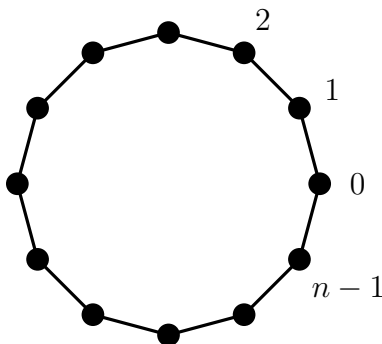


Math 391 – Mixing Times for Markov Chains
Summary of Computer Lab #3
Monday, Feb. 18

These are notes provided for the benefit of those students who cannot attend the computer lab at the regular hour. Starting next week, there will also be small assignments in the computer lab in the sense that students will report their data. At that time, students who cannot attend the lab at the regular hour will still be responsible for emailing me their data. But for now, that is not necessary.

Goal: Generate a simulation for the random walk on the cycle.



Problem 1: Note that to produce a random variable U which is uniformly distributed on the interval $[0, 1]$, you use the command “rand”. The typical output produced is as follows:

```

>> U = rand
U =
    0.8147

```

In order to generate a random step for the lazy random walk we want a random variable W which takes the values $0, 1$ and -1 with the probabilities $P(W = 0) = 0.5, P(W = 1) = P(W = -1) = 0.25$. How can we do this?

Answer: We have already seen the floor function last week. The floor of x is the largest integer k satisfying $k \leq x$. So, if $x = 1.1$ then the floor of x is 1 . But if $x = 2$ then the floor of x is 2 . Note that if $x = -0.1$ then the floor of x is -1 .

```

>> floor(-0.1)
ans =
    -1

```

Therefore, we can take $W = \text{floor}(V)$ if we can generate a random variable V such that $P(-1 < V < 0) = 0.25, P(0 < V < 1) = 0.5, P(1 < V < 2) = 0.25$.

Q1: Suppose we take $V = aU - b$ for some numbers $a, b > 0$, where U is uniform. What should we choose for a and b in order to get the probabilities above?

A1: Since U is uniform, we have

$$P(aU - b < 0) = P(aU < b) = P(U < b/a) = P(0 < U < b/a) = b/a,$$

as long as we assume that $b/a < 1$, i.e., that $b < a$. Similarly,

$$P(aU - b > 1) = P(aU > 1 + b) = P(U > (1 + b)/a) = P(1 > U > (1 + b)/a) = 1 - \frac{1 + b}{a},$$

assuming $(1 + b)/a < 1$, i.e., that $1 + b < a$. So we need to solve

$$\frac{b}{a} = \frac{1}{4} \quad \text{and} \quad 1 - \frac{1 + b}{a} = \frac{1}{4},$$

whose solution is $a = 2$ and $b = 0.5$. So taking $V = 2U - 0.5$ gives the right probabilities.

So the answer to the first problem is to take:

```
>> U = rand
U =
    0.9058
>> V = 2*U - 0.5
V =
    1.3116
>> W = floor(V)
W =
    1
```

Task 1: Make a script to generate this random step. Call it “step.m”.

First do the following

```
>> edit step.m
```

This will bring up a dialog box asking you to agree to create a new m-file called step.m. Click the “Yes” box. This will take you to a new text file which is “step.m”. There you should simply type what we just did, but probably use semicolons at the end of the lines to silence the output. Thus, I suggest writing

```
linewalk.m: U = rand;
            V = 2*U-0.5;
            W = floor(V);
```

At the end save by pressing control-s (or option-s on Mac’s) or by going up to the file menu and selecting save. Then go back to the main command window and type the following to make sure the script works. (As usual, the output is random, I just show my output to give an example of what you should see.)

```
>> step
>> W
W =
   -1
```

Note that, because I used semicolons in the m-file script, the output for W was not automatically typed. But a new random variable W is made by the m-file (which wipes out the old version of W). One can see it by just typing W , as I did. The next step is to use step.m to generate a random walk on the line.

Task 2: Make a script to generate a random walk on the line. Call it “linewalk.m”.

To get into the m-file type the following in the main command window, just as we did to make step.m.

```
>> edit linewalk.m
```

Again, click the “Yes” box, to create a new m-file. In the new m-file, I suggest writing

```
step.m: T = 1000;
        X = 0;
        for t=1:T,
            step
            X = X + W;
        end
```

At the end, remember to save. You can check your m-file by going back to the main command window and typing the following.

```
>> linewalk
>> X
X =
   -38
```

This is the result of a lazy random walk, with 1000 time steps. The output could be any integer between -1000 and 1000 . This is *not* the random walk on the cycle. This is the random walk on the line.

Problem 2: We have a random walk on the line. Find a way to use this to produce a random walk on the cycle of length L .

The answer is to use the mod function. Basically, the mod function does the following

$$\text{mod}(x, n) = x - n * \text{floor}(x/n)$$

Q2: If n is an integer and x ranges over integers 1 to n , what will be the outputs of $\text{mod}(x, n)$?

