```
!-----------------------------------------------------------------------
! Skeleton 2-1/2D Electromagnetic MPI PIC code
! written by Viktor K. Decyk, UCLA
      program pbpic2
      use pbpush2_h
      use pplib2_h
      implicit none
      integer, parameter :: indx =   9, indy =   9
      integer, parameter :: npx =  3072, npy =   3072
      integer, parameter :: ndim = 3
      real, parameter :: tend = 10.0, dt = 0.04, qme = -1.0
!     real, parameter :: tend = 10.0, dt = 0.025, qme = -1.0
      real, parameter :: vtx = 1.0, vty = 1.0, vx0 = 0.0, vy0 = 0.0
      real, parameter :: vtz = 1.0, vz0 = 0.0
      real :: ax = .912871, ay = .912871, ci = 0.1
! idimp = dimension of phase space = 5
! sortime = number of time steps between standard electron sorting
! relativity = (no,yes) = (0,1) = relativity is used
      integer :: idimp = 5, ipbc = 1, sortime = 50, relativity = 1
! idps = number of partition boundaries
      integer :: idps = 2
      real :: wke = 0.0, we = 0.0, wf = 0.0, wm = 0.0, wt = 0.0
! declare scalars for standard code
      integer :: nx, ny, nxh, nyh, nxe, nye, nxeh, nnxe, nxyh, nxhy
      integer :: ny1, ntime, nloop, isign, ierr
      real :: qbme, affp, dth
      double precision :: np
!
! declare scalars for MPI code
      integer :: ntpose = 1
      integer :: nvp, idproc, kstrt, npmax, kxp, kyp, nypmx, nypmn
      integer :: nyp, noff, npp, nps, nbmax, ntmax
!
! declare arrays for standard code
      real, dimension(:,:), pointer :: part, part2, tpart
      real, dimension(:,:), pointer :: qe
      real, dimension(:,:,:), pointer :: cue, fxyze, bxyze
      complex, dimension(:,:,:), pointer :: exyz, bxyz
      complex, dimension(:,:), pointer :: qt
      complex, dimension(:,:,:), pointer :: cut, fxyt, bxyt
      complex, dimension(:,:), pointer :: ffc
      integer, dimension(:), pointer :: mixup
      complex, dimension(:), pointer :: sct
      integer, dimension(:), pointer :: ihole
      integer, dimension(:), pointer :: npic
      real, dimension(7) :: wtot, work
      integer, dimension(7) :: info
!
! declare arrays for MPI code
     complex, dimension(:,:,:), pointer :: bs, br
      real, dimension(:,:), pointer :: sbufl, sbufr, rbufl, rbufr
      real, dimension(:), pointer  :: edges
      real, dimension(:), pointer  :: scr
!
```

```
! declare and initialize timing data
      real :: time
      integer, dimension(4) :: itime
      real :: tdpost = 0.0, tguard = 0.0, ttp = 0.0, tfield = 0.0
      real :: tdjpost = 0.0, tpush = 0.0, tsort = 0.0, tmov = 0.0
      real, dimension(2) :: tfft
      double precision :: dtime
!
! initialize scalars for standard code
      np =  dble(npx)*dble(npy)
      nx = 2**indx; ny = 2**indy; nxh = nx/2; nyh = ny/2
      nxe = nx + 2; nye = ny + 2; nxeh = nxe/2; nnxe = ndim*nxe
      nxyh = max(nx,ny)/2; nxhy = max(nxh,ny); ny1 = ny + 1
      nloop = tend/dt + .0001; ntime = 0
      qbme = qme
      affp = dble(nx)*dble(ny)/np
      dth = 0.0
!
