Computational Methods for Kinetic Processes in Plasma Physics

Ken Nishikawa

Department of Physics/UAH

Main program 1

June 3, 2015

Context

- Structure of Tristan code

  Subroutines

- Initial settings (inputperpN15nSS.f)

  Jet simulations

  Reconnections

- Main program

Structure of Tristan code: Subroutines

Main program

- main loop
- dump data for rerun and diagnostics
- accumulate data for radiation

input program (include)

- set parameters for simulations (density, jet velocity, magnetic field, etc)
- load particles as initial conditions

diagnostic program (include) (not active)

Initial settings

Jet simulations

```
c Version for the jet studies, based on 3D-periodic code for magnetic field
c generation studies at SNR shocks (warm beam case). Non-periodic boundary
c conditions for the jet direction (x-direction)

c DIFFERENCES with a serial version and an OpenMP version by Ken include:
c 1. Umeda 1-st order method for current deposition (can be easily changed
c    to Buneman-Villasenor method if required)
c 2. different output file format for file naming and data dump: now each
c    processor writes data to a separate file
c 3. data (double precision) coded with 16-bit integers, to save the disk
c    space - needs to be converted back to double prec. for post-processing
c 4. smoothing of currents is taken out of depsit routines and applied after
c    particle splitting
c 5. particle sorting in order they are stored initially in the memory is
c    applied for better performance
c number of processors
```

```
      parameter (Nproc=4)
      parameter (Npx=1,Npy=2,Npz=2)

c  VIRTUAL PARTICLE array is distributed among processes so that each proces
c  works on nFx*nFy*nFz cells (its subdomain);

c !!! y and z domain size MUST BE EQUAL in the present setup: nFy=nFz !!!

c  FIELD arrays have mFi (mFx,mFy,mFz) elements in each dimension:
c  nFi plus 3 ghost cells on the "Right" and 2 ghost cells on the "Left"
      parameter (mx=645,my=11,mz=11)

      parameter (nFx=(mx-5)/Npx,mFx=nFx+5)
      parameter (nFy=(my-5)/Npy,mFy=nFy+5)
      parameter (nFz=(mz-5)/Npz,mFz=nFz+5)
```

```
c depending the size of domain in a processor
      parameter (nptl=2000000)
      parameter (mb=nptl,mj=1800000)

c  size of particle communication buffer arrays
c ** size is set as for ambient particles that move in one general direction  **
c !! check for particle number moving perpendicular to jet flow in Left proc. !!
c **       mpass = nFy*nFz*c*DT*adens + 10%-20%                               **
c ** must be reset for different particle densities                           **

      parameter (mpass=500000)

      parameter (mdiag=2500)

c  2**15-1
      parameter (imax=32767)