A2: If $x = 1, \dots, n - 1$ then $x/n < 1$ which means $\text{floor}(x/n) = 0$. So for $x = 1, \dots, n - 1$, we obtain $\text{mod}(x, n) = x$. But if $x = n$ then $x/n = 1$ so that $n * \text{floor}(x/n) = n * 1 = n$. So, when $x = n$ we obtain $\text{mod}(x, n) = 0$. This is important to note: the outputs will be numbers $0, 1, \dots, n - 1$. That is why we drew the cycle with states $0, 1, 2$ up to $n - 1$, rather than 1 up to n .

Let us agree to make $n = 25$ for this numerical experiment.

Task 3: Make a script to generate a random walk on the cycle of length 25. Call it “circlewalk.m”.

To get into the m-file type the following in the main command window:

```
>> edit
circlewalk.m
```

Again, click the “Yes” box, to create a new m-file. In the new m-file, I suggest writing

```
circlewalk.m: L = 25;
               linewalk
               Y = mod(X,L);
```

At the end, remember to save. You can check your m-file by going back to the main command window and typing the following.

```
>> circlewalk
>> Y
Y =
    17
```

Note that you can also check what X was by typing:

```
>> X
X =
    17
```

I wanted to see an example where X is not in the range 0 to 24, so I repeat the experiment.

```
>> circlewalk
>> Y
Y =
     3
>> X
X =
    28
```

This is what I wanted to see. Even though X is outside the range of the cycle, since it is 28, by taking the mod-function, we obtain Y which is in this range, $Y = 3 = 28 - 25$.

Task 4: Generate statistics associated to this random walk by repeating the experiment 1000 times.

In the main command window type the following:

```
>> NumTrials =
1000;
>> Yarray=[];
>> for
nctr=1:NumTrials,
circlewalk,
Yarray =
[Yarray,Y];
end
```

Note that it will take a little bit of time (about 1 minute on my laptop) after you type return after the end command. That is because you are repeating a random walk 1000 times, and each random walk takes 1000 steps. So there are on the order of one million operations.

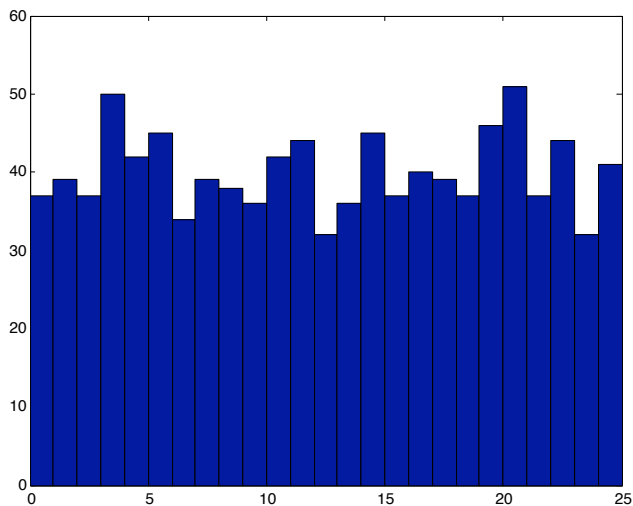
At the end, the vector “Yarray” has the last step for every one of the 1000 random walks. We need a way to see the data. A histogram is a good way to visualize it. In the command window type the following:

```
>> histc(Yarray,0:24)
ans =
  Columns 1 through 15
    37 39 37 50 42 45 34 39 38 36 42 44 32 36 45
  Columns 16 through 25
    37 40 39 37 46 51 37 44 32 41
```

Of course, as usual your data will look different than mine. You should type “help histc” to see what it does. One thing it does not do, that the usual “hist” command does do is to plot. So in order to see a plot type the following:

```
>> bar(0:24,histc(Yarray,0:24),'histc')
```

For me this produces the following output in a separate figure window.



Questions for thought: Here are some questions to think about, but not to hand in.

1. What did all the commands we used really mean? To find out you can type “help” and then the command name for any of the commands.
2. What is the stationary distribution for the random walk on the cycle?
3. Does the random walk on the cycle converge to its stationary distribution? (What are the requirements for that?)
4. What is the Markov generator for the lazy random walk on the cycle? Did we really simulate this correctly? (The answer is yes.)
5. Is there a stationary distribution for the random walk on the line? If not, what is the difference between the cycle and the line that accounts for this?
6. In the histogram, unless you are very lucky, there is a lot of fluctuation visible still away from the stationary distribution. The histogram represents the “empirical distribution”. It is not a perfect replica of the underlying distribution for μ_T because we are “sampling” so that there is the usual type of statistical fluctuations present. What can we do to reduce these fluctuations?
7. What is the effect of increasing “NumTrials”? What is the effect of increasing “T”? Which of these matter more? Or does one matter more than the other? Which one are we studying in this class?

Histogram for results obtained by adding an extra 10,000 trials, so that NumTrials is 11,000:

