Computational Methods for Kinetic Processes in Plasma Physics

Ken Nishikawa

Department of Physics/UAH

Main program 2

June 3, 2015

Main program

after the main loop data is written for diagnostics and rerun

```
            go to 7189    (skipped for diagnostics during the run)
c  partial output is stored every 100 time-step (regardless of the full data dump)
c       if(mod(nstep,100).eq.0) then
       if(nstep.eq.5 .or. mod(nstep,5000).eq.0) then
cc        if(nstep.eq.1 .or. nstep.eq.5 .or. mod(nstep,50).eq.0 .or.
cc     &       nstep.eq.470 .or. nstep.eq.485 .or.
cc     &     nstep.eq.970 .or. nstep.eq.985 ) then
c  convert "nstep" into character string
            step1 = char(int(nstep/10000.)+48)
            step2 = char(int((nstep-int(nstep/10000.)*10000)/1000.)+48)
            step3 = char(int((nstep-int(nstep/1000.)*1000)/100.)+48)
            step4 = char(int((nstep-int(nstep/100.)*100)/10.)+48)
            step5 = char(int(nstep-int(nstep/10.)*10)+48)
            step  = hyph//step1//step2//step3//step4//step5

            open(21,file=soutd//num//step,form='unformatted')
```

```
c  magnetic and electric fields
      call F_convert(bx,by,bz,ifx,ify,ifz,mFx,mFy,mFz,imax,
     &                  bxmax,bxmin,bymax,bymin,bzmax,bzmin)
      write(21) bxmax,bxmin,bymax,bymin,bzmax,bzmin
      write(21) ifx,ify,ifz
      call F_convert(ex,ey,ez,ifx,ify,ifz,mFx,mFy,mFz,imax,
     &                  exmax,exmin,eymax,eymin,ezmax,ezmin)
      write(21) exmax,exmin,eymax,eymin,ezmax,ezmin
      write(21) ifx,ify,ifz
cWR     write(21) bx,by,bz
cWR       close(21)
cWR       write(22) ex,ey,ez
cWR       close(22)
c  number density and current density for ambient ions
          write(21) ions,lecs,ionj,lecj
c       call densty(ions,rho,flx,fly,flz,mFx,mFy,mFz,
c    &              xi,yi,zi,ui,vi,wi,mb,DHDx,DHDy,DHDz)
      call densty_conv(ions,irho,ifx,ify,ifz,mFx,mFy,mFz,imax,
     &       rhomax,rhomin,fxmax,fxmin,fymax,fymin,fzmax,fzmin,
     &                    xi,yi,zi,ui,vi,wi,mb,DHDx,DHDy,DHDz)
```

```
         call veldist(ions,mb,mdiag,rselect,isel,Cvpar,Cvper,xpos,
     &                                   xi,yi,zi,ui,vi,wi,is)
         write(21) rhomax,rhomin,fxmax,fxmin,fymax,fymin,fzmax,fzmin
                write(21) irho,ifx,ify,ifz
                write(21) isel,Cvpar,Cvper,xpos
c  number density and current density for ambient electrons
c        call densty(lecs,rho,flx,fly,flz,mFx,mFy,mFz,
c     &               xe,ye,ze,ue,ve,we,mb,DHDx,DHDy,DHDz)
         call densty_conv(lecs,irho,ifx,ify,ifz,mFx,mFy,mFz,imax,
     &       rhomax,rhomin,fxmax,fxmin,fymax,fymin,fzmax,fzmin,
     &                      xe,ye,ze,ue,ve,we,mb,DHDx,DHDy,DHDz)
         call veldist(lecs,mb,mdiag,rselect,isel,Cvpar,Cvper,xpos,
     &                                   xe,ye,ze,ue,ve,we,is)
         write(21) rhomax,rhomin,fxmax,fxmin,fymax,fymin,fzmax,fzmin
                write(21) irho,ifx,ify,ifz
         write(21) isel,Cvpar,Cvper,xpos
c  number density and current density for jet ions
c ** if ionj=0 or lecj=0 the routine returns (iformation on ionj and lecj **
c ** necessary for dealing with these files                              **
         if (ionj.gt.0) then
```

```
c        call densty(ionj,rho,flx,fly,flz,mFx,mFy,mFz,
c     &              xij,yij,zij,uij,vij,wij,mj,DHDx,DHDy,DHDz)
         call densty_conv(ionj,irho,ifx,ify,ifz,mFx,mFy,mFz,imax,
     &        rhomax,rhomin,fxmax,fxmin,fymax,fymin,fzmax,fzmin,
     &              xij,yij,zij,uij,vij,wij,mj,DHDx,DHDy,DHDz)
cJET
         rseljet = rselect/0.7
              call veldist(ionj,mj,mdiag,rseljet,isel,Cvpar,Cvper,xpos,
     &                        xij,yij,zij,uij,vij,wij,is)
              write(21) rhomax,rhomin,fxmax,fxmin,fymax,fymin,fzmax,fzmin
              write(21) irho,ifx,ify,ifz
              write(21) isel,Cvpar,Cvper,xpos
              end if
c  number density and current density for jet electrons
              if (lecj.gt.0) then
c        call densty(lecj,rho,flx,fly,flz,mFx,mFy,mFz,
c     &              xej,yej,zej,uej,vej,wej,mj,DHDx,DHDy,DHDz)
         call densty_conv(lecj,irho,ifx,ify,ifz,mFx,mFy,mFz,imax,
     &        rhomax,rhomin,fxmax,fxmin,fymax,fymin,fzmax,fzmin,
     &              xej,yej,zej,uej,vej,wej,mj,DHDx,DHDy,DHDz)
```

```
cJET
        rseljet = rselect/0.7
               call veldist(lecj,mj,mdiag,rseljet,isel,Cvpar,Cvper,xpos,
     &                               xej,yej,zej,uej,vej,wej,is)
               write(21) rhomax,rhomin,fxmax,fxmin,fymax,fymin,fzmax,fzmin
               write(21) irho,ifx,ify,ifz
               write(21) isel,Cvpar,Cvper,xpos
              end if
       close(21)
c             close(24)
      end if

7189   continue         skipped to this line
c  partial output - magnetic field perturbations
cJET      if(nstep.eq.1 .or. mod(nstep,20).eq.0) then
cJET       open(26,file=nfield//num,status='old',position='append')

cJET      call bfield(bx,by,bz,b0x,c,mFx,mFy,mFz,bpar,bperp,
cJET    &           FBD_BLx,FBD_BRx,FBD_BLy,FBD_BRy,FBD_BLz,FBD_BRz)
```

```
cJET      write(26,*) nstep,bpar,bperp
cJET      close(26)
cJET      end if


c     print*,'particles',ions,lecs,ionj,lecj,' myid step',myid,nstep


      if (nstep.le.last) goto 1000
c *** MAIN LOOP ENDS HERE*****

c  dump the data
       do i = 1,4
        ipsend(1)=ions
        ipsend(2)=lecs
        ipsend(3)=ionj
        ipsend(4)=lecj
       end do
```

```
   call MPI_GATHER(ipsend,4,MPI_INTEGER,iprecv,4,MPI_INTEGER,
  &               1,lgrp,ierror)

  if (myid.eq.1) then
   j=1
        do i = 1,Nproc
         write(1,200) i-1
         write(1,201) iprecv(j),iprecv(j+1),iprecv(j+2),iprecv(j+3)
         j = j+4
   end do
        close(1)
  end if

  if(myid.eq.0) print *, 'after gather'


200   format(' myid=',i3)
201   format(' ions=',i10,' lecs=',i10,' ionj=',i10,' lecj=',i10/)
```

```
cCOL      if(last.gt.last0) nst = nst+1
      nst = nst+1

      st01 = char(int(nst/100.)+48)
      st02 = char(int((nst-int(nst/100.)*100)/10.)+48)
      st03 = char(int(nst-int(nst/10.)*10)+48)

      st0 = st01//st02//st03

c      st0 = char(int(nst/10.)+48)//char(nst-int(nst/10.)*10+48)
      st = hyph//st0

      open(7,file=strpd//num//st,form='unformatted')
      write(7) c
      write(7) ions,lecs,ionj,lecj
      write(7) PBLeft,PBRght,PBFrnt,PBRear,PBBot,PBTop
      call F_convert(bx,by,bz,ifx,ify,ifz,mFx,mFy,mFz,imax,
     &           bxmax,bxmin,bymax,bymin,bzmax,bzmin)
      write(7) bxmax,bxmin,bymax,bymin,bzmax,bzmin
      write(7) ifx,ify,ifz
```

```
      call F_convert(ex,ey,ez,ifx,ify,ifz,mFx,mFy,mFz,imax,
     &               exmax,exmin,eymax,eymin,ezmax,ezmin)
      write(7) exmax,exmin,eymax,eymin,ezmax,ezmin
      write(7) ifx,ify,ifz
c  write ambient ions
      call X_convert(ions,xi,yi,zi,ixx,iyy,izz,mb,mb,imax,
     &           PBLeft,PBRght,PBFrnt,PBRear,PBBot,PBTop)
      write(7) (ixx(i),i=1,ions),(iyy(i),i=1,ions),(izz(i),i=1,ions)
cNewJacek      call V_convert(ions,ui,vi,wi,ixx,iyy,izz,mb,mb,imax,
cNewJacek     &                   umax,umin,vmax,vmin,wmax,wmin)
cNewJacek changes V_convert into P_convert everywhere below
      call P_convert(ions,ui,vi,wi,ixx,iyy,izz,mb,mb,imax,
     &                   umax,umin,vmax,vmin,wmax,wmin,c)
      write(7) umax,umin,vmax,vmin,wmax,wmin
      write(7) (ixx(i),i=1,ions),(iyy(i),i=1,ions),(izz(i),i=1,ions)
c  write ambient electrons
      call X_convert(lecs,xe,ye,ze,ixx,iyy,izz,mb,mb,imax,
     &           PBLeft,PBRght,PBFrnt,PBRear,PBBot,PBTop)
      write(7) (ixx(i),i=1,lecs),(iyy(i),i=1,lecs),(izz(i),i=1,lecs)
```

```
      call P_convert(lecs,ue,ve,we,ixx,iyy,izz,mb,mb,imax,
     &                  umax,umin,vmax,vmin,wmax,wmin,c)
      write(7) umax,umin,vmax,vmin,wmax,wmin
      write(7) (ixx(i),i=1,lecs),(iyy(i),i=1,lecs),(izz(i),i=1,lecs)
c  write JET ions
      if (ionj.gt.0) then
       call X_convert(ionj,xij,yij,zij,ixx,iyy,izz,mj,mb,imax,
     &              PBLeft,PBRght,PBFrnt,PBRear,PBBot,PBTop)
       write(7) (ixx(i),i=1,ionj),(iyy(i),i=1,ionj),(izz(i),i=1,ionj)
       call P_convert(ionj,uij,vij,wij,ixx,iyy,izz,mj,mb,imax,
     &                    umax,umin,vmax,vmin,wmax,wmin,c)
       write(7) umax,umin,vmax,vmin,wmax,wmin
       write(7) (ixx(i),i=1,ionj),(iyy(i),i=1,ionj),(izz(i),i=1,ionj)
      end if
c  write JET electrons
      if (lecj.gt.0) then
       call X_convert(lecj,xej,yej,zej,ixx,iyy,izz,mj,mb,imax,
     &              PBLeft,PBRght,PBFrnt,PBRear,PBBot,PBTop)
       write(7) (ixx(i),i=1,lecj),(iyy(i),i=1,lecj),(izz(i),i=1,lecj)
```

```
  call P_convert(lecj,uej,vej,wej,ixx,iyy,izz,mj,mb,imax,
 &                     umax,umin,vmax,vmin,wmax,wmin,c)
   write(7) umax,umin,vmax,vmin,wmax,wmin
   write(7) (ixx(i),i=1,lecj),(iyy(i),i=1,lecj),(izz(i),i=1,lecj)
  end if

  write(7) c,DT,qi,qe,mi,me,qmi,qme,vithml,vethml,vijet,vejet,
 &        vithmj,vethmj,refli,refle,rselect,xj0,yj0,zj0,b0x,
c new Jacek
 &        b0y,e0z,
 &        dlxj,dlyj,dlzj,lyj1,lzj1,
 &        mc,mrl,mrh,mc2,mrh2,mcol,mrow,isis
c NewKen
 &        ,njskip

  write(7) GBLeft,GBRght,DHDx,DHDy,DHDz,FBD_BLx,FBD_BRx,
 &        FBD_BLy,FBD_BRy,FBD_BLz,FBD_BRz,FBD_ELx,FBD_ERx,
 &        FBD_ELy,FBD_ERy,FBD_ELz,FBD_ERz,FBDLx,FBDLy,FBDLz,
 &        FBDRx,FBDRy,FBDRz,FBDLxe,FBDRxe,FBDLxp,FBDRxp,
 &        PVLeft,PVRght,nsmooth,sm1,sm2,sm3,nfilt,nstep

  close(7)
```

```
 9999  call MPI_FINALIZE(ierror)
cCOL      return
       end
c  ****************************************************************
C  Data for smoothing: the currents fed into Maxwell's equations
C  are smoothed by convolving with the sequence  .25, .5, .25 in
C  each dimension. Generate array "sm" of the 27 weights.
      subroutine smoother(sm)

      dimension sm(-1:1,-1:1,-1:1)

      do nz = -1,1
       do ny = -1,1
              do nx = -1,1
               sm(nx,ny,nz)=0.015625*(2-nx*nx)*(2-ny*ny)*(2-nz*nz)
        end do
       end do
      end do

      return
      end
```

```
c  ******************************************************************
C  Data for smoothing for arbitrary digital filter profile
      subroutine smoother1(sm,ww)

      dimension sm(-1:1,-1:1,-1:1)

      den=1.0+2.0*ww
      factor=1.0/(den*den*den)

      do nz = -1,1
       do ny = -1,1
              do nx = -1,1
               sm(nx,ny,nz)=factor*(1.+(ww-1.0)*nx*nx)*(1.+(ww-1.0)*ny*ny)
     &                          *(1.+(ww-1.0)*nz*nz)
        end do
       end do
      end do

      return
      end
```

```
c *********************************************************************
c  initialize the fields, typically to uniform components, such as
c  the uniform magnetic field parallel to the x-axis
c new attention
      subroutine Field_init(bx,by,bz,ex,ey,ez,dex,dey,dez,
     &          mFx,mFy,mFz,b0x,b0y,e0z,c)

      dimension ex(mFx,mFy,mFz),ey(mFx,mFy,mFz),ez(mFx,mFy,mFz)
      dimension bx(mFx,mFy,mFz),by(mFx,mFy,mFz),bz(mFx,mFy,mFz)
      dimension dex(mFx,mFy,mFz),dey(mFx,mFy,mFz),dez(mFx,mFy,mFz)

      do k = 1,mFz
       do j = 1,mFy
         do i = 1,mFx
          ex(i,j,k)=0.0
        ey(i,j,k)=0.0
c new attention
```

```
c new Jacek
c        if(i.eq.25) then
c              ez(i,j,k)=e0z
c        else
              ez(i,j,k)=0.0
c        endif
        bx(i,j,k)=b0x*c
c new attention
c new Jacek
c        if(i.eq.25) then
c        by(i,j,k)=b0y
c        else
        by(i,j,k)=0.0
c        endif
              bz(i,j,k)=0.0
              dex(i,j,k) = 0.0
              dey(i,j,k) = 0.0
              dez(i,j,k) = 0.0
       end do
```

```
end do
end do
```

