, 3.0899424, 1.2067492, 0.2659732,
2 0.0360768, 0.0045813, 0.0000000, 0.0000000,
3 0.01328592, 0.00225319, -0.00157565, 0.00916281,
4 -0.02057706, 0.02635537, -0.01647633, 0.00392377,
5 0.01328592, 0.00225319, -0.00157565, 0.00916281,
6 -0.02057706, 0.02635537, -0.01647633, 0.00392377,
7 0.00301532, 0.00032411, 0.0000000, 0.0000000,
8 -0.03988024, -0.00362018, 0.00163801, -0.01031555,
9 0.02282967, -0.02895312, 0.01787654, -0.00420059,
10 0.42278420, 0.23069756, 0.03488590, 0.00262698,
DATA PI/3.1415926653589793238462643/

XI=DBLE(X)
YD=DBLE(Y)

CHECK FOR ARGUMENT TOO BIG FOR FORTRAN
COMPLEX EXPONENTIAL FUNCTION

IF(XD.EQ.0.000.AND.YD.GT.0.000) GO TO 100
IF(XD.EQ.0.000.AND.YD.LT.0.000) GO TO 200
ARGUT=YD/XD
TETA=DATAN(ARGUT)

print *,teta
GO TO 300
100 TETA=PI/2.0D0
GO TO 300
200 TETA=3.0D0*PI/2.0D0
300 CONTINUE
ARGu=XD**2D0+YD**2D0
R=DSQRT(ARGU)
IF(ABS(R).GT.88.0)) GO TO 999

I. CALCULATE BESSEL FUNCTION OF THE FIRST KIND (IO,I1)

IF(R.GT.3.75D0) GO TO 500

A=0.0D0
B=0.0D0
DO 400 I=1,6
CC=2D0*DFLOAT(I)
A=A+COEF(1,1)*((R/3.75D0)**(CC))*DCOS(CC*TETA)
400 B=B+COEF(1,1)*((R/3.75D0)**(CC))*DSIN(CC*TETA)
RIO=A+1.0D0
EIO=B
CC=DSIN(CC*TETA)
C
C
GO TO 700
C
C
500  CONTINUE
A=0.0D0
B=0.0D0
   DO 600 I=1,8
   A=A+COEF(2,I)*(3.75D0/R)**I*DCOS(DFLOAT(I)*TETA)
600  B=B-COEF(2,I)*(3.75D0/R)**I*DSIN(DFLOAT(I)*TETA)
   RIO=DEXP(R*DCOS(TETA))/DSQRT(R)*(DCOS(R*DSIN(TETA)-TETA/2.0D0)
       *(A+.39894228D0)-B*DSIN(R*DSIN(TETA)-TETA/2.0D0))
   EIO=DEXP(R*DCOS(TETA))/DSQRT(R)*((A+.398894228D0)*DSIN(R*
       DSIN(TETA)-TETA/2.0D0)+B*DCOS(R*DSIN(TETA)-TETA/2.0D0))
C
   IF(R.GT.3.75D0) GO TO 900
C
700  CONTINUE
C
A=0.0D0
B=0.0D0
   DO 800 I=1,6
   A=A+COEF(3,I)*(R/3.75D0)**I*DCOS(DFLOAT(I)*TETA)
800  B=B+COEF(3,I)*(R/3.75D0)**I*DSIN(DFLOAT(I)*TETA)
   RI1=(A+.5D0)*R*DCOS(TETA)-R*B*DSIN(TETA)
   EI1=B*R*DCOS(TETA)+(A+.5D0)*R*DSIN(TETA)
C
C
GO TO 1100
C
C
900  CONTINUE
A=0.0D0
B=0.0D0
   DO 1000 I=1,8
   A=A+COEF(4,I)*(3.75D0/R)**I*DCOS(DFLOAT(I)*TETA)
1000 B=B-COEF(4,I)*(3.75D0/R)**I*DSIN(DFLOAT(I)*TETA)
   RI1=DEXP(R*DCOS(TETA))/DSQRT(R)*(DCOS(R*DSIN(TETA)-TETA/2.0D0)
       *(A+.39894228D0)-B*DSIN(R*DSIN(TETA)-TETA/2.0D0))
   EI1=DEXP(R*DCOS(TETA))/DSQRT(R)*((A+.39894228D0)*DSIN(R*
       DSIN(TETA)-TETA/2.0D0)+B*DCOS(R*DSIN(TETA)-TETA/2.0D0))
C
1100 CONTINUE
C
II THIS PART OF THE PROGRAM CALCULATES THE FUNCTIONS K0 AND K1