      integer size,myid,ierror
      integer lgrp,comm3d
      integer topol

      integer dims(3),coords(3)
      logical isperiodic(3),reorder

      integer FBDRx,FBDRy,FBDRz
      integer FBDLx,FBDLy,FBDLz
      integer FBDRxe,FBDLxe
      integer FBDRxp,FBDLxp
      integer FBD_BRx,FBD_BRy,FBD_BRz
      integer FBD_BLx,FBD_BLy,FBD_BLz
      integer FBD_ERx,FBD_ERy,FBD_ERz
      integer FBD_ELx,FBD_ELy,FBD_ELz

c  temporary arrays in 16-bit integers for data dump
      integer*2 irho,ifx,ify,ifz,ixx,iyy,izz

      real mi,me
      character strb1*6,strb2*6,strb3*6,strb4*6,strb5*6,strb6*6
      character strj1*6,strj2*6,strj3*6,strj4*6,strj5*6,strj6*6
      character strfb*5,strfe*5
      character strin*8
      character dir*28,num*3,st01,st02,st03,st0*3,st*4,hyph
      character step*6,step1,step2,step3,step4,step5
      character soutb*3,soute*3,soutd*5,soutv*4
      character ndiag*6,nfield*7
      character num1,num2,num3
      character strpd*6

c  electric and magnetic field arrays
      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  electric field longitudinal increaments (currents) arrays
      dimension dex(mFx,mFy,mFz),dey(mFx,mFy,mFz),dez(mFx,mFy,mFz)

c  ambient ions and electrons positions and velocities arrays
      dimension xi(mb),yi(mb),zi(mb),ui(mb),vi(mb),wi(mb)
      dimension xe(mb),ye(mb),ze(mb),ue(mb),ve(mb),we(mb)
```

```
c  jet ions and electrons positions and velocities arrays
      dimension xij(mj),yij(mj),zij(mj),uij(mj),vij(mj),wij(mj)
      dimension xej(mj),yej(mj),zej(mj),uej(mj),vej(mj),wej(mj)

c  diagnostic arrays (for quantities recorded on the grid)
cWR      dimension flx(mFx,mFy,mFz),fly(mFx,mFy,mFz),flz(mFx,mFy,mFz)
cWR      dimension rho(mFx,mFy,mFz)

c  diagnostic arrays for velocity distribution
      dimension Cvpar(mdiag),Cvper(mdiag),xpos(mdiag)

c  integer*2 arrays
      dimension ifx(mFx,mFy,mFz),ify(mFx,mFy,mFz),ifz(mFx,mFy,mFz)
      dimension irho(mFx,mFy,mFz)
      dimension ixx(mb),iyy(mb),izz(mb)

c  temporary arrays for jet injection
c       dimension Cseli(msel),Csele(msel)

      dimension ipsend(4),iprecv(Nproc*4)
```

```
 cCOL embed "topology" calculation
      dimension itops(3),itopr(Nproc*3)
      dimension topol(0:(Nproc-1),3)

c  smoother array
      dimension sm(-1:1,-1:1,-1:1)
c  smoother arrays for combined digital filtering
      dimension sm1(-1:1,-1:1,-1:1),sm2(-1:1,-1:1,-1:1)
      dimension sm3(-1:1,-1:1,-1:1)
      dimension nfilt(16)

c  initialize MPI
c ** "size" must be equal "Nproc"; "myid" is a ID for each process **
      common /pparms/ lgrp,comm3d

      lgrp = MPI_COMM_WORLD

      call MPI_INIT(ierror)
      call MPI_COMM_SIZE(lgrp,size,ierror)
      call MPI_COMM_RANK(lgrp,myid,ierror)
```