$$b_x^{new}(i,j,k) = b_x^{old}(i,j,k) \\ \qquad +c[\, e_y(i,j,k+1) - e_y(i,j,k) - e_z(i,j+1,k) + e_z(i,j,k)\,].$$

subroutine B_field_push(bx,by,bz,ex,ey,ez,mFx,mFy,mFz,DT,c

$$b_y^{new}(i,j,k) = b_y^{old}(i,j,k) \\ \qquad +c[\, e_z(i+1,j,k) - e_z(i,j,k) - e_x(i,j,k+1) + e_x(i,j,k)\,],$$

integer FBD_BLx,FBD_BLy,FBD_BLz

$$b_z^{new}(i,j,k) = b_z^{old}(i,j,k) \\ \qquad +c[\, e_x(i,j+1,k) - e_x(i,j,k) - e_y(i+1,j,k) + e_y(i,j,k)].$$

```
do k = FBD_BLz,FBD_BRz
 do j = FBD_BLy,FBD_BRy
  do i = FBD_BLx,FBD_BRx
          bx(i,j,k)=bx(i,j,k) + DT*(0.5*c)*
&         (ey(i,j,k+1)-ey(i,j,k)-ez(i,j+1,k)+ez(i,j,k))
```

```
              by(i,j,k)=by(i,j,k) + DT*(0.5*c)*
     &        (ez(i+1,j,k)-ez(i,j,k)-ex(i,j,k+1)+ex(i,j,k))
              bz(i,j,k)=bz(i,j,k) + DT*(0.5*c)*
     &        (ex(i,j+1,k)-ex(i,j,k)-ey(i+1,j,k)+ey(i,j,k))
      end do
      end do
      end do

      return
      end
c  **************************************************************
      subroutine B_field_push4(bx,by,bz,ex,ey,ez,mFx,mFy,mFz,DT,
     &    c,FBD_BLx,FBD_BRx,FBD_BLy,FBD_BRy,FBD_BLz,FBD_BRz,
     &                          dims,coords)

      integer FBD_BRx,FBD_BRy,FBD_BRz
      integer FBD_BLx,FBD_BLy,FBD_BLz
      integer dims(3),coords(3)
```

```
   dimension ex(mFx,mFy,mFz),ey(mFx,mFy,mFz),ez(mFx,mFy,mFz)
   dimension bx(mFx,mFy,mFz),by(mFx,mFy,mFz),bz(mFx,mFy,mFz)

   if(dims(1).eq.1)then
   i=1
   do k = FBD_BLz,FBD_BRz
   do j = FBD_BLy,FBD_BRy
          bx(i,j,k)=bx(i,j,k) + DT*(0.5*c)*
&         (ey(i,j,k+1)-ey(i,j,k)-ez(i,j+1,k)+ez(i,j,k))
          by(i,j,k)=by(i,j,k) + DT*(0.5*c)*
&         (ez(i+1,j,k)-ez(i,j,k)-ex(i,j,k+1)+ex(i,j,k))
          bz(i,j,k)=bz(i,j,k) + DT*(0.5*c)*
&         (ex(i,j+1,k)-ex(i,j,k)-ey(i+1,j,k)+ey(i,j,k))
   end do
   end do

   do k = FBD_BLz,FBD_BRz
   do j = FBD_BLy,FBD_BRy
   do i = FBD_BLx+1,FBD_BRx-1
          bx(i,j,k)=bx(i,j,k) + DT*(0.5*c)*
```

```
&     (1.125*(ey(i,j,k+1)-ey(i,j,k)-ez(i,j+1,k)+ez(i,j,k))
&   -(ey(i,j,k+2)-ey(i,j,k-1)-ez(i,j+2,k)+ez(i,j-1,k))/24.)
   by(i,j,k)=by(i,j,k) + DT*(0.5*c)*
&     (1.125*(ez(i+1,j,k)-ez(i,j,k)-ex(i,j,k+1)+ex(i,j,k))
&   -(ez(i+2,j,k)-ez(i-1,j,k)-ex(i,j,k+2)+ex(i,j,k-1))/24.)
            bz(i,j,k)=bz(i,j,k) + DT*(0.5*c)*
&     (1.125*(ex(i,j+1,k)-ex(i,j,k)-ey(i+1,j,k)+ey(i,j,k))
&   -(ex(i,j+2,k)-ex(i,j-1,k)-ey(i+2,j,k)+ey(i-1,j,k))/24.)
  end do
  end do
  end do

  i=FBD_BRx
  do k = FBD_BLz,FBD_BRz
  do j = FBD_BLy,FBD_BRy
            bx(i,j,k)=bx(i,j,k) + DT*(0.5*c)*
&           (ey(i,j,k+1)-ey(i,j,k)-ez(i,j+1,k)+ez(i,j,k))
   by(i,j,k)=by(i,j,k) + DT*(0.5*c)*
&           (ez(i+1,j,k)-ez(i,j,k)-ex(i,j,k+1)+ex(i,j,k))
            bz(i,j,k)=bz(i,j,k) + DT*(0.5*c)*
&           (ex(i,j+1,k)-ex(i,j,k)-ey(i+1,j,k)+ey(i,j,k))
```

```
  end do
  end do

  else

   if(coords(1).eq.0)then
   i=1
   do k = FBD_BLz,FBD_BRz
   do j = FBD_BLy,FBD_BRy
             bx(i,j,k)=bx(i,j,k) + DT*(0.5*c)*
&            (ey(i,j,k+1)-ey(i,j,k)-ez(i,j+1,k)+ez(i,j,k))
    by(i,j,k)=by(i,j,k) + DT*(0.5*c)*
&            (ez(i+1,j,k)-ez(i,j,k)-ex(i,j,k+1)+ex(i,j,k))
             bz(i,j,k)=bz(i,j,k) + DT*(0.5*c)*
&            (ex(i,j+1,k)-ex(i,j,k)-ey(i+1,j,k)+ey(i,j,k))
   end do
   end do
```

```
 do k = FBD_BLz,FBD_BRz
 do j = FBD_BLy,FBD_BRy
  do i = FBD_BLx+1,FBD_BRx
             bx(i,j,k)=bx(i,j,k) + DT*(0.5*c)*
&      (1.125*(ey(i,j,k+1)-ey(i,j,k)-ez(i,j+1,k)+ez(i,j,k))
&   -(ey(i,j,k+2)-ey(i,j,k-1)-ez(i,j+2,k)+ez(i,j-1,k))/24.)
   by(i,j,k)=by(i,j,k) + DT*(0.5*c)*
&      (1.125*(ez(i+1,j,k)-ez(i,j,k)-ex(i,j,k+1)+ex(i,j,k))
&   -(ez(i+2,j,k)-ez(i-1,j,k)-ex(i,j,k+2)+ex(i,j,k-1))/24.)
             bz(i,j,k)=bz(i,j,k) + DT*(0.5*c)*
&      (1.125*(ex(i,j+1,k)-ex(i,j,k)-ey(i+1,j,k)+ey(i,j,k))
&   -(ex(i,j+2,k)-ex(i,j-1,k)-ey(i+2,j,k)+ey(i-1,j,k))/24.)
  end do
 end do
end do

else if(coords(1).eq.(dims(1)-1))then
do k = FBD_BLz,FBD_BRz
 do j = FBD_BLy,FBD_BRy
  do i = FBD_BLx,FBD_BRx-1
```

```
              bx(i,j,k)=bx(i,j,k) + DT*(0.5*c)*
&     (1.125*(ey(i,j,k+1)-ey(i,j,k)-ez(i,j+1,k)+ez(i,j,k))
&   -(ey(i,j,k+2)-ey(i,j,k-1)-ez(i,j+2,k)+ez(i,j-1,k))/24.)
   by(i,j,k)=by(i,j,k) + DT*(0.5*c)*
&     (1.125*(ez(i+1,j,k)-ez(i,j,k)-ex(i,j,k+1)+ex(i,j,k))
&   -(ez(i+2,j,k)-ez(i-1,j,k)-ex(i,j,k+2)+ex(i,j,k-1))/24.)
              bz(i,j,k)=bz(i,j,k) + DT*(0.5*c)*
&     (1.125*(ex(i,j+1,k)-ex(i,j,k)-ey(i+1,j,k)+ey(i,j,k))
&   -(ex(i,j+2,k)-ex(i,j-1,k)-ey(i+2,j,k)+ey(i-1,j,k))/24.)
  end do
  end do
  end do

  i=FBD_BRx
  do k = FBD_BLz,FBD_BRz
  do j = FBD_BLy,FBD_BRy
              bx(i,j,k)=bx(i,j,k) + DT*(0.5*c)*
&           (ey(i,j,k+1)-ey(i,j,k)-ez(i,j+1,k)+ez(i,j,k))
   by(i,j,k)=by(i,j,k) + DT*(0.5*c)*
&           (ez(i+1,j,k)-ez(i,j,k)-ex(i,j,k+1)+ex(i,j,k))
```

```
            bz(i,j,k)=bz(i,j,k) + DT*(0.5*c)*
&           (ex(i,j+1,k)-ex(i,j,k)-ey(i+1,j,k)+ey(i,j,k))
  end do
  end do

 else
 do k = FBD_BLz,FBD_BRz
 do j = FBD_BLy,FBD_BRy
  do i = FBD_BLx,FBD_BRx
            bx(i,j,k)=bx(i,j,k) + DT*(0.5*c)*
&     (1.125*(ey(i,j,k+1)-ey(i,j,k)-ez(i,j+1,k)+ez(i,j,k))
&   -(ey(i,j,k+2)-ey(i,j,k-1)-ez(i,j+2,k)+ez(i,j-1,k))/24.)
   by(i,j,k)=by(i,j,k) + DT*(0.5*c)*
&     (1.125*(ez(i+1,j,k)-ez(i,j,k)-ex(i,j,k+1)+ex(i,j,k))
&   -(ez(i+2,j,k)-ez(i-1,j,k)-ex(i,j,k+2)+ex(i,j,k-1))/24.)
            bz(i,j,k)=bz(i,j,k) + DT*(0.5*c)*
&     (1.125*(ex(i,j+1,k)-ex(i,j,k)-ey(i+1,j,k)+ey(i,j,k))
&   -(ex(i,j+2,k)-ex(i,j-1,k)-ey(i+2,j,k)+ey(i-1,j,k))/24.)
  end do
  end do
 end do
```

```
      end if

      end if

      return
      end
c new attention
c  ************************************************************
      subroutine B_field_boun(bx,by,bz,ex,ey,ez,mFx,mFy,mFz,DT,c,
     &  b0x,b0y,e0z,FBD_BLx,FBD_BRx,FBD_BLy,FBD_BRy,FBD_BLz,FBD_BRz)

      integer FBD_BRx,FBD_BRy,FBD_BRz
      integer FBD_BLx,FBD_BLy,FBD_BLz

      dimension ex(mFx,mFy,mFz),ey(mFx,mFy,mFz),ez(mFx,mFy,mFz)
      dimension bx(mFx,mFy,mFz),by(mFx,mFy,mFz),bz(mFx,mFy,mFz)

c new Jacek
      do k = FBD_BLz-1,FBD_BRz+1
      do j = FBD_BLy-1,FBD_BRy+1
```

```
c       do i = FBD_BLx,FBD_BRx
c        bx(i,j,k)=bx(i,j,k) + DT*(0.5*c)*
c    &            (ey(i,j,k+1)-ey(i,j,k)-ez(i,j+1,k)+ez(i,j,k))
c NewKen
         by(24,j,k)=by(24,j,k) + b0y
         by(25,j,k)=by(25,j,k) + b0y
         by(26,j,k)=by(26,j,k) + b0y
c    &            (ez(i+1,j,k)-ez(i,j,k)-ex(i,j,k+1)+ex(i,j,k))
c NewKenw
         ez(24,j,k)=ez(24,j,k) + e0z
         ez(25,j,k)=ez(25,j,k) + e0z
         ez(26,j,k)=ez(26,j,k) + e0z
c    &            (ex(i,j+1,k)-ex(i,j,k)-ey(i+1,j,k)+ey(i,j,k))
c       end do
       end do
      end do

      return
      end
```

```
c  ********************************************************************
      subroutine E_field_push(bx,by,bz,ex,ey,ez,mFx,mFy,mFz,DT,c,
     &           FBD_ELx,FBD_ERx,FBD_ELy,FBD_ERy,FBD_ELz,FBD_ERz)

      integer FBD_ERx,FBD_ERy,FBD_ERz
      integer FBD_ELx,FBD_ELy,FBD_ELz
```