! nvp = number of distributed memory nodes
! initialize for distributed memory parallel processing
      call PPINIT2(idproc,nvp)
      kstrt = idproc + 1
! check if too many processors
      if (nvp > ny) then
         if (kstrt==1) then
         write (*,*) 'Too many processors requested: ny, nvp=', ny, nvp
         endif
         go to 3000
      endif

! initialize data for MPI code
      allocate(edges(idps))
! calculate partition variables: edges, nyp, noff, nypmx
! edges(1:2) = lower:upper boundary of particle partition
! nyp = number of primary (complete) gridpoints in particle partition
! noff = lowermost global gridpoint in particle partition
! nypmx = maximum size of particle partition, including guard cells
! nypmn = minimum value of nyp
      call PDICOMP2L(edges,nyp,noff,nypmx,nypmn,ny,kstrt,nvp,idps)
      if (nypmn < 1) then
         if (kstrt==1) then
            write (*,*) 'combination not supported nvp, ny =',nvp,ny
         endif
         go to 3000
      endif
!
! initialize additional scalars for MPI code
! kxp = number of complex grids in each field partition in x direction
      kxp = (nxh - 1)/nvp + 1
! kyp = number of complex grids in each field partition in y direction
      kyp = (ny - 1)/nvp + 1
! npmax = maximum number of electrons in each partition
      npmax = (np/nvp)*1.25
! nbmax = size of buffer for passing particles between processors
```

```
      nbmax = 0.1*npmax
! ntmax = size of ihole buffer for particles leaving processor
      ntmax = 2*nbmax
!
! allocate and initialize data for standard code
      allocate(part(idimp,npmax),part2(idimp,npmax))
      allocate(qe(nxe,nypmx),fxyze(ndim,nxe,nypmx))
      allocate(cue(ndim,nxe,nypmx),bxyze(ndim,nxe,nypmx))
      allocate(exyz(ndim,nye,kxp),bxyz(ndim,nye,kxp))
      allocate(qt(nye,kxp),fxyt(ndim,nye,kxp))
      allocate(cut(ndim,nye,kxp),bxyt(ndim,nye,kxp))
      allocate(ffc(nyh,kxp),mixup(nxhy),sct(nxyh))
      allocate(ihole(ntmax+1),npic(nypmx))
!
! allocate and initialize data for MPI code
      allocate(bs(ndim,kxp,kyp),br(ndim,kxp,kyp))
      allocate(sbufl(idimp,nbmax),sbufr(idimp,nbmax))
      allocate(rbufl(idimp,nbmax),rbufr(idimp,nbmax))
      allocate(scr(ndim*nxe))
!
! prepare fft tables
      call WPFFT2RINIT(mixup,sct,indx,indy,nxhy,nxyh)
! calculate form factors
      isign = 0
      call PPOIS23(qt,fxyt,isign,ffc,ax,ay,affp,we,nx,ny,kstrt,nye,kxp, &
     &nyh)
! initialize electrons
      nps = 1
      npp = 0
      call PDISTR2H(part,edges,npp,nps,vtx,vty,vtz,vx0,vy0,vz0,npx,npy, &
     &nx,ny,idimp,npmax,idps,ipbc,ierr)
! check for particle initialization error
      if (ierr /= 0) then
         if (kstrt==1) then
            write (*,*) 'particle initialization error: ierr=', ierr
         endif
         go to 3000
      endif
!
! initialize transverse electromagnetic fields
      exyz = cmplx(0.0,0.0)
      bxyz = cmplx(0.0,0.0)
!
      if (dt > 0.45*ci) then
         if (kstrt==1) then
            write (*,*) 'Warning: Courant condition may be exceeded!'
         endif
      endif
!
! * * * start main iteration loop * * *
!
  500 if (nloop <= ntime) go to 2000
!     if (kstrt==1) write (*,*) 'ntime = ', ntime
!