Main program

```
C***********************************************************************
C  TRIdimensional STANford code, TRISTAN, fully electromagnetic,
C  with full relativistic particle dynamics. Written during spring
C  1990 by OSCAR BUNEMAN, with help from TORSTEN NEUBERT and
C  KEN NISHIKAWA.
C
C  Modified for study of jets by KEN NISHIKAWA.
C
C  A Parallel Version of TRISTAN rewritten for Jet simulations by
C  JACEK NIEMIEC based on parallel code for Space-Weather simulations
C  rewritten by Mr.TAO referencing the HPF based code.
C  February 2006
C
C  3D parallel version written by JACEK NIEMIEC with setup for
C  the shock wave precursor studies
C  March-April 2006
C
C  3D parallel version written by JACEK NIEMIEC with setup for
C  jet simulations and multiple modifications introduced during
C  studies of magnetic field generation at SNR shocks
C  February 2008
```

```
C
C***************************************************************
      program TrisMPI3D
      include 'mpif.h'

c  include initialization file
      include 'inputperpN15nSS.f'


c *** MAIN LOOP BEGINS HERE *****
1000  nstep = nstep + 1
       if(myid.eq.1) write(1,*) 'nstep=',nstep
       if(myid.eq.0) print *,'nstep=',nstep
c new attention
c adding perpendicular magnetic field with convective field
c4push      if (coords(1).eq.0) then
c4push      call B_field_boun(bx,by,bz,ex,ey,ez,mFx,mFy,mFz,DT,c,
c4push     &  b0x,b0y,e0z,FBD_BLx,FBD_BRx,FBD_BLy,FBD_BRy,FBD_BLz,FBD_
c4push      endif
```

```
c  first Maxwell-advance of the magnetic field by half a time-step
c       if(myid.eq.0) print *,'before b-field pusher'
c4push       call B_field_push(bx,by,bz,ex,ey,ez,mFx,mFy,mFz,DT,c,
c4push     &              FBD_BLx,FBD_BRx,FBD_BLy,FBD_BRy,FBD_BLz,FBD_BRz)
        call 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)


c  B-field passing and periodic boundary conditions for B-field
c ** periodic boundary conditions ("copylayr") combined in passing subroutine**
c ** only two layers must be made periodic to ensure the same conditions     **
c ** in MOVER for particles crossing the transverse boundaries                **
c       if(myid.eq.0) print *,'before b-field passing'
        call Field_passing(bx,by,bz,mFx,mFy,mFz,mc,mrl,mrh,
     &              dims,coords,FBDLx,FBDLy,FBDLz,FBDRx,FBDRy,FBDRz,
     &                        nleft,nright,nfront,nrear,nbottom,ntop)
cU1       call Field_passing2(bx,by,bz,mFx,mFy,mFz,mc2,mrh2,
cU1     &             dims,coords,FBDLxp,FBDLy,FBDLz,FBDRxp,FBDRy,FBDRz,
cU1     &                       nleft,nright,nfront,nrear,nbottom,ntop)

        if(myid.eq.0) print *,'after b-field passing’
```

```
c  MOVE PARTICLES
c  ambient ions and electrons
c        if(myid.eq.0) print *, 'before mover ambient'
       call mover(ions,xi,yi,zi,ui,vi,wi,mb,ex,ey,ez,bx,by,bz,
     &                    mFx,mFy,mFz,DHDx,DHDy,DHDz,qmi,DT,c)
       call mover(lecs,xe,ye,ze,ue,ve,we,mb,ex,ey,ez,bx,by,bz,
     &                    mFx,mFy,mFz,DHDx,DHDy,DHDz,qme,DT,c)
cU1       call mover2(ions,xi,yi,zi,ui,vi,wi,mb,ex,ey,ez,bx,by,bz,