$$
\begin{aligned}
e_x^{new}(i,j,k) = {} & e_x^{old}(i,j,k) \\
& +c[\, b_y(i,j,k-1) - b_y(i,j,k) - b_z(i,j-1,k) + b_z(i,j,k)\,],
\end{aligned}
$$

```
      do k = FBD_ELz,FBD_ERz
       do j = FBD_ELy,FBD_ERy
        do i = FBD_ELx,FBD_ERx
         ex(i,j,k)=ex(i,j,k) + DT*c*
     &           (by(i,j,k-1)-by(i,j,k)-bz(i,j-1,k)+bz(i,j,k))
         ey(i,j,k)=ey(i,j,k) + DT*c*
     &           (bz(i-1,j,k)-bz(i,j,k)-bx(i,j,k-1)+bx(i,j,k))
                ez(i,j,k)=ez(i,j,k) + DT*c*
     &           (bx(i,j-1,k)-bx(i,j,k)-by(i-1,j,k)+by(i,j,k))
        end do
```

```
      end do
      end do

      return
      end
c  ****************************************************************
      subroutine E_field_push4(bx,by,bz,ex,ey,ez,mFx,mFy,mFz,DT,
     &        c,FBD_ELx,FBD_ERx,FBD_ELy,FBD_ERy,FBD_ELz,FBD_ERz,
     &                                    dims,coords)

      integer FBD_ERx,FBD_ERy,FBD_ERz
      integer FBD_ELx,FBD_ELy,FBD_ELz
      integer dims(3),coords(3)

      dimension ex(mFx,mFy,mFz),ey(mFx,mFy,mFz),ez(mFx,mFy,mFz)
      dimension bx(mFx,mFy,mFz),by(mFx,mFy,mFz),bz(mFx,mFy,mFz)

      if(dims(1).eq.1) then
       i=2
```

```
   do k = FBD_ELz,FBD_ERz
   do j = FBD_ELy,FBD_ERy
    ex(i,j,k)=ex(i,j,k) + DT*c*
&            (by(i,j,k-1)-by(i,j,k)-bz(i,j-1,k)+bz(i,j,k))
    ey(i,j,k)=ey(i,j,k) + DT*c*
&            (bz(i-1,j,k)-bz(i,j,k)-bx(i,j,k-1)+bx(i,j,k))
             ez(i,j,k)=ez(i,j,k) + DT*c*
&            (bx(i,j-1,k)-bx(i,j,k)-by(i-1,j,k)+by(i,j,k))
   end do
  end do

 do k = FBD_ELz,FBD_ERz
  do j = FBD_ELy,FBD_ERy
   do i = FBD_ELx+1,FBD_ERx-1
    ex(i,j,k)=ex(i,j,k) + DT*c*
&       (1.125*(by(i,j,k-1)-by(i,j,k)-bz(i,j-1,k)+bz(i,j,k))
&   -(by(i,j,k-2)-by(i,j,k+1)-bz(i,j-2,k)+bz(i,j+1,k))/24.)
    ey(i,j,k)=ey(i,j,k) + DT*c*
&       (1.125*(bz(i-1,j,k)-bz(i,j,k)-bx(i,j,k-1)+bx(i,j,k))
&   -(bz(i-2,j,k)-bz(i+1,j,k)-bx(i,j,k-2)+bx(i,j,k+1))/24.)
```

```
              ez(i,j,k)=ez(i,j,k) + DT*c*
&      (1.125*(bx(i,j-1,k)-bx(i,j,k)-by(i-1,j,k)+by(i,j,k))
&    -(bx(i,j-2,k)-bx(i,j+1,k)-by(i-2,j,k)+by(i+1,j,k))/24.)
   end do
  end do
 end do

  i=FBD_ERx
  do k = FBD_ELz,FBD_ERz
   do j = FBD_ELy,FBD_ERy
    ex(i,j,k)=ex(i,j,k) + DT*c*
&           (by(i,j,k-1)-by(i,j,k)-bz(i,j-1,k)+bz(i,j,k))
    ey(i,j,k)=ey(i,j,k) + DT*c*
&           (bz(i-1,j,k)-bz(i,j,k)-bx(i,j,k-1)+bx(i,j,k))
              ez(i,j,k)=ez(i,j,k) + DT*c*
&           (bx(i,j-1,k)-bx(i,j,k)-by(i-1,j,k)+by(i,j,k))
   end do
  end do
```

```
 else

if(coords(1).eq.0)then
 i=2
 do k = FBD_ELz,FBD_ERz
  do j = FBD_ELy,FBD_ERy
   ex(i,j,k)=ex(i,j,k) + DT*c*
&          (by(i,j,k-1)-by(i,j,k)-bz(i,j-1,k)+bz(i,j,k))
   ey(i,j,k)=ey(i,j,k) + DT*c*
&          (bz(i-1,j,k)-bz(i,j,k)-bx(i,j,k-1)+bx(i,j,k))
            ez(i,j,k)=ez(i,j,k) + DT*c*
&          (bx(i,j-1,k)-bx(i,j,k)-by(i-1,j,k)+by(i,j,k))
  end do
 end do

do k = FBD_ELz,FBD_ERz
 do j = FBD_ELy,FBD_ERy
  do i = FBD_ELx+1,FBD_ERx
   ex(i,j,k)=ex(i,j,k) + DT*c*
&      (1.125*(by(i,j,k-1)-by(i,j,k)-bz(i,j-1,k)+bz(i,j,k))
&   -(by(i,j,k-2)-by(i,j,k+1)-bz(i,j-2,k)+bz(i,j+1,k))/24.)
```

```
              ey(i,j,k)=ey(i,j,k) + DT*c*
&      (1.125*(bz(i-1,j,k)-bz(i,j,k)-bx(i,j,k-1)+bx(i,j,k))
&    -(bz(i-2,j,k)-bz(i+1,j,k)-bx(i,j,k-2)+bx(i,j,k+1))/24.)
              ez(i,j,k)=ez(i,j,k) + DT*c*
&      (1.125*(bx(i,j-1,k)-bx(i,j,k)-by(i-1,j,k)+by(i,j,k))
&    -(bx(i,j-2,k)-bx(i,j+1,k)-by(i-2,j,k)+by(i+1,j,k))/24.)
   end do
  end do
 end do
 else if(coords(1).eq.(dims(1)-1))then
 do k = FBD_ELz,FBD_ERz
  do j = FBD_ELy,FBD_ERy
   do i = FBD_ELx,FBD_ERx-1
    ex(i,j,k)=ex(i,j,k) + DT*c*
&      (1.125*(by(i,j,k-1)-by(i,j,k)-bz(i,j-1,k)+bz(i,j,k))
&    -(by(i,j,k-2)-by(i,j,k+1)-bz(i,j-2,k)+bz(i,j+1,k))/24.)
    ey(i,j,k)=ey(i,j,k) + DT*c*
&      (1.125*(bz(i-1,j,k)-bz(i,j,k)-bx(i,j,k-1)+bx(i,j,k))
&    -(bz(i-2,j,k)-bz(i+1,j,k)-bx(i,j,k-2)+bx(i,j,k+1))/24.)
```

```
          ez(i,j,k)=ez(i,j,k) + DT*c*
&     (1.125*(bx(i,j-1,k)-bx(i,j,k)-by(i-1,j,k)+by(i,j,k))
&   -(bx(i,j-2,k)-bx(i,j+1,k)-by(i-2,j,k)+by(i+1,j,k))/24.)
   end do
  end do
 end do

 i=FBD_ERx
  do k = FBD_ELz,FBD_ERz
   do j = FBD_ELy,FBD_ERy
    ex(i,j,k)=ex(i,j,k) + DT*c*
&          (by(i,j,k-1)-by(i,j,k)-bz(i,j-1,k)+bz(i,j,k))
    ey(i,j,k)=ey(i,j,k) + DT*c*
&          (bz(i-1,j,k)-bz(i,j,k)-bx(i,j,k-1)+bx(i,j,k))
          ez(i,j,k)=ez(i,j,k) + DT*c*
&          (bx(i,j-1,k)-bx(i,j,k)-by(i-1,j,k)+by(i,j,k))
   end do
  end do
```

```
  else
 do k = FBD_ELz,FBD_ERz
  do j = FBD_ELy,FBD_ERy
   do i = FBD_ELx,FBD_ERx
    ex(i,j,k)=ex(i,j,k) + DT*c*
&       (1.125*(by(i,j,k-1)-by(i,j,k)-bz(i,j-1,k)+bz(i,j,k))
&    -(by(i,j,k-2)-by(i,j,k+1)-bz(i,j-2,k)+bz(i,j+1,k))/24.)
    ey(i,j,k)=ey(i,j,k) + DT*c*
&       (1.125*(bz(i-1,j,k)-bz(i,j,k)-bx(i,j,k-1)+bx(i,j,k))
&    -(bz(i-2,j,k)-bz(i+1,j,k)-bx(i,j,k-2)+bx(i,j,k+1))/24.)
              ez(i,j,k)=ez(i,j,k) + DT*c*
&       (1.125*(bx(i,j-1,k)-bx(i,j,k)-by(i-1,j,k)+by(i,j,k))
&    -(bx(i,j-2,k)-bx(i,j+1,k)-by(i-2,j,k)+by(i+1,j,k))/24.)
   end do
  end do
 end do
 end if

 end if

 return
 end
```

```
c  ********************************************************************
c !! changes made compared to 1D parallel version: see comments !!
      subroutine Surface_Byzx(bx,by,bz,ex,ey,ez,mFx,mFy,mFz,DT,c,
     &                    FBD_BLy,FBD_BRy,FBD_BLz,FBD_BRz)

      integer FBD_BRy,FBD_BRz
      integer FBD_BLy,FBD_BLz

      dimension ex(mFx,mFy,mFz),ey(mFx,mFy,mFz),ez(mFx,mFy,mFz)
      dimension bx(mFx,mFy,mFz),by(mFx,mFy,mFz),bz(mFx,mFy,mFz)

c  update Right layer of B-field on VIRTUAL box
c ** j and k field elements updated within regular domain limits: 3,nFi+2 **
      i = mFx

c  time-step must be incorporated!
      cdt = c*DT

      rs=2.*cdt/(1.+cdt)
      s=.4142136
      os=.5*(1.-s)*rs
```

```
c ** here k and j start from k=j=2 because update of By and Bz requires **
c ** Bx(mFx,2,k) and Bx(mFx,j,2) elements, respectively;              **
c ** all E-field elements necessary for Bx push are accessible !!!      **
      do k = FBD_BLz-1,FBD_BRz
       do j = FBD_BLy-1,FBD_BRy
             bx(i,j,k)=bx(i,j,k)
     &       +.5*cdt*(ey(i,j,k+1)-ey(i,j,k)
     &            -ez(i,j+1,k)+ez(i,j,k))
       end do
      end do

c ** k-loop is done separately (compared to combined j-k-loops in 1D parallel  **
c ** version) because access to By(mFx-1,3,2) requires unnecessary communication*
      do k = FBD_BLz,FBD_BRz
       do j = FBD_BLy,FBD_BRy
             by(i,j,k)=by(i,j,k)
     &      +rs*(by(i-1,j,k)-by(i,j,k)
     &          +s*(bx(i,j,k)-bx(i,j-1,k)))
     &      -os*(ex(i,j,k+1)-ex(i,j,k))
```

```
&        -(os-cdt)*(ex(i-1,j,k+1)-ex(i-1,j,k))
&             -cdt*(ez(i,j,k)-ez(i-1,j,k))
  end do
 end do

 do j = FBD_BLy,FBD_BRy
  do k = FBD_BLz,FBD_BRz
            bz(i,j,k)=bz(i,j,k)
&        +rs*(bz(i-1,j,k)-bz(i,j,k)
&              +s*(bx(i,j,k)-bx(i,j,k-1)))
&        +os*(ex(i,j+1,k)-ex(i,j,k))
&        +(os-cdt)*(ex(i-1,j+1,k)-ex(i-1,j,k))
&        +cdt*(ey(i,j,k)-ey(i-1,j,k))
  end do

c ** second-half advance of Bx: regular j and k limits - 2 and nFi+3 elements **
c ** obtained from field communication                                       **
  do k = FBD_BLz,FBD_BRz
            bx(i,j,k)=bx(i,j,k)
&          +.5*cdt*(ey(i,j,k+1)-ey(i,j,k)
&                  -ez(i,j+1,k)+ez(i,j,k))
```

```
      end do
      end do

      return
      end
c ********************************************************************
      subroutine Surface_Eyzx(bx,by,bz,ex,ey,ez,mFx,mFy,mFz,DT,c,
     &                        FBD_ELy,FBD_ERy,FBD_ELz,FBD_ERz)

      integer FBD_ERy,FBD_ERz
      integer FBD_ELy,FBD_ELz

      dimension ex(mFx,mFy,mFz),ey(mFx,mFy,mFz),ez(mFx,mFy,mFz)
      dimension bx(mFx,mFy,mFz),by(mFx,mFy,mFz),bz(mFx,mFy,mFz)

c  rear layer of E-field on VIRTUAL array
      i = 1

c  time-step must be incorporated!
      cdt = c*DT
```

```
 rs=2.*cdt/(1.+cdt)
 s=.4142136
 os=.5*(1.-s)*rs

 do k = FBD_ELz,FBD_ERz+1
  do j = FBD_ELy,FBD_ERy+1
             ex(i,j,k)=ex(i,j,k)
&            +.5*cdt*(by(i,j,k-1)-by(i,j,k)
&                    -bz(i,j-1,k)+bz(i,j,k))
  end do
 end do

 do k = FBD_ELz,FBD_ERz
  do j = FBD_ELy,FBD_ERy
             ey(i,j,k)=ey(i,j,k)
&           +rs*(ey(i+1,j,k)-ey(i,j,k)
&                 +s*(ex(i,j,k)-ex(i,j+1,k)))
&           -os*(bx(i,j,k-1)-bx(i,j,k))
&           -(os-cdt)*(bx(i+1,j,k-1)-bx(i+1,j,k))
&               -cdt*(bz(i,j,k)-bz(i+1,j,k))
```

```
 end do
 end do

 do j = FBD_ELy,FBD_ERy
  do k = FBD_ELz,FBD_ERz
             ez(i,j,k)=ez(i,j,k)
&           +rs*(ez(i+1,j,k)-ez(i,j,k)
&                 +s*(ex(i,j,k)-ex(i,j,k+1)))
&           +os*(bx(i,j-1,k)-bx(i,j,k))
&           +(os-cdt)*(bx(i+1,j-1,k)-bx(i+1,j,k))
&          +cdt*(by(i,j,k)-by(i+1,j,k))
  end do

  do k = FBD_ELz,FBD_ERz
             ex(i,j,k)=ex(i,j,k)
&             +.5*cdt*(by(i,j,k-1)-by(i,j,k)
&                   -bz(i,j-1,k)+bz(i,j,k))
  end do
 end do

 return
 end
```

