```
c-----------------------------------------------------------------------
      subroutine VGPUSH2LT(part,fxy,qbm,dt,ek,idimp,nop,npe,nx,ny,nxv,
     1nyv,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 vectorizable version using guard cells
c 44 flops/particle, 12 loads, 4 stores
c input: all, output: part, 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(n,1) = position x of particle n
c part(n,2) = position y of particle n
c part(n,3) = velocity vx of particle n
c part(n,4) = velocity vy of particle n
c fxy(1,j,k) = x component of force/charge at grid (j,k)
c fxy(2,j,k) = y component of force/charge at grid (j,k)
c that is, convolution of electric field over particle shape
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 idimp = size of phase space = 4
c nop = number of particles
c npe = first dimension of particle array
c nx/ny = system length in x/y direction
c nxv = second dimension of field array, must be >= nx+1
c nyv = third dimension of field array, must be >= ny+1
c ipbc = particle boundary condition = (0,1,2,3) =
c (none,2d periodic,2d reflecting,mixed reflecting/periodic)
      implicit none
      integer idimp, nop, npe, nx, ny, nxv, nyv, ipbc
      real qbm, dt, ek
      real part, fxy
      dimension part(npe,idimp), fxy(2,nxv*nyv)
c local data
      integer npblk, lvect
      parameter(npblk=32,lvect=4)
      integer i, j, k, ipp, joff, nps, nn, mm
      real qtm, edgelx, edgely, edgerx, edgery, dxp, dyp, amx, amy
      real x, y, dx, dy, vx, vy
c scratch arrays
      integer n
      real s, t
```

```
      dimension n(npblk), s(npblk,lvect), t(npblk,2)
      double precision sum1
      qtm = qbm*dt
      sum1 = 0.0d0
c set boundary values
      edgelx = 0.0
      edgely = 0.0
      edgerx = real(nx)
      edgery = real(ny)
      if (ipbc.eq.2) then
         edgelx = 1.0
         edgely = 1.0
         edgerx = real(nx-1)
         edgery = real(ny-1)
      else if (ipbc.eq.3) then
         edgelx = 1.0
         edgerx = real(nx-1)
      endif
      ipp = nop/npblk
c outer loop over number of full blocks
      do 60 k = 1, ipp
      joff = npblk*(k - 1)
c inner loop over particles in block
      do 10 j = 1, npblk
c find interpolation weights
      x = part(j+joff,1)
      y = part(j+joff,2)
      nn = x
      mm = y
      dxp = x - real(nn)
      dyp = y - real(mm)
      n(j) = nn + nxv*mm
      amx = 1.0 - dxp
      amy = 1.0 - dyp
      s(j,1) = amx*amy
      s(j,2) = dxp*amy
      s(j,3) = amx*dyp
      s(j,4) = dxp*dyp
      t(j,1) = x
      t(j,2) = y
   10 continue
c find acceleration
      do 30 j = 1, npblk
      nn = n(j)
      mm = nn + nxv - 2
      dx = 0.0
      dy = 0.0
!dir$ ivdep
      do 20 i = 1, lvect
      if (i.gt.2) nn = mm
      dx = dx + fxy(1,i+nn)*s(j,i)
      dy = dy + fxy(2,i+nn)*s(j,i)
   20 continue
      s(j,1) = dx
```

```
      s(j,2) = dy
   30 continue
c new velocity
      do 40 j = 1, npblk
      x = t(j,1)
      y = t(j,2)
      dxp = part(j+joff,3)
      dyp = part(j+joff,4)
      vx = dxp + qtm*s(j,1)
      vy = dyp + qtm*s(j,2)
c average kinetic energy
      dxp = dxp + vx
      dyp = dyp + vy
      sum1 = sum1 + (dxp*dxp + dyp*dyp)
c new position
      s(j,1) = x + vx*dt
      s(j,2) = y + vy*dt
      s(j,3) = vx
      s(j,4) = vy
   40 continue
! check boundary conditions
!dir$ novector
      do 50 j = 1, npblk
      dx = s(j,1)
      dy = s(j,2)
      vx = s(j,3)
      vy = s(j,4)
c periodic boundary conditions
      if (ipbc.eq.1) then
         if (dx.lt.edgelx) dx = dx + edgerx
         if (dx.ge.edgerx) dx = dx - edgerx
         if (dy.lt.edgely) dy = dy + edgery
         if (dy.ge.edgery) dy = dy - edgery
c reflecting boundary conditions
      else if (ipbc.eq.2) then
         if ((dx.lt.edgelx).or.(dx.ge.edgerx)) then
            dx = t(j,1)
            vx = -vx
         endif
         if ((dy.lt.edgely).or.(dy.ge.edgery)) then
            dy = t(j,2)
            vy = -vy
         endif
c mixed reflecting/periodic boundary conditions
      else if (ipbc.eq.3) then
         if ((dx.lt.edgelx).or.(dx.ge.edgerx)) then
            dx = t(j,1)
            vx = -vx
         endif
         if (dy.lt.edgely) dy = dy + edgery
         if (dy.ge.edgery) dy = dy - edgery
      endif
c set new position
      part(j+joff,1) = dx
```

```
      part(j+joff,2) = dy
c set new velocity
      part(j+joff,3) = vx
