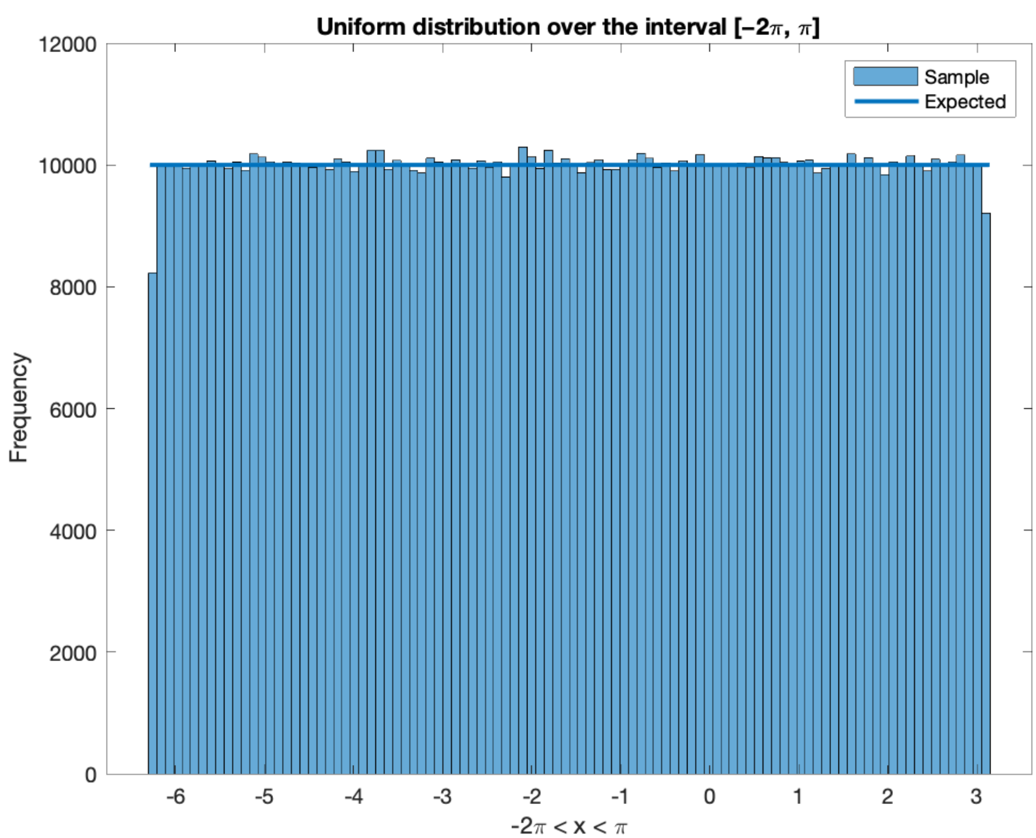
Assignment 2

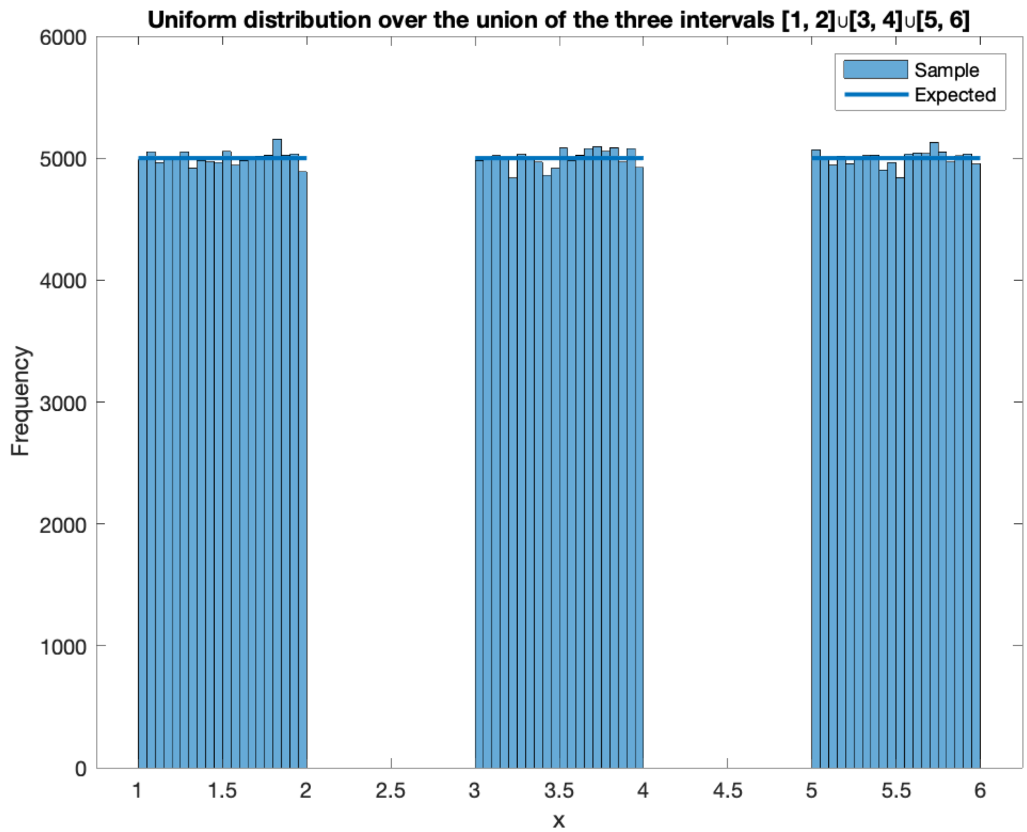
January 4, 2020

Question 1.