```

```fortran
! deposit current with standard procedure: updates part, cue, ihole
      call dtimer(dtime,itime,-1)
      cue = 0.0
      if (relativity==1) then
         call PPGRJPOST2L(part,cue,edges,npp,noff,ihole,qme,dth,ci,nx,ny&
     &,idimp,npmax,nxe,nypmx,idps,ntmax,ipbc)
!        call PPGSRJPOST2L(part,cue,edges,npp,noff,ihole,qme,dth,ci,nx, &
!    &ny,idimp,npmax,nxe,nxe*nypmx,idps,ntmax,ipbc)
      else
         call PPGJPOST2L(part,cue,edges,npp,noff,ihole,qme,dth,nx,ny,   &
     &idimp,npmax,nxe,nypmx,idps,ntmax,ipbc)
!        call PPGSJPOST2L(part,cue,edges,npp,noff,ihole,qme,dth,nx,ny,  &
!    &idimp,npmax,nxe,nxe*nypmx,idps,ntmax,ipbc)
      endif
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tdjpost = tdjpost + time
! check for ihole overflow error
      if (ihole(1) < 0) then
         ierr = -ihole(1)
         write (*,*) kstrt,'ihole overflow error: ntmax,ih=', ntmax,ierr
         call PPABORT
         go to 3000
      endif
!
! move electrons into appropriate spatial regions: updates part, npp
      call dtimer(dtime,itime,-1)
      call PPMOVE2(part,edges,npp,sbufr,sbufl,rbufr,rbufl,ihole,ny,kstrt&
     &,nvp,idimp,npmax,idps,nbmax,ntmax,info)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tmov = tmov + time
! check for particle manager error
      if (info(1) /= 0) then
         ierr = info(1)
         if (kstrt==1) then
            write (*,*) 'current particle manager error: ierr=', ierr
         endif
         go to 3000
      endif
!
! deposit charge with standard procedure: updates qe
      call dtimer(dtime,itime,-1)
      qe = 0.0
      call PPGPOST2L(part,qe,npp,noff,qme,idimp,npmax,nxe,nypmx)
!     call PPGSPOST2L(part,qe,npp,noff,qme,idimp,npmax,nxe,nxe*nypmx)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tdpost = tdpost + time
!
! add guard cells with standard procedure: updates cue, qe
      call dtimer(dtime,itime,-1)
      call PPACGUARD2XL(cue,nyp,nx,ndim,nxe,nypmx)
      call PPNACGUARD2L(cue,scr,nyp,nx,ndim,kstrt,nvp,nxe,nypmx)
      call PPAGUARD2XL(qe,nyp,nx,nxe,nypmx)
      call PPNAGUARD2L(qe,scr,nyp,nx,kstrt,nvp,nxe,nypmx)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tguard = tguard + time
!
! transform charge to fourier space with standard procedure: updates qt
! modifies qe
      call dtimer(dtime,itime,-1)
      isign = -1
      call WPPFFT2R(qe,qt,bs,br,isign,ntpose,mixup,sct,ttp,indx,indy,   &
     &kstrt,nvp,nxeh,nye,kxp,kyp,nypmx,nxhy,nxyh)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tfft(1) = tfft(1) + time
      tfft(2) = tfft(2) + ttp
!
! transform current to fourier space with standard procedure: updates cut
! modifies cue
      call dtimer(dtime,itime,-1)
      isign = -1
      call WPPFFT2R3(cue,cut,bs,br,isign,ntpose,mixup,sct,ttp,indx,indy,&
     &kstrt,nvp,nxeh,nye,kxp,kyp,nypmx,nxhy,nxyh)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tfft(1) = tfft(1) + time
      tfft(2) = tfft(2) + ttp
!
! take transverse part of current with standard procedure: updates cut
      call dtimer(dtime,itime,-1)
      call PPCUPERP2(cut,nx,ny,kstrt,nye,kxp)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tfield = tfield + time
!
! calculate electromagnetic fields in fourier space with standard
! procedure: updates exyz, bxyz
      call dtimer(dtime,itime,-1)
      if (ntime==0) then
         call IPPBPOISP23(cut,bxyz,ffc,ci,wm,nx,ny,kstrt,nye,kxp,nyh)
         wf = 0.0
         dth = 0.5*dt
      else
         call PPMAXWEL2(exyz,bxyz,cut,ffc,affp,ci,dt,wf,wm,nx,ny,kstrt, &
     &nye,kxp,nyh)
      endif
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tfield = tfield + time
!
