In-Class Work: Random Walks

Outline:
At the beginning of class, I will walk you through the tasks and solutions of last class's steps 2a-2d. We will do then some theory, and we will test the theoretical prediction with step 2e.

2. Random Walk in One Dimension

Next we will do important analysis on the random walk. To simplify the task (and yet being able to get the main concept) let us start with the random walk in one dimension.

2a. Write a program for a random walk in one dimension. In all following we assume $p = q = 0.5$, i.e. there is equal probability to jump to the right or to the left with jump size 1. Initialize with $x(t = 0) = 0$ (so x=0) and print t and x for each time step. Use `NSTEPS` = 500 time steps.
You find the solution to this in

```
~kvollmay/share.dir/inclass2023.dir/classrndwalk2a.py
```

(After copying the program, you can look at the resulting $x(t)$ with

```
./classrndwalk2a.py | xmgrace -pipe
```

2b. The next step is a preparation for 2c. Instead of printing every time step, print only once after $N =$`NSTEPS` time steps (i.e. after the time-loop) the resulting $x(N)$ and $(x(N))^2$. Increase `NSTEPS` to `NSTEPS=5000`.
You find the solution to this in

```
~kvollmay/share.dir/inclass2023.dir/classrndwalk2b.py
```

and you can confirm what it does with

```
./classrndwalk2b.py
```

2c. Now add a loop over simulation runs to your program from 2b. Each simulation run starts with $x = 0$ and gives you an $x($`NSTEPS`$)$ and an $x^2($`NSTEPS`$)$. First set `NSIMRUNS=10` and print for each simulation run $x($`NSTEPS`$)$ and $x^2($`NSTEPS`$)$. Next change your program such that it determines the average over `NSIMRUNS=10000` simulation runs of x and an x^2 and prints out the resulting averages $\langle x \rangle$ and $\langle x^2 \rangle$.
You find the solution to this in

```
~kvollmay/share.dir/inclass2023.dir/classrndwalk2c.py
```

You can run it with

```
./classrndwalk2c.py
```

2d. Next no longer use the number of steps `NSTEPS` as constant but instead add a loop over N=`nsteps` to your program of 2c. Loop `nsteps` from 100 to 2000 in steps of 100. For each `nstep` print out N=`nstep`, $\langle x \rangle$ and $\langle x^2 \rangle$. Look at the resulting $\langle x(N)$ and also $\langle x^2(N) \rangle$.
You find the solution to this in

```
~kvollmay/share.dir/inclass2023.dir/classrndwalk2d.py
```

This time the program takes a few minutes, so safe the data in a file,e.g.

```
./classrndwalk2d.py > dat2dout
```

Have a quick look at the data with

```
cat dat2dout
```

Now think about what you would predict for $\langle x(N) \rangle$ and for $\langle x^2(x) \rangle$. We will derive a theoretical prediction next in class.

Theoretical Prediction in class

2e. In the following steps we will use `gnuplot` to fit the resulting $\langle x^2(N) \rangle$ simulation data. Assuming that your data are in `dat2dout`
Type the command

```
gnuplot
```

This will start a session in the graphics-tool `gnuplot`. To do a power law fit and to look at the comparison of the fitted line and your simulation data type

```
a=1;b=1;f(x)=a*x**b;fit f(x) "dat2dout" using 1:3 via a,b;
plot "dat2dout" using 1:3,f(x)
```

When you are finished with gnuplot, you can get out of it with `quit`

Fractal Growth

1. Fractal Growth: Random Walk in Two Dimensions

1a. For the fractal growth DLA model we will need a random walk in two dimensions. Write a python-program for a random walker on a two dimensional lattice (all four directions being equally likely), starting at $x = 0$ and $y = 0$ and (print and) look at $x(t)$ and $y(t)$. You may use the solution program `~kvollmay/share.dir/inclass2023.dir/classrndwalk2a.py`.
Note: We do not want all the analysis we did for the one dimensional random walk, that's why you might want to recycle `classrndwalk2a.py`
For looking at $x(t)$ and $y(t)$ in the same figure, you can use the command (assuming that your program is called `classfractal1a.py` and assuming that your program prints three columns: t, x, y)

```
./classfractal1a.py > j; xmgrace -block j -bxy 1:2 -bxy 1:3
```

1b. Movie

Next let's make a movie of your random walk. Define a lattice (`lattice`) of size 30x30 and initialize it for all sites equal to zero. Put your initial walker at site $x = 15$ and $y = 15$. We want to make a movie of the random walker where we mark on the lattice the current random

walker site with the lattice value 2 and we mark any previously visited site with 1 (This is just for our fun.). To make a movie we first make an image for every random-walk step. **(So please use only** `NSTEPS=50` **random walk steps!)** To see how to make these pictures see the example
`~kvollmay/share.dir/pythonsamples.dir/sample_latticemovie.py` Once you have all pictures in `frame*` you can run the movie with

```
animate -delay 30 -pause 5 frame*png
```