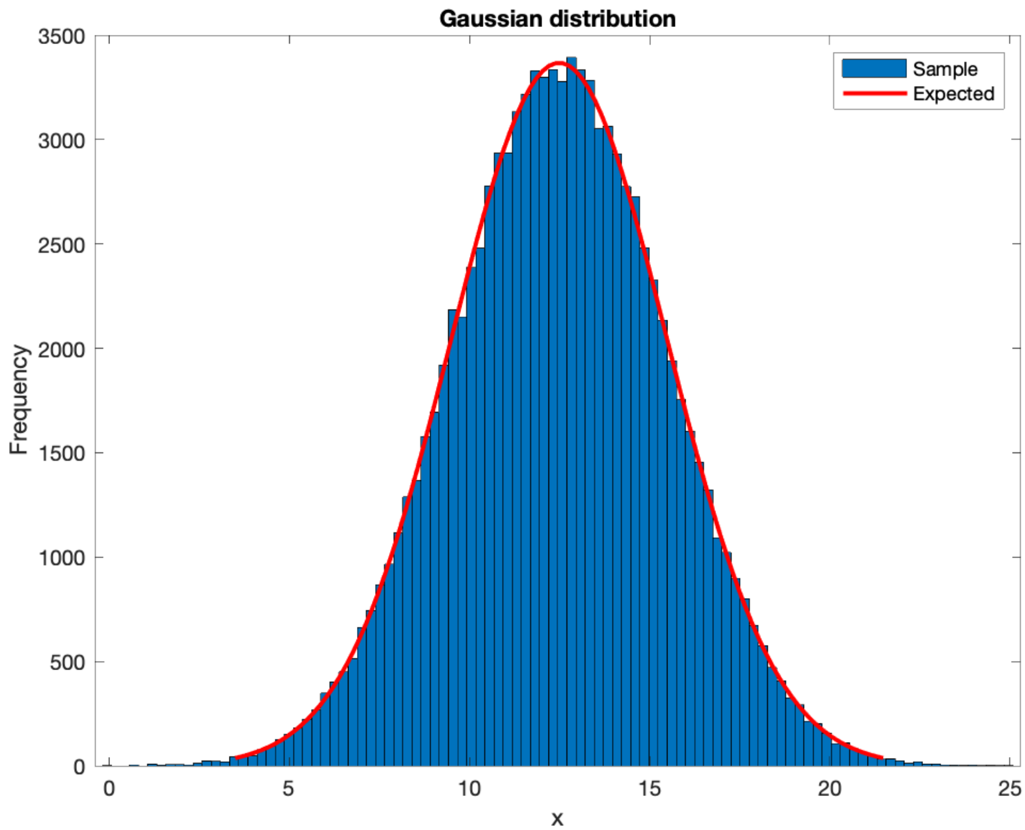
(a)



(b)

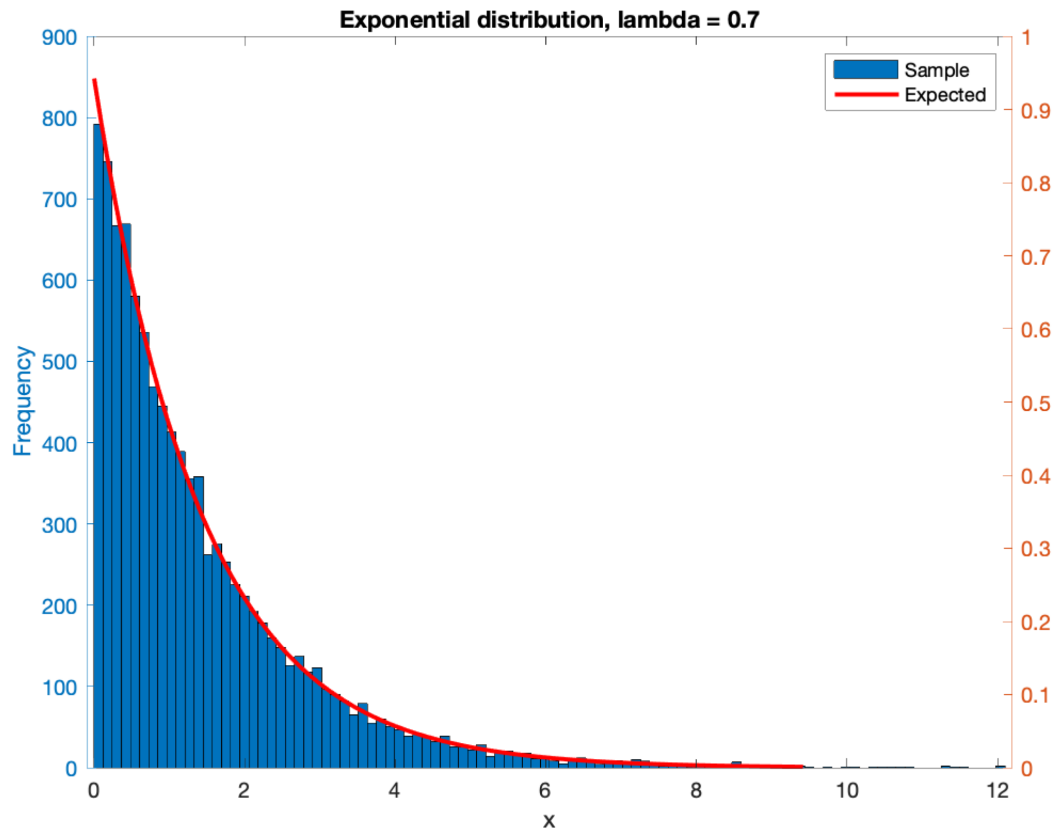

Uniform distribution over the union of the three intervals [1, 2]∪[3, 4]∪[5, 6]
Sample
Expected
Frequency
x
0
1000
2000
3000
4000
5000
6000
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5
6

