```
c nufdtd.f: Northeastern University 3-D
c Finite Difference Time Domain Code
c Work by          Adnan Sahin (8/12/97), Carey Rappaport (5/1998),
c John Talbot (8/1999), Panos Kosmas (6/2000), M. Farid (8/2004)
c Seth Baum (8/2004-8/2006)

        PROGRAM nufdtd
        INCLUDE "nufdtd_params.f"

! This code assumes mu=mu0.
! All parameters are defined in MKS units.

! If you add a new soil type, you must adjust code
!  everywhere a !nwsltp marker exists (in both files)

! If you add a new excitation type, you must adjust code
!  everywhere a !nwexcntp marker exists (all are in this file)

! If you add a new excitation shape, you must adjust code
!  everywhere a !nwexcnshp marker exists (all are in this file)

c *** BEGIN ENTERING INPUT DATA
        print*,'Input data.  Corresponding variable in (parentheses).'

c *** SPACE DIMENSIONS
        print*,''
        print*,'*** SPACE DIMENSIONS'
        print*,'Enter dimension configuration'
        print*,' 1 - 2D: TEy (Ex,Hy,Ez on x-z plane)'
        print*,' 2 - 2D: TMy (Hx,Ey,Hz on x-z plane)'
        print*,' 3 - 3D: (Ex,Ey,Ez,Hx,Hy,Hz)'
                read*,ndim  ! Dimension Configuration
        do while(ndim<1.or.ndim>3)
                print*,'Dimension configuration entry out of range.'
                print*,'Enter dimension configuration'
                        read*,ndim  ! built-in object type
        enddo
        open(unit=1,file='input.txt',status='unknown')  ! input.txt
         write(1,"(i1,a)")ndim,'  ! Dimension configuration'  ! input.txt
        close(unit=1)  ! input.txt
        if(ndim<3)then  ! 2D
                ny=1
                print*,'Enter grid size (nx,nz)'
                do while(nx==0.or.nz==0.or.nx>npx.or.nz>npz)
                        if(nx>npx.or.nz>npz)then
                                print*,'Error: Values too large. Decrease them,'
                                print*,'or increase grid size in the parameters file.'
                        endif
                        read*,nx,nz
                enddo
        open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
         write(1,"(i5,i5,a)")nx,nz,
   $  '  ! Grid size (see also params file)'  ! input.txt
        close(unit=1)  ! input.txt
                dely=1
                print*,'Enter grid cell size in cm (delx,delz)'
```

```
              do while(delx==0.or.delz==0)
                      read*,delx,delz
              enddo
      open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
       write(1,"(f7.3,f7.3,a)")delx,delz,
   $  '  ! Grid cell size in cm'  ! input.txt
      close(unit=1)  ! input.txt
      else  ! 3D
              print*,'Enter grid size (nx,ny,nz)'
      do while(nx==0..or.ny==0..or.nz==0..or.nx>npx.or.ny>npy.or.nz>npz)
                      if(nx>npx.or.ny>npy.or.nz>npz)then
                              print*,'Error: Values too large. Decrease them,'
                              print*,'or increase grid size in the parameters file.'
                      endif
                      read*,nx,ny,nz
              enddo
      open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
       write(1,"(i5,i5,i5,a)")nx,ny,nz,
   $  '  ! Grid size (see also params file)'  ! input.txt
      close(unit=1)  ! input.txt
              print*,'Enter grid cell size in cm (delx,dely,delz)'
              do while(delx==0..or.dely==0..or.delz==0.)
                      read*,delx,dely,delz
              enddo
      open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
       write(1,"(f7.3,f7.3,f7.3,a)")delx,dely,delz,
   $  '  ! Grid cell size in cm'  ! input.txt
      close(unit=1)  ! input.txt
      endif
      nx1=nx-1
      ny1=ny-1
      nz1=nz-1
      delx=delx/100.  ! Conversion to meters
      dely=dely/100.
      delz=delz/100.
      delr=sqrt(delx**2+dely**2+delz**2)  ! For Points Per Wavelength
      dtmax=1/sqrt(1/delx**2+1/dely**2+1/delz**2)/c0  ! For Courant Condition

c *** TIME DIMENSION
      print*,''
      print*,'*** TIME DIMENSION'
      print*,'Enter total number of time steps to run the code (nts)'
      do while(nstop==0)
              read*,nstop
      enddo
      open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
       write(1,"(i6,a)")nstop,'  ! number of time steps'  ! input.txt
      close(unit=1)  ! input.txt
      print*,'Enter time step size in sec (dt)'
      dt=dtmax+1
      do while(dt>dtmax)
      print*,'To statisfy Courant Condition, need dt < ',dtmax
              read*,dt
      enddo
      open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
       write(1,"(e10.4,a)")dt,'  ! Time step size (sec)'  ! input.txt
      close(unit=1)  ! input.txt

c *** CREATE NEW MATERIALS
      print*,''
      print*,'*** CREATE NEW MATERIALS'
      print*,'Enter number of new materials.'
      print*,'Type -1 to view built-in materials.'
            read*,nnewmat
            if(nnewmat==-1) CALL MATPRINT  ! Display material options
            do while(nnewmat<0.or.nnewmat>nemt)
                  print*,'Need entry to be between 0 and ',nemt
                        read*,nnewmat
            enddo
      open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
       write(1,"(i2,a)")nnewmat,'  ! Number of new materials'  ! input.txt
      close(unit=1)  ! input.txt
      namt=nbmt+nnewmat  ! Number of available materials
      do n=nbmt+1,namt  ! For each built-in material
            print*,' *** MATERIAL ID #',n
            print*,'Enter relative permittivity'
                  read*,epsr(n)
                  do while(epsr(n)<1)
                        print*,'Need relative permittivity >= 1.'
                        read*,epsr(n)
                  enddo
      open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
       write(1,"(f7.3,a,i3)")epsr(n),
  $  '  ! Relative permittivity, id #',n  ! input.txt
      close(unit=1)  ! input.txt
            print*,'Type 0 for non-dispersive or 1 for dispersive'
                  read*,ndisp
                  do while(ndisp<0.or.ndisp>1)
                        print*,'Entry must = 0 or 1.'
                        read*,ndisp
                  enddo
      open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
       write(1,"(i2,a,i3)")ndisp,
  $  '  ! 0 for non-dispersive or 1 for dispersive, id #',n  ! input.txt
      close(unit=1)  ! input.txt
            if(ndisp==0)then  ! Non-dispersive
                  print*,'Enter conductivity'
                        read*,b0(n)
      open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
       write(1,"(f7.3,a,i3)")b0(n),
  $  '  ! conductivity for id #',n  ! input.txt
      close(unit=1)  ! input.txt
            elseif(ndisp==1)then  ! Dispersive
                  print*,'Enter a1, b0, b1, & b2'
                        read*,a1(n),b0(n),b1(n),b2(n)
      open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
       write(1,"(f7.3,f7.3,f7.3,f7.3,a,i3)")a1(n),b0(n),b1(n),b2(n),
  $  '  ! a1, b0, b1, & b2 for id #',n  ! input.txt
      close(unit=1)  ! input.txt
            endif
      enddo
      CALL SETUP  ! Initializes certain problem parameters

c *** BACKGROUND LAYERS
      print*,''
      print*,'*** BACKGROUND LAYERS'
      print*,'Enter number of background layers:'
            read*,nbglr
            do while(nbglr<1.or.nbglr>nz)
                  print*,'Need entry to be between 1 and ',nz
                        read*,nbglr
            enddo
      open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
       write(1,"(i3,a)")nbglr,'  ! Number of background layers'  ! input.txt
      close(unit=1)  ! input.txt
      do i=1,nbglr  ! For each layer
            if(i==nbglr)then  ! Top layer
                  if(nbglr==1)then
                        nbgth(i)=nz  ! Thickness layer
                  else
                        nbgth(i)=nz-nbght(i-1)  ! Thickness of top layer
                  endif
                  nbght(i)=nz  ! Height of top layer
            else  ! Not top layer
                  print*,'Enter thickness of layer',i,' (in grid cells)'
                        read*,nbgth(i)
                  do while(nbgth(i)<=0)
                        print*,'Need thickness > 0'
                        print*,'Enter thickness of layer',i,' (in grid cells)'
                              read*,nbgth(i)
                  enddo
                  if(i==1)then  ! Bottom layer
                        nbght(i)=nbgth(i)
                  else
                        nbght(i)=nbght(i-1)+nbgth(i) ! Height of layer i
                  endif
                  do while(nbght(i)>nz-nbglr+i)
                        print*,'Need thickness <=',nz-nbglr+i-nbght(i-1)
                        print*,'(Not enough room left for remaining layers)'
              print*,'Enter thickness of soil layer',i,' (in grid cells)'
                              read*,nbgth(i)
                        do while(nbgth(i)<0)
                    print*,'Need thickness > 0 (No infinitely thin layers)'
              print*,'Enter thickness of soil layer',i,' (in grid cells)'
                                    read*,nbgth(i)
                        enddo
                  enddo
      open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
       write(1,"(i4,a,i3)")nbgth(i),'  ! Thickness of soil layer',i  ! input.txt
      close(unit=1)  ! input.txt
            endif
            print*,'Enter material type of layer',i
            print*,'Type 0 to view options.'
                  read*,ibgtp(i)
                  if(ibgtp(i)==0)CALL MATPRINT  ! Display material options
                  do while(ibgtp(i)<1.or.ibgtp(i)>namt)
                        print*,'Need entry to be between 1 and ',namt
```

```
                              read*,ibgtp(i)
                  enddo
      open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
       write(1,"(i3,a,i3)")ibgtp(i),'  ! Material type of layer',i  ! input.txt
      close(unit=1)  ! input.txt
      enddo   ! (For each background layer)
      do n=1,nbglr  ! Defining background layers
            CALL SHP_RP(1,1,nbght(n)-nbgth(n)+1,nx,ny,nbgth(n),ibgtp(n))
      enddo

c *** FOREGROUND OBJECTS
      print*,''
      print*,'*** FOREGROUND OBJECTS'
      print*,'Enter number of foreground objects:'
      print*,' Note: If objects overlap, then those inputted later'
      print*,' will overwrite those inputted earlier.'
            read*,nobj  ! number of foreground objects
      open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
       write(1,"(i3,a)")nobj,
     $  '  ! Number of foreground objects'  ! input.txt
      close(unit=1)  ! input.txt
      if(ndim<3)then  ! 2D
        do n=1,nobj  ! For each foreground object
            print*,'Enter type of object # ',n
            print*,' 0 - Material Input File'
            print*,' 1 - Rectangle'
            print*,' 2 - Circle'
                  read*,nobjtyp  ! built-in object type
            do while(nobjtyp<0.or.nobjtyp>2)
                  print*,'Object type out of range.'
                  print*,'Need 0 <= Object type <= 2.'
                  print*,'Enter object type:'
                        read*,nobjtyp  ! built-in object type
            enddo
      open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
       write(1,"(i1,a)")nobjtyp,
     $  '  ! Object type'  ! input.txt
      close(unit=1)  ! input.txt
            if(nobjtyp==0)then          ! Read in a Material Input File
                  CALL FOBJ_MT_INFILE
            elseif(nobjtyp==1)then   ! Build a Rectangular Prism
                  CALL FOBJ_RECT_PRISM
            elseif(nobjtyp==2)then   ! Build a Cylinder
                  CALL FOBJ_CIRCLE
            else  ! Error in built-in object type #
                  print*,'Error in built-in object type #'
            endif
        enddo  ! (For each foreground object)
      else  ! 3D
        do n=1,nobj  ! For each foreground object
            print*,'Enter type of object # ',n
            print*,' 0 - Material Input File'
            print*,' 1 - Rectangular Prism'
            print*,' 2 - Cylinder (Use for landmines)'
            print*,' 3 - Sphere'
            print*,' 4 - Monopole Antenna'
```

```
                  read*,nobjtyp  ! built-in object type
            do while(nobjtyp<0.or.nobjtyp>4)
                  print*,'Object type out of range.'
                  print*,'Need 0 <= Object type <= 4.'
                  print*,'Enter object type:'
                        read*,nobjtyp  ! built-in object type
            enddo
      open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
       write(1,"(i1,a)")nobjtyp,
  $  '  ! Object type'  ! input.txt
      close(unit=1)  ! input.txt
            if(nobjtyp==0)then          ! Read in a Material Input File
                  CALL FOBJ_MT_INFILE
            elseif(nobjtyp==1)then   ! Build a Rectangular Prism
                  CALL FOBJ_RECT_PRISM
            elseif(nobjtyp==2)then   ! Build a Cylinder
                  CALL FOBJ_CYLINDER
            elseif(nobjtyp==3)then   ! Build a Sphere
                  CALL FOBJ_SPHERE
            elseif(nobjtyp==4)then   ! Build a Monopole Antenna
                  CALL FOBJ_MONOPOLE
            else  ! Error in built-in object type #
                  print*,'Error in built-in object type #'
            endif
       enddo  ! (For each foreground object)
      endif

c *** MATERIAL OUTPUT FILES
      print*,''
      print*,'*** MATERIAL OUTPUT FILES'
      if(ndim<3)then  ! 2D
            CALL WRITE_MT(2,1)
      elseif(ndim==3)then  ! 3D
            print*,'How many material output files to print?'
                  read*,nmf  ! number of material output files printed
      open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
       write(1,"(i5,a)")nmf,
  $  '  ! Number of material output files printed'  ! input.txt
      close(unit=1)  ! input.txt
            do n=1,nmf  ! For each material output file
                  print*,'Enter direction, location of mat. out. file # ',n
             print*,' First number: direction: 1=x (y-z plane); 2=y; 3=z'
                  print*,' Second number: location: slice coordinate'
                        read*,mtdir,mtloc  ! direction, location of material output file
                  if(mtdir==1)then
                        do while(mtloc<1.or.mtloc>nx)
                              print*,'Need 0 < location < ',nx
                              print*,'Enter location only'
                                    read*,mtloc  ! location of material output file
                        enddo
                  elseif(mtdir==2)then
                        do while(mtloc<1.or.mtloc>ny)
                              print*,'Need 0 < location < ',ny
                              print*,'Enter location only'
                                    read*,mtloc  ! location of material output file
                        enddo
                        elseif(mtdir==3)then
                                do while(mtloc<1.or.mtloc>nz)
                                        print*,'Need 0 < location < ',nz
                                        print*,'Enter location only'
                                                read*,mtloc  ! location of material output file
                                enddo
                        else
                                print*,'Error in direction.'
                        endif
        open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
         write(1,"(i3,i5,a)")mtdir,mtloc,
     $  '  ! direction, location of material output file'  ! input.txt
        close(unit=1)  ! input.txt
                CALL WRITE_MT(mtdir,mtloc)
                enddo
        endif

c *** EXCITATION
        print*,''
        print*,'*** EXCITATION'
        print*,'Enter source type:'
        print*,' 1 - User-defined hard source'
        print*,' 2 - User-defined soft source'
        if(ndim==3)then  ! 3D
                print*,' 3 - Monopole hard source'
                print*,' 4 - Monopole soft source' !nwexcntp
        endif
        do while(nsrctyp==0.)
                read*,nsrctyp
        enddo
        open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
         write(1,"(i1,a)")nsrctyp,'  ! Source type'  ! input.txt
        close(unit=1)  ! input.txt
        if(nsrctyp==1.or.nsrctyp==2)then
                print*,'Enter number of excitation points'
                        read*,nexcnpts
        open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
         write(1,"(i3,a)")nexcnpts,'  ! Number of excitation points'  ! input.txt
        close(unit=1)  ! input.txt
                do nep=1,nexcnpts         ! For each excitation point
                        print *,'Enter excitation point coordinates #',nep
                        if(ndim<3)then  ! 2D
                                my(nep)=1
                                read *,mx(nep),mz(nep)
        open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
         write(1,"(i5,i6,a,i3)")mx(nep),mz(nep),
     $  '  ! Excitation point coords #',nep  ! input.txt
        close(unit=1)  ! input.txt
                        else  ! 3D
                                read *,mx(nep),my(nep),mz(nep)
        open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
         write(1,"(i5,i6,i6,a,i3)")mx(nep),my(nep),mz(nep),
     $  '  ! Excitation point coords #',nep  ! input.txt
        close(unit=1)  ! input.txt
                        endif
                        print *,'Enter directional excitation strengths #',nep
                                read *,esx(nep),esy(nep),esz(nep)
        open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
         write(1,"(f4.2,f5.2,f5.2,a)")esx(nep),esy(nep),esz(nep),
     $  '  ! Directional exn strengths'  ! input.txt
        close(unit=1)  ! input.txt
                enddo  ! (For each excitation point)
        endif
        print*,''
        print*,'Enter pulse shape'
        print*,' 1 - Narrow-Width Gaussian'
        print*,' 2 - Cosine-Modulated Gaussian'   !nwexcnshp
        print*,' 3 - Narrow-Width Half-Gaussian'  !nwexcnshp
        print*,' 4 - Cosine-Modulated Half-Gaussian'        !nwexcnshp
                read*,npulseshape
        do while(npulseshape<1.or.npulseshape>4)
                print*,'Pulse type out of range.'
                print*,'Enter pulse shape'
                        read*,npulseshape
        enddo
        open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
         write(1,"(i1,a)")npulseshape,
     $  '  ! Pulse shape'  ! input.txt
        close(unit=1)  ! input.txt
        print*,'Enter Gaussian pulse width in time steps'
                read*,gaussw
        open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
         write(1,"(f6.1,a)")gaussw,'  ! Gaussian pulse width (time steps)'  ! input.txt
        close(unit=1)  ! input.txt
        print*,'Enter Gaussian pulse peak time in time steps'
        do while(gausspt==0.)
                read*,gausspt
        enddo
        open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
         write(1,"(f6.1,a)")gausspt,'  ! Gaussian peak time (time steps)'  ! input.txt
        close(unit=1)  ! input.txt
        if(npulseshape==2.or.npulseshape==4)then
                print*,'Enter the pulse frequency in Hertz'
                do while(freqf==0.)
