```
c-----------------------------------------------------------------------
      subroutine PPGPUSH2L(part,fxy,edges,npp,noff,ihole,qbm,dt,ek,nx,ny
     1,idimp,npmax,nxv,nypmx,idps,ntmax,ipbc)
c for 2d code, this subroutine updates particle co-ordinates and
c velocities using leap-frog scheme in time and first-order linear
c interpolation in space, with various boundary conditions
c also determines list of particles which are leaving this processor
c scalar version using guard cells, for distributed data
c 42 flops/particle, 12 loads, 4 stores
c input: all except ihole, output: part, ihole, ek
c equations used are:
c vx(t+dt/2) = vx(t-dt/2) + (q/m)*fx(x(t),y(t))*dt,
c vy(t+dt/2) = vy(t-dt/2) + (q/m)*fy(x(t),y(t))*dt,
c where q/m is charge/mass, and
c x(t+dt) = x(t) + vx(t+dt/2)*dt, y(t+dt) = y(t) + vy(t+dt/2)*dt
c fx(x(t),y(t)) and fy(x(t),y(t)) are approximated by interpolation from
c the nearest grid points:
c fx(x,y) = (1-dy)*((1-dx)*fx(n,m)+dx*fx(n+1,m)) + dy*((1-dx)*fx(n,m+1)
c    + dx*fx(n+1,m+1))
c fy(x,y) = (1-dy)*((1-dx)*fy(n,m)+dx*fy(n+1,m)) + dy*((1-dx)*fy(n,m+1)
c    + dx*fy(n+1,m+1))
c where n,m = leftmost grid points and dx = x-n, dy = y-m
c part(1,n) = position x of particle n in partition
c part(2,n) = position y of particle n in partition
c part(3,n) = velocity vx of particle n in partition
c part(4,n) = velocity vy of particle n in partition
c fxy(1,j,k) = x component of force/charge at grid (j,kk)
c fxy(2,j,k) = y component of force/charge at grid (j,kk)
c in other words, fxy are the convolutions of the electric field
c over the particle shape, where kk = k + noff - 1
c edges(1:2) = lower:upper boundary of particle partition
c npp = number of particles in partition
c noff = lowermost global gridpoint in particle partition.
c ihole = location of hole left in particle arrays
c ihole(1) = ih, number of holes left (error, if negative)
c qbm = particle charge/mass
c dt = time interval between successive calculations
c kinetic energy/mass at time t is also calculated, using
c ek = .125*sum((vx(t+dt/2)+vx(t-dt/2))**2+(vy(t+dt/2)+vy(t-dt/2))**2)
c nx/ny = system length in x/y direction
c idimp = size of phase space = 4
c npmax = maximum number of particles in each partition
c nxv = first dimension of field array, must be >= nx+1
c nypmx = maximum size of particle partition, including guard cells.
c idps = number of partition boundaries
c ntmax = size of hole array for particles leaving processors
c ipbc = particle boundary condition = (0,1,2,3) =
c (none,2d periodic,2d reflecting,mixed reflecting/periodic)
      implicit none
      integer npp, noff, nx, ny, idimp, npmax, idps, ntmax, nxv, nypmx
      integer ipbc
      real qbm, dt, ek
      real part, fxy, edges
      integer ihole
```

```
      dimension part(idimp,npmax), fxy(2,nxv,nypmx)
      dimension edges(idps), ihole(ntmax+1)
c local data
      integer mnoff, j, nn, mm, np, mp, ih, nh
      real qtm, edgelx, edgely, edgerx, edgery, dxp, dyp, amx, amy
      real dx, dy
      double precision sum1
      qtm = qbm*dt
      sum1 = 0.0d0
c set boundary values
      edgelx = 0.0
      edgely = 1.0
      edgerx = real(nx)
      edgery = real(ny-1)
      if ((ipbc.eq.2).or.(ipbc.eq.3)) then
         edgelx = 1.0
         edgerx = real(nx-1)
      endif
      mnoff = noff - 1
      ih = 0
      nh = 0
      do 10 j = 1, npp
c find interpolation weights
      nn = part(1,j)
      mm = part(2,j)
      dxp = part(1,j) - real(nn)
      dyp = part(2,j) - real(mm)
      nn = nn + 1
      mm = mm - mnoff
      amx = 1.0 - dxp
      mp = mm + 1
      amy = 1.0 - dyp