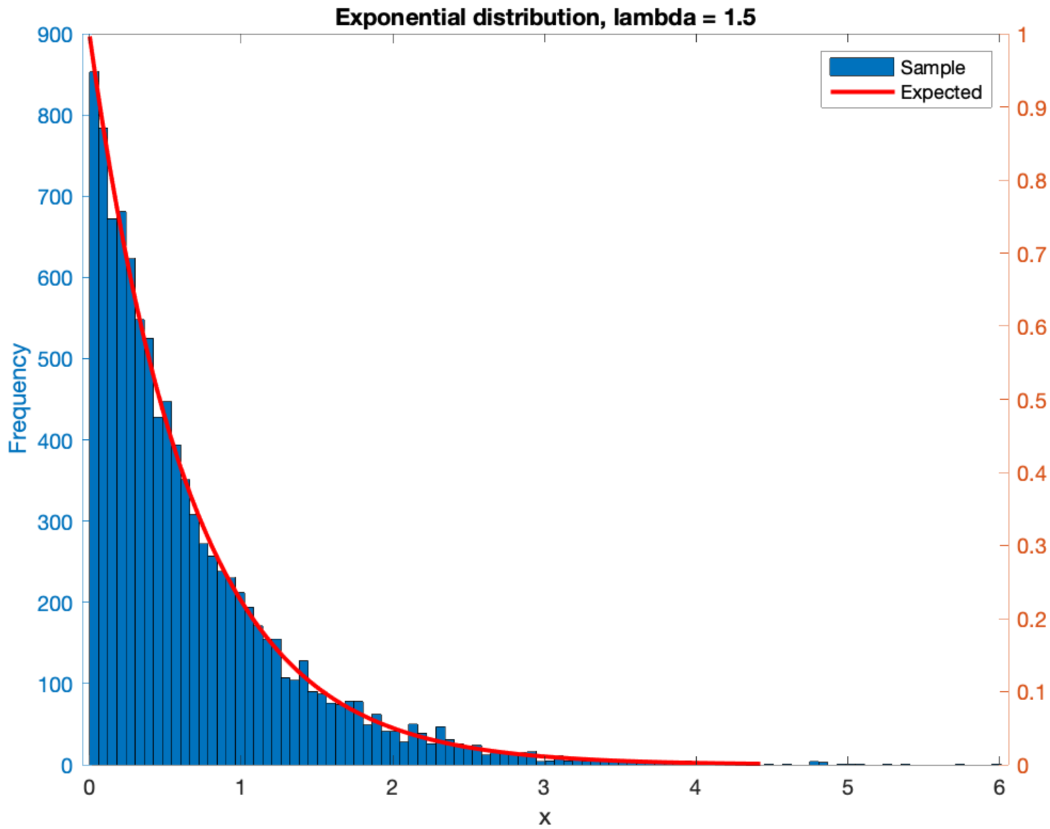
(c)

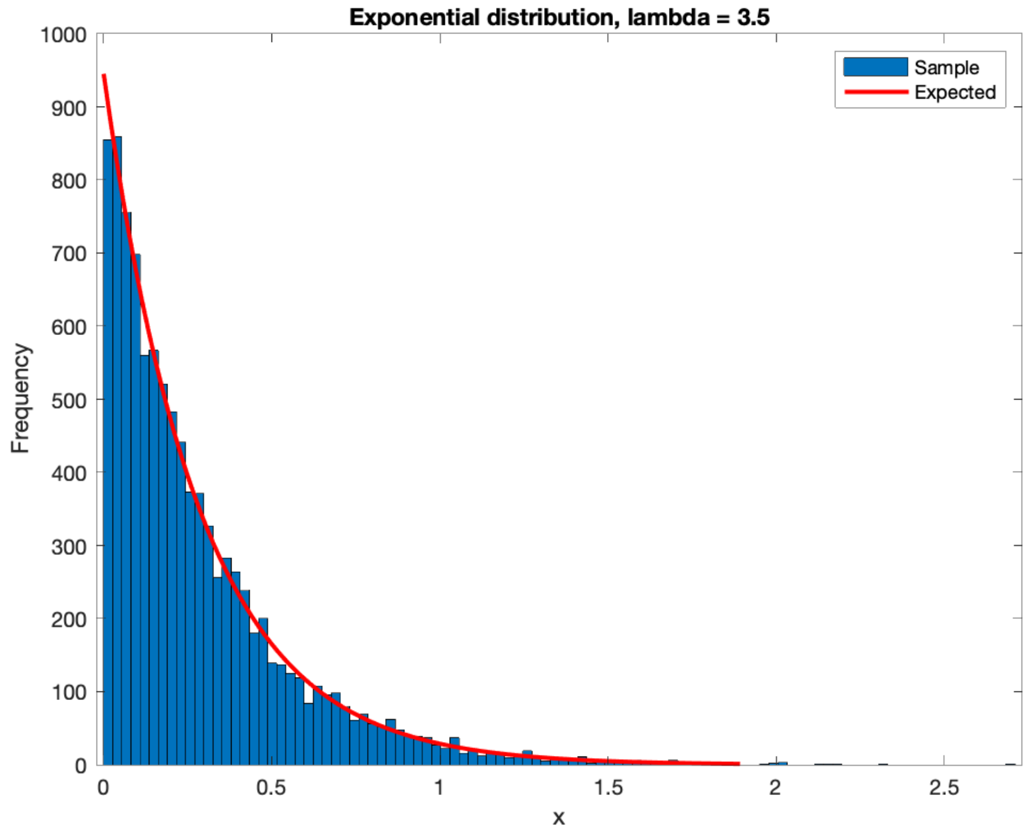


Gaussian distribution
3500
3000
2500
2000
1500
1000
500
0
Frequency
0
5
10
15
20
25
x
Sample
Expected