cU1     &                    mFx,mFy,mFz,DHDx,DHDy,DHDz,qmi,DT,c)
cU1       call mover2(lecs,xe,ye,ze,ue,ve,we,mb,ex,ey,ez,bx,by,bz,
cU1     &                    mFx,mFy,mFz,DHDx,DHDy,DHDz,qme,DT,c)

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

c  jet ions and electrons
c        if(myid.eq.0) print *, 'before mover JET'
       if (ionj.ne.0) then
cJET "qmi" for a pair jet injected into electron-ion plasma
```

```
c for ion jet
c       qmi2 = qi/me
        qmi2 = qi/mi
        call mover(ionj,xij,yij,zij,uij,vij,wij,mj,ex,ey,ez,bx,by,bz,
     &                        mFx,mFy,mFz,DHDx,DHDy,DHDz,qmi2,DT,c)
cU1        call mover2(ionj,xij,yij,zij,uij,vij,wij,mj,ex,ey,ez,bx,by,bz,
cU1     &                        mFx,mFy,mFz,DHDx,DHDy,DHDz,qmi,DT,c)
       end if

       if (lecj.ne.0) then
        call mover(lecj,xej,yej,zej,uej,vej,wej,mj,ex,ey,ez,bx,by,bz,
     &                        mFx,mFy,mFz,DHDx,DHDy,DHDz,qme,DT,c)
cU1        call mover2(lecj,xej,yej,zej,uej,vej,wej,mj,ex,ey,ez,bx,by,bz,
cU1     &                        mFx,mFy,mFz,DHDx,DHDy,DHDz,qme,DT,c)
       end if

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

c  second Maxwell-advance of the magnetic field by half a time-step
       if(myid.eq.0) print *,'before b-field pusher'
```

```
c4push       call B_field_push(bx,by,bz,ex,ey,ez,mFx,mFy,mFz,DT,c,
c4oush     &             FBD_BLx,FBD_BRx,FBD_BLy,FBD_BRy,FBD_BLz,FBD_BRz)
        call 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)

c  radiating boundary conditions for B-field at the front layer of VIRTUAL
c  box based on Lindman's method
cJET
        if (coords(1).eq.(Npx-1)) then
                call Surface_Byzx(bx,by,bz,ex,ey,ez,mFx,mFy,mFz,DT,c,
      &                    FBD_BLy,FBD_BRy,FBD_BLz,FBD_BRz)
        end if

c  B-field passing and periodic boundary conditions for B-field
c        if(myid.eq.0) print *,'before b-field passing'
c4push       call Field_passing(bx,by,bz,mFx,mFy,mFz,mc,mrl,mrh,
c4push     &             dims,coords,FBDLx,FBDLy,FBDLz,FBDRx,FBDRy,FBDRz,
c4push     &                     nleft,nright,nfront,nrear,nbottom,ntop)
        call Field_passing2(bx,by,bz,mFx,mFy,mFz,mc2,mrh2,
      &       dims,coords,FBDLxp,FBDLy,FBDLz,FBDRxp,FBDRy,FBDRz,
      &                nleft,nright,nfront,nrear,nbottom,ntop)
```

```
          if(myid.eq.0) print *, 'after b-field passing'

c  full Maxwell-advance of the electric field
        if(myid.eq.0) print *,'before e-field pusher'
c4push!!
        call 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)

c  radiating boundary conditions for E-field at the rear layer of VIRTUAL box
cJET
        if (coords(1).eq.0) then
                    call Surface_Eyzx(bx,by,bz,ex,ey,ez,mFx,mFy,mFz,DT,c,
      &                          FBD_ELy,FBD_ERy,FBD_ELz,FBD_ERz)
        end if


c  PARTICLE SORTING AND PASSING
c ** sorting is included in passing subroutine **
```

```
c  ambient ions
c       if(myid.eq.0) print *, 'before passing ambient ions'
       call Particle_passing(ions,mpass,mb,GBLeft,GBRght,
     &                             PBFrnt,PBRear,PBBot,PBTop,