                        read*,freqf
                enddo
        open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
         write(1,"(e9.4,a)")freqf,'  ! Pulse frequency'  ! input.txt
        close(unit=1)  ! input.txt
                do i=1,nx
                 do j=1,ny
                  do k=1,nz
                        if(epsr(mtx(i,j,k))>epsrmax)then  ! If biggest epsr yet found
                                epsrmax=epsr(mtx(i,j,k))  ! Set it to epsrmax
                        endif
                  enddo
                 enddo
                enddo
                cmin=c0/sqrt(epsrmax)  ! Minimum wave speed in simulation
                wmin=c0/freqf  ! Minimum wavelength
                print*,'Minimum wavelength = ',wmin
                print*,' Min points per wavelength in x-direction:',
   $            wmin/delx
                print*,' Min points per wavelength in y-direction:',
   $            wmin/dely
                print*,' Min points per wavelength in z-direction:',
   $            wmin/delz
                print*,' Min points per wavelength along diagonal:',
   $            wmin/delr
                print*,''
        endif

c *** FIELD COMPONENT OUTPUT
        print*,''
        print*,'*** FIELD COMPONENT OUTPUT'
        print*,'Enter number of time steps between outputs'
        do while(ntime==0)
                read*,ntime
        enddo
        open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
         write(1,"(i6,a)")ntime,'  ! Number of time steps between outputs'  ! input.txt
        close(unit=1)  ! input.txt
        if(ndim==1)then  ! TEy
                nout=3
                mfld(1)=4
                mdir(1)=2
                mloc(1)=1
                mfld(2)=2
                mdir(2)=2
                mloc(2)=1
                mfld(3)=6
                mdir(3)=2
                mloc(3)=1
        elseif(ndim==2)then  ! TMy
                nout=3
                mfld(1)=1
                mdir(1)=2
                mloc(1)=1
                mfld(2)=5
                mdir(2)=2
                mloc(2)=1
                mfld(3)=3
                mdir(3)=2
                mloc(3)=1
        elseif(ndim==3)then  ! 3D
                print*,'Enter number of field component slice series'
                        read*,nout
        open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
         write(1,"(i3,a)")nout,'  ! # field component slice series'  ! input.txt
        close(unit=1)  ! input.txt
                print*,'Enter field component slice variables ',
   $     '(field,direction,location)'
                print*,' field: Field to output.'
                print*,'  1 - Hx; 2- Hy; 3 - Hz; 4 - Ex; 5 - Ey; 6 - Ez'
                print*,' direction: Direction of slice.'
                print*,'  1 - x; 2 - y; 3 - z'
                print*,' location: Location (coordinate) of slice.'
                do no=1,nout
```

```
                        print*,'Enter variables (integers only) for slice',no
                        read*,mfld(no),mdir(no),mloc(no)
        open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
         write(1,"(i2,i2,i5,a,i3)")mfld(no),mdir(no),mloc(no),
   $  '  ! Specs, field component slice #',no  ! input.txt
        close(unit=1)  ! input.txt
                enddo
        endif

        iii=1
c****************************************************************
C    BEGIN MAIN LOOP FOR FIELD COMPUTATIONS AND DATA SAVING
c****************************************************************
        do nsts=1,nstop  ! For each simulation time step

C *** ADVANCE SCATTERED FIELDS
        if(ndim==1)then  ! TEy
                CALL HYSFLD  ! Hz update equation
        elseif(ndim==2)then  ! TMy
                CALL HXSFLD  ! Hx update equation
                CALL HZSFLD  ! Hy update equation
        elseif(ndim==3)then  ! 3D
                CALL HXSFLD  ! Hx update equation
                CALL HYSFLD  ! Hz update equation
                CALL HZSFLD  ! Hy update equation
        endif
        if(ndim==1)then  ! TEy
                CALL EXSFLD  ! Ex update equation
                CALL RADEXZ  ! Ex absorbing boundary at z=1, z=nz
                CALL EZSFLD  ! Ez update equation
                CALL RADEZX  ! Ez absorbing boundary at x=1, x=nx
        elseif(ndim==2)then  ! TMy
                CALL EYSFLD  ! Ey update equation
                CALL RADEYX  ! Ey absorbing boundary at x=1, x=nx
                CALL RADEYZ  ! Ey absorbing boundary at z=1, z=nz
        elseif(ndim==3)then  ! 3D
                CALL EXSFLD  ! Ex update equation
                CALL RADEXY  ! Ex absorbing boundary at y=1, y=ny
                CALL RADEXZ  ! Ex absorbing boundary at z=1, z=nz
                CALL EYSFLD  ! Ey update equation
                CALL RADEYX  ! Ey absorbing boundary at x=1, x=nx
                CALL RADEYZ  ! Ey absorbing boundary at z=1, z=nz
                CALL EZSFLD  ! Ez update equation
                CALL RADEZX  ! Ez absorbing boundary at x=1, x=nx
                CALL RADEZY  ! Ez absorbing boundary at y=1, y=ny
        endif

c *** ANTENNA (PULSE) EXCITATION
        if(npulseshape==1)then  ! Narrow-Width Gaussian
                temp=exp(-((nsts-gausspt)/gaussw)**2)  ! Excitation strength
        elseif(npulseshape==2)then  ! Cosine-Modulated Gaussian
                temp=exp(-((nsts-gausspt)/gaussw)**2)
   $             *cos(2.*pi*freqf*real(nsts)*dt)  ! Excitation strength
        elseif(npulseshape==3)then  ! Narrow-Width Half-Gaussian
                if(nsts<gausspt)then
                        temp=exp(-((nsts-gausspt)/gaussw)**2)  ! Excitation strength
```

```
              else
                      temp=1  ! Excitation strength
              endif
      elseif(npulseshape==4)then  ! Cosine-Modulated Half-Gaussian
              if(nsts<gausspt)then
                      temp=exp(-((nsts-gausspt)/gaussw)**2)
$              *cos(2.*pi*freqf*real(nsts)*dt)  ! Excitation strength
              else
                      temp=cos(2.*pi*freqf*real(nsts)*dt)  ! Excitation strength
              endif
      else !nwexcnshp
              print*,'Error in pulse type.'
      endif
      ! Print pulse shape
      if(nsts==1)then  ! First Time Step
              open(unit=1,file='pulse.dat',status='unknown')
                      write(1,"(f6.3)")temp
              close(unit=no)
      else
              open(unit=1,file='pulse.dat',access='append',status='old')
                      write(1,"(f6.3)")temp
              close(unit=no)
      endif
      if(nsrctyp==1)then  ! User-defined hard source
              do m=1,nexcnpts
                      exs(mx(m),my(m),mz(m))=temp*esx(m)
                      eys(mx(m),my(m),mz(m))=temp*esy(m)
                      ezs(mx(m),my(m),mz(m))=temp*esz(m)
              enddo
      elseif(nsrctyp==2)then  ! User-defined soft source
              do m=1,nexcnpts
                      exs(mx(m),my(m),mz(m))=temp*esx(m)+exs(mx(m),my(m),mz(m))
                      eys(mx(m),my(m),mz(m))=temp*esy(m)+eys(mx(m),my(m),mz(m))
                      ezs(mx(m),my(m),mz(m))=temp*esz(m)+ezs(mx(m),my(m),mz(m))
              enddo
      elseif(nsrctyp==3)then  ! Monopole Antenna hard source
       do i=iantctr-ishieldrad,iantctr+ishieldrad  ! The x-y plane square
        do j=jantctr-ishieldrad,jantctr+ishieldrad  ! enclosing the antenna
              r=sqrt(real(i-iantctr)**2+real(j-jantctr)**2)
              if(icorerad<r.and.r<=idierad)then  ! Dielectric
                      adj=real(i-iantctr)
                      opp=real(j-jantctr)
                      hyp=sqrt(adj**2+opp**2)
                      costheta=adj/hyp
                      sintheta=opp/hyp
                      exs(i,j,kanttop-1)=temp*costheta
                      eys(i,j,kanttop-1)=temp*sintheta
              endif
        enddo
       enddo
      elseif(nsrctyp==4)then  ! Monopole Antenna soft source
       do i=iantctr-ishieldrad,iantctr+ishieldrad  ! The x-y plane square
        do j=jantctr-ishieldrad,jantctr+ishieldrad  ! enclosing the antenna
              r=sqrt(real(i-iantctr)**2+real(j-jantctr)**2)
              if(icorerad<r.and.r<=idierad)then  ! Dielectric
                      adj=real(i-iantctr)
```

```
                  opp=real(j-jantctr)
                  hyp=sqrt(adj**2+opp**2)
                  costheta=adj/hyp
                  sintheta=opp/hyp
                  exs(i,j,kanttop-1)=temp*costheta+exs(i,j,kanttop-1)
                  eys(i,j,kanttop-1)=temp*sintheta+eys(i,j,kanttop-1)
            endif
       enddo
       enddo !nwexcntp
      endif

c *** WRITE OUTPUT
      if(mod(nsts,ntime)==0)then  ! Output this time step
            CALL WRITEF_FC
            print*,'Output time step ',iii
            iii=iii+1  ! SelfNote: move this line to a more logical place?
      endif

      enddo  ! (For each time step)
c**********************************************************
C    END MAIN LOOP FOR FIELD COMPUTATIONS AND DATA SAVING
c**********************************************************

      STOP
      END


c*******************************************************
c*******************************************************
      SUBROUTINE MATPRINT
      INCLUDE "nufdtd_params.f"

! Displays the available material choices

      print*,'   '
      print*,' *** List of available materials *** '
      print*,'   '
      print*,' [0]: non-dispersive; [1]: dispersive'
      print*,' tss: "time step size"; f: "frequency"'
      print*,'   '
      print*,' If tss & f values are provided, they must be used'
      print*,' for the simulation to run properly.'
      print*,'   '
      print*,'  1=Free Space'
      print*,'  2=Metal (PEC)'
      print*,'  3=Dielectric; relative permittivity = 2.3 [0]'
      print*,'  4=TNT (Use for landmines) [0]'
      print*,'  5=Dielectric modeled as dispersive material [1]'
      print*,'    (relative permittivity = 2.3)'
      print*,'  6=Lossy Puerto Rican soil; tss=20ps; f=1.5GHz [1]'
      print*,'  7=Lossy Puerto Rican soil; tss=10ps; f=1.5GHz [1]'
      print*,'  8=Lossy Bosnian soil;  tss=20ps; f=1.5GHz [1]'
      print*,'  9=Lossy Bosnian soil; tss=100ps; f=1.5GHz [1]'
      print*,' 10=Lossy Bosnian soil;   tss=2ps; f=1.5GHz [1]'
      print*,' 11=Water; tss=50 ps; f=100MHz [1]'
      print*,' 12=Bosnian soil,  2.5% water; tss=50ps; f=100MHz [1]'
      print*,' 13=Bosnian soil,  5.0% water; tss=50ps; f=100MHz [1]'
      print*,' 14=Bosnian soil, 10.0% water; tss=50ps; f=100MHz [1]'
      print*,' 15=Bosnian soil, 20.0% water; tss=50ps; f=100MHz [1]'
      print*,' 16=Sandy soil,  4.0% water; tss=2ps; f=1.3GHz [1]'
      print*,' 17=Sandy soil, 17.0% water; tss=2ps; f=1.3GHz [1]'
      print*,' 18=Sandy soil,  4.0% water; tss=20ps; f=1.3GHz [1]'
      print*,' 19=Sandy soil, 17.0% water; tss=20ps; f=1.3GHz [1]'
      print*,' 20=Sandy soil, 17.0% water; tss=6ps; f=1.3GHz [1]'
      print*,' 21=Teflon (Use for antenna subustrate) [1]'  !nwsltp
      print*,' 22-34=Optional user-defined material'
      print*,'   '

      RETURN
      END


c************************************************************
c************************************************************
      SUBROUTINE SETUP
      INCLUDE "nufdtd_params.f"

! Initializes built-in material properties & performs calculations
! for derived properties for built-in & user-inputted materials.

! Note Fortran automatically defaults variable values to 0.
! Note if a time step size and frequency are specified for a material,
! they must be used for the simulation to run properly.