IF(R.GT.2.0D0) GO TO 1300

AA=0.0D0
BB=0.0D0
DO 1200 I=1,6
  AA=COEF(5,I)*(R/2.0D0)**(2*I)*
      DCOS(2.0D0*DFLOAT(I)*TETA)+AA
  BB=COEF(5,I)*(R/2.0D0)**(2*I)*
      DSIN(2.0D0*DFLOAT(I)*TETA)+BB
1200
RKO=AA-RIO*DLOG(R/Z.0D0)+TETA*EIO-.5771572
EKO=BB-RIO*TETA-EIO*DLOG(R/2.0D0)

GO TO 1500
1300 CONTINUE
AA=0.0D0
BB=0.0D0
DO 1400 I=1,6
  AA=COEF(6,I)*(2.0D0/R)**I*DCOS(DFLOAT(I)*TETA)+AA
  BB=BB-COEF(6,I)*(2.0D0/R)**I*DSIN(DFLOAT(I)*TETA)
1400
RKO=DEXP(-R*DCOS(TETA))/DSQRT(R)*
     (DCOS(R*DSIN(TETA)+TETA/Z.0D0)*(AA+1.25331414D0)+
      BB*DSIN(R*DSIN(TETA)+TETA/2.0D0))
EKO=DEXP(-R*DCOS(TETA))/DSQRT(R)*
     -(AA+1.25331414D0)*DSIN(R*DSIN(TETA)+TETA/2.0D0))

IF(R.GT.2.0D0) GO TO 1700
1500 CONTINUE

AK=0.0D0
AL=0.0D0
AA=0.0D0
BB=0.0D0
DO 1600 I=1,6
  AA=AA+COEF(7,I)*(R/Z.0D0)**I*DCOS(DFLOAT(I)*TETA)
  BB=BB+COEF(7,I)*(R/Z.0D0)**I*DSIN(DFLOAT(I)*TETA)
1600
RK1=(AA+AK+1.0D0)/R*DCOS(TETA)+(BB+AL)/R*DSIN(TETA)
EK1=(BB+AL)/R*DCOS(TETA)-(AA+AK+1.0D0)/R*DSIN(TETA)
C
RETURN
C
C
1700 CONTINUE
C
C
AA=0.0D0
BB=0.0D0
DO 1800 I=1,6
AA=AA+COEF(8,I)*(2.0D0/R)**I*DCOS(DFLOAT(I)*TETA)
1800 BB=BB-COEF(8,I)*(2.0D0/R)**I*DSIN(DFLOAT(I)*TETA)
RK1=DEXP(-R*DCOS(TETA))/DSQRT(R)*((AA+1.25331414D0)*DCOS(R*DSIN(TETA)+TETA/2.0D0)+BB*DSIN(R*DSIN(TETA)+TETA/2.0D0))
EK1=DEXP(-R*DCOS(TETA))/DSQRT(R)*(-(AA+1.2533141400)*DSIN(R*DSIN(TETA)+TETA/2.000)+BB*DCOS(R*DSIN(TETA)+TETA/2.0D0))
RETURN
C
C
C
CHECK FOR ERRORS
C
C
999 CONTINUE
WRITE(5,10)
10 FORMAT(' ARGUMENT TOO LARGE...EROR!')
RETURN
END