Main program (continued)

```
c  ***************************************************************
      subroutine mover2(ipar,x,y,z,u,v,w,mh,ex,ey,ez,bx,by,bz,
     &                mFx,mFy,mFz,DHDx,DHDy,DHDz,qm,DT,c)

      dimension ex(mFx,mFy,mFz),ey(mFx,mFy,mFz),ez(mFx,mFy,mFz)
      dimension bx(mFx,mFy,mFz),by(mFx,mFy,mFz),bz(mFx,mFy,mFz)
      dimension x(mh),y(mh),z(mh)
      dimension u(mh),v(mh),w(mh)

      dimension wi(-1:1),wj(-1:1),wk(-1:1)

       DO n = 1,ipar
c  cell index and displacement in cell
             i  = nint(x(n)) - DHDx
             dx = x(n)-i -DHDx
       j  = nint(y(n)) - DHDy
             dy = y(n)-j -DHDy
       k  = nint(z(n)) - DHDz
             dz = z(n)-k -DHDz
```

```
wi(-1)= 0.5*(0.5-dx)*(0.5-dx)
       wi(0) = 0.75-dx*dx
       wi(1) = 0.5*(0.5+dx)*(0.5+dx)

wi02 = wi(0)+wi(1)
wi01 = wi(0)+wi(-1)

       wj(-1)= 0.5*(0.5-dy)*(0.5-dy)
       wj(0) = 0.75-dy*dy
       wj(1) = 0.5*(0.5+dy)*(0.5+dy)

wj02 = wj(0)+wj(1)
wj01 = wj(0)+wj(-1)

      wk(-1)= 0.5*(0.5-dz)*(0.5-dz)
      wk(0) = 0.75-dz*dz
      wk(1) = 0.5*(0.5+dz)*(0.5+dz)

wk02 = wk(0)+wk(1)
wk01 = wk(0)+wk(-1)
```

```
c  0 - (xp,j,k),  1 - (xp,j-1,k-1), 2 - (xp,j+1,k+1); e.g. f12 - ex(xp,j-1,k+1)
   f00=      wi02*ex(i,j,k)     +  wi01*ex(i-1,j,k)
  &     + wi(-1)*ex(i-2,j,k)     + wi(1)*ex(i+1,j,k)
   f10=      wi02*ex(i,j-1,k)   +  wi01*ex(i-1,j-1,k)
  &     + wi(-1)*ex(i-2,j-1,k)   + wi(1)*ex(i+1,j-1,k)
   f20=      wi02*ex(i,j+1,k)   +  wi01*ex(i-1,j+1,k)
  &     + wi(-1)*ex(i-2,j+1,k)   + wi(1)*ex(i+1,j+1,k)

    f0 = wj(-1)*f10 + wj(0)*f00 + wj(1)*f20

   f01=      wi02*ex(i,j,k-1)   +  wi01*ex(i-1,j,k-1)
  &     + wi(-1)*ex(i-2,j,k-1)   + wi(1)*ex(i+1,j,k-1)
   f11=      wi02*ex(i,j-1,k-1) +  wi01*ex(i-1,j-1,k-1)
  &     + wi(-1)*ex(i-2,j-1,k-1) + wi(1)*ex(i+1,j-1,k-1)
   f21=      wi02*ex(i,j+1,k-1) +  wi01*ex(i-1,j+1,k-1)
  &     + wi(-1)*ex(i-2,j+1,k-1) + wi(1)*ex(i+1,j+1,k-1)

    f1 = wj(-1)*f11 + wj(0)*f01 + wj(1)*f21
```

```
 f02=     wi02*ex(i,j,k+1)   +  wi01*ex(i-1,j,k+1)
&     + wi(-1)*ex(i-2,j,k+1)   + wi(1)*ex(i+1,j,k+1)
 f12=     wi02*ex(i,j-1,k+1) +  wi01*ex(i-1,j-1,k+1)
&     + wi(-1)*ex(i-2,j-1,k+1) + wi(1)*ex(i+1,j-1,k+1)
 f22=     wi02*ex(i,j+1,k+1) +  wi01*ex(i-1,j+1,k+1)
&     + wi(-1)*ex(i-2,j+1,k+1) + wi(1)*ex(i+1,j+1,k+1)

  f2 = wj(-1)*f12 + wj(0)*f02 + wj(1)*f22

 ex0 = (wk(-1)*f1 + wk(0)*f0 + wk(1)*f2)*0.25*qm


 f00=     wj02*ey(i,j,k)     +  wj01*ey(i,j-1,k)
&     + wj(-1)*ey(i,j-2,k)     + wj(1)*ey(i,j+1,k)
 f10=     wj02*ey(i,j,k-1)   +  wj01*ey(i,j-1,k-1)
&     + wj(-1)*ey(i,j-2,k-1)   + wj(1)*ey(i,j+1,k-1)
 f20=     wj02*ey(i,j,k+1)   +  wj01*ey(i,j-1,k+1)
&     + wj(-1)*ey(i,j-2,k+1)   + wj(1)*ey(i,j+1,k+1)

  f0 = wk(-1)*f10 + wk(0)*f00 + wk(1)*f20
```

```
 f01=     wj02*ey(i-1,j,k)   +  wj01*ey(i-1,j-1,k)
&     + wj(-1)*ey(i-1,j-2,k)   + wj(1)*ey(i-1,j+1,k)
 f11=     wj02*ey(i-1,j,k-1) +  wj01*ey(i-1,j-1,k-1)
&    +  wj(-1)*ey(i-1,j-2,k-1) + wj(1)*ey(i-1,j+1,k-1)
 f21=     wj02*ey(i-1,j,k+1) +  wj01*ey(i-1,j-1,k+1)
&     + wj(-1)*ey(i-1,j-2,k+1) + wj(1)*ey(i-1,j+1,k+1)

  f1 = wk(-1)*f11 + wk(0)*f01 + wk(1)*f21

 f02=     wj02*ey(i+1,j,k)   +  wj01*ey(i+1,j-1,k)
&     + wj(-1)*ey(i+1,j-2,k)   + wj(1)*ey(i+1,j+1,k)
 f12=     wj02*ey(i+1,j,k-1) +  wj01*ey(i+1,j-1,k-1)
&     + wj(-1)*ey(i+1,j-2,k-1) + wj(1)*ey(i+1,j+1,k-1)
 f22=     wj02*ey(i+1,j,k+1) +  wj01*ey(i+1,j-1,k+1)
&     + wj(-1)*ey(i+1,j-2,k+1) + wj(1)*ey(i+1,j+1,k+1)

  f2 = wk(-1)*f12 + wk(0)*f02 + wk(1)*f22

 ey0 = (wi(-1)*f1 + wi(0)*f0 + wi(1)*f2)*0.25*qm
```

```
  f00=    wk02*ez(i,j,k)       +  wk01*ez(i,j,k-1)
&     + wk(-1)*ez(i,j,k-2)     + wk(1)*ez(i,j,k+1)
  f10=    wk02*ez(i-1,j,k)     +  wk01*ez(i-1,j,k-1)
&     + wk(-1)*ez(i-1,j,k-2)   + wk(1)*ez(i-1,j,k+1)
  f20=    wk02*ez(i+1,j,k)     +  wk01*ez(i+1,j,k-1)
&     + wk(-1)*ez(i+1,j,k-2)   + wk(1)*ez(i+1,j,k+1)

   f0 = wi(-1)*f10 + wi(0)*f00 + wi(1)*f20

  f01=    wk02*ez(i,j-1,k)     +  wk01*ez(i,j-1,k-1)
&     + wk(-1)*ez(i,j-1,k-2)   + wk(1)*ez(i,j-1,k+1)
  f11=    wk02*ez(i-1,j-1,k)   +  wk01*ez(i-1,j-1,k-1)
&     + wk(-1)*ez(i-1,j-1,k-2) + wk(1)*ez(i-1,j-1,k+1)
  f21=    wk02*ez(i+1,j-1,k)   +  wk01*ez(i+1,j-1,k-1)
&     + wk(-1)*ez(i+1,j-1,k-2) + wk(1)*ez(i+1,j-1,k+1)

   f1 = wi(-1)*f11 + wi(0)*f01 + wi(1)*f21
```

```
      f02=   wk02*ez(i,j+1,k)     +  wk01*ez(i,j+1,k-1)
     &    + wk(-1)*ez(i,j+1,k-2)   + wk(1)*ez(i,j+1,k+1)
      f12=   wk02*ez(i-1,j+1,k)   +  wk01*ez(i-1,j+1,k-1)
     &    + wk(-1)*ez(i-1,j+1,k-2) + wk(1)*ez(i-1,j+1,k+1)
      f22=   wk02*ez(i+1,j+1,k)   +  wk01*ez(i+1,j+1,k-1)
     &    + wk(-1)*ez(i+1,j+1,k-2) + wk(1)*ez(i+1,j+1,k+1)

       f2 = wi(-1)*f12 + wi(0)*f02 + wi(1)*f22

      ez0 = (wj(-1)*f1 + wj(0)*f0 + wj(1)*f2)*0.25*qm

c Bx
      f00=   wk02*(bx(i,j,k)     + bx(i,j-1,k))
     &    +   wk01*(bx(i,j,k-1)   + bx(i,j-1,k-1))
     &    + wk(-1)*(bx(i,j,k-2)   + bx(i,j-1,k-2))
     &    +  wk(1)*(bx(i,j,k+1)   + bx(i,j-1,k+1))

      f10=   wk02*(bx(i,j-1,k)   + bx(i,j-2,k))
     &    +   wk01*(bx(i,j-1,k-1) + bx(i,j-2,k-1))
     &    + wk(-1)*(bx(i,j-1,k-2) + bx(i,j-2,k-2))
     &    +  wk(1)*(bx(i,j-1,k+1) + bx(i,j-2,k+1))
```

```
  f20=    wk02*(bx(i,j+1,k)   + bx(i,j,k))
&     +   wk01*(bx(i,j+1,k-1) + bx(i,j,k-1))
&     + wk(-1)*(bx(i,j+1,k-2) + bx(i,j,k-2))
&     +  wk(1)*(bx(i,j+1,k+1) + bx(i,j,k+1))

   f0 = wj(-1)*f10 + wj(0)*f00 + wj(1)*f20

  f01=    wk02*(bx(i-1,j,k)     + bx(i-1,j-1,k))
&     +   wk01*(bx(i-1,j,k-1)   + bx(i-1,j-1,k-1))
&     + wk(-1)*(bx(i-1,j,k-2)   + bx(i-1,j-1,k-2))
&     +  wk(1)*(bx(i-1,j,k+1)   + bx(i-1,j-1,k+1))

  f11=    wk02*(bx(i-1,j-1,k)   + bx(i-1,j-2,k))
&     +   wk01*(bx(i-1,j-1,k-1) + bx(i-1,j-2,k-1))
&     + wk(-1)*(bx(i-1,j-1,k-2) + bx(i-1,j-2,k-2))
&     +  wk(1)*(bx(i-1,j-1,k+1) + bx(i-1,j-2,k+1))

  f21=    wk02*(bx(i-1,j+1,k)   + bx(i-1,j,k))
&     +   wk01*(bx(i-1,j+1,k-1) + bx(i-1,j,k-1))
&     + wk(-1)*(bx(i-1,j+1,k-2) + bx(i-1,j,k-2))
&     +  wk(1)*(bx(i-1,j+1,k+1) + bx(i-1,j,k+1))
```

```
  f1 = wj(-1)*f11 + wj(0)*f01 + wj(1)*f21

 f02=    wk02*(bx(i+1,j,k)     + bx(i+1,j-1,k))
&    +   wk01*(bx(i+1,j,k-1)   + bx(i+1,j-1,k-1))
&    + wk(-1)*(bx(i+1,j,k-2)   + bx(i+1,j-1,k-2))
&    +  wk(1)*(bx(i+1,j,k+1)   + bx(i+1,j-1,k+1))

 f12=    wk02*(bx(i+1,j-1,k)   + bx(i+1,j-2,k))