c *** BUILT-IN MATERIAL INITIALIZATIONS
 ! epsr: relative permittivity
 ! b0: conductivity (also used in dispersion approximation)
 ! a1, b1, b2: other dispersion approximation parameters
!  1 Free Space
      epsr(1)=1.0
!  2 Metal/Perfect Electric Conductor (PEC)
      epsr(2)=1.0
!  3 Non-dispersive dielectric
      epsr(3)=2.3
      b0(3)=0.00000056
!  4 TNT
      epsr(4)=2.9
      b0(4)=0.0001
! 5 Dielectric
      epsr(5)=2.3
! 6 Lossy Puerto Rican Soil at 20 ps; f=1.5GHz
      epsr(6)=4.167
      a1(6)=-0.88
      b0(6)=0.916249
      b1(6)=-1.67662
      b2(6)=0.761072
! 7 Lossy Puerto Rican Soil at 10 ps; f=1.5GHz
      epsr(7)=1.95563
      a1(7)=-0.95
      b0(7)=3.76795
      b1(7)=-7.30659
      b2(7)=3.53892
```

```
! 8 Lossy Bosnian Soil at 20 ps; f=1.5GHz
        epsr(8)=5.03815
        a1(8)=-0.925
        b0(8)=1.76106
        b1(8)=-3.32102
        b2(8)=1.56193
! 9 Lossy Bosnian Soil at 100 ps; f=1.5GHz
        epsr(9)=7.0
        a1(9)=-0.925
        b0(9)=1.76106
        b1(9)=-3.32102
        b2(9)=1.56193
! 10 Lossy Bosnian Soil at 2 ps; f=1.5GHz
        epsr(10)=2.08814
        a1(10)=-0.9555
        b0(10)=12.9552
        b1(10)=-25.0192
        b2(10)=12.0648
! 11 Water at 50 ps; f=100MHz
        epsr(11)=75.3619813
        a1(11)=-0.9685
        b0(11)=0.358437
        b1(11)=-0.690476
        b2(11)=0.332181
! 12 Bosnian soil at 50 ps; Moisture content=2.5%; f=100MHz
        epsr(12)=7.69
        a1(12)=-0.9085
        b0(12)=-0.2375335
        b1(12)=0.46763
        b2(12)=-0.229884
! 13 Bosnian soil at 50 ps; Moisture content=5%; f=100MHz
        epsr(13)=8.462
        a1(13)=-0.8685
        b0(13)=-0.408516
        b1(13)=0.789749
        b2(13)=-0.380641
! 14 Bosnian soil at 50 ps; Moisture content=10%; f=100MHz
        epsr(14)=9.40
        a1(14)=-0.8585
        b0(14)=-0.611398
        b1(14)=1.17886
        b2(14)=-0.566334
! 15 Bosnian soil at 50 ps; Moisture content=20%; f=100MHz
        epsr(15)=6.912
        a1(15)=-0.8685
        b0(15)=-0.242174
        b1(15)=0.554974
        b2(15)=-0.308386
! 16 Sandy soil at  2 ps; Moisture content=4%; f=1.3GHz
        epsr(16)=4.9508098
        a1(16)=-0.8285
        b0(16)=-6.4532
        b1(16)=12.5242
        b2(16)=-6.07063
! 17 Sandy Soil at 2 ps; Moisture content=17%; f=1.3GHz
        epsr(17)=20.9
```

```
          a1(17)=-0.8985
          b0(17)=-34.3627
          b1(17)=68.7577
          b2(17)=-34.3945
! 18 Sandy Soil at 20 ps; Moisture content=4%; f=1.3GHz
          epsr(18)=3.64315259
          a1(18)=-0.4785
          b0(18)=0.323846
          b1(18)=-0.458101
          b2(18)=0.135271
! 19 Sandy Soil at 20 ps; Moisture content=17%; f=1.3GHz
          epsr(19)=13.60445
          a1(19)=-0.8585
          b0(19)=2.88719
          b1(19)=-5.1354
          b2(19)=2.24691
! 20 Sandy Soil at 6 ps; Moisture content=17%; f=1.3GHz
          epsr(20)=3.69898497
          a1(20)=-0.8785
          b0(20)=21.7478
          b1(20)=-40.2878
          b2(20)=18.5404
! 21 Teflon: Substrate for Spiral Antenna
          epsr(21)=2.5  !nwsltp

c *** CALCULATIONS FOR ALL MATERIALS
 ! namt: # available materials = (# built-in) + (# user-defined)
          do n=1,namt  ! For each available material
                    eps(n)=epsr(n)*eps0  ! Permittivity
          enddo

c *** Maxwell Equation Coefficients for H@SFLD
          dtxmu=dt/(xmu0*delx)
          dtymu=dt/(xmu0*dely)
          dtzmu=dt/(xmu0*delz)

c *** Maxwell Equation Coefficients for E@SFLD
          do n=1,namt  ! For each available material
                    dtxeps(n)=dt/(eps(n)*delx)  ! dt, delx, eps coefficient
                    dtyeps(n)=dt/(eps(n)*dely)
                    dtzeps(n)=dt/(eps(n)*delz)
                    dsp0(n)=1+(b0(n)*dt)/(2*eps(n))  ! Dispersion coefficients
                    dsp1(n)=(1-a1(n))-(b0(n)+b1(n))*dt/(2*eps(n))
                    dsp2(n)=a1(n)-(b1(n)+b2(n))*dt/(2*eps(n))
                    dsp3(n)=-b2(n)*dt/(2*eps(n))
          enddo

c *** 1st order orbc (abc) constants
! Dispersive
          xx1X=1.0/delx
          xx1Y=1.0/dely
          xx1Z=1.0/delz

c *** 2nd order orbc (abc) constants
! Dispersive
          do n=1,namt
```

```
                xx2(n)=-sqrt(epsr(n))/(c0*dt)
                xx3(n)=-eta0/(2.0*sqrt(epsr(n)))
                uu1X(n)=sqrt(epsr(n))/(c0*dt*delx)
                uu1Y(n)=sqrt(epsr(n))/(c0*dt*dely)
                uu1Z(n)=sqrt(epsr(n))/(c0*dt*delz)
                uu2(n)=-epsr(n)/(c0*c0*dt*dt)
        enddo
        uu3=-xmu0/(2.0*dt)

        RETURN
        END


c***********************************************************
c***********************************************************
        SUBROUTINE FOBJ_MT_INFILE
        INCLUDE "nufdtd_params.f"

! Reads a material input file into the material distribution
! as a foreground object (2D or 3D)

        print*,'Enter material input filename'  !  file name
        print*,' Include extension; Need # chars <=20'
                read *,matname
        open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
         write(1,"(a,a,i3)")matname,'  ! Material input filename'  ! input.txt
        close(unit=1)  ! input.txt
        if(ndim<3)then  ! 2D
                norient=2  ! infile must be oriented in the y-direction (x-z plane)
                nheight=1  ! infile must be at y=1
        else  ! 3D
                print*,'Enter orientation (1:x; 2:y; 3:z)'      !  orientation
                        read *,norient
        open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
         write(1,"(i4,a,i3)")norient,'  ! Orientation'  ! input.txt
        close(unit=1)  ! input.txt
                if(norient==1)then  ! Material input file oriented in y-z direction
                        print*,'Enter height (nheight)'
                                read *,nheight
                        do while(nheight<1.or.nheight>nx)
                                print*,'nheight out of range'
                                print*,'Enter height (nheight)'
                                        read *,nheight
                        enddo
                elseif(norient==2)then  ! Material input file oriented in x-z direction
                        print*,'Enter height (nheight)'
                                read *,nheight
                        do while(nheight<1.or.nheight>ny)
                                print*,'nheight out of range'
                                print*,'Enter height (nheight)'
                                        read *,nheight
                        enddo
                elseif(norient==3)then  ! Material input file oriented in x-y direction
                        print*,'Enter height (nheight)'
                                read *,nheight
                        do while(nheight<1.or.nheight>nz)
                                print*,'nheight out of range'
                                print*,'Enter height (nheight)'
                                        read *,nheight
                        enddo
                endif
        open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
         write(1,"(i4,a,i3)")nheight,'  ! Height'  ! input.txt
        close(unit=1)  ! input.txt
        endif
        if(norient==1)then  ! Material input file oriented in y-z direction
                istart=nheight
                print*,'Enter starting point (y,z)'    ! Material input file starting coordinate
                        read*,jstart,kstart
                do while(jstart<1.or.jstart>ny)
                        print*,'jstart out of range'
                        print*,'Enter starting point (y,z)'    ! Material input file starting coordinate
                                read*,jstart,kstart
                enddo
                do while(kstart<1.or.kstart>ny)
                        print*,'kstart out of range'
                        print*,'Enter starting point (y,z)'    ! Material input file starting coordinate
                                read*,jstart,kstart
                enddo
        open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
         write(1,"(i4,i4,a)")jstart,kstart,'  ! Starting point (y,z)'  ! input.txt
        close(unit=1)  ! input.txt
                print*,'Enter width (y,z)'  ! Material input file width
                        read*,jwidth,kwidth
                do while(jwidth<1.or.jstart+jwidth-1>ny)
                        print*,'jwidth out of range'
                        print*,'Enter width (y,z)'  ! Material input file width
                                read*,jwidth,kwidth
                enddo
                do while(kwidth<1.or.kstart+kwidth-1>nz)
                        print*,'kwidth out of range'
                        print*,'Enter width (y,z)'  ! Material input file width
                                read*,jwidth,kwidth
                enddo
        open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
         write(1,"(i5,i5,a)")jwidth,kwidth,'  ! Width (y,z)'  ! input.txt
        close(unit=1)  ! input.txt
                imax=nheight
                jmax=jstart+jwidth-1
                kmax=kstart+kwidth-1
                open(unit=52,file=matname,status='unknown')
                        do kk=kstart,kmax
                         read(52,*)(mtx(nheight,jj,kmax-kk+1),jj=jstart,jmax)
                        enddo
                close(unit=52)
                open(unit=53,file=matname,status='unknown')
                        do kk=kstart,kmax
                         read(53,*)(mty(nheight,jj,kmax-kk+1),jj=jstart,jmax)
                        enddo
                close(unit=53)
                open(unit=54,file=matname,status='unknown')
                        do kk=kstart,kmax
```

```
                 read(54,*)(mtz(nheight,jj,kmax-kk+1),jj=jstart,jmax)
                enddo
        close(unit=54)
elseif(norient==2)then  ! Material input file oriented in x-z direction
        jstart=nheight
        print*,'Enter starting point (x,z)'    ! Material input file starting coordinate
                read*,istart,kstart
        do while(istart<1.or.istart>nx)
                print*,'istart out of range'
                print*,'Enter starting point (x,z)'    ! Material input file starting coordinate
                        read*,istart,kstart
        enddo
        do while(kstart<1.or.kstart>ny)
                print*,'kstart out of range'
                print*,'Enter starting point (x,z)'    ! Material input file starting coordinate
                        read*,istart,kstart
        enddo
open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
 write(1,"(i5,i5,a)")istart,kstart,'  ! Starting point (x,z)'  ! input.txt
close(unit=1)  ! input.txt
        print*,'Enter width (x,z)'  ! Material input file width
                read*,iwidth,kwidth
        do while(iwidth<1.or.istart+iwidth-1>nx)
                print*,'iwidth out of range'
                print*,'Enter width (x,z)'  ! Material input file width
                        read*,iwidth,kwidth
        enddo
        do while(kwidth<1.or.kstart+kwidth-1>nz)
                print*,'kwidth out of range'
                print*,'Enter width (x,z)'  ! Material input file width
                        read*,iwidth,kwidth
        enddo
open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
 write(1,"(i5,i5,a)")iwidth,kwidth,'  ! Width (x,z)'  ! input.txt
close(unit=1)  ! input.txt
        imax=istart+iwidth-1
        jmax=nheight
        kmax=kstart+kwidth-1
        open(unit=52,file=matname,status='unknown')
                do kk=kstart,kmax
                 read(52,*)(mtx(ii,nheight,kmax-kk+1),ii=istart,imax)
                enddo
        close(unit=52)
        open(unit=53,file=matname,status='unknown')
                do kk=kstart,kmax
                 read(53,*)(mty(ii,nheight,kmax-kk+1),ii=istart,imax)
                enddo
        close(unit=53)
        open(unit=54,file=matname,status='unknown')
                do kk=kstart,kmax
                 read(54,*)(mtz(ii,nheight,kmax-kk+1),ii=istart,imax)
                enddo
        close(unit=54)
elseif(norient==3)then  ! Material input file oriented in x-y direction
        kstart=nheight
        print*,'Enter starting point (x,y)'    ! Material input file starting coordinate
```

```
                  read*,istart,jstart
            do while(istart<1.or.istart>nx)
                  print*,'istart=',istart
                  print*,'istart out of range'
                  print*,'Enter starting point (x,y)'    ! Material input file starting coordinate
                        read*,istart,jstart
            enddo
            do while(jstart<1.or.jstart>ny)
                  print*,'jstart out of range'
                  print*,'Enter starting point (x,y)'    ! Material input file starting coordinate
                        read*,istart,jstart
            enddo
      open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
       write(1,"(i5,i5,a)")istart,jstart,'  ! Starting point (x,y)'  ! input.txt
      close(unit=1)  ! input.txt
            print*,'Enter width (y,z)'  ! Material input file width
                  read*,iwidth,jwidth
            do while(iwidth<1.or.istart+iwidth-1>nx)
                  print*,'iwidth out of range'
                  print*,'Enter width (y,z)'  ! Material input file width
                        read*,iwidth,jwidth
            enddo
            do while(jwidth<1.or.jstart+jwidth-1>ny)
                  print*,'jwidth out of range'
                  print*,'Enter width (y,z)'  ! Material input file width
                        read*,iwidth,jwidth
            enddo
      open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
       write(1,"(i5,i5,a)")iwidth,jwidth,'  ! Width (x,y)'  ! input.txt
      close(unit=1)  ! input.txt
            imax=istart+iwidth-1
            jmax=jstart+jwidth-1
            kmax=nheight
            open(unit=52,file=matname,status='unknown')
                  do jj=jstart,jmax
                   read(52,*)(mtx(ii,jmax-jj+1,nheight),ii=istart,imax)
                  enddo
            close(unit=52)
            open(unit=53,file=matname,status='unknown')
                  do jj=jstart,jmax
                   read(53,*)(mty(ii,jmax-jj+1,nheight),ii=istart,imax)
                  enddo
            close(unit=53)
            open(unit=54,file=matname,status='unknown')
                  do jj=jstart,jmax
                   read(54,*)(mtz(ii,jmax-jj+1,nheight),ii=istart,imax)
                  enddo
            close(unit=54)
      endif

      RETURN
      END


c************************************************************
c************************************************************
```

```
          SUBROUTINE FOBJ_RECT_PRISM
          INCLUDE "nufdtd_params.f"

! Builds a rectangular prism (3D only) or a rectangle (2D only)
! as a foreground object

          if(ndim<3)then  ! 2D
                    jstart=1
                    print*,'Enter coordinates (i,k) of bottom corner'
                              read*,istart,kstart
                    do while(istart<1.or.istart>nx)
                              print*,'istart out of range'
                              print*,'Enter coordinates (i,k) of bottom corner'
                                        read*,istart,kstart
                    enddo
                    do while(kstart<1.or.kstart>nz)
                              print*,'kstart out of range'
                              print*,'Enter coordinates (i,k) of bottom corner'
                                        read*,istart,kstart
                    enddo
          open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
           write(1,"(i5,i5,a)")istart,kstart,
    $  '  ! Rectangle starting coordinates (bottom corner)'  ! input.txt
          close(unit=1)  ! input.txt
                    jwidth=1
                    print*,'Enter width in each direction'
                              read*,iwidth,kwidth
                    do while(istart+iwidth>nx)
                              print*,'iwidth too large'
                              print*,'Enter width in each direction'
                                        read*,iwidth,kwidth
                    enddo
                    do while(kstart+kwidth>nz)
                              print*,'kwidth too large'
                              print*,'Enter width in each direction'
                                        read*,iwidth,kwidth
                    enddo
          open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
           write(1,"(i5,i5,a)")iwidth,kwidth,
    $  '  ! Rectangle width in each direction'  ! input.txt
          close(unit=1)  ! input.txt
          else  ! 3D
                    print*,'Enter coordinates (i,j,k) of bottom corner'
                              read*,istart,jstart,kstart
                    do while(istart<1.or.istart>nx)
                              print*,'istart out of range'
                              print*,'Enter coordinates (i,j,k) of bottom corner'
                                        read*,istart,jstart,kstart
                    enddo
                    do while(jstart<1.or.jstart>ny)
                              print*,'jstart out of range'
                              print*,'Enter coordinates (i,j,k) of bottom corner'
                                        read*,istart,jstart,kstart
                    enddo
                    do while(kstart<1.or.kstart>nz)
                              print*,'kstart out of range'
```

```
              print*,'Enter coordinates (i,j,k) of bottom corner'
                  read*,istart,jstart,kstart
          enddo
      open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
       write(1,"(i5,i5,i5,a)")istart,jstart,kstart,
     $  '  ! Rectangular prism starting coordinates (bottom corner)'  ! input.txt
      close(unit=1)  ! input.txt
          print*,'Enter width in each direction'
              read*,iwidth,jwidth,kwidth
          do while(istart+iwidth>nx)
              print*,'iwidth too large'
              print*,'Enter width in each direction'
                  read*,iwidth,jwidth,kwidth
          enddo
          do while(jstart+jwidth>ny)
              print*,'jwidth too large'
              print*,'Enter width in each direction'
                  read*,iwidth,jwidth,kwidth
          enddo
          do while(kstart+kwidth>nz)
              print*,'kwidth too large'
              print*,'Enter width in each direction'
                  read*,iwidth,jwidth,kwidth
          enddo
      open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
       write(1,"(i5,i5,i5,a)")iwidth,jwidth,kwidth,
     $  '  ! Rectangular prism width in each direction'  ! input.txt
      close(unit=1)  ! input.txt
      endif
      print*,'Enter material type'
      print*,' Type 0 to view options.'
          read*,mtnum
          if(mtnum==0)CALL MATPRINT  ! Display material options
      do while(mtnum<1.or.mtnum>namt)
          print*,'Need entry to be between 1 and ',namt
          print*,'Enter material type'
              read*,mtnum
      enddo
      open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
       write(1,"(i3,a,i3)")mtnum,'  ! Material type'  ! input.txt
      close(unit=1)  ! input.txt
      CALL SHP_RP(istart,jstart,kstart,iwidth,jwidth,kwidth,mtnum)

      RETURN
      END


c************************************************************
c************************************************************
      SUBROUTINE FOBJ_CIRCLE
      INCLUDE "nufdtd_params.f"

! Builds a circle as a foreground object (2D only)

      print*,'Enter coordinates (i,k) of center'
          read*,ictr,kctr
```

```
        do while(ictr<1.or.ictr>nx)
                print*,'x-direction coordinate out of range'
                print*,'Enter coordinates (i,k) of center'
                        read*,ictr,kctr
        enddo
        do while(kctr<1.or.kctr>nz)
                print*,'z-direction coordinate out of range'
                print*,'Enter coordinates (i,k) of center'
                        read*,ictr,kctr
        enddo
        open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
         write(1,"(i5,i5,i5,a)")ictr,kctr,
    $  '  ! Circle center coordinates'  ! input.txt
        close(unit=1)  ! input.txt
        print*,'Enter radius'
                read*,irad
        do while(ictr-irad<1.or.ictr+irad>nx)
                print*,'radius too large'
                print*,'Enter radius'
                        read*,irad
        enddo
        do while(kctr-irad<1.or.kctr+irad>nz)
                print*,'radius too large'
                print*,'Enter radius'
                        read*,irad
        enddo
        open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
         write(1,"(i5,i5,a)")irad,
    $  '  ! Circle radius'  ! input.txt
        close(unit=1)  ! input.txt
        print*,'Enter material type'
        print*,' Type 0 to view options.'
                read*,mtnum
                if(mtnum==0)CALL MATPRINT  ! Display material options
        do while(mtnum<1.or.mtnum>namt)
                print*,'Need entry to be between 1 and ',namt
                print*,'Enter material type'
                        read*,mtnum
        enddo
        open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
         write(1,"(i3,a,i3)")mtnum,'  ! Material type'  ! input.txt
        close(unit=1)  ! input.txt
        CALL SHP_CIRCLE(ictr,kctr,irad,mtnum)

        RETURN
        END


c**************************************************************
c**************************************************************
        SUBROUTINE FOBJ_CYLINDER
        INCLUDE "nufdtd_params.f"

! Builds a cylinder as a foreground object (3D only)

        print*,'Enter coordinates (i,j,k) of bottom-center'
```

```
              read*,ictr,jctr,kbtm
      do while(ictr<1.or.ictr>nx)
              print*,'x-direction coordinate out of range'
              print*,'Enter coordinates (i,j,k) of bottom-center'
                      read*,ictr,jctr,kbtm
      enddo
      do while(jctr<1.or.jctr>ny)
              print*,'y-direction coordinate out of range'
              print*,'Enter coordinates (i,j,k) of bottom-center'
                      read*,ictr,jctr,kbtm
      enddo
      do while(kbtm<1.or.kbtm>nz)
              print*,'z-direction coordinate out of range'
              print*,'Enter coordinates (i,j,k) of bottom-center'
                      read*,ictr,jctr,kbtm
      enddo
      open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
       write(1,"(i5,i5,i5,a)")ictr,jctr,kbtm,
$  '  ! Cylinder bottom-center coordinates'  ! input.txt
      close(unit=1)  ! input.txt
      print*,'Enter radius & height'
              read*,irad,kheight
      do while(ictr-irad<1.or.ictr+irad>nx)
              print*,'radius too large'
              print*,'Enter radius & height'
                      read*,irad,kheight
      enddo
      do while(jctr-irad<1.or.jctr+irad>ny)
              print*,'radius too large'
              print*,'Enter radius & height'
                      read*,irad,kheight
      enddo
      do while(kbtm+kheight>nz)
              print*,'height too large'
              print*,'Enter radius & height'
                      read*,irad,kheight
      enddo
      open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
       write(1,"(i5,i5,a)")irad,kheight,
$  '  ! Cylinder radius & height'  ! input.txt
      close(unit=1)  ! input.txt
      print*,'Enter material type'
      print*,' Type 0 to view options.'
