The Code for Parallel and Random-sequential update rules.

(a) For Parallel Update rule

```
!define variables
integer :: N, lx, ntime
integer :: ii, jj, kk, iseed, temp, mm, iie
integer :: old, new, cnt, itime
real(8) :: xx(N,2), spin(N,2)
real(8) :: ran2, dx, dr, ff, nb, arg, B
real(8) :: r0, v0, pp, M, mag, r1, r2, zz, op

r0 = 0.01; v0 = 0.001

B = 2.4  ! inverse temperature

! Initialization of spins
do ii = 1, N
   xx(ii,old) = ran2(iseed)*lx
   r1 = ran2(iseed)*lx
   if(r1 .le. 0.5) then
      spin(ii,old) = 1.0
   else
      spin(ii,old) = -1.0
   end if
end do

! time loop
do itime = 1, ntime

  do ii = 1, N
    ff = 0.0
    cnt = 0
    do jj = 1, N
       dx = xx(jj,old) - xx(ii,old)
       if (abs(dx) .gt. lx*0.5) dx = lx - abs(dx)
       dr = sqrt(dx*dx)
       if(dr  .le.  r0) then
          ff = ff + spin(jj,old)
          cnt = cnt + 1
       end if
     end do
    nb = ff/dfloat(cnt) ! Net spin
    arg = spin(ii,old)*nb

    ! spin orientation update according to Metropolis algorithm
    if(arg .le. 0.0) then
       spin(ii,new) = -spin(ii,old)
    else
       pp = exp(-arg*B)
       r2 = ran2(iseed)*lx
       if(r2 .lt. pp) then
```

```
          spin(ii,new) = -spin(ii,old)
        else
          spin(ii,new) = spin(ii,old)
        end if
      end if
  end do


 ! Position update and calculation of order parameter
  M = 0.0
  do kk = 1, N
    M = M + spin(kk,new)
    xx(kk,new) = xx(kk,old) + v0*spin(kk,new)
   ! Periodic Boundary condition
    if(xx(kk,new) .ge. lx)  xx(kk,new) = xx(kk,new) - lx
    if(xx(kk,new) .lt. 0.0) xx(kk,new) = xx(kk,new) + lx
  end do

  mag = M/dfloat(N)

  temp = old
  old  = new
  new  = temp

end do  ! end time loop

END program
```

(b) For Random-sequential update

```
integer :: N,lx
integer :: ii, jj, kk, iseed, temp, cnt
integer :: itime, ntime, mm, iie
real(8) :: xx(N), spin(N)
real(8) :: ran2, dx, dr, ff, nb, arg, B
real(8) :: r0, v0, pp, M, mag, r1, r2, op

r0 = 0.01; v0 = 0.001
B = 2.0

! Initialization of spins
do ii = 1, N
    xx(ii) = ran2(iseed)*lx
    r1 = ran2(iseed)*lx
    if(r1  .le.  0.5) then
        spin(ii) = 1.0
    else
        spin(ii) = -1.0
  end if
end do
```

```
! time loop
do itime = 1, ntime

  M = 0.0
  do kk = 1, N
    ii = int(ran2(iseed)*N+1)

    ff = 0.0
    cnt = 0
    do jj = 1, N
       dx = xx(jj) - xx(ii)
       if (abs(dx) .gt. lx*0.5) dx = lx - abs(dx)
       dr = sqrt(dx*dx)
       if(dr .le. r0) then
          ff = ff + spin(jj)
          cnt = cnt + 1
       end if
    end do

    nb = ff/dfloat(cnt)  ! Net spin
    arg = spin(ii)*nb

    ! spin orientation update according to Metropolis algorithm
    if(arg .lt. 0.0) then
       spin(ii) = -spin(ii)
    else
       pp = exp(-arg*B)
       r2 = ran2(iseed)
       if(r2 .lt. pp) then
          spin(ii) = -spin(ii)
       else
          spin(ii) = spin(ii)
       end if
    end if

   ! Position update and calculation of order parameter
    xx(ii) = xx(ii) + v0*spin(ii)
    if(xx(ii) .ge. lx)  xx(ii) = xx(ii) - lx
    if(xx(ii) .lt. 0.0) xx(ii) = xx(ii) + lx

    M = M + spin(ii)
  end do
  mag = M/dfloat(N)

end do
end do

END program
```