     &            nleft,nright,nfront,nrear,nbottom,ntop,
     &                               xi,yi,zi,ui,vi,wi)

       if(myid.eq.0) print *, 'after passing ambient ions'

c       if (nstep.eq.569) then
c        print *, myid,'ions',ions,lecs,ionj,lecj
c       stop
c       end if

c  ambient electrons
c       if(myid.eq.0) print *, 'before passing ambient electrons'
       call Particle_passing(lecs,mpass,mb,GBLeft,GBRght,
     &                             PBFrnt,PBRear,PBBot,PBTop,
     &            nleft,nright,nfront,nrear,nbottom,ntop,
     &                               xe,ye,ze,ue,ve,we)
```

```
        if(myid.eq.0) print *, 'after passing ambient electrons'
c       if (nstep.eq.569) then
c        print *, myid,'lecs',ions,lecs,ionj,lecj
c       stop
c       end if
c  jet ions
c       if(myid.eq.0) print *, 'before passing jet ions'
      call Particle_passing(ionj,mpass,mj,GBLeft,GBRght,
     &                              PBFrnt,PBRear,PBBot,PBTop,
     &            nleft,nright,nfront,nrear,nbottom,ntop,
     &                           xij,yij,zij,uij,vij,wij)

      if(myid.eq.0) print *, 'after passing jet ions'
c        if (nstep.eq.569) then
c        print *, myid,'ionj',ions,lecs,ionj,lecj
c        stop
c      end if
```

```
c  jet electrons
c       if(myid.eq.0) print *,'before passing jet electrons'
      call Particle_passing(lecj,mpass,mj,GBLeft,GBRght,
     &                              PBFrnt,PBRear,PBBot,PBTop,
     &            nleft,nright,nfront,nrear,nbottom,ntop,
     &                           xej,yej,zej,uej,vej,wej)

      if(myid.eq.0) print *,'after passing jet electrons'

c       if (nstep.eq.569) then
c        print *, myid,'lecj',ions,lecs,ionj,lecj
c        stop
c       end if


c  BOUNDARY CONDITIONS FOR PARTICLES AND CURRENT DEPOSITION
c  current smoothing (filtering) is applied here after current deposition
c  from all particles; this saves computational time because particle number
c  is much larger than grid points number and particles are uniformly distributed
c  increaments to the electric fields from current deposition are stored in
c  "dex, dey, dez" arrays as referenced now by "Split_..." subroutines, and
```

```
c  by "xsplit,..., depsit", "Split_..." updated for boundary conditions handling
c  ** there is no need for "Clear_ghost" routine now **

c  ambient ions and electrons (periodicity in y- and z-direction applied)
c       if(myid.eq.0) print *, 'before split ambient'
      call Split_Ambient(ions,xi,yi,zi,ui,vi,wi,mb,dex,dey,dez,
     &                   mFx,mFy,mFz,mx,my,mz,qi,DHDx,DHDy,DHDz,
     &                  PVLeft,PVRght,PBFrnt,PBRear,PBBot,PBTop,
     &                           vithml,c,DT,refli,elosi,mi,is)
c        print *, myid,'after split ambient ions'
      call Split_Ambient(lecs,xe,ye,ze,ue,ve,we,mb,dex,dey,dez,
     &                   mFx,mFy,mFz,mx,my,mz,qe,DHDx,DHDy,DHDz,
     &                  PVLeft,PVRght,PBFrnt,PBRear,PBBot,PBTop,
     &                           vethml,c,DT,refle,elose,me,is)