              read*,mtnum
              if(mtnum==0)CALL MATPRINT  ! Display material options
      do while(mtnum<1.or.mtnum>namt)
              print*,'Need entry to be between 1 and ',namt
              print*,'Enter material type'
                      read*,mtnum
      enddo
      open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
       write(1,"(i3,a,i3)")mtnum,'  ! Material type'  ! input.txt
      close(unit=1)  ! input.txt
      CALL SHP_CYL(ictr,jctr,kbtm,irad,kheight,mtnum)

      RETURN
```

```
        END


c********************************************************
c********************************************************
        SUBROUTINE FOBJ_SPHERE
        INCLUDE "nufdtd_params.f"

! Builds a sphere         as a foreground object (3D only)

        print*,'Enter coordinates (i,j,k) of center'
                read*,ictr,jctr,kctr
        do while(ictr<1.or.ictr>nx)
                print*,'x-direction coordinate out of range'
                print*,'Enter coordinates (i,j,k) of center'
                        read*,ictr,jctr,kctr
        enddo
        do while(jctr<1.or.jctr>ny)
                print*,'y-direction coordinate out of range'
                print*,'Enter coordinates (i,j,k) of center'
                        read*,ictr,jctr,kctr
        enddo
        do while(kctr<1.or.kctr>nz)
                print*,'z-direction coordinate out of range'
                print*,'Enter coordinates (i,j,k) of center'
                        read*,ictr,jctr,kctr
        enddo
        open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
         write(1,"(i5,i5,i5,a)")ictr,jctr,kctr,
   $  '  ! Sphere center coordinates'  ! input.txt
        close(unit=1)  ! input.txt
        print*,'Enter radius'
                read*,irad
        do while(ictr-irad<1.or.ictr+irad>nx)
                print*,'radius too large'
                print*,'Enter radius'
                        read*,irad
        enddo
        do while(jctr-irad<1.or.jctr+irad>ny)
                print*,'radius too large'
                print*,'Enter radius'
                        read*,irad
        enddo
        do while(kctr-irad<1.or.kctr+irad>nz)
                print*,'radius too large'
                print*,'Enter radius'
                        read*,irad
        enddo
        open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
         write(1,"(i5,a)")irad,
   $  '  ! Sphere radius'  ! input.txt
        close(unit=1)  ! input.txt
        print*,'Enter material type'
        print*,' Type 0 to view options.'
                read*,mtnum
        if(mtnum==0)CALL MATPRINT  ! Display material options
        do while(mtnum<1.or.mtnum>namt)
                print*,'Need entry to be between 1 and ',namt
                print*,'Enter material type'
                        read*,mtnum
        enddo
        open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
         write(1,"(i3,a,i3)")mtnum,'  ! Material type'  ! input.txt
        close(unit=1)  ! input.txt
        CALL SHP_SPHERE(ictr,jctr,kctr,irad,mtnum)

        RETURN
        END


c***********************************************************
c***********************************************************
        SUBROUTINE FOBJ_MONOPOLE
        INCLUDE "nufdtd_params.f"

! Builds a monopole antenna as a foreground object (3D only)
! Antenna exposed dielectric points towards z=1
! Uses concentric cylinders: core, dielectric, shield = metal, dielectric, metal
! Also makes the top layer metal so no fields escape through the top
! If there is a monopole antenna excitation, then the last
!  antenna built will be the antenna used for the excitation

        print*,'Give (i,j,k) of antenna top-center in grid points'
        print*,' k should be coordinate of top metal layer.'
                read*,iantctr,jantctr,kanttop
        open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
         write(1,"(i5,i5,i5,a)")iantctr,jantctr,kanttop,
   $  '  ! (i,j,k) of antenna top-center'  ! input.txt
        close(unit=1)  ! input.txt
        print*,'Give core radius, dielectric thickness ',
   $            '& shield thickness in grid points'
                read*,icorerad,idieth,ishieldth
        open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
         write(1,"(i5,i5,i5,a)")icorerad,idieth,ishieldth,
   $'  ! Core radius, dielectric thickness & shield thickness'  ! input.txt
        close(unit=1)  ! input.txt
        print*,'Dielectric/core length & sheild length in grid points'
        print*,' (Not counting top metal layer.)'
                read*,kcorelen,kshieldlen
        open(unit=1,file='input.txt',access='append',status='old')  ! input.txt
         write(1,"(i5,i5,a)")kcorelen,kshieldlen,
   $  '  ! Dielectric/core length, sheild length'  ! input.txt
        close(unit=1)  ! input.txt

        idierad=icorerad+idieth  ! dielectric radius = core radius + dielectric thickness
        ishieldrad=idierad+ishieldth  ! shield radius = dielectric radius + shield thickness
        ! Note ishieldrad is used for both the x (i) & y (j) directions
        kcorebtm=kanttop-kcorelen  ! core bottom = core top - core length
        kshieldbtm=kanttop-kshieldlen  ! shield bottom = core top - shield length

        CALL SHP_CYL(iantctr,jantctr,kshieldbtm,ishieldrad,kshieldlen+1,2)  ! Shield & top metal layer
        CALL SHP_CYL(iantctr,jantctr,kcorebtm,idierad,kcorelen,5)          ! Dielectric
```

```
      CALL SHP_CYL(iantctr,jantctr,kcorebtm,icorerad,kcorelen,2)  ! Core

      RETURN
      END


c************************************************************
c************************************************************
      SUBROUTINE SHP_RP(ibtm,jbtm,kbtm,iwidth,jwidth,kwidth,mtnum)
      INCLUDE "nufdtd_params.f"

! Builds a rectangular prism shape.
! Note CALL SHP_RP(i,j,k,1,1,1,m) is an easy way
! to assign material m at point (i,j,k)

! ibtm,jbtm,kbtm: cube bottom corner coordinate
! iwidth,jwidth,kwidth: cube width
! mtnum: material id

      ! Error catches:
      if(ibtm<1.or.ibtm>nx)print*,
     $      'bottom coordinate in x-direction out of range'
      if(jbtm<1.or.jbtm>ny)print*,
     $      'bottom coordinate in y-direction out of range'
      if(kbtm<1.or.kbtm>nz)print*,
     $      'bottom coordinate in z-direction out of range'
      if(ibtm+iwidth-1>nx)print*,'width in x-direction out of range'
      if(jbtm+jwidth-1>ny)print*,'width in y-direction out of range'
      if(kbtm+kwidth-1>nz)print*,'width in z-direction out of range'
      if(mtnum<1.or.mtnum>namt)print*,'material type out of range'

      itop=ibtm+iwidth-1  ! cube top corner coordinate
      jtop=jbtm+jwidth-1
      ktop=kbtm+kwidth-1

      do k=kbtm,ktop
       do j=jbtm,jtop
        do i=ibtm,itop
            mtx(i,j,k)=mtnum
            mty(i,j,k)=mtnum
            mtz(i,j,k)=mtnum
        enddo
       enddo
      enddo

      RETURN
      END


c************************************************************
c************************************************************
      SUBROUTINE SHP_CYL(ictr,jctr,kbtm,irad,kheight,mtnum)
      INCLUDE "nufdtd_params.f"

! Builds a cylinder shape.
! Cylinder fits within the rectangular prism defined by the
! starting and ending coordinates.  These coordinates are used
! to avoid scanning through the entire computational domain.

! ictr,jctr,kbtm: cylinder bottom-center coordinate
! irad,kheight: cylinder radius and height
! mtnum: material id

      ! Error catches:
      if(ictr<1.or.ictr>nx)print*,
     $      'center coordinate in x-direction out of range'
      if(jctr<1.or.jctr>ny)print*,
     $      'center coordinate in y-direction out of range'
      if(kbtm<1.or.kbtm>nz)print*,
     $      'bottom coordinate in z-direction out of range'
      if(ictr-irad<1.or.ictr+irad>nx)print*,'radius too large'
      if(jctr-irad<1.or.jctr+irad>ny)print*,'radius too large'
      if(kheight<1.or.kbtm+kheight>nz)print*,
     $      'height in z-direction out of range'
      if(mtnum<1.or.mtnum>namt)print*,'material type out of range'

      istart=ictr-irad+1  !Starting coordinates
      jstart=jctr-irad+1
      kstart=kbtm

      imax=ictr+irad-1  !Endinging coordinates
      jmax=jctr+irad-1
      kmax=kbtm+kheight-1

      do k=kstart,kmax
       do j=jstart,jmax
        do i=istart,imax
              if(sqrt(real((i-ictr)**2+(j-jctr)**2))<irad)then
                      mtx(i,j,k)=mtnum
                      mty(i,j,k)=mtnum
                      mtz(i,j,k)=mtnum
              endif
        enddo
       enddo
      enddo

      RETURN
      END


c************************************************************
c************************************************************
      SUBROUTINE SHP_CIRCLE(ictr,kctr,irad,mtnum)
      INCLUDE "nufdtd_params.f"

! Builds a circle shape in the x-z plane.

! ictr,kctr: circle center coordinate
! irad: circle radius
! mtnum: material id
```

```
      ! Error catches:
      if(ictr<1.or.ictr>nx)print*,
     $ 'center coordinate in x-direction out of range'
      if(kctr<1.or.kctr>nz)print*,
     $ 'center coordinate in z-direction out of range'
      if(ictr-irad<1.or.ictr+irad>nx)print*,'radius too large'
      if(kctr-irad<1.or.kctr+irad>nz)print*,'radius too large'
      if(mtnum<1.or.mtnum>namt)print*,'material type out of range'

      istart=ictr-irad+1  !Starting coordinates
      kstart=kctr-irad+1

      imax=ictr+irad-1  !Ending coordinates
      kmax=kctr+irad-1

      do k=kstart,kmax
        do i=istart,imax
              if(sqrt(real((i-ictr)**2+(k-kctr)**2))<irad)then
                    mtx(i,1,k)=mtnum
                    mty(i,1,k)=mtnum
                    mtz(i,1,k)=mtnum
              endif
        enddo
      enddo

      RETURN
      END


c************************************************************
c************************************************************
      SUBROUTINE SHP_SPHERE(ictr,jctr,kctr,irad,mtnum)
      INCLUDE "nufdtd_params.f"

! Builds a sphere shape.

! Sphere fits within the rectangular prism defined by the
! starting and ending coordinates.  These coordinates are used
! to avoid scanning through the entire computational domain.

! ictr,jctr,kctr: sphere center coordinate
! irad: sphere radius
! mtnum: material id

      ! Error catches:
      if(ictr<1.or.ictr>nx)print*,
     $ 'center coordinate in x-direction out of range'
      if(jctr<1.or.jctr>ny)print*,
     $ 'center coordinate in y-direction out of range'
      if(kctr<1.or.kctr>nz)print*,
     $ 'center coordinate in z-direction out of range'
      if(ictr-irad<1.or.ictr+irad>nx)print*,'radius too large'
      if(jctr-irad<1.or.jctr+irad>ny)print*,'radius too large'
      if(kctr-irad<1.or.kctr+irad>nz)print*,'radius too large'
      if(mtnum<1.or.mtnum>namt)print*,'material type out of range'
```

```
        istart=ictr-irad+1  !Starting coordinates
        jstart=jctr-irad+1
        kstart=kctr-irad+1

        imax=ictr+irad-1  !Ending coordinates
        jmax=jctr+irad-1
        kmax=kctr+irad-1

        do k=kstart,kmax
         do j=jstart,jmax
          do i=istart,imax
                if(sqrt(real((i-ictr)**2+(j-jctr)**2+(k-kctr)**2))<irad)then
                        mtx(i,j,k)=mtnum
                        mty(i,j,k)=mtnum
                        mtz(i,j,k)=mtnum
                endif
          enddo
         enddo
        enddo

        RETURN
        END


c*********************************************************
c*********************************************************
        SUBROUTINE HXSFLD
        INCLUDE "nufdtd_params.f"

C Updates the HX scattered field.

        if(ndim<3)then  ! 2D
                ny2=1
        else  ! 3D
                ny2=ny1
        endif

C Save past values
        do k=1,nz1
         do j=1,ny2
          do i=1,nx
                hxs1(i,j,k)=hxs(i,j,k)
          enddo
         enddo
        enddo
        do k=1,nz1
         do j=1,ny2
          do i=1,nx
                hxs(i,j,k)=hxs(i,j,k)-(ezs(i,j+1,k)-ezs(i,j,k))*dtymu
   $                    +(eys(i,j,k+1)-eys(i,j,k))*dtzmu
          enddo
         enddo
        enddo

        RETURN
        END
```

```
c************************************************************
c************************************************************
      SUBROUTINE HYSFLD
      INCLUDE "nufdtd_params.f"

C Updates the HY scattered field.

C Save past values
      do k=1,nz1
       do j=1,ny
        do i=1,nx1
            hys1(i,j,k)=hys(i,j,k)
        enddo
       enddo
      enddo
      do k=1,nz1
       do j=1,ny
        do i=1,nx1
            hys(i,j,k)=hys(i,j,k)-(exs(i,j,k+1)-exs(i,j,k))*dtzmu
  $                                 +(ezs(i+1,j,k)-ezs(i,j,k))*dtxmu
        enddo
       enddo
      enddo

      RETURN
      END


c************************************************************
c************************************************************
      SUBROUTINE HZSFLD
      INCLUDE "nufdtd_params.f"

C Updates the HZ scattered field.

      if(ndim<3)then  ! 2D
            ny2=1
      else  ! 3D
            ny2=ny1
      endif

C Save past values
      do k=1,nz
       do j=1,ny2
        do i=1,nx1
            hzs1(i,j,k)=hzs(i,j,k)
        enddo
       enddo
      enddo
      do k=1,nz
       do j=1,ny2
        do i=1,nx1
            hzs(i,j,k)=hzs(i,j,k)-(eys(i+1,j,k)-eys(i,j,k))*dtxmu
  $                                 +(exs(i,j+1,k)-exs(i,j,k))*dtymu
```

```
        enddo
       enddo
      enddo

      RETURN
      END


c**********************************************************
c**********************************************************
      SUBROUTINE EXSFLD
      INCLUDE "nufdtd_params.f"

C Updates the EX scattered field.

      if(ndim<3)then  ! 2D
            ny0=1
            ny2=1
      else  ! 3D
            ny0=2
            ny2=ny1
      endif

C Save past values
      do k=2,nz1
       do j=ny0,ny2
        do i=1,nx1
            exs3(i,j,k)=exs2(i,j,k)
            exs2(i,j,k)=exs1(i,j,k)
            exs1(i,j,k)=exs(i,j,k)
        enddo
       enddo
      enddo

      do k=2,nz1
       do j=ny0,ny2
        do i=1,nx1
C Determine material type
! Note exs(i,j,k) references material (i,j,k)
            if(mtx(i,j,k)==2)then  ! (PEC)
                  exs(i,j,k)=0
            else
                  exs(i,j,k)=(1/dsp0(mtx(i,j,k)))*(
  $          dsp1(mtx(i,j,k))*exs(i,j,k)
  $         +dsp2(mtx(i,j,k))*exs2(i,j,k)
  $         +dsp3(mtx(i,j,k))*exs3(i,j,k)
  $         +dtyeps(mtx(i,j,k))*(hzs(i,j,k)-hzs(i,j-1,k))
  $         -dtzeps(mtx(i,j,k))*(hys(i,j,k)-hys(i,j,k-1))
  $         +dtyeps(mtx(i,j,k))*(hzs1(i,j,k)-hzs1(i,j-1,k))
  $           *a1(mtx(i,j,k))
  $         -dtzeps(mtx(i,j,k))*(hys1(i,j,k)-hys1(i,j,k-1))
  $           *a1(mtx(i,j,k)))
            endif
        enddo
       enddo
      enddo
      RETURN
      END


c**********************************************************
c**********************************************************
      SUBROUTINE EYSFLD
      INCLUDE "nufdtd_params.f"

C Updates the EY scattered field.

      if(ndim<3)then  ! 2D
            ny2=1
      else  ! 3D
            ny2=ny1
      endif

C Save past values
      do k=2,nz1
       do j=1,ny2
        do i=2,nx1
            eys3(i,j,k)=eys2(i,j,k)
            eys2(i,j,k)=eys1(i,j,k)
            eys1(i,j,k)=eys(i,j,k)
        enddo
       enddo
      enddo

      do k=2,nz1
       do j=1,ny2
        do i=2,nx1
C Determine material type
! Note eys(i,j,k) references material (i,j,k)
            if(mty(i,j,k)==2)then  ! (PEC)
                  eys(i,j,k)=0
            else
                  eys(i,j,k)=(1/dsp0(mty(i,j,k)))*(
     $             dsp1(mty(i,j,k))*eys(i,j,k)
     $            +dsp2(mty(i,j,k))*eys2(i,j,k)
     $            +dsp3(mty(i,j,k))*eys3(i,j,k)
     $            +dtzeps(mty(i,j,k))*(hxs(i,j,k)-hxs(i,j,k-1))
     $            -dtxeps(mty(i,j,k))*(hzs(i,j,k)-hzs(i-1,j,k))
     $            +dtzeps(mty(i,j,k))*(hxs1(i,j,k)-hxs1(i,j,k-1))
     $              *a1(mty(i,j,k))
     $            -dtxeps(mty(i,j,k))*(hzs1(i,j,k)-hzs1(i-1,j,k))
     $              *a1(mty(i,j,k)))
            endif
        enddo
       enddo
      enddo

      RETURN
      END
```

```
c*********************************************************
c*********************************************************
      SUBROUTINE EZSFLD
      INCLUDE "nufdtd_params.f"

C Updates the EZ scattered field.