&    +   wk01*(bx(i+1,j-1,k-1) + bx(i+1,j-2,k-1))
&    + wk(-1)*(bx(i+1,j-1,k-2) + bx(i+1,j-2,k-2))
&    +  wk(1)*(bx(i+1,j-1,k+1) + bx(i+1,j-2,k+1))

 f22=    wk02*(bx(i+1,j+1,k)   + bx(i+1,j,k))
&    +   wk01*(bx(i+1,j+1,k-1) + bx(i+1,j,k-1))
&    + wk(-1)*(bx(i+1,j+1,k-2) + bx(i+1,j,k-2))
&    +  wk(1)*(bx(i+1,j+1,k+1) + bx(i+1,j,k+1))

  f2 = wj(-1)*f12 + wj(0)*f02 + wj(1)*f22
```

```
      bx0 = (wi(-1)*f1 + wi(0)*f0 + wi(1)*f2)*0.125*qm/c

c By
     f00=      wi02*(by(i,j,k)   +   by(i,j,k-1))
    &     +   wi01*(by(i-1,j,k)   + by(i-1,j,k-1))
    &     + wi(-1)*(by(i-2,j,k)   + by(i-2,j,k-1))
    &     +  wi(1)*(by(i+1,j,k)   + by(i+1,j,k-1))

     f10=      wi02*(by(i,j,k-1) +   by(i,j,k-2))
    &     +   wi01*(by(i-1,j,k-1) + by(i-1,j,k-2))
    &     + wi(-1)*(by(i-2,j,k-1) + by(i-2,j,k-2))
    &     +  wi(1)*(by(i+1,j,k-1) + by(i+1,j,k-2))

     f20=      wi02*(by(i,j,k+1) +   by(i,j,k))
    &     +   wi01*(by(i-1,j,k+1) + by(i-1,j,k))
    &     + wi(-1)*(by(i-2,j,k+1) + by(i-2,j,k))
    &     +  wi(1)*(by(i+1,j,k+1) + by(i+1,j,k))

      f0 = wk(-1)*f10 + wk(0)*f00 + wk(1)*f20
```

```
  f01=      wi02*(by(i,j-1,k)   +   by(i,j-1,k-1))
&     +   wi01*(by(i-1,j-1,k)   + by(i-1,j-1,k-1))
&     + wi(-1)*(by(i-2,j-1,k)   + by(i-2,j-1,k-1))
&     +  wi(1)*(by(i+1,j-1,k)   + by(i+1,j-1,k-1))

  f11=      wi02*(by(i,j-1,k-1) +   by(i,j-1,k-2))
&     +   wi01*(by(i-1,j-1,k-1) + by(i-1,j-1,k-2))
&     + wi(-1)*(by(i-2,j-1,k-1) + by(i-2,j-1,k-2))
&     +  wi(1)*(by(i+1,j-1,k-1) + by(i+1,j-1,k-2))

  f21=      wi02*(by(i,j-1,k+1) +   by(i,j-1,k))
&     +   wi01*(by(i-1,j-1,k+1) + by(i-1,j-1,k))
&     + wi(-1)*(by(i-2,j-1,k+1) + by(i-2,j-1,k))
&     +  wi(1)*(by(i+1,j-1,k+1) + by(i+1,j-1,k))

  f1 = wk(-1)*f11 + wk(0)*f01 + wk(1)*f21

  f02=      wi02*(by(i,j+1,k)   +   by(i,j+1,k-1))
&     +   wi01*(by(i-1,j+1,k)   + by(i-1,j+1,k-1))
&     + wi(-1)*(by(i-2,j+1,k)   + by(i-2,j+1,k-1))
&     +  wi(1)*(by(i+1,j+1,k)   + by(i+1,j+1,k-1))
```

```
  f12=     wi02*(by(i,j+1,k-1) +   by(i,j+1,k-2))
 &     +   wi01*(by(i-1,j+1,k-1) + by(i-1,j+1,k-2))
 &     + wi(-1)*(by(i-2,j+1,k-1) + by(i-2,j+1,k-2))
 &     +  wi(1)*(by(i+1,j+1,k-1) + by(i+1,j+1,k-2))

  f22=     wi02*(by(i,j+1,k+1) +   by(i,j+1,k))
 &     +   wi01*(by(i-1,j+1,k+1) + by(i-1,j+1,k))
 &     + wi(-1)*(by(i-2,j+1,k+1) + by(i-2,j+1,k))
 &     +  wi(1)*(by(i+1,j+1,k+1) + by(i+1,j+1,k))

  f2 = wk(-1)*f12 + wk(0)*f02 + wk(1)*f22

  by0 = (wj(-1)*f1 + wj(0)*f0 + wj(1)*f2)*0.125*qm/c

c Bz
  f00=    wj02*(bz(i,j,k)     + bz(i-1,j,k))
 &     +   wj01*(bz(i,j-1,k)   + bz(i-1,j-1,k))
 &     + wj(-1)*(bz(i,j-2,k)   + bz(i-1,j-2,k))
 &     +  wj(1)*(bz(i,j+1,k)   + bz(i-1,j+1,k))
```

```
  f10=    wj02*(bz(i-1,j,k)   + bz(i-2,j,k))
&     +   wj01*(bz(i-1,j-1,k) + bz(i-2,j-1,k))
&     + wj(-1)*(bz(i-1,j-2,k) + bz(i-2,j-2,k))
&     +  wj(1)*(bz(i-1,j+1,k) + bz(i-2,j+1,k))

  f20=    wj02*(bz(i+1,j,k)   + bz(i,j,k))
&     +   wj01*(bz(i+1,j-1,k) + bz(i,j-1,k))
&     + wj(-1)*(bz(i+1,j-2,k) + bz(i,j-2,k))
&     +  wj(1)*(bz(i+1,j+1,k) + bz(i,j+1,k))

   f0 = wi(-1)*f10 + wi(0)*f00 + wi(1)*f20

  f01=    wj02*(bz(i,j,k-1)     + bz(i-1,j,k-1))
&     +   wj01*(bz(i,j-1,k-1)   + bz(i-1,j-1,k-1))
&     + wj(-1)*(bz(i,j-2,k-1)   + bz(i-1,j-2,k-1))
&     +  wj(1)*(bz(i,j+1,k-1)   + bz(i-1,j+1,k-1))
```

```
 f11=    wj02*(bz(i-1,j,k-1)   + bz(i-2,j,k-1))
&     +   wj01*(bz(i-1,j-1,k-1) + bz(i-2,j-1,k-1))
&     + wj(-1)*(bz(i-1,j-2,k-1) + bz(i-2,j-2,k-1))
&     +  wj(1)*(bz(i-1,j+1,k-1) + bz(i-2,j+1,k-1))

 f21=    wj02*(bz(i+1,j,k-1)   + bz(i,j,k-1))
&     +   wj01*(bz(i+1,j-1,k-1) + bz(i,j-1,k-1))
&     + wj(-1)*(bz(i+1,j-2,k-1) + bz(i,j-2,k-1))
&     +  wj(1)*(bz(i+1,j+1,k-1) + bz(i,j+1,k-1))

  f1 = wi(-1)*f11 + wi(0)*f01 + wi(1)*f21

 f02=    wj02*(bz(i,j,k+1)     + bz(i-1,j,k+1))
&     +   wj01*(bz(i,j-1,k+1)   + bz(i-1,j-1,k+1))
&     + wj(-1)*(bz(i,j-2,k+1)   + bz(i-1,j-2,k+1))
&     +  wj(1)*(bz(i,j+1,k+1)   + bz(i-1,j+1,k+1))

 f12=    wj02*(bz(i-1,j,k+1)   + bz(i-2,j,k+1))
&     +   wj01*(bz(i-1,j-1,k+1) + bz(i-2,j-1,k+1))
&     + wj(-1)*(bz(i-1,j-2,k+1) + bz(i-2,j-2,k+1))
&     +  wj(1)*(bz(i-1,j+1,k+1) + bz(i-2,j+1,k+1))
```

```
      f22=    wj02*(bz(i+1,j,k+1)   + bz(i,j,k+1))
    &     +   wj01*(bz(i+1,j-1,k+1) + bz(i,j-1,k+1))
    &     + wj(-1)*(bz(i+1,j-2,k+1) + bz(i,j-2,k+1))
    &     +  wj(1)*(bz(i+1,j+1,k+1) + bz(i,j+1,k+1))

      f2 = wi(-1)*f12 + wi(0)*f02 + wi(1)*f22

      bz0 = (wk(-1)*f1 + wk(0)*f0 + wk(1)*f2)*0.125*qm/c

c  multiplication by time-step
           ex0=DT*ex0
           ey0=DT*ey0
           ez0=DT*ez0
           bx0=DT*bx0
           by0=DT*by0
           bz0=DT*bz0
```

```
c  first-half electric acceleration, with relativity's gamma
           g=c/sqrt(c**2-u(n)**2-v(n)**2-w(n)**2)
      u0=g*u(n)+ex0
      v0=g*v(n)+ey0
      w0=g*w(n)+ez0

c  first-half magnetic rotation, with relativity's gamma
           g=c/sqrt(c**2+u0**2+v0**2+w0**2)
      bx0=g*bx0
      by0=g*by0
      bz0=g*bz0
      f=2.0/(1.0+bx0*bx0+by0*by0+bz0*bz0)
      u1=(u0+v0*bz0-w0*by0)*f
      v1=(v0+w0*bx0-u0*bz0)*f
      w1=(w0+u0*by0-v0*bx0)*f

c  second-half magnetic rotation and electric acceleration
      u0=u0+v1*bz0-w1*by0+ex0
      v0=v0+w1*bx0-u1*bz0+ey0
      w0=w0+u1*by0-v1*bx0+ez0
```

```
c  relativity's gamma
        g=c/sqrt(c**2+u0**2+v0**2+w0**2)
        u(n)=g*u0
        v(n)=g*v0
        w(n)=g*w0
c  position advance
        x(n)=x(n)+DT*u(n)
        y(n)=y(n)+DT*v(n)
                z(n)=z(n)+DT*w(n)
       END DO

      return
      end
```

```
c  ****************************************************************
      subroutine mover(ipar,x,y,z,u,v,w,mh,ex,ey,ez,bx,by,bz,
     &                   mFx,mFy,mFz,DHDx,DHDy,DHDz,qm,DT,c)

      dimension ex(mFx,mFy,mFz),ey(mFx,mFy,mFz),ez(mFx,mFy,mFz)
      dimension bx(mFx,mFy,mFz),by(mFx,mFy,mFz),bz(mFx,mFy,mFz)
      dimension x(mh),y(mh),z(mh)
      dimension u(mh),v(mh),w(mh)

       DO n = 1,ipar
c  cell index and displacement in cell
        i  = x(n)- DHDx
        dx = x(n)-i -DHDx
        j  = y(n)- DHDy
        dy = y(n)-j -DHDy
        k  = z(n)- DHDz
        dz = z(n)-k -DHDz

C Field interpolations are tri-linear (linear in x times linear in y
C times linear in z). This amounts to the 3-D generalisation of "area
```