(d)

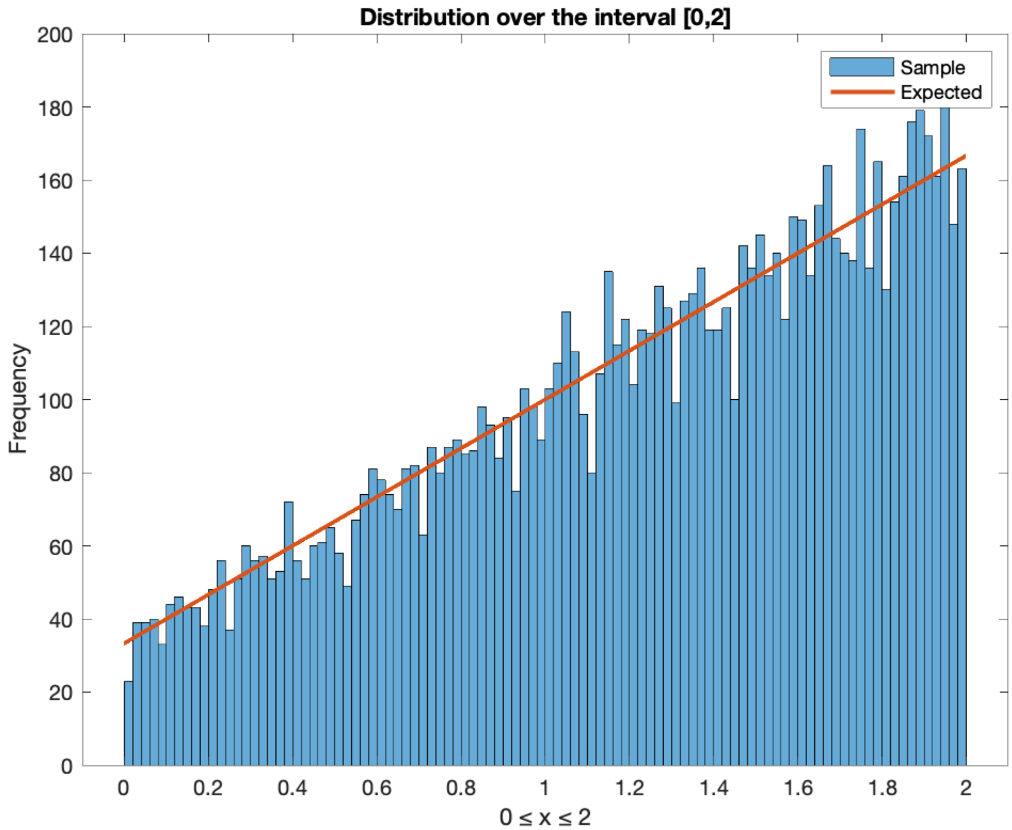

Exponential distribution, lambda = 0.7
Sample
Expected
Frequency
x
900
800
700
600
500
400
300
200
100
0
0
2
4
6
8
10
12
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0


Exponential distribution, lambda = 1.5
Sample
Expected
Frequency
x
0
1
2
3
4
5
6
0
100
200
300
400
500
600
700
800
900
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1



(e)

$$
\begin{aligned}
u &= F(x) && (1)\\
&= \frac{1}{6}\left(x^2 + x\right) && (2)\\
6u &= x^2 + x + \frac{1}{4} - \frac{1}{4} && (3)\\
6u + \frac{1}{4} &= (x + \frac{1}{2})^2 \qquad \text{x is non-negative.} && (4)\\
x &= \sqrt{6u + \frac{1}{4}} - 1/2 && (5)\\
& && (6)
\end{aligned}
$$



Question 2.

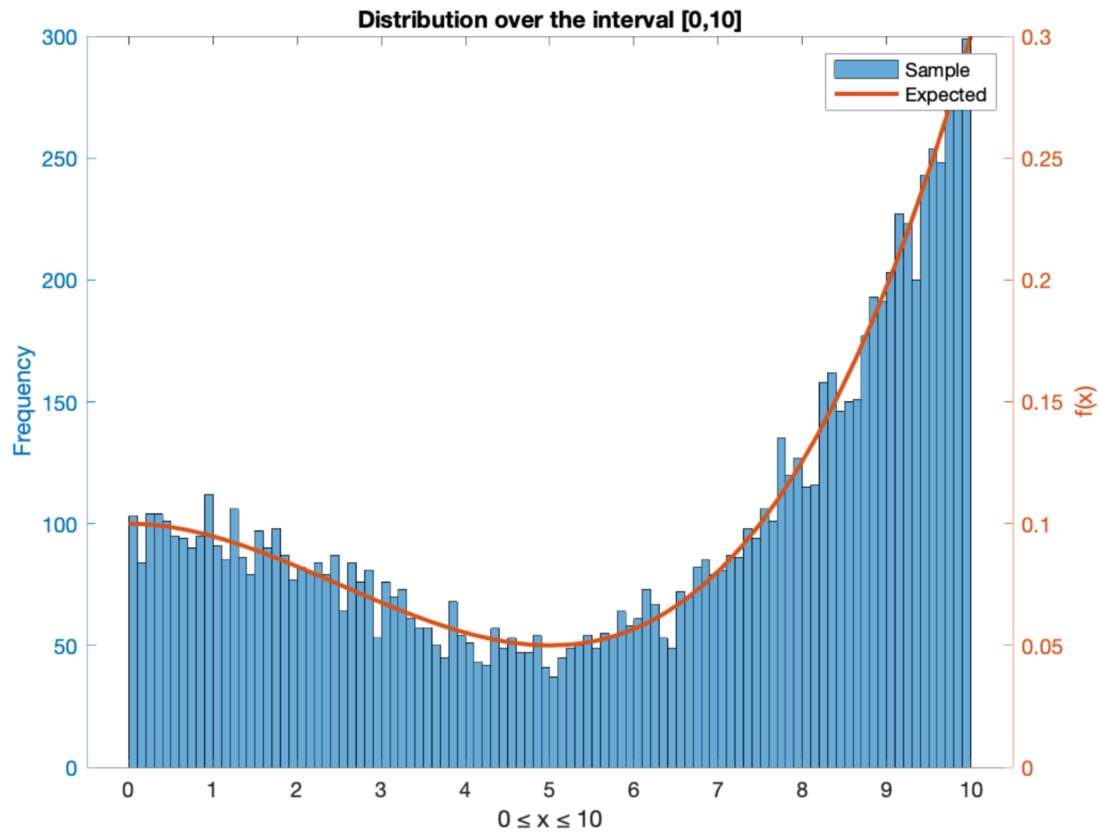
(a)