      if(ndim<3)then  ! 2D
            ny0=1
            ny2=1
      else  ! 3D
            ny0=2
            ny2=ny1
      endif

C Save past values
      do k=1,nz1
       do j=ny0,ny2
        do i=2,nx1
            ezs3(i,j,k)=ezs2(i,j,k)
            ezs2(i,j,k)=ezs1(i,j,k)
            ezs1(i,j,k)=ezs(i,j,k)
        enddo
       enddo
      enddo

      do k=1,nz1
       do j=ny0,ny2
        do i=2,nx1
C Determine material type
! Note ezs(i,j,k) references material (i,j,k)
            if(mtz(i,j,k)==2)then  ! (PEC)
                  ezs(i,j,k)= 0
            else
                  ezs(i,j,k)=(1/dsp0(mtz(i,j,k)))*(
  $          dsp1(mtz(i,j,k))*ezs(i,j,k)
  $         +dsp2(mtz(i,j,k))*ezs2(i,j,k)
  $         +dsp3(mtz(i,j,k))*ezs3(i,j,k)
  $         +dtxeps(mtz(i,j,k))*(hys(i,j,k)-hys(i-1,j,k))
  $         -dtyeps(mtz(i,j,k))*(hxs(i,j,k)-hxs(i,j-1,k))
  $         +dtxeps(mtz(i,j,k))*(hys1(i,j,k)-hys1(i-1,j,k))
  $           *a1(mtz(i,j,k))
  $         -dtyeps(mtz(i,j,k))*(hxs1(i,j,k)-hxs1(i,j-1,k))
  $           *a1(mtz(i,j,k)))
            endif
        enddo
       enddo
      enddo

      RETURN
      END


c*********************************************************
c*********************************************************
      SUBROUTINE RADEXY
```

```
            INCLUDE "nufdtd_params.f"
! ABC for Ex at y=1, y=ny

            do k=1,nz  ! z-direction edges & corners
             do ii=0,1
              i=ii*(nx-2)+1  ! i={1,nx-1}
                         n=mtx(i,1,k)
                         exs(i,1,k)=(1.0/(xx1Y-xx2(n)))*(
     $         exs(i,2,k)*(xx1Y+xx2(n))
     $        +exsY1(i,2,k)*
     $          ((1.0+a1(n))*xx1Y-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
     $        +exsY1(i,1,k)*
     $          (-(1.0+a1(n))*xx1Y-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
     $        +exsY2(i,2,k)*(a1(n)*xx1Y-a1(n)*xx2(n)+b1(n)*xx3(n))
     $        +exsY2(i,1,k)*(-a1(n)*xx1Y-a1(n)*xx2(n)+b1(n)*xx3(n))
     $        +exsY3(i,2,k)*(b2(n)*xx3(n))
     $        +exsY3(i,1,k)*(b2(n)*xx3(n))
     $        )
                         n=mtx(i,ny,k)
                         exs(i,ny,k)=(1.0/(xx1Y-xx2(n)))*(
     $         exs(i,ny1,k)*(xx1Y+xx2(n))
     $        +exsY1(i,3,k)*
     $          ((1.0+a1(n))*xx1Y-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
     $        +exsY1(i,4,k)*
     $          (-(1.0+a1(n))*xx1Y-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
     $        +exsY2(i,3,k)*(a1(n)*xx1Y-a1(n)*xx2(n)+b1(n)*xx3(n))
     $        +exsY2(i,4,k)*(-a1(n)*xx1Y-a1(n)*xx2(n)+b1(n)*xx3(n))
     $        +exsY3(i,3,k)*(b2(n)*xx3(n))
     $        +exsY3(i,4,k)*(b2(n)*xx3(n))
     $        )
             enddo
            enddo

            do i=2,nx-2  ! x-direction edges
             do kk=0,1
              k=kk*(nz-1)+1   ! k={1,nz}
                         n=mtx(i,1,k)
                         exs(i,1,k)=(1.0/(xx1Y-xx2(n)))*(
     $         exs(i,2,k)*(xx1Y+xx2(n))
     $        +exsY1(i,2,k)*
     $          ((1.0+a1(n))*xx1Y-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
     $        +exsY1(i,1,k)*
     $          (-(1.0+a1(n))*xx1Y-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
     $        +exsY2(i,2,k)*(a1(n)*xx1Y-a1(n)*xx2(n)+b1(n)*xx3(n))
     $        +exsY2(i,1,k)*(-a1(n)*xx1Y-a1(n)*xx2(n)+b1(n)*xx3(n))
     $        +exsY3(i,2,k)*(b2(n)*xx3(n))
     $        +exsY3(i,1,k)*(b2(n)*xx3(n))
     $        )
                         n=mtx(i,ny,k)
                         exs(i,ny,k)=(1.0/(xx1Y-xx2(n)))*(
     $         exs(i,ny1,k)*(xx1Y+xx2(n))
     $        +exsY1(i,3,k)*
     $          ((1.0+a1(n))*xx1Y-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
     $        +exsY1(i,4,k)*
     $          (-(1.0+a1(n))*xx1Y-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
     $        +exsY2(i,3,k)*(a1(n)*xx1Y-a1(n)*xx2(n)+b1(n)*xx3(n))
$     +exsY2(i,4,k)*(-a1(n)*xx1Y-a1(n)*xx2(n)+b1(n)*xx3(n))
$     +exsY3(i,3,k)*(b2(n)*xx3(n))
$     +exsY3(i,4,k)*(b2(n)*xx3(n))
$     )
     enddo
    enddo

    do k=2,nz1  ! x-z faces
     do i=2,nx-2
           n=mtx(i,1,k)
           exs(i,1,k)=(1.0/(uu1Y(n)-uu2(n)-b0(n)*uu3))*(
$      exs(i,2,k)*(uu1Y(n)+uu2(n)+b0(n)*uu3)
$     +exsY1(i,2,k)*
$      (a1(n)*uu1Y(n)+(-2.0+a1(n))*uu2(n)+(-b0(n)+b1(n))*uu3)
$     +exsY1(i,1,k)*
$      (-a1(n)*uu1Y(n)+(-2.0+a1(n))*uu2(n)+(-b0(n)+b1(n))*uu3)
$     +exsY2(i,2,k)*
$      (-uu1Y(n)+(1-2*a1(n))*uu2(n)+(-b1(n)+b2(n))*uu3)
$     +exsY2(i,1,k)*
$      (uu1Y(n)+(1-2*a1(n))*uu2(n)+(-b1(n)+b2(n))*uu3)
$     +exsY3(i,2,k)*(-a1(n)*uu1Y(n)+a1(n)*uu2(n)-b2(n)*uu3)
$     +exsY3(i,1,k)*(a1(n)*uu1Y(n)+a1(n)*uu2(n)-b2(n)*uu3)
$     +(1.0/(2.0*delx**2))*( ! The d/dx**2 term:
$      +exsY1(i+1,1,k)-2.*exsY1(i,1,k)+exsY1(i-1,1,k)
$      +exsY1(i+1,2,k)-2.*exsY1(i,2,k)+exsY1(i-1,2,k)
$      +a1(n)*(
$      +exsY2(i+1,1,k)-2.*exsY2(i,1,k)+exsY2(i-1,1,k)
$      +exsY2(i+1,2,k)-2.*exsY2(i,2,k)+exsY2(i-1,2,k)
$       )
$      ) ! end of d/dx**2 term
$     +(1.0/(2.0*delz**2))*( ! The d/dz**2 term:
$      +exsY1(i,1,k+1)-2.*exsY1(i,1,k)+exsY1(i,1,k-1)
$      +exsY1(i,2,k+1)-2.*exsY1(i,2,k)+exsY1(i,2,k-1)
$      +a1(n)*(
$      +exsY2(i,1,k+1)-2.*exsY2(i,1,k)+exsY2(i,1,k-1)
$      +exsY2(i,2,k+1)-2.*exsY2(i,2,k)+exsY2(i,2,k-1)
$       )
$      ) ! end of d/dz**2 term
$     )
           n=mtx(i,ny,k)
           exs(i,ny,k)=(1.0/(uu1Y(n)-uu2(n)-b0(n)*uu3))*(
$      exs(i,ny1,k)*(uu1Y(n)+uu2(n)+b0(n)*uu3)
$     +exsY1(i,3,k)*
$      (a1(n)*uu1Y(n)+(-2.0+a1(n))*uu2(n)+(-b0(n)+b1(n))*uu3)
$     +exsY1(i,4,k)*
$      (-a1(n)*uu1Y(n)+(-2.0+a1(n))*uu2(n)+(-b0(n)+b1(n))*uu3)
$     +exsY2(i,3,k)*
$      (-uu1Y(n)+(1-2*a1(n))*uu2(n)+(-b1(n)+b2(n))*uu3)
$     +exsY2(i,4,k)*
$      (uu1Y(n)+(1-2*a1(n))*uu2(n)+(-b1(n)+b2(n))*uu3)
$     +exsY3(i,3,k)*(-a1(n)*uu1Y(n)+a1(n)*uu2(n)-b2(n)*uu3)
$     +exsY3(i,4,k)*(a1(n)*uu1Y(n)+a1(n)*uu2(n)-b2(n)*uu3)
$     +(1.0/(2.0*delx**2))*( ! The d/dx**2 term:
$      +exsY1(i+1,4,k)-2.*exsY1(i,4,k)+exsY1(i-1,4,k)
$      +exsY1(i+1,3,k)-2.*exsY1(i,3,k)+exsY1(i-1,3,k)
$      +a1(n)*(
     $         +exsY2(i+1,4,k)-2.*exsY2(i,4,k)+exsY2(i-1,4,k)
     $         +exsY2(i+1,3,k)-2.*exsY2(i,3,k)+exsY2(i-1,3,k)
     $          )
     $        ) ! end of d/dx**2 term
     $       +(1.0/(2.0*delz**2))*( ! The d/dz**2 term:
     $         +exsY1(i,4,k+1)-2.*exsY1(i,4,k)+exsY1(i,4,k-1)
     $         +exsY1(i,3,k+1)-2.*exsY1(i,3,k)+exsY1(i,3,k-1)
     $         +a1(n)*(
     $         +exsY2(i,4,k+1)-2.*exsY2(i,4,k)+exsY2(i,4,k-1)
     $         +exsY2(i,3,k+1)-2.*exsY2(i,3,k)+exsY2(i,3,k-1)
     $          )
     $        ) ! end of d/dz**2 term
     $       )
              enddo
             enddo

             do k=1,nz  ! Save past values
               do i=1,nx1
                     exsY3(i,1,k)=exsY2(i,1,k)
                     exsY3(i,2,k)=exsY2(i,2,k)
                     exsY3(i,3,k)=exsY2(i,3,k)
                     exsY3(i,4,k)=exsY2(i,4,k)
                     exsY2(i,1,k)=exsY1(i,1,k)
                     exsY2(i,2,k)=exsY1(i,2,k)
                     exsY2(i,3,k)=exsY1(i,3,k)
                     exsY2(i,4,k)=exsY1(i,4,k)
                     exsY1(i,1,k)=exs(i,1,k)
                     exsY1(i,2,k)=exs(i,2,k)
                     exsY1(i,3,k)=exs(i,ny1,k)
                     exsY1(i,4,k)=exs(i,ny,k)
               end do
             end do

      RETURN
      END


c************************************************************
c************************************************************
             SUBROUTINE RADEXZ
             INCLUDE "nufdtd_params.f"
! ABC for Ex at z=1, z=nz

             do j=1,ny  ! y-direction edges & corners
              do ii=0,1
               i=ii*(nx-2)+1  ! i={1,nx-1}
                     n=mtx(i,j,1)
                     exs(i,j,1)=(1.0/(xx1Z-xx2(n)))*(
     $        exs(i,j,2)*(xx1Z+xx2(n))
     $       +exsZ1(i,j,2)*
     $         ((1.0+a1(n))*xx1Z-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
     $       +exsZ1(i,j,1)*
     $         (-(1.0+a1(n))*xx1Z-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
     $       +exsZ2(i,j,2)*(a1(n)*xx1Z-a1(n)*xx2(n)+b1(n)*xx3(n))
     $       +exsZ2(i,j,1)*(-a1(n)*xx1Z-a1(n)*xx2(n)+b1(n)*xx3(n))
     $       +exsZ3(i,j,2)*(b2(n)*xx3(n))
$       +exsZ3(i,j,1)*(b2(n)*xx3(n))
$       )
                n=mtx(i,j,nz)
                exs(i,j,nz)=(1.0/(xx1Z-xx2(n)))*(
$        exs(i,j,nz1)*(xx1Z+xx2(n))
$       +exsZ1(i,j,3)*
$         ((1.0+a1(n))*xx1Z-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
$       +exsZ1(i,j,4)*
$         (-(1.0+a1(n))*xx1Z-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
$       +exsZ2(i,j,3)*(a1(n)*xx1Z-a1(n)*xx2(n)+b1(n)*xx3(n))
$       +exsZ2(i,j,4)*(-a1(n)*xx1Z-a1(n)*xx2(n)+b1(n)*xx3(n))
$       +exsZ3(i,j,3)*(b2(n)*xx3(n))
$       +exsZ3(i,j,4)*(b2(n)*xx3(n))
$       )
       enddo
      enddo

      do i=2,nx-2  ! x-direction edges
       do jj=0,1
        j=jj*(ny-1)+1  ! j={1,ny}
                n=mtx(i,j,1)
                exs(i,j,1)=(1.0/(xx1Z-xx2(n)))*(
$        exs(i,j,2)*(xx1Z+xx2(n))
$       +exsZ1(i,j,2)*
$         ((1.0+a1(n))*xx1Z-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
$       +exsZ1(i,j,1)*
$         (-(1.0+a1(n))*xx1Z-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
$       +exsZ2(i,j,2)*(a1(n)*xx1Z-a1(n)*xx2(n)+b1(n)*xx3(n))
$       +exsZ2(i,j,1)*(-a1(n)*xx1Z-a1(n)*xx2(n)+b1(n)*xx3(n))
$       +exsZ3(i,j,2)*(b2(n)*xx3(n))
$       +exsZ3(i,j,1)*(b2(n)*xx3(n))
$       )
                n=mtx(i,j,nz)
                exs(i,j,nz)=(1.0/(xx1Z-xx2(n)))*(
$        exs(i,j,nz1)*(xx1Z+xx2(n))
$       +exsZ1(i,j,3)*
$         ((1.0+a1(n))*xx1Z-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
$       +exsZ1(i,j,4)*
$         (-(1.0+a1(n))*xx1Z-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
$       +exsZ2(i,j,3)*(a1(n)*xx1Z-a1(n)*xx2(n)+b1(n)*xx3(n))
$       +exsZ2(i,j,4)*(-a1(n)*xx1Z-a1(n)*xx2(n)+b1(n)*xx3(n))
$       +exsZ3(i,j,3)*(b2(n)*xx3(n))
$       +exsZ3(i,j,4)*(b2(n)*xx3(n))
$       )
       enddo
      enddo

      do j=2,ny1  ! x-y faces
       do i=2,nx-2
                n=mtx(i,j,1)
                exs(i,j,1)=(1.0/(uu1Z(n)-uu2(n)-b0(n)*uu3))*(
$       +exs(i,j,2)*(uu1Z(n)+uu2(n)+b0(n)*uu3)
$       +exsZ1(i,j,2)*
$         (a1(n)*uu1Z(n)+(-2.0+a1(n))*uu2(n)+(-b0(n)+b1(n))*uu3)
$       +exsZ1(i,j,1)*
$         (-a1(n)*uu1Z(n)+(-2.0+a1(n))*uu2(n)+(-b0(n)+b1(n))*uu3)
$     +exsZ2(i,j,2)*
$       (-uu1Z(n)+(1-2*a1(n))*uu2(n)+(-b1(n)+b2(n))*uu3)
$     +exsZ2(i,j,1)*
$       (uu1Z(n)+(1-2*a1(n))*uu2(n)+(-b1(n)+b2(n))*uu3)
$     +exsZ3(i,j,2)*(-a1(n)*uu1Z(n)+a1(n)*uu2(n)-b2(n)*uu3)
$     +exsZ3(i,j,1)*(a1(n)*uu1Z(n)+a1(n)*uu2(n)-b2(n)*uu3)
$     +(1.0/(2.0*delx**2))*( ! The d/dx**2 term:
$       +exsZ1(i+1,j,1)-2.*exsZ1(i,j,1)+exsZ1(i-1,j,1)
$       +exsZ1(i+1,j,2)-2.*exsZ1(i,j,2)+exsZ1(i-1,j,2)
$       +a1(n)*(
$       +exsZ2(i+1,j,1)-2.*exsZ2(i,j,1)+exsZ2(i-1,j,1)
$       +exsZ2(i+1,j,2)-2.*exsZ2(i,j,2)+exsZ2(i-1,j,2)
$        )
$      ) ! end of d/dx**2 term
$     +(1.0/(2.0*dely**2))*( ! The d/dy**2 term:
$       +exsZ1(i,j+1,1)-2.*exsZ1(i,j,1)+exsZ1(i,j-1,1)
$       +exsZ1(i,j+1,2)-2.*exsZ1(i,j,2)+exsZ1(i,j-1,2)
$       +a1(n)*(
$       +exsZ2(i,j+1,1)-2.*exsZ2(i,j,1)+exsZ2(i,j-1,1)
$       +exsZ2(i,j+1,2)-2.*exsZ2(i,j,2)+exsZ2(i,j-1,2)
$        )
$      ) ! end of d/dy**2 term
$    )
            n=mtx(i,j,nz)
            exs(i,j,nz)=(1.0/(uu1Z(n)-uu2(n)-b0(n)*uu3))*(
$     +exs(i,j,nz1)*(uu1Z(n)+uu2(n)+b0(n)*uu3)
$     +exsZ1(i,j,3)*
$       (a1(n)*uu1Z(n)+(-2.0+a1(n))*uu2(n)+(-b0(n)+b1(n))*uu3)
$     +exsZ1(i,j,4)*
$       (-a1(n)*uu1Z(n)+(-2.0+a1(n))*uu2(n)+(-b0(n)+b1(n))*uu3)
$     +exsZ2(i,j,3)*
$       (-uu1Z(n)+(1-2*a1(n))*uu2(n)+(-b1(n)+b2(n))*uu3)
