In-Class Work: Random Walk

1. Random Walk in Two Dimensions

Our goal is for today to write a program for a random walk. We focus here on the random walk in two dimensions (instead of one dimension), because the random walk in two dimensions will be part of the fractal growth model, which we will program in the following classes.

1a. Flow Chart

Draw a flow chart for a program of a random walk in two dimensions for N timesteps, which prints out at each time step t, x, y. Be as detailed as possible.

1b. Program

Write a program for a random walk in two dimensions for $N = 50000$ timesteps. As a start copy into your working directory the sample program
`~kvollmay/classes.dir/capstone_s2011.dir/fractal.dir/rand_sample.cc` This program includes the necessary random number generator (same as we have used already for traffic flow) and main structure of the program (including printing). Add to the program at the indicated lines the initialization ($x = 0$ and $y = 0$) and the random walk step. The program prints for each time step t,x and y, so you end up with three columns and 50000 lines. Have a look at $x(t)$ and $y(t)$ with

```
rand1b.out | xgraph -m          and
rand1b.out | gawk '{print $1,$3}' | xgraph -m
```

1c. Movie

Next let's make a movie of your random walk. Define a lattice of size 100x100 and initialize it with all sites equal to zero. Put your initial walker at $x = 50$ and $y = 50$. For each timestep print the lattice such that you can pipe your output into DynamicLattice, that means that for each time step you print on screen the whole lattice (100x100) and one empty line in between whole lattices. Set the current lattice site (x, y) equal to 2 and all sites which were visited in the past equal to 1. You should obtain a white polymer with a red tail and blue background. Run your program for about 300 timesteps.
Hint: Please notice that `rand_sample.cc` includes already a function which allows you to print a lattice of size `LATSIZExLATSIZE`
To look at the movie you use

```
rand1c.out | DynamicLattice -nx 100 -ny 100 -matrix -z 0 2
```

2. Random Walk in One Dimension (if time)

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 use the random walk in one dimension. Before you start with 2a, please get me. I would like to discuss the following analysis with you.

2a. Use your program from 1b. and change it to a program for the random walk in one dimension, initialize with $x(0) = 0$ and print t and x for each time step. Use `NSTEPS = 5000` time steps. As before look at the resulting $x(t)$ with `xgraph`.

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 $(x(N))^2$.

2c. Now add a loop over simulation runs to your program from 2b. Each simulation run starts with $x = 0, y = 0$ and gives you an $x($`NSTEPS`$)$ and an $x^2($`NSTEPS`$)$. Take the average over `NSIMRUNS=10000` simulation runs of x and an x^2 and print out the resulting averages $\langle x \rangle$ and $\langle x^2 \rangle$.

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. For each `NSTEP` print out $N=$`NSTEPS` and $\langle x^2 \rangle$. Look at the resulting $\langle x^2(N) \rangle$. Please get me when you got this done, so that we can interpret your results with the class. I will also explain to you an important analytical result.

2e. In the following steps we will use `gnuplot` to fit the resulting $\langle x^2(N) \rangle$ simulation data. First save your data with `rand2d.out > rand2d.data`, Then 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) "rand2d.data" using 1:3 via a,b; plot "rand2d.data" using
```

Upcoming Deadlines:

- April 14: Results Final Version
- April 21: Abstract
- April 26: Second Paper 1st Version
- May 3: Second Paper Final Version
- April 22: Second Paper (first version)
- April 27 – May 4: Symposium Talks