      part(j+joff,4) = vy
   50 continue
   60 continue
      nps = npblk*ipp + 1
c loop over remaining particles
      do 70 j = nps, nop
c find interpolation weights
      x = part(j,1)
      y = part(j,2)
      nn = x
      mm = y
      dxp = x - real(nn)
      dyp = y - real(mm)
      nn = nn + nxv*mm + 1
      amx = 1.0 - dxp
      amy = 1.0 - dyp
c find acceleration
      dx = amx*fxy(1,nn)
      dy = amx*fxy(2,nn)
      dx = amy*(dxp*fxy(1,nn+1) + dx)
      dy = amy*(dxp*fxy(2,nn+1) + dy)
      vx = amx*fxy(1,nn+nxv)
      vy = amx*fxy(2,nn+nxv)
      dx = dx + dyp*(dxp*fxy(1,nn+1+nxv) + vx)
      dy = dy + dyp*(dxp*fxy(2,nn+1+nxv) + vy)
c new velocity
      dxp = part(j,3)
      dyp = part(j,4)
      vx = dxp + qtm*dx
      vy = dyp + qtm*dy
c average kinetic energy
      dxp = dxp + vx
      dyp = dyp + vy
      sum1 = sum1 + (dxp*dxp + dyp*dyp)
c new position
      dx = x + vx*dt
      dy = y + vy*dt
c periodic boundary conditions
      if (ipbc.eq.1) then
         if (dx.lt.edgelx) dx = dx + edgerx
         if (dx.ge.edgerx) dx = dx - edgerx
         if (dy.lt.edgely) dy = dy + edgery
         if (dy.ge.edgery) dy = dy - edgery
c reflecting boundary conditions
      else if (ipbc.eq.2) then
         if ((dx.lt.edgelx).or.(dx.ge.edgerx)) then
            dx = x
            vx = -vx
         endif
         if ((dy.lt.edgely).or.(dy.ge.edgery)) then
            dy = y
```

```
            vy = -vy
         endif
c mixed reflecting/periodic boundary conditions
      else if (ipbc.eq.3) then
         if ((dx.lt.edgelx).or.(dx.ge.edgerx)) then
            dx = x
            vx = -vx
         endif
         if (dy.lt.edgely) dy = dy + edgery
         if (dy.ge.edgery) dy = dy - edgery
      endif
c set new position
      part(j,1) = dx
      part(j,2) = dy
c set new velocity
      part(j,3) = vx
      part(j,4) = vy
   70 continue
c normalize kinetic energy
      ek = ek + 0.125*sum1
      return
      end
```

```
c-----------------------------------------------------------------------
      subroutine VGPOST2LT(part,q,qm,nop,npe,idimp,nxv,nyv)
c for 2d code, this subroutine calculates particle charge density
c using first-order linear interpolation, periodic boundaries
c vectorizable version using guard cells
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(n,1) = position x of particle n
c part(n,2) = position y of particle n
c q(j,k) = charge density at grid point j,k
c qm = charge on particle, in units of e
c nop = number of particles
c npe = first dimension of particle array
c idimp = size of phase space = 4
c nxv = first dimension of charge array, must be >= nx+1
c nyv = second dimension of charge array, must be >= ny+1
      implicit none
      integer nop, npe, idimp, nxv, nyv
      real qm
      real part, q
      dimension part(npe,idimp), q(nxv*nyv)
c local data
      integer npblk, lvect
      parameter(npblk=32,lvect=4)
      integer i, j, k, ipp, joff, nps, nn, mm
      real x, y, dxp, dyp, amx, amy
c scratch arrays
      integer n
      real s
      dimension n(npblk), s(npblk,lvect)
      ipp = nop/npblk
c outer loop over number of full blocks
      do 40 k = 1, ipp
      joff = npblk*(k - 1)
c inner loop over particles in block
      do 10 j = 1, npblk
c find interpolation weights
      x = part(j+joff,1)
      y = part(j+joff,2)
      nn = x
      mm = y
      dxp = qm*(x - real(nn))
      dyp = y - real(mm)
      n(j) = nn + nxv*mm
      amx = qm - dxp
      amy = 1.0 - dyp
      s(j,1) = amx*amy
      s(j,2) = dxp*amy
```

```
      s(j,3) = amx*dyp
      s(j,4) = dxp*dyp
   10 continue
c deposit charge
      do 30 j = 1, npblk
      nn = n(j)
      mm = nn + nxv - 2
!dir$ ivdep
      do 20 i = 1, lvect
      if (i.gt.2) nn = mm
      q(i+nn) = q(i+nn) + s(j,i)
   20 continue
   30 continue
   40 continue
      nps = npblk*ipp + 1
c loop over remaining particles
      do 50 j = nps, nop
c find interpolation weights
      x = part(j,1)
      y = part(j,2)
      nn = x
      mm = y
      dxp = qm*(x - real(nn))
      dyp = y - real(mm)
      nn = nn + nxv*mm + 1
      amx = qm - dxp
      amy = 1.0 - dyp
c deposit charge
      x = q(nn) + amx*amy
      y = q(nn+1) + dxp*amy
      q(nn) = x
      q(nn+1) = y
      x = q(nn+nxv) + amx*dyp
      y = q(nn+nxv+1) + dxp*dyp
      q(nn+nxv) = x
      q(nn+nxv+1) = y
   50 continue
      return
      end
```