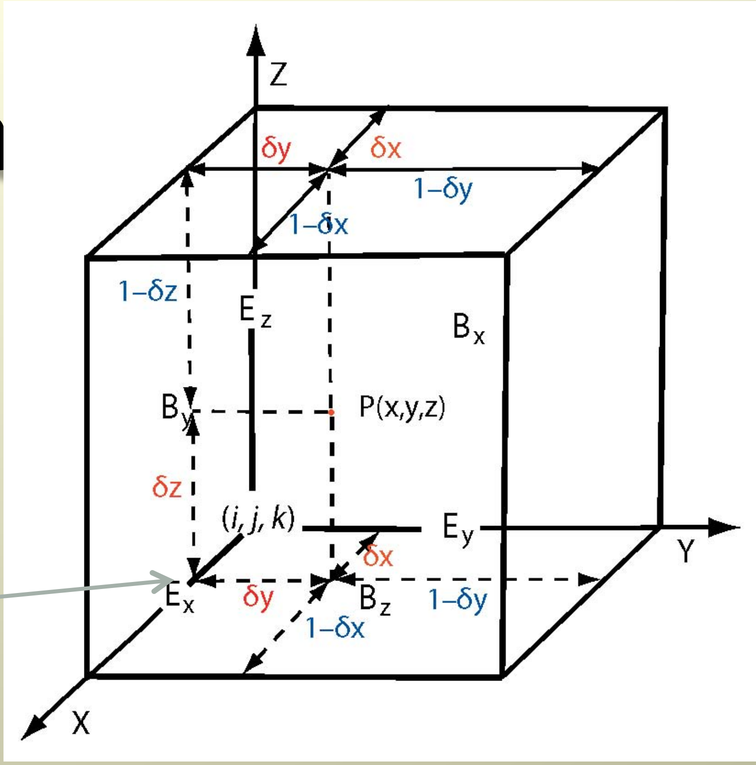
```
C weighting". A modification of the simple linear interpolation formula
C        f(i+dx) = f(i) + dx * (f(i+1)-f(i))
C is needed since fields are recorded at half-integer locations in certain
C dimensions: see comments and illustration with the Maxwell part of this
C code. One then has first to interpolate from "midpoints" to "gridpoints"
C by averaging neighbors. Then one proceeds with normal interpolation.
C Combining these two steps leads to:
C    f at location i+dx  = half of f(i)+f(i-1) + dx*(f(i+1)-f(i-1))
C where now f(i) means f at location i+1/2. The halving is absorbed
C in the final scaling (e.g, in ex0 etc.).

c  E-component interpolations:
      f=ex(i,j,k)+ex(i-1,j,k)+dx*(ex(i+1,j,k)-ex(i-1,j,k))
      f=f+dy*(ex(i,j+1,k)+ex(i-1,j+1,k)+dx*(ex(i+1,j+1,k)-
     &         ex(i-1,j+1,k))-f)
      g=ex(i,j,k+1)+ex(i-1,j,k+1)+dx*(ex(i+1,j,k+1)-ex(i-1,j,k+1))
      g=g+dy*(ex(i,j+1,k+1)+ex(i-1,j+1,k+1)+dx*(ex(i+1,j+1,k+1)-
     &         ex(i-1,j+1,k+1))-g)
      ex0=(f+dz*(g-f))*(.25*qm)
```

