February 9, 2015, 9:30–11:00 PHYS 338 Spring 2015

In-Class Work: Random Walks

1. Copy into your working directory the sample python-file for random numbers

`~kvollmay/classes.dir/phys338.dir/phys338_s2015.dir/samples.dir/sample_rndnumbers.py`

1a. Run the program and scan it to approximately know what the sample-program does. The following steps guide you through the sample-program.

1b. Write your python program which prints 50 lines, each line with the loop-index i and the random float number in the range $[0, 1)$. You can either use python-plotting tools or `xmgrace` for which I will show to you several tools for making nice figures for your paper in a class much later in the course. Let's say your python-program is named `classrndwalk1.py` and has the two columns, then type in the commandline of a terminal:

```
./classrndwalk1.py | xmgrace -pipe
```

Another way of plotting the data would have been to first write them into a file, let's say named `dat1` and then using xmgrace, that means in the commandline:

```
./classrndwalk1.py > dat1 ; xmgrace dat1
```

1c. Run your python-program again, but this time write the data instead into the file `dat2`. Compare `dat1` and `dat2`. What is going on? How can you ensure to get instead each time when you run the program the same random numbers? (Reproducibility is crucial e.g. for program testing). Yes, this is the `sp.random.seed(15)`. Use a different integer than 15. To confirm that `dat1` and `dat2` are the same, you may use in the command-line the linux-command

```
diff dat1 dat2
```

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. Iinitialize with $x(0) = 0$ and print t and x for each time step. Use `NSTEPS = 5000` time steps. Look at the resulting $x(t)$ with `xmgrace`.

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 $x = 0$ 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. Loop nsteps from 100 to 2000 in steps of 100. For each nstep print out N=nstep and $\langle x^2 \rangle$. Look at the resulting $\langle x(N) \rangle$ and also $\langle x^2(N) \rangle$. Please get me when you got this done, so that we can interpret your results with the class. Try if you can derive a theoretical prediction.

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 `./classrndwalk2d.py > dat2d`, 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) "dat2d" using 1:3 via a,b;
plot "dat2d" using 1:3,f(x)
```

2f. In case you have time left, read in our textbook §7.2 and pages 249–251.