Random walk

```
  ! Program to perform a random walk in two dimensions
! repeats for many realizations and accmulates r**2 statistics
! Also compute entropy vs. time

    IMPLICIT NONE
    INTEGER, PARAMETER :: Prec14=SELECTED_REAL_KIND(14)
    INTEGER :: i,j,is,ix,iy,it
    INTEGER ::  ne,nt
    REAL(Kind=Prec14) :: rnd,rnds,r2,entropy
    INTEGER, PARAMETER :: nx=200, ny=200, ixmin=-99, ixmax=100, iymin=-99, iymax=100 ! Lat
    INTEGER, PARAMETER :: nxg=8, nyg=8 ! grid sites
    INTEGER, PARAMETER :: ntmax=10000 ! maximum number of time steps
    INTEGER, PARAMETER :: nemax=100000  ! number of walks in ensemble
    INTEGER, PARAMETER :: ient=10 ! Number of steps to update p array
    INTEGER, PARAMETER :: ipmax=2000 ! Maximum number of time elements in p
    REAL(Kind=Prec14), DIMENSION(ntmax) ::r2a(ntmax) ! accumulated average of r**2
    REAL(Kind=Prec14), DIMENSION(nxg,nyg,ipmax) :: p ! Probability of visiting a site
    REAL(Kind=Prec14) :: t

!  set r2a=0, p=0

    r2a=0.0d0
    p=0
```

Random number generator

```
! initialize random number generator
        call RANDOM_SEED

50         call RANDOM_NUMBER(rnd)
        call RANDOM_NUMBER(rnds)
```

One random number (rnds) can be used to decide on +/- Step

Other random number (rnd) can be used to decide whether we move walker in x,y, or z direction

*Average r**2 for many random walks*

```
   do ne=1,nemax ! Nemax realizations of random walk

    do nt=1,ntmax

   …

60       r2=real(i)**2+real(j)**2
         r2a(nt)=r2a(nt)+r2

  enddo ! Loop over steps

 enddo ! Loop over random walks
```

Entropy averaged over many walks

```
if(mod(nt,ient).eq.0) then ! update array
 it=nt/ient+1
 ix=int(dfloat(i+99)*dfloat(nxg)/dfloat(nx))+1
 iy=int(dfloat(j+99)*dfloat(nyg)/dfloat(ny))+1
 p(ix,iy,it)=p(ix,iy,it)+1.0d0
endif
```

- First line to decide if we should update probability array p
- Need to find particular grid point ix,iy to update
- Increment p array for grid point ix,iy
- After ensemble of runs, accumulate and output the entropy
- Sum entropy up at each time it, over all nxg,nyg grid points:

```
if(p(ix,iy,it).gt.0.0d0) entropy=entropy-(p(ix,iy,it)/dfloat(nemax))*dlog(p(ix,iy,it)/dfloat(nemax))
```

Modification of homework 4….

- For problem 1, random walk, use declarations:

```
INTEGER, PARAMETER :: ntmax=1000 ! maximum number of time steps
INTEGER, PARAMETER :: nemax=100000  ! number of walks in ensemble
```

- In problem 1, output r2a averaged over the ensemble of walks at each time step
- Should be 1000 time points
- For problem 2, random walk with entropy calculation, use:

```
INTEGER, PARAMETER :: ntmax=10000 ! maximum number of time steps
INTEGER, PARAMETER :: nemax=100000  ! number of walks in ensemble
INTEGER, PARAMETER :: ient=10 ! Number of steps to update p array
```

- Hence, only include 10000 steps, compute probability and entropy every 10 steps
- Still 100000 walks in the ensemble