Lab VII : Random systems and random walk (in two dimensions)

Example 1 : Random walk in two dimensions with discrete steps

In the lecture, we saw how we can write a one dimensional random walk with a fixed step size of 1. In this exercise we are going to extend this to two dimensions. Just like we have done in the lecture, we are going to run the code for different numbers of step sizes and calculate the mean square displacement over a large set of random walks.

First let's write a function that outputs a single random walk. Because we are now in two dimensions, we need an x and a y coordinate of each displacement. So we'll have our function return two arrays, one for each coordinate. It would also be nice to see the particle execute the random walk, however because we are going to be calling this function several times, it's a good idea to make the plotting optional. Otherwise it would become terribly annoying. Let's give our function a rather lengthy but self-explanatory name and call it `random_walk_2d_single`.

Before you create the file `random_walk_2d_single.m`, download `rand_disc.m` from the Web because we going to be using this in our code.

```
## Produce a single random walk in two dimensions
## Usage : [x,y]=random_walk_single(Nsteps,plot_or_not)

function [x,y]=random_walk_single(Nsteps,plot_or_not)

  x=rand_disc(Nsteps,0.5);
  y=rand_disc(Nsteps,0.5);

  if ( strcmp(plot_or_not,"yes") )
    cumx=cumsum(x);
    cumy=cumsum(y);

    axis([min(cumx),max(cumx),min(cumy),max(cumy)]);
    for n=1:length(cumx)
      plot(cumx(1:n),cumy(1:n),'b-*');
      pause(0.1)
    endfor

  elseif ( strcmp(cplot,"no") )
    continue
  else
    printf("Wrong plot option. Available options are : yes and no.\n");
  endif

endfunction
```

What this function really does is executed on the first two lines where the random walk is generated, however, it performs a lot of additional tasks before returning the output. Let's examine the highlighted parts of the code :

- The argument `plot_or_not` is a string that accepts two values `yes` and `no`. String arguments cannot simply be compared using the `==` but a specialized string function, `strcmp` must be called.
- If a plot has been asked for, we first find the cumulative sum of all the steps using `cumsum`. This will help us to figure out the dimensions of the plot, which we set using the `axis` command.
- We then loop over `n`, creating a plot of the first `n` points at each step. This produces a plot which really looks like the time evolution of a random walk with connected steps.
- If no plot has been asked for the code does nothing but just `continue`s.
- If any option has been entered other than `yes` or `no`, then the code prints an error sign.