My function in Matlab is pretty neat, you might want to check it out. It goes something like this:

```
function X = accreja(n)
f = @(x) (100+4/5*x^3-6*x^2)/1000
X = zeros(1,n);
for i=1:n
```

```
        accept = false;
        while accept == false
            gx = 0.3 * rand;
            range = 10 * rand;
            if gx <= f(range)
                X(i) = range;
                accept = true;
            end
        end
    end
end
```

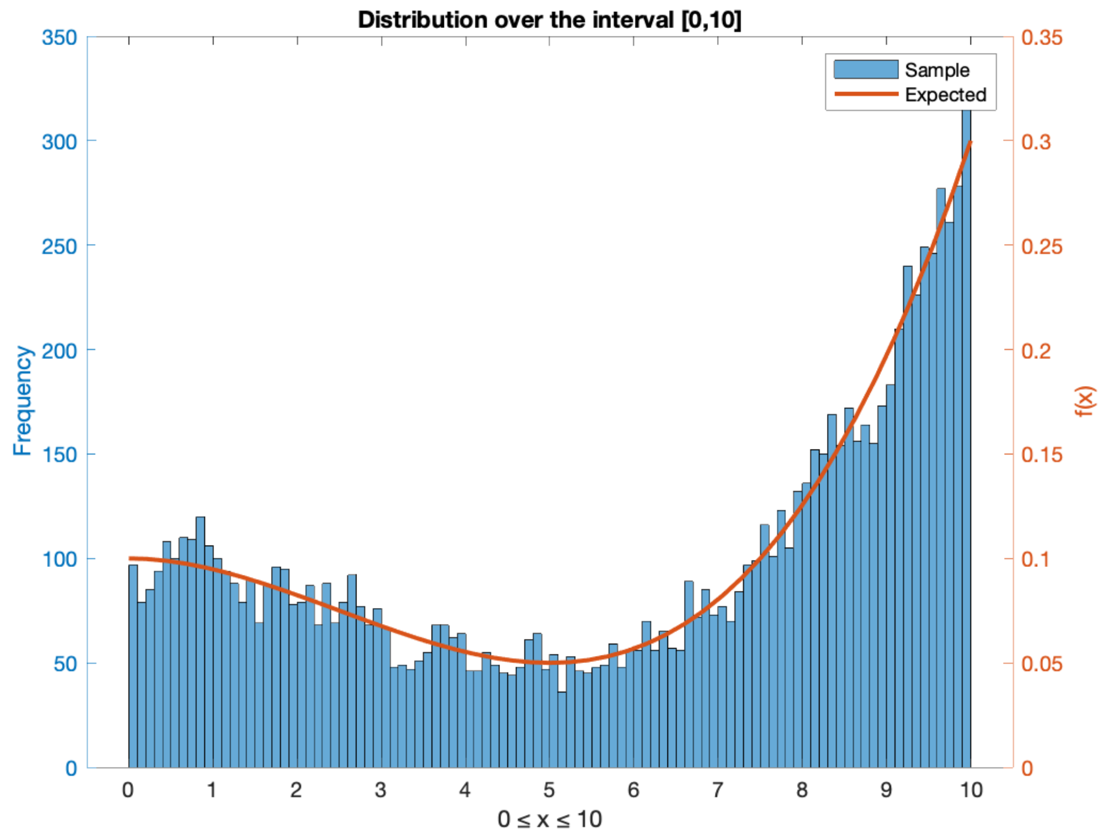
(b)



(c)

My function is in the bottom of my Matlab file.

(d)



Rejection rate: 0.24

Question 3.

(a)

Absolute value of the estimate: 0.010; Mean Squared Error: 0.00272

(b)

Estimate: 0.100; Standard error = 0.00237

(c)

f_1: Estimate: 0.102; Standard error: 0.00142

f_2: Estimate: 0.102; Standard error: 0.00090

f_3: Estimate: 0.101; Standard error: 0.00128

Like the tale of Goldilocks, the middling estimate brought about by function $f_2(x) = 2x^2$ is the best estimate here.

Part (a) brought a wildly varying estimate with not very good standard error.