$     +exsZ2(i,j,4)*
$       (uu1Z(n)+(1-2*a1(n))*uu2(n)+(-b1(n)+b2(n))*uu3)
$     +exsZ3(i,j,3)*(-a1(n)*uu1Z(n)+a1(n)*uu2(n)-b2(n)*uu3)
$     +exsZ3(i,j,4)*(a1(n)*uu1Z(n)+a1(n)*uu2(n)-b2(n)*uu3)
$     +(1.0/(2.0*delx**2))*( ! The d/dx**2 term:
$       +exsZ1(i+1,j,4)-2.*exsZ1(i,j,4)+exsZ1(i-1,j,4)
$       +exsZ1(i+1,j,3)-2.*exsZ1(i,j,3)+exsZ1(i-1,j,3)
$       +a1(n)*(
$       +exsZ2(i+1,j,4)-2.*exsZ2(i,j,4)+exsZ2(i-1,j,4)
$       +exsZ2(i+1,j,3)-2.*exsZ2(i,j,3)+exsZ2(i-1,j,3)
$        )
$      ) ! end of d/dx**2 term
$     +(1.0/(2.0*dely**2))*( ! The d/dy**2 term:
$       +exsZ1(i,j+1,4)-2.*exsZ1(i,j,4)+exsZ1(i,j-1,4)
$       +exsZ1(i,j+1,3)-2.*exsZ1(i,j,3)+exsZ1(i,j-1,3)
$       +a1(n)*(
$       +exsZ2(i,j+1,4)-2.*exsZ2(i,j,4)+exsZ2(i,j-1,4)
$       +exsZ2(i,j+1,3)-2.*exsZ2(i,j,3)+exsZ2(i,j-1,3)
$        )
$      ) ! end of d/dy**2 term
$    )
     enddo
    enddo
```

```
       do j=1,ny  ! Save past values
        do i=1,nx1
              exsZ3(i,j,1)=exsZ2(i,j,1)
              exsZ3(i,j,2)=exsZ2(i,j,2)
              exsZ3(i,j,3)=exsZ2(i,j,3)
              exsZ3(i,j,4)=exsZ2(i,j,4)
              exsZ2(i,j,1)=exsZ1(i,j,1)
              exsZ2(i,j,2)=exsZ1(i,j,2)
              exsZ2(i,j,3)=exsZ1(i,j,3)
              exsZ2(i,j,4)=exsZ1(i,j,4)
              exsZ1(i,j,1)=exs(i,j,1)
              exsZ1(i,j,2)=exs(i,j,2)
              exsZ1(i,j,3)=exs(i,j,nz1)
              exsZ1(i,j,4)=exs(i,j,nz)
        end do
       end do

   RETURN

   END


c************************************************************
c************************************************************
       SUBROUTINE RADEYX
       INCLUDE "nufdtd_params.f"
! ABC for Ey at x=1, x=nx

       if(ndim<3)then  ! 2D
              ny2=2
       else  ! 3D
              ny2=ny
       endif

       do k=1,nz  ! z-direction edges & corners
        do jj=0,1
        j=jj*(ny2-2)+1  ! j={1,ny-1}
              n=mtx(1,j,k)
              eys(1,j,k)=(1.0/(xx1X-xx2(n)))*(
   $    +eys(2,j,k)*(xx1X+xx2(n))
   $    +eysX1(2,j,k)*
   $     ((1.0+a1(n))*xx1X-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
   $    +eysX1(1,j,k)*
   $     (-(1.0+a1(n))*xx1X-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
   $    +eysX2(2,j,k)*(a1(n)*xx1X-a1(n)*xx2(n)+b1(n)*xx3(n))
   $    +eysX2(1,j,k)*(-a1(n)*xx1X-a1(n)*xx2(n)+b1(n)*xx3(n))
   $    +eysX3(2,j,k)*(b2(n)*xx3(n))
   $    +eysX3(1,j,k)*(b2(n)*xx3(n))
   $    )
              n=mtx(nx,j,k)
              eys(nx,j,k)=(1.0/(xx1X-xx2(n)))*(
   $     eys(nx1,j,k)*(xx1X+xx2(n))
   $    +eysX1(3,j,k)*
   $     ((1.0+a1(n))*xx1X-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
   $    +eysX1(4,j,k)*
   $     (-(1.0+a1(n))*xx1X-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
$       +eysX2(3,j,k)*(a1(n)*xx1X-a1(n)*xx2(n)+b1(n)*xx3(n))
$       +eysX2(4,j,k)*(-a1(n)*xx1X-a1(n)*xx2(n)+b1(n)*xx3(n))
$       +eysX3(3,j,k)*(b2(n)*xx3(n))
$       +eysX3(4,j,k)*(b2(n)*xx3(n))
$       )
       enddo
      enddo

      do j=2,ny-2  ! y-direction edges
       do kk=0,1
        k=kk*(nz-1)+1   ! k={1,nz}
                n=mtx(1,j,k)
                eys(1,j,k)=(1.0/(xx1X-xx2(n)))*(
$        eys(2,j,k)*(xx1X+xx2(n))
$       +eysX1(2,j,k)*
$         ((1.0+a1(n))*xx1X-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
$       +eysX1(1,j,k)*
$         (-(1.0+a1(n))*xx1X-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
$       +eysX2(2,j,k)*(a1(n)*xx1X-a1(n)*xx2(n)+b1(n)*xx3(n))
$       +eysX2(1,j,k)*(-a1(n)*xx1X-a1(n)*xx2(n)+b1(n)*xx3(n))
$       +eysX3(2,j,k)*(b2(n)*xx3(n))
$       +eysX3(1,j,k)*(b2(n)*xx3(n))
$       )
                n=mtx(nx,j,k)
                eys(nx,j,k)=(1.0/(xx1X-xx2(n)))*(
$        eys(nx1,j,k)*(xx1X+xx2(n))
$       +eysX1(3,j,k)*
$         ((1.0+a1(n))*xx1X-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
$       +eysX1(4,j,k)*
$         (-(1.0+a1(n))*xx1X-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
$       +eysX2(3,j,k)*(a1(n)*xx1X-a1(n)*xx2(n)+b1(n)*xx3(n))
$       +eysX2(4,j,k)*(-a1(n)*xx1X-a1(n)*xx2(n)+b1(n)*xx3(n))
$       +eysX3(3,j,k)*(b2(n)*xx3(n))
$       +eysX3(4,j,k)*(b2(n)*xx3(n))
$       )
       enddo
      enddo

      do k=2,nz1  ! y-z faces
       do j=2,ny-2
                n=mtx(1,j,k)
                eys(1,j,k)=(1.0/(uu1X(n)-uu2(n)-b0(n)*uu3))*(
$        eys(2,j,k)*(uu1X(n)+uu2(n)+b0(n)*uu3)
$       +eysX1(2,j,k)*
$         (a1(n)*uu1X(n)+(-2.0+a1(n))*uu2(n)+(-b0(n)+b1(n))*uu3)
$       +eysX1(1,j,k)*
$         (-a1(n)*uu1X(n)+(-2.0+a1(n))*uu2(n)+(-b0(n)+b1(n))*uu3)
$       +eysX2(2,j,k)*
$         (-uu1X(n)+(1-2*a1(n))*uu2(n)+(-b1(n)+b2(n))*uu3)
$       +eysX2(1,j,k)*
$         (uu1X(n)+(1-2*a1(n))*uu2(n)+(-b1(n)+b2(n))*uu3)
$       +eysX3(2,j,k)*(-a1(n)*uu1X(n)+a1(n)*uu2(n)-b2(n)*uu3)
$       +eysX3(1,j,k)*(a1(n)*uu1X(n)+a1(n)*uu2(n)-b2(n)*uu3)
$       +(1.0/(2.0*dely**2))*( ! The d/dy**2 term:
$          eysX1(1,j+1,k)-2.*eysX1(1,j,k)+eysX1(1,j-1,k)
$         +eysX1(2,j+1,k)-2.*eysX1(2,j,k)+eysX1(2,j-1,k)
$           +a1(n)*(
$            eysX2(1,j+1,k)-2.*eysX2(1,j,k)+eysX2(1,j-1,k)
$           +eysX2(2,j+1,k)-2.*eysX2(2,j,k)+eysX2(2,j-1,k)
$            )
$          ) !end of d/dy**2 term
$         +(1.0/(2.0*delz**2))*( ! The d/dz**2 term:
$            eysX1(1,j,k+1)-2.*eysX1(1,j,k)+eysX1(1,j,k-1)
$           +eysX1(2,j,k+1)-2.*eysX1(2,j,k)+eysX1(2,j,k-1)
$           +a1(n)*(
$           +eysX2(1,j,k+1)-2.*eysX2(1,j,k)+eysX2(1,j,k-1)
$           +eysX2(2,j,k+1)-2.*eysX2(2,j,k)+eysX2(2,j,k-1)
$            )
$          ) !end of d/dz**2 term
$         )
                    n=mtx(nx,j,k)
                    eys(nx,j,k)=(1.0/(uu1X(n)-uu2(n)-b0(n)*uu3))*(
$          eys(nx1,j,k)*(uu1X(n)+uu2(n)+b0(n)*uu3)
$         +eysX1(3,j,k)*
$           (a1(n)*uu1X(n)+(-2.0+a1(n))*uu2(n)+(-b0(n)+b1(n))*uu3)
$         +eysX1(4,j,k)*
$           (-a1(n)*uu1X(n)+(-2.0+a1(n))*uu2(n)+(-b0(n)+b1(n))*uu3)
$         +eysX2(3,j,k)*
$           (-uu1X(n)+(1-2*a1(n))*uu2(n)+(-b1(n)+b2(n))*uu3)
$         +eysX2(4,j,k)*
$           (uu1X(n)+(1-2*a1(n))*uu2(n)+(-b1(n)+b2(n))*uu3)
$         +eysX3(3,j,k)*(-a1(n)*uu1X(n)+a1(n)*uu2(n)-b2(n)*uu3)
$         +eysX3(4,j,k)*(a1(n)*uu1X(n)+a1(n)*uu2(n)-b2(n)*uu3)
$         +(1.0/(2.0*dely**2))*( ! The d/dy**2 term:
$            eysX1(4,j+1,k)-2.*eysX1(4,j,k)+eysX1(4,j-1,k)
$           +eysX1(3,j+1,k)-2.*eysX1(3,j,k)+eysX1(3,j-1,k)
$           +a1(n)*(
$           +eysX2(4,j+1,k)-2.*eysX2(4,j,k)+eysX2(4,j-1,k)
$           +eysX2(3,j+1,k)-2.*eysX2(3,j,k)+eysX2(3,j-1,k)
$            )
$          ) !end of d/dy**2 term
$         +(1.0/(2.0*delz**2))*( ! The d/dz**2 term:
$           +eysX1(4,j,k+1)-2.*eysX1(4,j,k)+eysX1(4,j,k-1)
$           +eysX1(3,j,k+1)-2.*eysX1(3,j,k)+eysX1(3,j,k-1)
$           +a1(n)*(
$           +eysX2(4,j,k+1)-2.*eysX2(4,j,k)+eysX2(4,j,k-1)
$           +eysX2(3,j,k+1)-2.*eysX2(3,j,k)+eysX2(3,j,k-1)
$            )
$          ) !end of d/dz**2 term
$         )
         enddo
        enddo

        do k=1,nz  ! Save past values
          do j=1,ny2-1
                    eysX3(1,j,k)=eysX2(1,j,k)
                    eysX3(2,j,k)=eysX2(2,j,k)
                    eysX3(3,j,k)=eysX2(3,j,k)
                    eysX3(4,j,k)=eysX2(4,j,k)
                    eysX2(1,j,k)=eysX1(1,j,k)
                    eysX2(2,j,k)=eysX1(2,j,k)
                    eysX2(3,j,k)=eysX1(3,j,k)
```

```
                          eysX2(4,j,k)=eysX1(4,j,k)
                          eysX1(1,j,k)=eys(1,j,k)
                          eysX1(2,j,k)=eys(2,j,k)
                          eysX1(3,j,k)=eys(nx1,j,k)
                          eysX1(4,j,k)=eys(nx,j,k)
               end do
             end do

      RETURN
      END


c***********************************************************
c***********************************************************
             SUBROUTINE RADEYZ
             INCLUDE "nufdtd_params.f"
! ABC for Ey at z=1, z=nz

             if(ndim<3)then  ! 2D
                          ny2=2
             else  ! 3D
                          ny2=ny
             endif

             do i=1,nx  ! x-direction edges
              do jj=0,1
               j=jj*(ny2-2)+1  ! j={1,ny-1}
                          n=mtx(i,j,1)
                          eys(i,j,1)=(1.0/(xx1Z-xx2(n)))*(
     $         +eys(i,j,2)*(xx1Z+xx2(n))
     $         +eysZ1(i,j,2)*
     $           ((1.0+a1(n))*xx1Z-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
     $         +eysZ1(i,j,1)*
     $           (-(1.0+a1(n))*xx1Z-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
     $         +eysZ2(i,j,2)*(a1(n)*xx1Z-a1(n)*xx2(n)+b1(n)*xx3(n))
     $         +eysZ2(i,j,1)*(-a1(n)*xx1Z-a1(n)*xx2(n)+b1(n)*xx3(n))
     $         +eysZ3(i,j,2)*(b2(n)*xx3(n))
     $         +eysZ3(i,j,1)*(b2(n)*xx3(n))
     $         )
                          n=mtx(i,j,nz)
                          eys(i,j,nz)=(1.0/(xx1Z-xx2(n)))*(
     $         +eys(i,j,nz1)*(xx1Z+xx2(n))
     $         +eysZ1(i,j,3)*
     $           ((1.0+a1(n))*xx1Z-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
     $         +eysZ1(i,j,4)*
     $           (-(1.0+a1(n))*xx1Z-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
     $         +eysZ2(i,j,3)*(a1(n)*xx1Z-a1(n)*xx2(n)+b1(n)*xx3(n))
     $         +eysZ2(i,j,4)*(-a1(n)*xx1Z-a1(n)*xx2(n)+b1(n)*xx3(n))
     $         +eysZ3(i,j,3)*(b2(n)*xx3(n))
     $         +eysZ3(i,j,4)*(b2(n)*xx3(n))
     $         )
              enddo
             enddo

             do j=2,ny-2  ! y-direction edges
              do ii=0,1
```

```
       i=ii*(nx-1)+1      ! i={1,nx}
             n=mtx(i,j,1)
             eys(i,j,1)=(1.0/(xx1Z-xx2(n)))*(
$      +eys(i,j,2)*(xx1Z+xx2(n))
$      +eysZ1(i,j,2)*
$        ((1.0+a1(n))*xx1Z-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
$      +eysZ1(i,j,1)*
$        (-(1.0+a1(n))*xx1Z-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
$      +eysZ2(i,j,2)*(a1(n)*xx1Z-a1(n)*xx2(n)+b1(n)*xx3(n))
$      +eysZ2(i,j,1)*(-a1(n)*xx1Z-a1(n)*xx2(n)+b1(n)*xx3(n))
$      +eysZ3(i,j,2)*(b2(n)*xx3(n))
$      +eysZ3(i,j,1)*(b2(n)*xx3(n))
$      )
             n=mtx(i,j,nz)
             eys(i,j,nz)=(1.0/(xx1Z-xx2(n)))*(
$      +eys(i,j,nz1)*(xx1Z+xx2(n))
$      +eysZ1(i,j,3)*
$        ((1.0+a1(n))*xx1Z-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
$      +eysZ1(i,j,4)*
$        (-(1.0+a1(n))*xx1Z-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
$      +eysZ2(i,j,3)*(a1(n)*xx1Z-a1(n)*xx2(n)+b1(n)*xx3(n))
$      +eysZ2(i,j,4)*(-a1(n)*xx1Z-a1(n)*xx2(n)+b1(n)*xx3(n))
$      +eysZ3(i,j,3)*(b2(n)*xx3(n))
$      +eysZ3(i,j,4)*(b2(n)*xx3(n))
$      )
      enddo
     enddo

     do j=2,ny-2  ! x-y faces
      do i=2,nx1
             n=mtx(i,j,1)
             eys(i,j,1)=(1.0/(uu1Z(n)-uu2(n)-b0(n)*uu3))*(
$      +eys(i,j,2)*(uu1Z(n)+uu2(n)+b0(n)*uu3)
$      +eysZ1(i,j,2)*
$        (a1(n)*uu1Z(n)+(-2.0+a1(n))*uu2(n)+(-b0(n)+b1(n))*uu3)
$      +eysZ1(i,j,1)*
$        (-a1(n)*uu1Z(n)+(-2.0+a1(n))*uu2(n)+(-b0(n)+b1(n))*uu3)
$      +eysZ2(i,j,2)*
$        (-uu1Z(n)+(1-2*a1(n))*uu2(n)+(-b1(n)+b2(n))*uu3)
$      +eysZ2(i,j,1)*
$        (uu1Z(n)+(1-2*a1(n))*uu2(n)+(-b1(n)+b2(n))*uu3)
$      +eysZ3(i,j,2)*(-a1(n)*uu1Z(n)+a1(n)*uu2(n)-b2(n)*uu3)
$      +eysZ3(i,j,1)*(a1(n)*uu1Z(n)+a1(n)*uu2(n)-b2(n)*uu3)
$      +(1.0/(2.0*delx**2))*( ! The d/dx**2 term:
$         eysZ1(i+1,j,1)-2.*eysZ1(i,j,1)+eysZ1(i-1,j,1)
$        +eysZ1(i+1,j,2)-2.*eysZ1(i,j,2)+eysZ1(i-1,j,2)
$        +a1(n)*(
$        +eysZ2(i+1,j,1)-2.*eysZ2(i,j,1)+eysZ2(i-1,j,1)
$        +eysZ2(i+1,j,2)-2.*eysZ2(i,j,2)+eysZ2(i-1,j,2)
$         )
$       ) ! end of d/dx**2 term
$      +(1.0/(2.0*dely**2))*( ! The d/dy**2 part:
$         eysZ1(i,j+1,1)-2.*eysZ1(i,j,1)+eysZ1(i,j-1,1)
$        +eysZ1(i,j+1,2)-2.*eysZ1(i,j,2)+eysZ1(i,j-1,2)
$        +a1(n)*(