! calculate force/charge in fourier space with standard procedure:
! updates fxyt
      call dtimer(dtime,itime,-1)
      isign = -1
      call PPOIS23(qt,fxyt,isign,ffc,ax,ay,affp,we,nx,ny,kstrt,nye,kxp, &
     &nyh)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tfield = tfield + time
!
! add longitudinal and transverse electric fields with standard
! procedure: updates fxyt
      call dtimer(dtime,itime,-1)
      isign = 1
      call PPEMFIELD2(fxyt,exyz,ffc,isign,nx,ny,kstrt,nye,kxp,nyh)
! copy magnetic field with standard procedure: updates bxyt
      isign = -1
      call PPEMFIELD2(bxyt,bxyz,ffc,isign,nx,ny,kstrt,nye,kxp,nyh)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tfield = tfield + time
!
! transform force to real space with standard procedure: updates fxyze
! modifies fxyt
      call dtimer(dtime,itime,-1)
      isign = 1
      call WPPFFT2R3(fxyze,fxyt,bs,br,isign,ntpose,mixup,sct,ttp,indx,  &
     &indy,kstrt,nvp,nxeh,nye,kxp,kyp,nypmx,nxhy,nxyh)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tfft(1) = tfft(1) + time
      tfft(2) = tfft(2) + ttp
!
! transform magnetic field to real space with standard procedure:
! updates bxyze, modifies bxyt
      call dtimer(dtime,itime,-1)
      isign = 1
      call WPPFFT2R3(bxyze,bxyt,bs,br,isign,ntpose,mixup,sct,ttp,indx,  &
     &indy,kstrt,nvp,nxeh,nye,kxp,kyp,nypmx,nxhy,nxyh)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tfft(1) = tfft(1) + time
      tfft(2) = tfft(2) + ttp
!
! copy guard cells with standard procedure: updates fxyze, bxyze
      call dtimer(dtime,itime,-1)
      call PPNCGUARD2L(fxyze,nyp,kstrt,nvp,nnxe,nypmx)
      call PPCGUARD2XL(fxyze,nyp,nx,ndim,nxe,nypmx)
      call PPNCGUARD2L(bxyze,nyp,kstrt,nvp,nnxe,nypmx)
      call PPCGUARD2XL(bxyze,nyp,nx,ndim,nxe,nypmx)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tguard = tguard + time
!
! push particles: updates part, wke, and ihole
      call dtimer(dtime,itime,-1)
      wke = 0.0
      if (relativity==1) then
```

```
         call PPGRBPUSH23L(part,fxyze,bxyze,edges,npp,noff,ihole,qbme,dt&
     &,dth,ci,wke,nx,ny,idimp,npmax,nxe,nypmx,idps,ntmax,ipbc)
!        call PPGSRBPUSH23L(part,fxyze,bxyze,edges,npp,noff,ihole,qbme, &
!    &dt,dth,ci,wke,nx,ny,idimp,npmax,nxe,nxe*nypmx,idps,ntmax,ipbc)
      else
         call PPGBPUSH23L(part,fxyze,bxyze,edges,npp,noff,ihole,qbme,dt,&
     &dth,wke,nx,ny,idimp,npmax,nxe,nypmx,idps,ntmax,ipbc)
!        call PPGSBPUSH23L(part,fxyze,bxyze,edges,npp,noff,ihole,qbme,dt&
!    &,dth,wke,nx,ny,idimp,npmax,nxe,nxe*nypmx,idps,ntmax,ipbc)
      endif
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tpush = tpush + time
! check for ihole overflow error
      if (ihole(1) < 0) then
         ierr = -ihole(1)
         write (*,*) kstrt,'ihole overflow error: ntmax,ih=', ntmax,ierr
         call PPABORT
         go to 3000
      endif
!