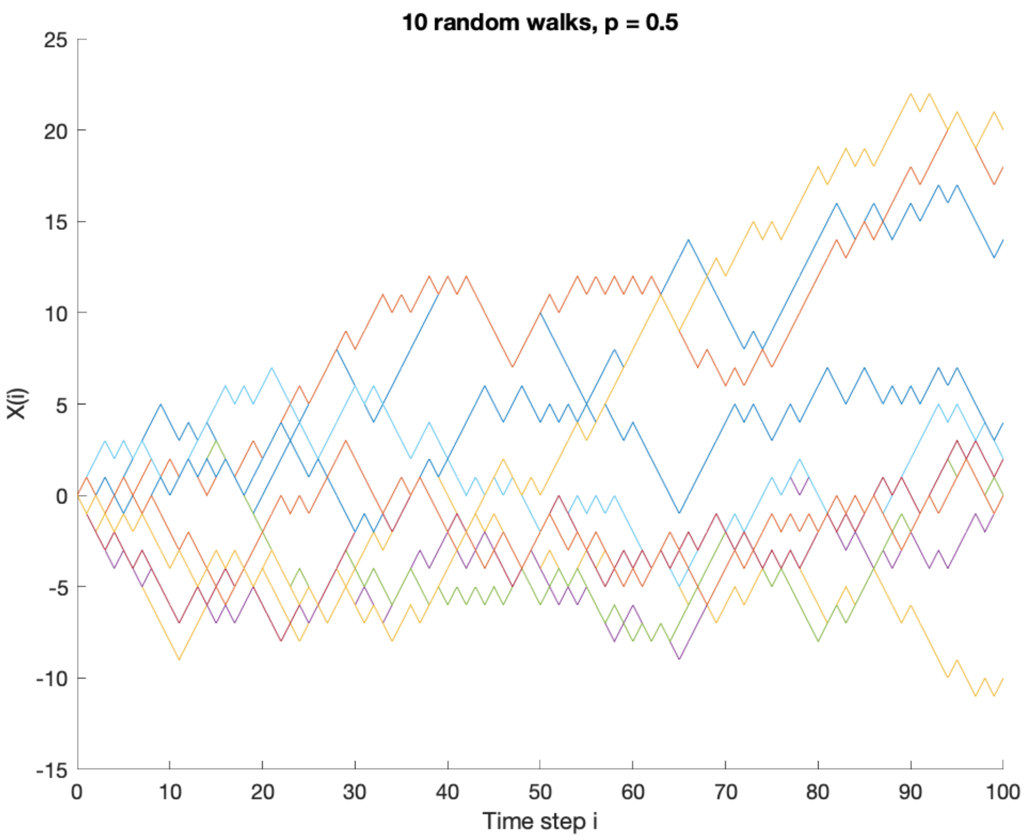
Part (b) improved on part (a)'s estimate and standard error.

In short, f_3 over-samples from the lower end of the distribution, and f_1 over-samples from the high end of the distribution, leaving f_2 as the best candidate for the role of the auxiliary sample distribution.

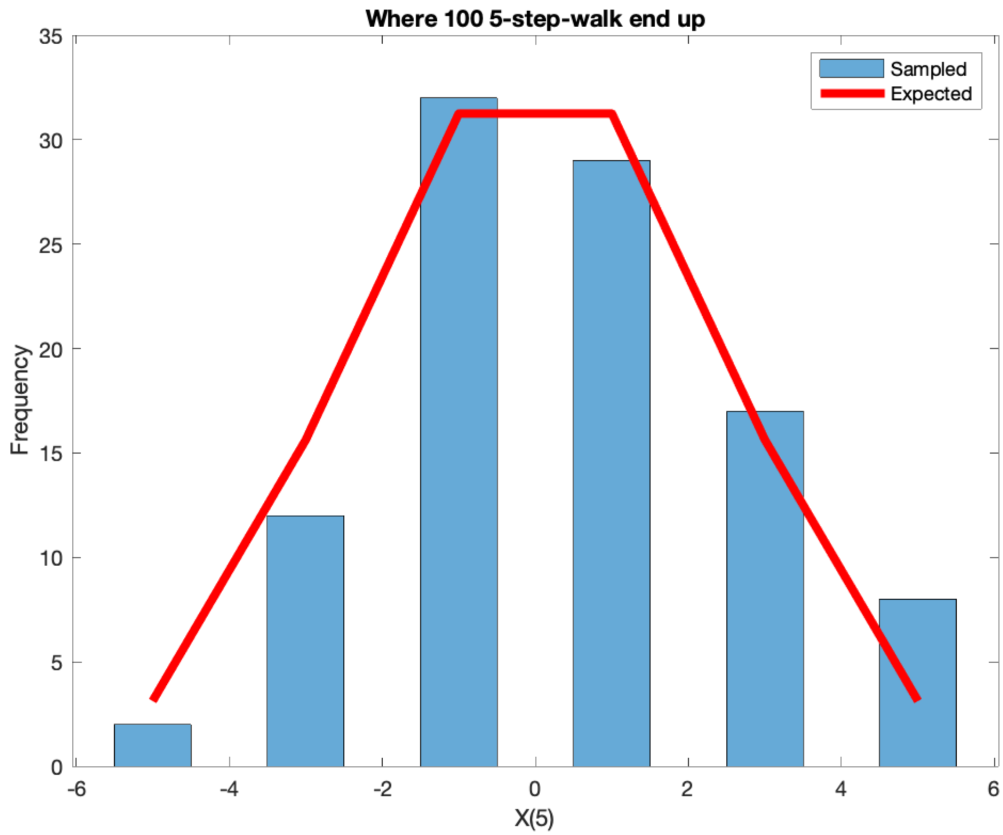
Question 4.