$        +eysZ2(i,j+1,1)-2.*eysZ2(i,j,1)+eysZ2(i,j-1,1)
$          +eysZ2(i,j+1,2)-2.*eysZ2(i,j,2)+eysZ2(i,j-1,2)
$           )
$         ) ! end of d/dy**2 term
$        )
                    n=mtx(i,j,nz)
                    eys(i,j,nz)=(1.0/(uu1Z(n)-uu2(n)-b0(n)*uu3))*(
$        +eys(i,j,nz1)*(uu1Z(n)+uu2(n)+b0(n)*uu3)
$        +eysZ1(i,j,3)*
$          (a1(n)*uu1Z(n)+(-2.0+a1(n))*uu2(n)+(-b0(n)+b1(n))*uu3)
$        +eysZ1(i,j,4)*
$          (-a1(n)*uu1Z(n)+(-2.0+a1(n))*uu2(n)+(-b0(n)+b1(n))*uu3)
$        +eysZ2(i,j,3)*
$          (-uu1Z(n)+(1-2*a1(n))*uu2(n)+(-b1(n)+b2(n))*uu3)
$        +eysZ2(i,j,4)*
$          (uu1Z(n)+(1-2*a1(n))*uu2(n)+(-b1(n)+b2(n))*uu3)
$        +eysZ3(i,j,3)*(-a1(n)*uu1Z(n)+a1(n)*uu2(n)-b2(n)*uu3)
$        +eysZ3(i,j,4)*(a1(n)*uu1Z(n)+a1(n)*uu2(n)-b2(n)*uu3)
$        +(1.0/(2.0*delx**2))*( ! The d/dx**2 term:
$           eysZ1(i+1,j,4)-2.*eysZ1(i,j,4)+eysZ1(i-1,j,4)
$          +eysZ1(i+1,j,3)-2.*eysZ1(i,j,3)+eysZ1(i-1,j,3)
$          +a1(n)*(
$          +eysZ2(i+1,j,4)-2.*eysZ2(i,j,4)+eysZ2(i-1,j,4)
$          +eysZ2(i+1,j,3)-2.*eysZ2(i,j,3)+eysZ2(i-1,j,3)
$           )
$         ) ! end of d/dx**2 term
$        +(1.0/(2.0*dely**2))*( ! The d/dy**2 part:
$           eysZ1(i,j+1,4)-2.*eysZ1(i,j,4)+eysZ1(i,j-1,4)
$          +eysZ1(i,j+1,3)-2.*eysZ1(i,j,3)+eysZ1(i,j-1,3)
$          +a1(n)*(
$          +eysZ2(i,j+1,4)-2.*eysZ2(i,j,4)+eysZ2(i,j-1,4)
$          +eysZ2(i,j+1,3)-2.*eysZ2(i,j,3)+eysZ2(i,j-1,3)
$           )
$         ) ! end of d/dy**2 term
$        )
        enddo
       enddo

       do j=1,ny2-1  ! Save past values
         do i=1,nx
                    eysZ3(i,j,1)=eysZ2(i,j,1)
                    eysZ3(i,j,2)=eysZ2(i,j,2)
                    eysZ3(i,j,3)=eysZ2(i,j,3)
                    eysZ3(i,j,4)=eysZ2(i,j,4)
                    eysZ2(i,j,1)=eysZ1(i,j,1)
                    eysZ2(i,j,2)=eysZ1(i,j,2)
                    eysZ2(i,j,3)=eysZ1(i,j,3)
                    eysZ2(i,j,4)=eysZ1(i,j,4)
                    eysZ1(i,j,1)=eys(i,j,1)
                    eysZ1(i,j,2)=eys(i,j,2)
                    eysZ1(i,j,3)=eys(i,j,nz1)
                    eysZ1(i,j,4)=eys(i,j,nz)
         end do
       end do

 RETURN
 END
```

```
c***********************************************************
c***********************************************************
      SUBROUTINE RADEZX
      INCLUDE "nufdtd_params.f"
! ABC for Ez at x=1, x=nx

      do k=2,nz-2  ! z-direction edges
       do jj=0,1
        j=jj*(ny-1)+1      ! j={1,ny}
             n=mtx(1,j,k)
             ezs(1,j,k)=(1.0/(xx1X-xx2(n)))*(
  $      ezs(2,j,k)*(xx1X+xx2(n))
  $     +ezsX1(2,j,k)*
  $       ((1.0+a1(n))*xx1X-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
  $     +ezsX1(1,j,k)*
  $       (-(1.0+a1(n))*xx1X-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
  $     +ezsX2(2,j,k)*(a1(n)*xx1X-a1(n)*xx2(n)+b1(n)*xx3(n))
  $     +ezsX2(1,j,k)*(-a1(n)*xx1X-a1(n)*xx2(n)+b1(n)*xx3(n))
  $     +ezsX3(2,j,k)*(b2(n)*xx3(n))
  $     +ezsX3(1,j,k)*(b2(n)*xx3(n))
  $     )
             n=mtx(nx,j,k)
             ezs(nx,j,k)=(1.0/(xx1X-xx2(n)))*(
  $      ezs(nx1,j,k)*(xx1X+xx2(n))
  $     +ezsX1(3,j,k)*
  $       ((1.0+a1(n))*xx1X-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
  $     +ezsX1(4,j,k)*
  $       (-(1.0+a1(n))*xx1X-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
  $     +ezsX2(3,j,k)*(a1(n)*xx1X-a1(n)*xx2(n)+b1(n)*xx3(n))
  $     +ezsX2(4,j,k)*(-a1(n)*xx1X-a1(n)*xx2(n)+b1(n)*xx3(n))
  $     +ezsX3(3,j,k)*(b2(n)*xx3(n))
  $     +ezsX3(4,j,k)*(b2(n)*xx3(n))
  $     )
       enddo
      enddo

      do j=1,ny  ! y-direction edges & corners
       do kk=0,1
        k=kk*(nz-2)+1  ! k={1,nz-1}
             n=mtx(1,j,k)
             ezs(1,j,k)=(1.0/(xx1X-xx2(n)))*(
  $      ezs(2,j,k)*(xx1X+xx2(n))
  $     +ezsX1(2,j,k)*
  $       ((1.0+a1(n))*xx1X-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
  $     +ezsX1(1,j,k)*
  $       (-(1.0+a1(n))*xx1X-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
  $     +ezsX2(2,j,k)*(a1(n)*xx1X-a1(n)*xx2(n)+b1(n)*xx3(n))
  $     +ezsX2(1,j,k)*(-a1(n)*xx1X-a1(n)*xx2(n)+b1(n)*xx3(n))
  $     +ezsX3(2,j,k)*(b2(n)*xx3(n))
  $     +ezsX3(1,j,k)*(b2(n)*xx3(n))
  $     )
             n=mtx(nx,j,k)
             ezs(nx,j,k)=(1.0/(xx1X-xx2(n)))*(
  $      ezs(nx1,j,k)*(xx1X+xx2(n))
$       +ezsX1(3,j,k)*
$         ((1.0+a1(n))*xx1X-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
$       +ezsX1(4,j,k)*
$         (-(1.0+a1(n))*xx1X-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
$       +ezsX2(3,j,k)*(a1(n)*xx1X-a1(n)*xx2(n)+b1(n)*xx3(n))
$       +ezsX2(4,j,k)*(-a1(n)*xx1X-a1(n)*xx2(n)+b1(n)*xx3(n))
$       +ezsX3(3,j,k)*(b2(n)*xx3(n))
$       +ezsX3(4,j,k)*(b2(n)*xx3(n))
$       )
       enddo
      enddo

      do k=2,nz-2  ! y-z faces
       do j=2,ny1
                 n=mtx(1,j,k)
                 ezs(1,j,k)=(1.0/(uu1X(n)-uu2(n)-b0(n)*uu3))*(
$        ezs(2,j,k)*(uu1X(n)+uu2(n)+b0(n)*uu3)
$       +ezsX1(2,j,k)*
$         (a1(n)*uu1X(n)+(-2.0+a1(n))*uu2(n)+(-b0(n)+b1(n))*uu3)
$       +ezsX1(1,j,k)*
$         (-a1(n)*uu1X(n)+(-2.0+a1(n))*uu2(n)+(-b0(n)+b1(n))*uu3)
$       +ezsX2(2,j,k)*
$         (-uu1X(n)+(1-2*a1(n))*uu2(n)+(-b1(n)+b2(n))*uu3)
$       +ezsX2(1,j,k)*
$         (uu1X(n)+(1-2*a1(n))*uu2(n)+(-b1(n)+b2(n))*uu3)
$       +ezsX3(2,j,k)*(-a1(n)*uu1X(n)+a1(n)*uu2(n)-b2(n)*uu3)
$       +ezsX3(1,j,k)*(a1(n)*uu1X(n)+a1(n)*uu2(n)-b2(n)*uu3)
$       +(1.0/(2.0*dely**2))*( ! The d/dy**2 term:
$          ezsX1(1,j+1,k)-2.*ezsX1(1,j,k)+ezsX1(1,j-1,k)
$         +ezsX1(2,j+1,k)-2.*ezsX1(2,j,k)+ezsX1(2,j-1,k)
$         +a1(n)*(
$         +ezsX2(1,j+1,k)-2.*ezsX2(1,j,k)+ezsX2(1,j-1,k)
$         +ezsX2(2,j+1,k)-2.*ezsX2(2,j,k)+ezsX2(2,j-1,k)
$          )
$        ) ! end of d/dy**2 term
$       +(1.0/(2.0*delz**2))*( ! The d/dz**2 term:
$          ezsX1(1,j,k+1)-2.*ezsX1(1,j,k)+ezsX1(1,j,k-1)
$         +ezsX1(2,j,k+1)-2.*ezsX1(2,j,k)+ezsX1(2,j,k-1)
$         +a1(n)*(
$         +ezsX2(1,j,k+1)-2.*ezsX2(1,j,k)+ezsX2(1,j,k-1)
$         +ezsX2(2,j,k+1)-2.*ezsX2(2,j,k)+ezsX2(2,j,k-1)
$          )
$        ) ! end of d/dz**2 term
$       )
                 n=mtx(nx,j,k)
                 ezs(nx,j,k)=(1.0/(uu1X(n)-uu2(n)-b0(n)*uu3))*(
$        ezs(nx1,j,k)*(uu1X(n)+uu2(n)+b0(n)*uu3)
$       +ezsX1(3,j,k)*
$         (a1(n)*uu1X(n)+(-2.0+a1(n))*uu2(n)+(-b0(n)+b1(n))*uu3)
$       +ezsX1(4,j,k)*
$         (-a1(n)*uu1X(n)+(-2.0+a1(n))*uu2(n)+(-b0(n)+b1(n))*uu3)
$       +ezsX2(3,j,k)*
$         (-uu1X(n)+(1-2*a1(n))*uu2(n)+(-b1(n)+b2(n))*uu3)
$       +ezsX2(4,j,k)*
$         (uu1X(n)+(1-2*a1(n))*uu2(n)+(-b1(n)+b2(n))*uu3)
$       +ezsX3(3,j,k)*(-a1(n)*uu1X(n)+a1(n)*uu2(n)-b2(n)*uu3)
```

```
     $       +ezsX3(4,j,k)*(a1(n)*uu1X(n)+a1(n)*uu2(n)-b2(n)*uu3)
     $       +(1.0/(2.0*dely**2))*( ! The d/dy**2 term:
     $         +ezsX1(4,j+1,k)-2.*ezsX1(4,j,k)+ezsX1(4,j-1,k)
     $         +ezsX1(3,j+1,k)-2.*ezsX1(3,j,k)+ezsX1(3,j-1,k)
     $         +a1(n)*(
     $         +ezsX2(4,j+1,k)-2.*ezsX2(4,j,k)+ezsX2(4,j-1,k)
     $         +ezsX2(3,j+1,k)-2.*ezsX2(3,j,k)+ezsX2(3,j-1,k)
     $          )
     $        ) ! end of d/dy**2 term
     $       +(1.0/(2.*delz**2))*( ! The d/dz**2 term:
     $         +ezsX1(4,j,k+1)-2.*ezsX1(4,j,k)+ezsX1(4,j,k-1)
     $         +ezsX1(3,j,k+1)-2.*ezsX1(3,j,k)+ezsX1(3,j,k-1)
     $         +a1(n)*(
     $         +ezsX2(4,j,k+1)-2.*ezsX2(4,j,k)+ezsX2(4,j,k-1)
     $         +ezsX2(3,j,k+1)-2.*ezsX2(3,j,k)+ezsX2(3,j,k-1)
     $          )
     $        ) ! end of d/dz**2 term
     $       )
             enddo
            enddo

            do k=1,nz1  ! Save past values
              do j=1,ny
                         ezsX3(1,j,k)=ezsX2(1,j,k)
                         ezsX3(2,j,k)=ezsX2(2,j,k)
                         ezsX3(3,j,k)=ezsX2(3,j,k)
                         ezsX3(4,j,k)=ezsX2(4,j,k)
                         ezsX2(1,j,k)=ezsX1(1,j,k)
                         ezsX2(2,j,k)=ezsX1(2,j,k)
                         ezsX2(3,j,k)=ezsX1(3,j,k)
                         ezsX2(4,j,k)=ezsX1(4,j,k)
                         ezsX1(1,j,k)=ezs(1,j,k)
                         ezsX1(2,j,k)=ezs(2,j,k)
                         ezsX1(3,j,k)=ezs(nx1,j,k)
                         ezsX1(4,j,k)=ezs(nx,j,k)
              end do
            end do

      RETURN
      END


c************************************************************
c************************************************************
            SUBROUTINE RADEZY
            INCLUDE "nufdtd_params.f"
! ABC for Ez at y=1, y=ny

            do k=2,nz-2  ! z-direction edges
             do ii=0,1
              i=ii*(nx-1)+1      ! i={1,nx}
                         n=mtx(i,1,k)
                         ezs(i,1,k)=(1.0/(xx1Y-xx2(n)))*(
     $       +ezs(i,2,k)*(xx1Y+xx2(n))
     $       +ezsY1(i,2,k)*
     $         ((1.0+a1(n))*xx1Y-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
$       +ezsY1(i,1,k)*
$         (-(1.0+a1(n))*xx1Y-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
$       +ezsY2(i,2,k)*(a1(n)*xx1Y-a1(n)*xx2(n)+b1(n)*xx3(n))
$       +ezsY2(i,1,k)*(-a1(n)*xx1Y-a1(n)*xx2(n)+b1(n)*xx3(n))
$       +ezsY3(i,2,k)*(b2(n)*xx3(n))
$       +ezsY3(i,1,k)*(b2(n)*xx3(n))
$       )
                   n=mtx(i,ny,k)
                   ezs(i,ny,k)=(1.0/(xx1Y-xx2(n)))*(
$        ezs(i,ny1,k)*(xx1Y+xx2(n))
$       +ezsY1(i,3,k)*
$         ((1.0+a1(n))*xx1Y-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
$       +ezsY1(i,4,k)*
$         (-(1.0+a1(n))*xx1Y-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
$       +ezsY2(i,3,k)*(a1(n)*xx1Y-a1(n)*xx2(n)+b1(n)*xx3(n))
$       +ezsY2(i,4,k)*(-a1(n)*xx1Y-a1(n)*xx2(n)+b1(n)*xx3(n))
$       +ezsY3(i,3,k)*(b2(n)*xx3(n))
$       +ezsY3(i,4,k)*(b2(n)*xx3(n))
$       )
       enddo
      enddo

      do i=1,nx  ! x-direction edges & corners
       do kk=0,1
        k=kk*(nz-2)+1  ! k={1,nz-1}
                   n=mtx(i,1,k)
                   ezs(i,1,k)=(1.0/(xx1Y-xx2(n)))*(
$       +ezs(i,2,k)*(xx1Y+xx2(n))
$       +ezsY1(i,2,k)*
$         ((1.0+a1(n))*xx1Y-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
$       +ezsY1(i,1,k)*
$         (-(1.0+a1(n))*xx1Y-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
$       +ezsY2(i,2,k)*(a1(n)*xx1Y-a1(n)*xx2(n)+b1(n)*xx3(n))
$       +ezsY2(i,1,k)*(-a1(n)*xx1Y-a1(n)*xx2(n)+b1(n)*xx3(n))
$       +ezsY3(i,2,k)*(b2(n)*xx3(n))
$       +ezsY3(i,1,k)*(b2(n)*xx3(n))
$       )
                   n=mtx(i,ny,k)
                   ezs(i,ny,k)=(1.0/(xx1Y-xx2(n)))*(
$        ezs(i,ny1,k)*(xx1Y+xx2(n))
$       +ezsY1(i,3,k)*
$         ((1.0+a1(n))*xx1Y-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
$       +ezsY1(i,4,k)*
$         (-(1.0+a1(n))*xx1Y-(1.0-a1(n))*xx2(n)+b0(n)*xx3(n))
$       +ezsY2(i,3,k)*(a1(n)*xx1Y-a1(n)*xx2(n)+b1(n)*xx3(n))
$       +ezsY2(i,4,k)*(-a1(n)*xx1Y-a1(n)*xx2(n)+b1(n)*xx3(n))
$       +ezsY3(i,3,k)*(b2(n)*xx3(n))
$       +ezsY3(i,4,k)*(b2(n)*xx3(n))
$       )
       enddo
      enddo

      do k=2,nz-2  ! x-y faces
       do i=2,nx1
                   n=mtx(i,1,k)
                   ezs(i,1,k)=(1.0/(uu1Y(n)-uu2(n)-b0(n)*uu3))*(
$      ezs(i,2,k)*(uu1Y(n)+uu2(n)+b0(n)*uu3)
$     +ezsY1(i,2,k)*
$       (a1(n)*uu1Y(n)+(-2.0+a1(n))*uu2(n)+(-b0(n)+b1(n))*uu3)
$     +ezsY1(i,1,k)*
$       (-a1(n)*uu1Y(n)+(-2.0+a1(n))*uu2(n)+(-b0(n)+b1(n))*uu3)
$     +ezsY2(i,2,k)*
$       (-uu1Y(n)+(1-2*a1(n))*uu2(n)+(-b1(n)+b2(n))*uu3)
$     +ezsY2(i,1,k)*
$       (uu1Y(n)+(1-2*a1(n))*uu2(n)+(-b1(n)+b2(n))*uu3)
$     +ezsY3(i,2,k)*(-a1(n)*uu1Y(n)+a1(n)*uu2(n)-b2(n)*uu3)
$     +ezsY3(i,1,k)*(a1(n)*uu1Y(n)+a1(n)*uu2(n)-b2(n)*uu3)
$     +(1.0/(2.0*delx**2))*( ! The d/dx**2 term:
$        ezsY1(i+1,1,k)-2.*ezsY1(i,1,k)+ezsY1(i-1,1,k)
$       +ezsY1(i+1,2,k)-2.*ezsY1(i,2,k)+ezsY1(i-1,2,k)