Force interpreta

"volume" weight

$$(i,j,k) \Leftarrow (1-\delta x)(1-\delta y)(1-\delta z) = cx \bullet cy \bullet cz$$

$$(i+1,j+1,k+1) \Leftarrow \delta x \bullet \delta y \bullet \delta z$$



$$\mathbf{F}_{e_x}^{(x,j,k)} = \overline{e}_x(i,j,k) + \left[\overline{e}_x(i+1,j,k) - \overline{e}_x(i,j,k)\right]\delta x$$

$$\overline{e}_x(i,j,k) = \frac{1}{2}\left\{e_x(i,j,k) + e_x(i-1,j,k)\right\} \qquad \overline{e}_x(i+1,j,k) = \frac{1}{2}\left\{e_x(i+1,j,k) + e_x(i,j,k)\right\}$$

on (x, j, k)

$$2\mathbf{F}_{e_x}^{(x,j,k)} = e_x(i,j,k) + e_x(i-1,j,k) + \left[e_x(i+1,j,k) - e_x(i-1,j,k)\right]\delta x$$

similarly on (x, *j*+1, *k*), (*x*, *j*, *k*+1), (*x*, *j*+1, *k*+1)

$$2\mathbf{F}_{e_x}^{(x,j+1,k)} = e_x(i,j+1,k) + e_x(i-1,j+1,k) + \left[e_x(i+1,j+1,k) - e_x(i-1,j+1,k)\right]\delta x$$

$$2\mathbf{F}_{e_x}^{(x,j,k+1)} = e_x(i,j,k+1) + e_x(i-1,j,k+1) + \left[e_x(i+1,j,k+1) - e_x(i-1,j,k+1)\right]\delta x$$

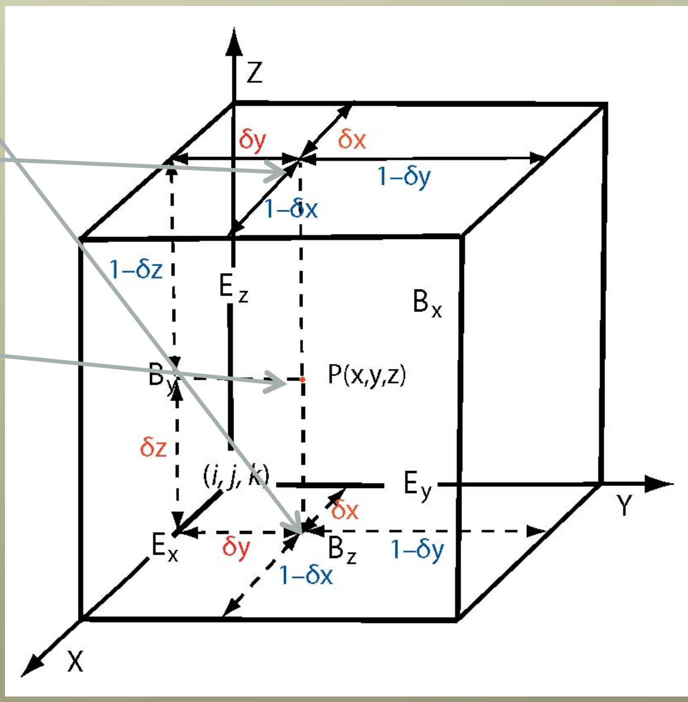
$$2\mathbf{F}_{e_x}^{(x,j+1,k+1)} = e_x(i,j+1,k+1) + e_x(i-1,j+1,k+1) + \left[e_x(i+1,j+1,k+1) - e_x(i-1,j+1,k+1)\right]\delta x$$

$$\mathbf{F}_{e_x}^{(x,y,k)} = \mathbf{F}_{e_x}^{(x,j,k)} + \left[\mathbf{F}_{e_x}^{(x,j+1,k)} - \mathbf{F}_{e_x}^{(x,j,k)}\right]\delta y$$

$$\mathbf{F}_{e_x}^{(x,y,k+1)} = \mathbf{F}_{e_x}^{(x,j,k+1)} + \left[\mathbf{F}_{e_x}^{(x,j+1,k+1)} - \mathbf{F}_{e_x}^{(x,j,k+1)}\right]\delta y$$

$$\mathbf{F}_{e_x}^{(x,y,z)} = \mathbf{F}_{e_x}^{(x,y,k)} + \left[\mathbf{F}_{e_x}^{(x,y,k+1)} - \mathbf{F}_{e_x}^{(x,y,k)}\right]\delta z$$