(a)

(i)

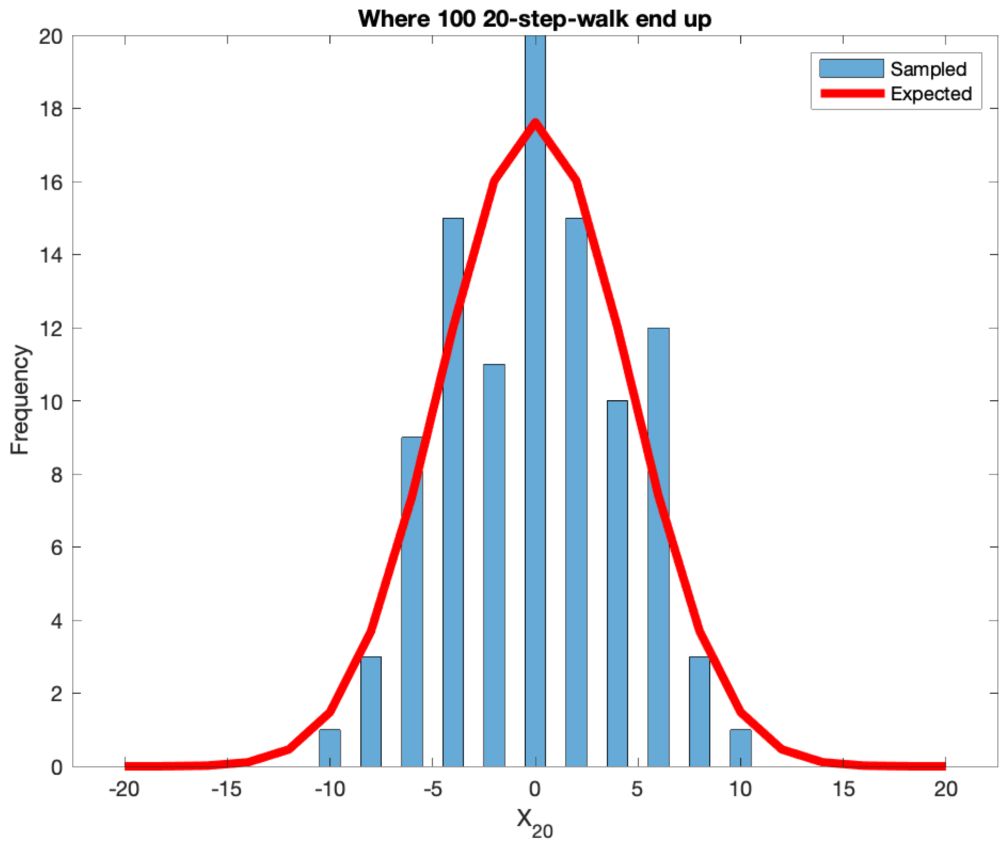


(ii)



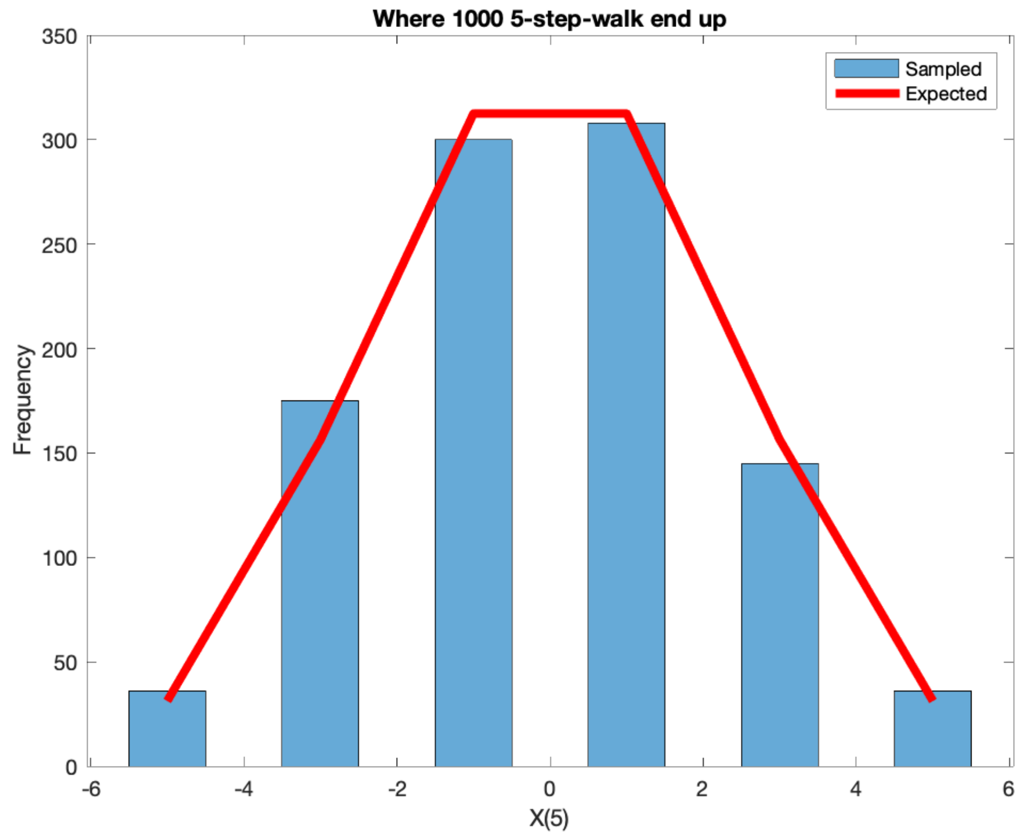
Well, it roughly fits.