      if(myid.eq.0) print *, 'after split ambient'

c  jet ions and electrons (periodicity in y- and z-direction applied)
      if (ionj.ne.0) then
       call Split_JET(ionj,xij,yij,zij,uij,vij,wij,mj,dex,dey,dez,
     &                     mFx,mFy,mFz,my,mz,qi,DHDx,DHDy,DHDz,DT)
      end if
```

```
       if (lecj.ne.0) then
       call Split_JET(lecj,xej,yej,zej,uej,vej,wej,mj,dex,dey,dez,
     &                  mFx,mFy,mFz,my,mz,qe,DHDx,DHDy,DHDz,DT)
      end if

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

c  pass contributions from current deposit to E-fields from ghost cells
c  to the cells in the appropriate domain's "particle core" before E-field
c  passing and applying periodic boundary conditions
c ** "addlayer" subroutine is embedded in here **
c !!! subroutine must be called with "FBDRxe" and "FBDLxe" !!!

c !!! ********************************************************** !!!
c  instead of E-field passing, current deposited in ghost cells is passed
c !!! ********************************************************** !!!
c  since smoothing is taken out of "depsit" subroutine, only 1 left and 2 right
c  guard cells are changed and need to be communicated - this is changed in
c  "E_Field_Passing_Add" in loops over "nguard": there is no need for 2+3 guard
c  cells anymore!!!
```

```
      call E_Field_Passing_Add(dex,dey,dez,mFx,mFy,mFz,mcol,mrow,
     &         dims,coords,FBDRxe,FBDRy,FBDRz,FBDLxe,FBDLy,FBDLz,
     &                  nleft,nright,nfront,nrear,nbottom,ntop)

c  multiple filtering
c !!! since smoothing spreads current to the neighboring cells, buffer zones!!!
c !!! in nonperiodic direction (x) must be larger to avoid accumulation of  !!!
c !!! currents at the Left and Right box boundaries - this depends on number!!!
c !!! of filterings: initial and boundary conditions must be changed        !!!

      do nsm = 1,nsmooth
c  current (longitudinal E-field increaments: DT*Ji) smoothing
c ** because current from ghost cells has been already passed, smoothing  **
c ** operates only on cells in the "particle core"; however, currents are **
c ** also spread to the nearest ghost cells, and additional passing is    **
c ** required every time the filter is applied -> "Current_Passing"       **

         if (nfilt(nsm) .eq. 1) then
          call Smooth_Current(dex,dey,dez,mFx,mFy,mFz,sm1,
     &                 FBDLx,FBDLy,FBDLz,FBDRx,FBDRy,FBDRz)
```

```
   else if (nfilt(nsm) .eq. 2) then
            call Smooth_Current(dex,dey,dez,mFx,mFy,mFz,sm2,
&                         FBDLx,FBDLy,FBDLz,FBDRx,FBDRy,FBDRz)
   else if (nfilt(nsm) .eq. 3) then
            call Smooth_Current(dex,dey,dez,mFx,mFy,mFz,sm3,
&                         FBDLx,FBDLy,FBDLz,FBDRx,FBDRy,FBDRz)
   end if

   call Current_Passing(dex,dey,dez,mFx,mFy,mFz,mcol,mrow,
&             dims,coords,FBDRxe,FBDRy,FBDRz,FBDLxe,FBDLy,FBDLz,
&                         nleft,nright,nfront,nrear,nbottom,ntop)
  end do

c  update longitudinal E-field components
  call E_field_update(ex,ey,ez,dex,dey,dez,mFx,mFy,mFz,
&                   FBDLx,FBDLy,FBDLz,FBDRx,FBDRy,FBDRz)
```