$$\mathbf{F}_{e_y}^{(x,y,z)},\ \mathbf{F}_{e_z}^{(x,y,z)},\ \mathbf{F}_{b_x}^{(x,y,z)},\ \mathbf{F}_{b_y}^{(x,y,z)},\ \mathbf{F}_{b_z}^{(x,y,z)}$$



```
      f=ey(i,j,k)+ey(i,j-1,k)+dy*(ey(i,j+1,k)-ey(i,j-1,k))
      f=f+dz*(ey(i,j,k+1)+ey(i,j-1,k+1)+dy*(ey(i,j+1,k+1)-
     &        ey(i,j-1,k+1))-f)
      g=ey(i+1,j,k)+ey(i+1,j-1,k)+dy*(ey(i+1,j+1,k)-ey(i+1,j-1,k))
      g=g+dz*(ey(i+1,j,k+1)+ey(i+1,j-1,k+1)+dy*(ey(i+1,j+1,k+1)-
     &        ey(i+1,j-1,k+1))-g)
      ey0=(f+dx*(g-f))*(.25*qm)

      f=ez(i,j,k)+ez(i,j,k-1)+dz*(ez(i,j,k+1)-ez(i,j,k-1))
      f=f+dx*(ez(i+1,j,k)+ez(i+1,j,k-1)+dz*(ez(i+1,j,k+1)-
     &        ez(i+1,j,k-1))-f)
      g=ez(i,j+1,k)+ez(i,j+1,k-1)+dz*(ez(i,j+1,k+1)-ez(i,j+1,k-1))
      g=g+dx*(ez(i+1,j+1,k)+ez(i+1,j+1,k-1)+dz*(ez(i+1,j+1,k+1)-
     &        ez(i+1,j+1,k-1))-g)
      ez0=(f+dy*(g-f))*(.25*qm)

c  B-component interpolations:    (Final assignment)
            f=bx(i,j-1,k)+bx(i,j-1,k-1)+dz*(bx(i,j-1,k+1)-bx(i,j-1,k-1))
      f=bx(i,j,k)+bx(i,j,k-1)+dz*(bx(i,j,k+1)-bx(i,j,k-1))+f+
     &  dy*(bx(i,j+1,k)+bx(i,j+1,k-1)+dz*(bx(i,j+1,k+1)-
     &  bx(i,j+1,k-1))-f)
```

```
  g=bx(i+1,j-1,k)+bx(i+1,j-1,k-1)+dz*(bx(i+1,j-1,k+1)-
&   bx(i+1,j-1,k-1))
  g=bx(i+1,j,k)+bx(i+1,j,k-1)+dz*(bx(i+1,j,k+1)-bx(i+1,j,k-1))+g+
&       dy*(bx(i+1,j+1,k)+bx(i+1,j+1,k-1)+dz*(bx(i+1,j+1,k+1)-
&   bx(i+1,j+1,k-1))-g)
  bx0=(f+dx*(g-f))*(.125*qm/c)

  f=by(i,j,k-1)+by(i-1,j,k-1)+dx*(by(i+1,j,k-1)-by(i-1,j,k-1))
   f=by(i,j,k)+by(i-1,j,k)+dx*(by(i+1,j,k)-by(i-1,j,k))+f+
&   dz*(by(i,j,k+1)+by(i-1,j,k+1)+dx*(by(i+1,j,k+1)-
&   by(i-1,j,k+1))-f)
  g=by(i,j+1,k-1)+by(i-1,j+1,k-1)+dx*(by(i+1,j+1,k-1)-
&   by(i-1,j+1,k-1))
  g=by(i,j+1,k)+by(i-1,j+1,k)+dx*(by(i+1,j+1,k)-by(i-1,j+1,k))+g+
&   dz*(by(i,j+1,k+1)+by(i-1,j+1,k+1)+dx*(by(i+1,j+1,k+1)-
&   by(i-1,j+1,k+1))-g)
  by0=(f+dy*(g-f))*(.125*qm/c)

  f=bz(i-1,j,k)+bz(i-1,j-1,k)+dy*(bz(i-1,j+1,k)-bz(i-1,j-1,k))
```

```
      f=bz(i,j,k)+bz(i,j-1,k)+dy*(bz(i,j+1,k)-bz(i,j-1,k))+f+
     &    dx*(bz(i+1,j,k)+bz(i+1,j-1,k)+dy*(bz(i+1,j+1,k)-
     &    bz(i+1,j-1,k))-f)
      g=bz(i-1,j,k+1)+bz(i-1,j-1,k+1)+dy*(bz(i-1,j+1,k+1)-
     &    bz(i-1,j-1,k+1))
      g=bz(i,j,k+1)+bz(i,j-1,k+1)+dy*(bz(i,j+1,k+1)-bz(i,j-1,k+1))+g+
     &    dx*(bz(i+1,j,k+1)+bz(i+1,j-1,k+1)+dy*(bz(i+1,j+1,k+1)-
     &    bz(i+1,j-1,k+1))-g)
      bz0=(f+dz*(g-f))*(.125*qm/c)

c  multiplication by time-step
            ex0=DT*ex0
            ey0=DT*ey0
            ez0=DT*ez0
            bx0=DT*bx0
            by0=DT*by0
            bz0=DT*bz0
```

```
c  first-half electric acceleration, with relativity's gamma
           g=c/sqrt(c**2-u(n)**2-v(n)**2-w(n)**2)
      u0=g*u(n)+ex0
      v0=g*v(n)+ey0
      w0=g*w(n)+ez0

c  first-half magnetic rotation, with relativity's gamma
           g=c/sqrt(c**2+u0**2+v0**2+w0**2)
      bx0=g*bx0
      by0=g*by0
      bz0=g*bz0
      f=2.0/(1.0+bx0*bx0+by0*by0+bz0*bz0)
      u1=(u0+v0*bz0-w0*by0)*f
      v1=(v0+w0*bx0-u0*bz0)*f
      w1=(w0+u0*by0-v0*bx0)*f

c  second-half magnetic rotation and electric acceleration
      u0=u0+v1*bz0-w1*by0+ex0
      v0=v0+w1*bx0-u1*bz0+ey0
      w0=w0+u1*by0-v1*bx0+ez0
```

```
c  relativity's gamma
      g=c/sqrt(c**2+u0**2+v0**2+w0**2)
      u(n)=g*u0
      v(n)=g*v0
      w(n)=g*w0
c  position advance
      x(n)=x(n)+DT*u(n)
      y(n)=y(n)+DT*v(n)
            z(n)=z(n)+DT*w(n)
     END DO

     return
     end
```

```
c  ****************************************************************
c  initialize ambient plasma particles to uniform distribution
c ** particles are kept 3 units away from the lower boundaries of the field **
c ** domain and 2 units away from the upper boundaries                       **
c ** each process has its portion(s) of a global VIRTUAL particle array(s)  **
      subroutine Particle_init(ions,lecs,mh,vithml,vethml,
     &                    PBLeft,PBRght,PBFrnt,PBRear,PBBot,PBTop,
     &                     x0,y0,z0,dlx,dly,dlz,lx1,ly1,lz1,
     &                 xi,yi,zi,ui,vi,wi,xe,ye,ze,ue,ve,we,c,is)

      dimension xi(mh),yi(mh),zi(mh),ui(mh),vi(mh),wi(mh)
      dimension xe(mh),ye(mh),ze(mh),ue(mh),ve(mh),we(mh)

      pi  = 4*atan(1.0d0)

c  variance of Maxwell distribution for individual velocity components
      sigi = vithml/sqrt(2.)
      sige = vethml/sqrt(2.)
```

```
c  initialize particle positions and velocities
      DO k = 1,lz1
       DO j = 1,ly1
        DO i = 1,lx1
         xx = x0 + dlx*i
         yy = y0 + dly*j
         zz = z0 + dlz*k

                   tx = xx + dlx*(ran1(is)-0.5)
                   ty = yy + dly*(ran1(is)-0.5)
                   tz = zz + dlz*(ran1(is)-0.5)

c ** "if" statement is not necessary since the algoritm should't place    **
c ** particles outside of the boundaries; this is to double check this    **
c ** and also to ensure correctness of relational operations ( .ge. .lt.) **
c ** which is important for MOVER, as it access B_i(i=2,j,k) and i=nFx+3  **
c ** elements - the only ones which are passed between processors (thus   **
c ** having, e.g., tx = PBRght would cause the obscure error)             **
                    if( ((tx.ge.PBLeft) .and. (tx.lt.PBRght)) .and.
     &              ((ty.ge.PBFrnt) .and. (ty.lt.PBRear)) .and.
```

```
 &      ((tz.ge.PBBot ) .and. (tz.lt.PBTop ))) then
              ions = ions + 1
              xi(ions) = tx
              yi(ions) = ty
              zi(ions) = tz
c  electrons in same locations as ions for zero initial charge density
              lecs = lecs + 1
              xe(lecs) = tx
              ye(lecs) = ty
              ze(lecs) = tz

c  thermal velocities of plasma particles
18        r11 = ran1(is)
        if(r11.EQ.1.0) goto 18

        r1 = sqrt(-2.0*log(1.0-r11))
              if(sigi*r1.ge.c) goto 18
              r2 = 2.0*pi*ran1(is)
        uion = sigi*r1*cos(r2)
        vion = sigi*r1*sin(r2)
```

```
19      r33 = ran1(is)
      if(r33.EQ.1.0) goto 19

      r3 = sqrt(-2.0*log(1.0-r33))
            if(sige*r3.ge.c) goto 19

            r4 = 2.0*pi*ran1(is)
            wion = sigi*r3*cos(r4)

      uele = sige*r3*sin(r4)

20      r55 = ran1(is)
      if(r55.EQ.1.0) goto 20

      r5 = sqrt(-2.0*log(1.0-r55))
            if(sige*r5.ge.c) goto 20

            r6 = 2.0*pi*ran1(is)
            vele = sige*r5*cos(r6)

      wele = sige*r5*sin(r6)
```

```
cJET
            ui(ions) = uion
      vi(ions) = vion
      wi(ions) = wion

            ue(lecs) = uele
      ve(lecs) = vele
      we(lecs) = wele
           end if
          END DO
    END DO
   END DO

   return
   end
```

```
c ***************************************************************
  subroutine Jet_injection(ionj,lecj,mh,xinj,rjt,yjc,zjc,
    &                    PBFrnt,PBRear,PBBot,PBTop,
    &                    vijet,vejet,vithmj,vethmj,
    &                    x0,y0,z0,dlx,dly,dlz,ly1,lz1,
    &      xij,yij,zij,uij,vij,wij,xej,yej,zej,uej,vej,wej,
    &                                   selj,c,is)

c             Model the dirfting relativistic Maxwellians
c             for the relativsitic Harris sheet
c             INPUT
c               beta: the (modulus of the) drift speed of each
c                     component with respect to the "lab" frame
c               temp: the temperature T in units of mc^2/kB
c                     In the co-drifting frames the temperature
c                     is \Theta=\Gamma*T - see notes in KS03

c     Use the methods described by Pozdnyakov and Sobol (1983) for
c     the relativistic Maxwellian

      dimension xij(mh),yij(mh),zij(mh),uij(mh),vij(mh),wij(mh)
      dimension xej(mh),yej(mh),zej(mh),uej(mh),vej(mh),wej(mh)
```

```
      pi  = 4*atan(1.0d0)
      gamijet = 1.0/sqrt(1.0-(vijet/c)**2)
      gamejet = 1.0/sqrt(1.0-(vejet/c)**2)

c  variance of Maxwell distribution for individual velocity components
      sigi = vithmj/sqrt(2.)
      sige = vethmj/sqrt(2.)
      theti = sigi*gamijet
      thete = sige*gamejet

c  initialize particle positions and velocities
      DO k = 1,lz1
       DO j = 1,ly1
        yy = y0 + dly*j
        zz = z0 + dlz*k

             tx = xinj + dlx*(ran1(is)-0.5)
             ty = yy + dly*(ran1(is)-0.5)
             tz = zz + dlz*(ran1(is)-0.5)
```

```
              if(((ty.ge.PBFrnt) .and. (ty.lt.PBRear)) .and.
    &     ((tz.ge.PBBot ) .and. (tz.lt.PBTop ))) then
             ionj = ionj + 1
             xij(ionj) = tx
             yij(ionj) = ty
             zij(ionj) = tz
c  electrons in same locations as ions for zero initial charge density
             lecj = lecj + 1
             xej(lecj) = tx
             yej(lecj) = ty
             zej(lecj) = tz

c  ion thermal
c        Find eta=p/mc
c ran1 dran()
     if(theti.lt.0.29)then
 1      r1=ran1(is)
        r2=ran1(is)
        zetap=-1.5*log(r1)
```

```
  if(
 1      r2**2.lt.0.51*(1.+theti*zetap)**2*zetap*(2+theti*zetap)*r1
 2    )then
      etai=sqrt(theti*zetap*(2.+theti*zetap))
    else
      go to 1
    end if
  else
2      r1=ran1(is)
    r2=ran1(is)
    r3=ran1(is)
    r4=ran1(is)
    etap=-theti*log(r1*r2*r3)
    etapp=-theti*log(r1*r2*r3*r4)
    if(etapp**2-etap**2.gt.1.)then
      etai=etap
    else
      go to 2
    end if
  end if
```

```
c       Draw a random direction

 3   r1=2.*(ran1(is)  -.5)
     r2=2.*(ran1(is)  -.5)
     r3=2.*(ran1(is)  -.5)
     radius=sqrt(r1**2+r2**2+r3**2)
     if(radius.lt.1.)then
       uion=etai*r1/radius
       vion=etai*r2/radius
       wion=etai*r3/radius
     else
       go to 3
     end if


c  electron thermal
c         Find eta=p/mc
c ran1 dran()
     if(thete.lt.0.29)then
```

```
5       r1=ran1(is)
        r2=ran1(is)
        zetap=-1.5*log(r1)
        if(
     1    r2**2.lt.0.51*(1.+thete*zetap)**2*zetap*(2+thete*zetap)*r1
     2  )then
          etae=sqrt(thete*zetap*(2.+thete*zetap))
        else
          go to 5
        end if
      else
6       r1=ran1(is)
        r2=ran1(is)
        r3=ran1(is)
        r4=ran1(is)
        etap=-thete*log(r1*r2*r3)
        etapp=-thete*log(r1*r2*r3*r4)
        if(etapp**2-etap**2.gt.1.)then
          etae=etap
        else
```

```
     go to 6
       end if
     end if

*      Draw a random direction

 7   r1=2.*(ran1(is)  -.5)
     r2=2.*(ran1(is)  -.5)
     r3=2.*(ran1(is)  -.5)
     radius=sqrt(r1**2+r2**2+r3**2)
     if(radius.lt.1.)then
       uele=etae*r1/radius
       vele=etae*r2/radius
       wele=etae*r3/radius
     else
       go to 7
     end if
```

```
c       Now compute the lab frame output quantities

cJET
c ** thermal spread is added in the jet plasma rest frame     **
c ** transform velocity components to the upstream rest frame **
c ** (should preserve causality)                              **
c  ions
        gammapi=sqrt(1.0+etai**2)
        vden=gamijet*(gammapi+vijet*uion)

        uij(ionj) = gamijet*(uion+vijet*gammapi)/vden
        vij(ionj) = vion/vden
        wij(ionj) = wion/vden

c  electrons
        gammape=sqrt(1.0+etae**2)
        vden=gamejet*(gammape+vejet*uele)
```

```
    uej(lecj) = gamejet*(uele+vejet*gammape)/vden
    vej(lecj) = vele/vden
    wej(lecj) = wele/vden
         end if
 END DO
END DO

return
end
```