      np = nn + 1
c find acceleration
      dx = dyp*(dxp*fxy(1,np,mp) + amx*fxy(1,nn,mp))
     1   + amy*(dxp*fxy(1,np,mm) + amx*fxy(1,nn,mm))
      dy = dyp*(dxp*fxy(2,np,mp) + amx*fxy(2,nn,mp))
     1   + amy*(dxp*fxy(2,np,mm) + amx*fxy(2,nn,mm))
c new velocity
      dx = part(3,j) + qtm*dx
      dy = part(4,j) + qtm*dy
c average kinetic energy
      sum1 = sum1 + (dx + part(3,j))**2 + (dy + part(4,j))**2
      part(3,j) = dx
      part(4,j) = dy
c new position
      dx = part(1,j) + dx*dt
      dy = part(2,j) + dy*dt
c periodic boundary conditions in x
      if (ipbc.eq.1) then
         if (dx.lt.edgelx) dx = dx + edgerx
         if (dx.ge.edgerx) dx = dx - edgerx
c reflecting boundary conditions
      else if (ipbc.eq.2) then
```

```
         if ((dx.lt.edgelx).or.(dx.ge.edgerx)) then
            dx = part(1,j)
            part(3,j) = -part(3,j)
         endif
         if ((dy.lt.edgely).or.(dy.ge.edgery)) then
            dy = part(2,j)
            part(4,j) = -part(4,j)
         endif
c mixed reflecting/periodic boundary conditions
      else if (ipbc.eq.3) then
         if ((dx.lt.edgelx).or.(dx.ge.edgerx)) then
            dx = part(1,j)
            part(3,j) = -part(3,j)
         endif
      endif
c find particles out of bounds
      if ((dy.lt.edges(1)).or.(dy.ge.edges(2))) then
         ih = ih + 1
         if (ih.le.ntmax) then
            ihole(ih+1) = j
         else
            nh = 1
         endif
      endif
c set new position
      part(1,j) = dx
      part(2,j) = dy
   10 continue
c set end of file flag
      if (nh.gt.0) ih = -ih
      ihole(1) = ih
c normalize kinetic energy
      ek = ek + 0.125*sum1
      return
      end
```

```
c-----------------------------------------------------------------------
      subroutine PPGPOST2L(part,q,npp,noff,qm,idimp,npmax,nxv,nypmx)
c for 2d code, this subroutine calculates particle charge density
c using first-order linear interpolation, periodic boundaries
c scalar version using guard cells, for distributed data
c 17 flops/particle, 6 loads, 4 stores
c input: all, output: q
c charge density is approximated by values at the nearest grid points
c q(n,m)=qm*(1.-dx)*(1.-dy)
c q(n+1,m)=qm*dx*(1.-dy)
c q(n,m+1)=qm*(1.-dx)*dy
c q(n+1,m+1)=qm*dx*dy
c where n,m = leftmost grid points and dx = x-n, dy = y-m
c part(1,n) = position x of particle n in partition
c part(2,n) = position y of particle n in partition
c q(j,k) = charge density at grid point (j,kk),
c where kk = k + noff - 1
c npp = number of particles in partition
c noff = lowermost global gridpoint in particle partition.
c qm = charge on particle, in units of e
c idimp = size of phase space = 4
c npmax = maximum number of particles in each partition
c nxv = first dimension of charge array, must be >= nx+1
c nypmx = maximum size of particle partition, including guard cells.
      implicit none
      integer npp, noff, idimp, npmax, nxv, nypmx
      real qm
      real part, q
      dimension part(idimp,npmax), q(nxv,nypmx)
c local data
      integer mnoff, j, nn, np, mm, mp
      real dxp, dyp, amx, amy
      mnoff = noff - 1
      do 10 j = 1, npp
c find interpolation weights
      nn = part(1,j)
      mm = part(2,j)
      dxp = qm*(part(1,j) - real(nn))
      dyp = part(2,j) - real(mm)
      nn = nn + 1
      mm = mm - mnoff
      amx = qm - dxp
      mp = mm + 1
      amy = 1.0 - dyp
      np = nn + 1
c deposit charge
      q(np,mp) = q(np,mp) + dxp*dyp
      q(nn,mp) = q(nn,mp) + amx*dyp
      q(np,mm) = q(np,mm) + dxp*amy
      q(nn,mm) = q(nn,mm) + amx*amy
   10 continue
      return
      end
```