$       +a1(n)*(
$       +ezsY2(i+1,1,k)-2.*ezsY2(i,1,k)+ezsY2(i-1,1,k)
$       +ezsY2(i+1,2,k)-2.*ezsY2(i,2,k)+ezsY2(i-1,2,k)
$        )
$      ) ! end of d/dx**2 term
$     +(1.0/(2.0*delz**2))*( ! The d/dz**2 term:
$       +ezsY1(i,1,k+1)-2.*ezsY1(i,1,k)+ezsY1(i,1,k-1)
$       +ezsY1(i,2,k+1)-2.*ezsY1(i,2,k)+ezsY1(i,2,k-1)
$       +a1(n)*(
$       +ezsY2(i,1,k+1)-2.*ezsY2(i,1,k)+ezsY2(i,1,k-1)
$       +ezsY2(i,2,k+1)-2.*ezsY2(i,2,k)+ezsY2(i,2,k-1)
$        )
$      ) ! end of d/dz**2 term
$     )
            n=mtx(i,ny,k)
            ezs(i,ny,k)=(1.0/(uu1Y(n)-uu2(n)-b0(n)*uu3))*(
$      ezs(i,ny1,k)*(uu1Y(n)+uu2(n)+b0(n)*uu3)
$     +ezsY1(i,3,k)*
$       (a1(n)*uu1Y(n)+(-2.0+a1(n))*uu2(n)+(-b0(n)+b1(n))*uu3)
$     +ezsY1(i,4,k)*
$       (-a1(n)*uu1Y(n)+(-2.0+a1(n))*uu2(n)+(-b0(n)+b1(n))*uu3)
$     +ezsY2(i,3,k)*
$       (-uu1Y(n)+(1-2*a1(n))*uu2(n)+(-b1(n)+b2(n))*uu3)
$     +ezsY2(i,4,k)*
$       (uu1Y(n)+(1-2*a1(n))*uu2(n)+(-b1(n)+b2(n))*uu3)
$     +ezsY3(i,3,k)*(-a1(n)*uu1Y(n)+a1(n)*uu2(n)-b2(n)*uu3)
$     +ezsY3(i,4,k)*(a1(n)*uu1Y(n)+a1(n)*uu2(n)-b2(n)*uu3)
$     +(1.0/(2.0*delx**2))*( ! The d/dx**2 term:
$       +ezsY1(i+1,4,k)-2.*ezsY1(i,4,k)+ezsY1(i-1,4,k)
$       +ezsY1(i+1,3,k)-2.*ezsY1(i,3,k)+ezsY1(i-1,3,k)
$       +a1(n)*(
$       +ezsY2(i+1,4,k)-2.*ezsY2(i,4,k)+ezsY2(i-1,4,k)
$       +ezsY2(i+1,3,k)-2.*ezsY2(i,3,k)+ezsY2(i-1,3,k)
$        )
$      ) ! end of d/dx**2 term
$     +(1.0/(2.0*delz**2))*( ! The d/dz**2 part:
$       +ezsY1(i,4,k+1)-2.*ezsY1(i,4,k)+ezsY1(i,4,k-1)
$       +ezsY1(i,3,k+1)-2.*ezsY1(i,3,k)+ezsY1(i,3,k-1)
$       +a1(n)*(
$       +ezsY2(i,4,k+1)-2.*ezsY2(i,4,k)+ezsY2(i,4,k-1)
$       +ezsY2(i,3,k+1)-2.*ezsY2(i,3,k)+ezsY2(i,3,k-1)
$        )
```

```
     $        ) ! end of d/dz**2 term
     $       )
             enddo
            enddo

            do k=1,nz1  ! Save past values
              do i=1,nx
                        ezsY3(i,1,k)=ezsY2(i,1,k)
                        ezsY3(i,2,k)=ezsY2(i,2,k)
                        ezsY3(i,3,k)=ezsY2(i,3,k)
                        ezsY3(i,4,k)=ezsY2(i,4,k)
                        ezsY2(i,1,k)=ezsY1(i,1,k)
                        ezsY2(i,2,k)=ezsY1(i,2,k)
                        ezsY2(i,3,k)=ezsY1(i,3,k)
                        ezsY2(i,4,k)=ezsY1(i,4,k)
                        ezsY1(i,1,k)=ezs(i,1,k)
                        ezsY1(i,2,k)=ezs(i,2,k)
                        ezsY1(i,3,k)=ezs(i,ny1,k)
                        ezsY1(i,4,k)=ezs(i,ny,k)
              end do
            end do

      RETURN
      END


c**********************************************************
c**********************************************************
            SUBROUTINE WRITE_MT(mtdir,mtloc)
            INCLUDE "nufdtd_params.f"

! Writes material files.

12180 format(999999999(i3))  ! Format statement used in all write statements

            i2=mod(mtloc,10)  ! Location Ones Digit
            i1=mod((mtloc-i2)/10,10)  ! Location Tens Digit
            i0=mod((mtloc-i1*10-i2)/100,10)  ! Location Hundreds Digit

            loc=char(48+i0)//char(48+i1)//char(48+i2)//'.dat'  ! Slice location

            if(mtdir==1)then  ! x-direction slice (yz plane)
                        outname='mt_x'//loc
                        open(unit=1,file=outname,status='unknown')
                          do kk=1,nz
                                    write(1,12180)(mtx(mtloc,jj,nz-kk+1),jj=1,ny)
                          enddo
                        close(unit=no)
            elseif(mtdir==2)then  ! y-direction slice (xz plane)
                        outname='mt_y'//loc
                        open(unit=1,file=outname,status='unknown')
                          do kk=1,nz
                                    write(1,12180)(mtx(ii,mtloc,nz-kk+1),ii=1,nx)
                          enddo
                        close(unit=no)
            elseif(mtdir==3)then  ! z-direction slice (xy plane)
```

```
                outname='mt_z'//loc
                open(unit=1,file=outname,status='unknown')
                  do jj=1,ny
                        write(1,12180)(mtx(ii,ny-jj+1,mtloc),ii=1,nx)
                  enddo
                close(unit=no)
        endif


        RETURN
        END


c***********************************************************
c***********************************************************
        SUBROUTINE WRITEF_FC
        INCLUDE "nufdtd_params.f"

! Writes field component output files.

        i2=mod(iii,10)  ! Time Step Ones Digit
        i1=mod((iii-i2)/10,10)  ! Time Step Tens Digit
        i0=mod((iii-i1*10-i2)/100,10)  ! Time Step Hundreds Digit

        tsn='_t'//char(48+i0)//char(48+i1)//char(48+i2)//'.dat'  ! Time Step Number

02115 format(999999999(1x,f30.16))  ! Format statement used in all write statements

        do no=1,nout  ! For each output file per time step
                j2=mod(mloc(no),10)  ! Slice Coordinate Ones Digit
                j1=mod((mloc(no)-j2)/10,10)  ! Slice Coordinate Tens Digit
                j0=mod((mloc(no)-j1*10-j2)/100,10)  ! Slice Coordinate Hundreds Digit
        scn=char(48+j0)//char(48+j1)//char(48+j2)//tsn  ! Slice Coordinate Number
                if(mfld(no)==1)then  ! Hx
                        if(mdir(no)==1)then  ! x-direction slice (yz plane)
                                outname='hx'//'_x'//scn
                                open(unit=no,file=outname,status='unknown')
                                  do kk=1,nz
                                        write(no,02115)(hxs(mloc(no),jj,nz-kk+1),jj=1,ny)
                                  enddo
                                close(unit=no)
                        elseif(mdir(no)==2)then  ! y-direction slice (xz plane)
                                outname='hx'//'_y'//scn
                                open(unit=no,file=outname,status='unknown')
                                  do kk=1,nz
                                        write(no,02115)(hxs(ii,mloc(no),nz-kk+1),ii=1,nx)
                                  enddo
                                close(unit=no)
                        elseif(mdir(no)==3)then  ! z-direction slice (xy plane)
                                outname='hx'//'_z'//scn
                                open(unit=no,file=outname,status='unknown')
                                  do jj=1,ny
                                        write(no,02115)(hxs(ii,ny-jj+1,mloc(no)),ii=1,nx)
                                  enddo
                                close(unit=no)
                        endif
```

```
elseif(mfld(no)==2)then  ! Hy
        if(mdir(no)==1)then  ! x-direction slice (yz plane)
                outname='hy'//'_x'//scn
                open(unit=no,file=outname,status='unknown')
                 do kk=1,nz
                        write(no,02115)(hys(mloc(no),jj,nz-kk+1),jj=1,ny)
                 enddo
                close(unit=no)
        elseif(mdir(no)==2)then  ! y-direction slice (xz plane)
                outname='hy'//'_y'//scn
                open(unit=no,file=outname,status='unknown')
                 do kk=1,nz
                        write(no,02115)(hys(ii,mloc(no),nz-kk+1),ii=1,nx)
                 enddo
                close(unit=no)
        elseif(mdir(no)==3)then  ! z-direction slice (xy plane)
                outname='hy'//'_z'//scn
                open(unit=no,file=outname,status='unknown')
                 do jj=1,ny
                        write(no,02115)(hys(ii,ny-jj+1,mloc(no)),ii=1,nx)
                 enddo
                close(unit=no)
        endif
elseif(mfld(no)==3)then  ! Hz
        if(mdir(no)==1)then  ! x-direction slice (yz plane)
                outname='hz'//'_x'//scn
                open(unit=no,file=outname,status='unknown')
                 do kk=1,nz
                        write(no,02115)(hzs(mloc(no),jj,nz-kk+1),jj=1,ny)
                 enddo
                close(unit=no)
        elseif(mdir(no)==2)then  ! y-direction slice (xz plane)
                outname='hz'//'_y'//scn
                open(unit=no,file=outname,status='unknown')
                 do kk=1,nz
                        write(no,02115)(hzs(ii,mloc(no),nz-kk+1),ii=1,nx)
                 enddo
                close(unit=no)
        elseif(mdir(no)==3)then  ! z-direction slice (xy plane)
                outname='hz'//'_z'//scn
                open(unit=no,file=outname,status='unknown')
                 do jj=1,ny
                        write(no,02115)(hzs(ii,ny-jj+1,mloc(no)),ii=1,nx)
                 enddo
                close(unit=no)
        endif
elseif(mfld(no)==4)then  ! Ex
        if(mdir(no)==1)then  ! x-direction slice (yz plane)
                outname='ex'//'_x'//scn
                open(unit=no,file=outname,status='unknown')
                 do kk=1,nz
                        write(no,02115)(exs(mloc(no),jj,nz-kk+1),jj=1,ny)
                 enddo
                close(unit=no)
        elseif(mdir(no)==2)then  ! y-direction slice (xz plane)
                outname='ex'//'_y'//scn
                open(unit=no,file=outname,status='unknown')
                  do kk=1,nz
                          write(no,02115)(exs(ii,mloc(no),nz-kk+1),ii=1,nx)
                  enddo
                close(unit=no)
        elseif(mdir(no)==3)then  ! z-direction slice (xy plane)
                outname='ex'//'_z'//scn
                open(unit=no,file=outname,status='unknown')
                  do jj=1,ny
                          write(no,02115)(exs(ii,ny-jj+1,mloc(no)),ii=1,nx)
                  enddo
                close(unit=no)
        endif
elseif(mfld(no)==5)then  ! Ey
        if(mdir(no)==1)then  ! x-direction slice (yz plane)
                outname='ey'//'_x'//scn
                open(unit=no,file=outname,status='unknown')
                  do kk=1,nz
                          write(no,02115)(eys(mloc(no),jj,nz-kk+1),jj=1,ny)
                  enddo
                close(unit=no)
        elseif(mdir(no)==2)then  ! y-direction slice (xz plane)
                outname='ey'//'_y'//scn
                open(unit=no,file=outname,status='unknown')
                  do kk=1,nz
                          write(no,02115)(eys(ii,mloc(no),nz-kk+1),ii=1,nx)
                  enddo
                close(unit=no)
        elseif(mdir(no)==3)then  ! z-direction slice (xy plane)
                outname='ey'//'_z'//scn
                open(unit=no,file=outname,status='unknown')
                  do jj=1,ny
                          write(no,02115)(eys(ii,ny-jj+1,mloc(no)),ii=1,nx)
                  enddo
                close(unit=no)
        endif
elseif(mfld(no)==6)then  ! Ez
        if(mdir(no)==1)then  ! x-direction slice (yz plane)
                outname='ez'//'_x'//scn
                open(unit=no,file=outname,status='unknown')
                  do kk=1,nz
                          write(no,02115)(ezs(mloc(no),jj,nz-kk+1),jj=1,ny)
                  enddo
                close(unit=no)
        elseif(mdir(no)==2)then  ! y-direction slice (xz plane)
                outname='ez'//'_y'//scn
                open(unit=no,file=outname,status='unknown')
                  do kk=1,nz
                          write(no,02115)(ezs(ii,mloc(no),nz-kk+1),ii=1,nx)
                  enddo
                close(unit=no)
        elseif(mdir(no)==3)then  ! z-direction slice (xy plane)
                outname='ez'//'_z'//scn
                open(unit=no,file=outname,status='unknown')
                  do jj=1,ny
                          write(no,02115)(ezs(ii,ny-jj+1,mloc(no)),ii=1,nx)
```

```
                              enddo
                             close(unit=no)
                        endif
                   endif
              enddo


              RETURN
              END
```