c4push "conduct" used near the left and right planes

```
C  Current density components jy=s*ey, jz=s*ez are spread laterally in the
C  proportion .25,.5,.25 required by our smoothing convention for all field
C  sources.
C=======  0.0625=0.25*0.25

      if (coords(1).eq.0) then
       Dratio=0.0625

       i=3
       do k=FBDLz,FBDRz
             do j=FBDLy,FBDRy
         Dcurr=Dratio*ey(i,j,k)
        ey(i-1,j,k)=ey(i-1,j,k)-Dcurr
        ey(i+1,j,k)=ey(i+1,j,k)-Dcurr
        ey(i,  j,k)=ey(i,  j,k)-2.0*Dcurr
         Dcurr=Dratio*ez(i,j,k)
        ez(i-1,j,k)=ez(i-1,j,k)-Dcurr
        ez(i+1,j,k)=ez(i+1,j,k)-Dcurr
                 ez(i,  j,k)=ez(i,  j,k)-2.0*Dcurr
```

```
            end do
           end do
          end if

if (coords(1).eq.(Npx-1)) then
 Dratio=0.0625

 i=nFx+3
 do k=FBDLz,FBDRz
        do j=FBDLy,FBDRy
    Dcurr=Dratio*ey(i,j,k)
   ey(i-1,j,k)=ey(i-1,j,k)-Dcurr
   ey(i+1,j,k)=ey(i+1,j,k)-Dcurr
   ey(i,  j,k)=ey(i,  j,k)-2.0*Dcurr
    Dcurr=Dratio*ez(i,j,k)
   ez(i-1,j,k)=ez(i-1,j,k)-Dcurr
   ez(i+1,j,k)=ez(i+1,j,k)-Dcurr
             ez(i,  j,k)=ez(i,  j,k)-2.0*Dcurr
            end do
           end do
          end if
```

```
c  E-field passing and periodic boundary conditions for E-field
c ** periodic boundary conditions ("copylayr") combined in passing subroutine **
c !! in 1D parallel version periodicity was applied to all the guard layers  !!
c !! because they were cleared before - this is not necessary as only two    !!
c !! layers must be made periodic to ensure the same conditions in MOVER     !!
c !! for particles crossing the transversal boundaries, and there is no need !!
c !! for pushing field elements in y and z-dir from 1 (up to mFi) in side    !!
c !! domains because the box is periodic in these directions                 !!
c       if(myid.eq.0) print *,'before e-field passing'
c4push       call Field_passing(ex,ey,ez,mFx,mFy,mFz,mc,mrl,mrh,
c4push     &               dims,coords,FBDLx,FBDLy,FBDLz,FBDRx,FBDRy,FBDRz,
c4push     &                        nleft,nright,nfront,nrear,nbottom,ntop)
      call Field_passing2(ex,ey,ez,mFx,mFy,mFz,mc2,mrh2,
     &            dims,coords,FBDLxp,FBDLy,FBDLz,FBDRxp,FBDRy,FBDRz,
     &                     nleft,nright,nfront,nrear,nbottom,ntop)

      if(myid.eq.0) print *, 'after e-field passing'

cJET
c  injects jet, according to the density, velocity, and time-step settings
c  injection every 3rd time-step, starting from nstep=1
```

```
c ** only most lefthand processors calculate this part **
c      if(mod((nstep-1),3).eq.0) then
c NewKen
      if(mod((nstep-1),njskip).eq.0) then
       if (coords(1).eq.0) then
        call Jet_injection(ionj,lecj,mj,PBFrnt,PBRear,PBBot,PBTop,
     &                  vijet,vejet,vithmj,vethmj,
     &                  xj0,yj0,zj0,dlxj,dlyj,dlzj,lyj1,lzj1,
     &     xij,yij,zij,uij,vij,wij,xej,yej,zej,uej,vej,wej,c,is)
       end if
      end if

c  write particle number
             open(25,file=ndiag//num,status='old',position='append')
             write(25,111) nstep,ions,lecs,ionj,ionj-ionj0,
     &                            lecj,lecj-lecj0
             close(25)
             ionj0=ionj
             lecj0=lecj

            if ((ions .gt. mb) .or. (lecs .gt. mb)) stop 'ambient crash'
            if ((ionj .gt. mj) .or. (lecj .gt. mj)) stop 'CR crash'
```

```
c  sorting particles in order they were arranged initially
        if(nstep.gt.400 .and. mod(nstep,100).eq.0) then
         if(myid.eq.0) print *, 'sorting of particles '
     call Sort_particles(ions,xi,yi,zi,ui,vi,wi,mb,nFy,nFz,
   &                           DHDx,DHDy,DHDz)
     call Sort_particles(lecs,xe,ye,ze,ue,ve,we,mb,nFy,nFz,
   &                           DHDx,DHDy,DHDz)
     if (ionj.ne.0) then
         call Sort_particles(ionj,xij,yij,zij,uij,vij,wij,mj,
   &                      nFy,nFz,DHDx,DHDy,DHDz)
     end if
         if (lecj.ne.0) then
      call Sort_particles(lecj,xej,yej,zej,uej,vej,wej,mj,
   &                      nFy,nFz,DHDx,DHDy,DHDz)
     end if
        end if

c  data are dumped here every 100 time step, together with partial output for
c  the fields and particle densities, currents, velocity distributions etc.
```

```
 if(mod(nstep,2).eq.0 .and. (last-nstep).ge.2) then
    if(myid.eq.0) print *, 'partial dumping of data'
    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

c ** each processor associates the same unit number with a different file **
c ** these files need to be deleted by hand if do not needed **
    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
c       write(7) bx,by,bz
      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(8) ex,ey,ez
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)
ccPconv_corrected
      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)
ccPconv_corrected
        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)
 end if
```