! move electrons into appropriate spatial regions: updates part, npp
      call dtimer(dtime,itime,-1)
      call PPMOVE2(part,edges,npp,sbufr,sbufl,rbufr,rbufl,ihole,ny,kstrt&
     &,nvp,idimp,npmax,idps,nbmax,ntmax,info)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tmov = tmov + time
! check for particle manager error
      if (info(1) /= 0) then
         ierr = info(1)
         if (kstrt==1) then
            write (*,*) 'push particle manager error: ierr=', ierr
         endif
         go to 3000
      endif
!
! sort particles for standard code: updates part
      if (sortime > 0) then
         if (mod(ntime,sortime)==0) then
            call dtimer(dtime,itime,-1)
            call PPDSORTP2YL(part,part2,npic,npp,noff,nyp,idimp,npmax,  &
     &nypmx)
! exchange pointers
            tpart => part
            part => part2
            part2 => tpart
            call dtimer(dtime,itime,1)
            time = real(dtime)
            tsort = tsort + time
         endif
      endif
!
! energy diagnostic
```

```
      wt = we + wf + wm
      wtot(1) = wt
      wtot(2) = wke
      wtot(3) = 0.0
      wtot(4) = wke + wt
      wtot(5) = we
      wtot(6) = wf
      wtot(7) = wm
      call PPSUM(wtot,work,7)
      wke = wtot(2)
      we = wtot(5)
      wf = wtot(6)
      wm = wtot(7)
      if (ntime==0) then
         if (kstrt.eq.1) then
            wt = we + wf + wm
            write (*,*) 'Initial Total Field, Kinetic and Total Energies&
     &:'
            write (*,'(3e14.7)') wt, wke, wke + wt
            write (*,*) 'Initial Electrostatic, Transverse Electric and &
     &Magnetic Field Energies:'
            write (*,'(3e14.7)') we, wf, wm
         endif
      endif
      ntime = ntime + 1
      go to 500
 2000 continue
!
! * * * end main iteration loop * * *
!
      if (kstrt.eq.1) then
         write (*,*) 'ntime, relativity = ', ntime, relativity
         write (*,*) 'MPI nodes nvp = ', nvp
         wt = we + wf + wm
         write (*,*) 'Final Total Field, Kinetic and Total Energies:'
         write (*,'(3e14.7)') wt, wke, wke + wt
         write (*,*) 'Final Electrostatic, Transverse Electric and Magne&
     &tic Field Energies:'
         write (*,'(3e14.7)') we, wf, wm
!
         write (*,*)
         write (*,*) 'deposit time = ', tdpost
         write (*,*) 'current deposit time = ', tdjpost
         tdpost = tdpost + tdjpost
         write (*,*) 'total deposit time = ', tdpost
         write (*,*) 'guard time = ', tguard
         write (*,*) 'solver time = ', tfield
         write (*,*) 'fft and transpose time = ', tfft(1), tfft(2)
         write (*,*) 'push time = ', tpush
         write (*,*) 'particle move time = ', tmov
         write (*,*) 'sort time = ', tsort
         tfield = tfield + tguard + tfft(1)
         write (*,*) 'total solver time = ', tfield
         time = tdpost + tpush + tmov + tsort
         write (*,*) 'total particle time = ', time
         wt = time + tfield
         write (*,*) 'total time = ', wt
         write (*,*)
!
         wt = 1.0e+09/(real(nloop)*real(np))
         write (*,*) 'Push Time (nsec) = ', tpush*wt
         write (*,*) 'Deposit Time (nsec) = ', tdpost*wt
         write (*,*) 'Sort Time (nsec) = ', tsort*wt
         write (*,*) 'Total Particle Time (nsec) = ', time*wt
      endif
!
 3000 continue
      call PPEXIT()
      stop
      end program
```