(iii)

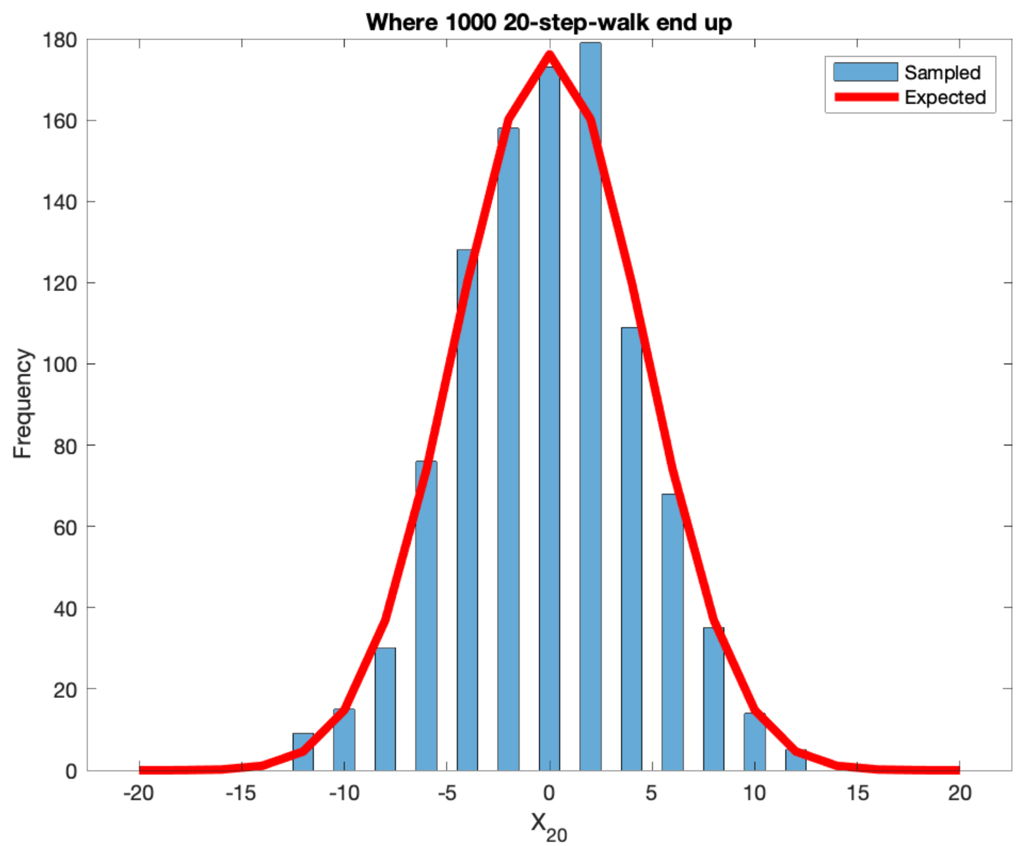


Well, it roughly fits.

(iv)



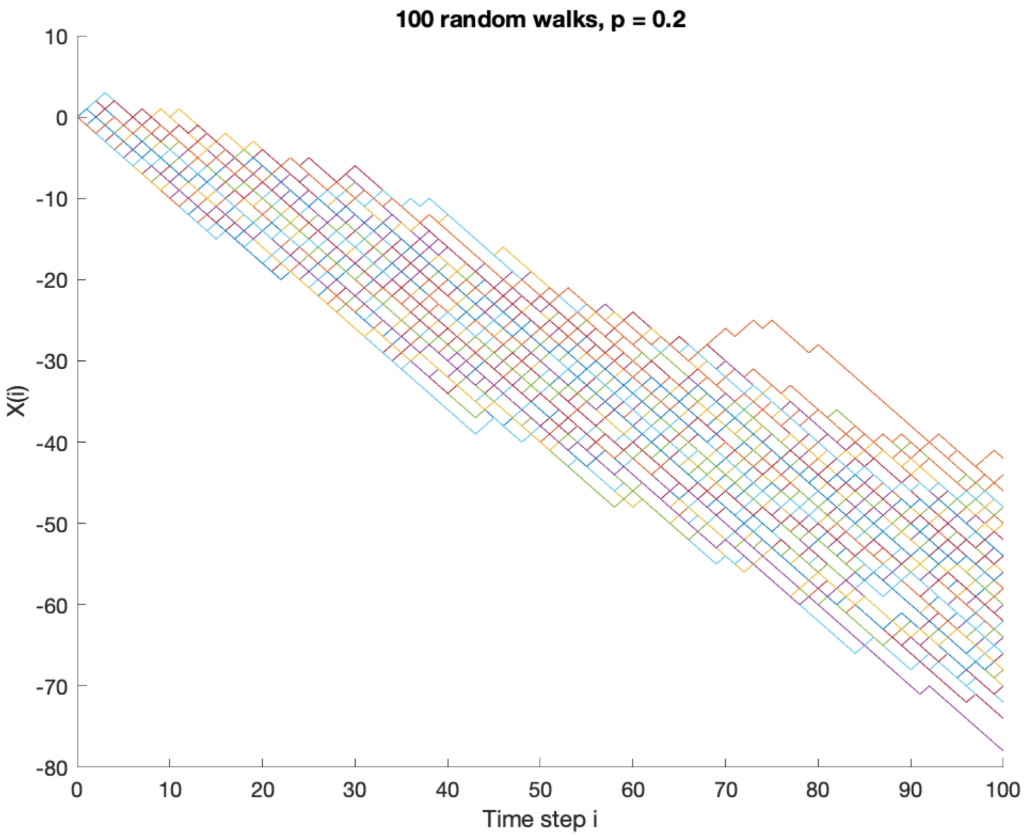
Now it fits better than (ii) thanks to the increased amount of walkers which makes the sample converges towards its expected distribution. Mass amount of independent walkers help turns randomness into predictability.



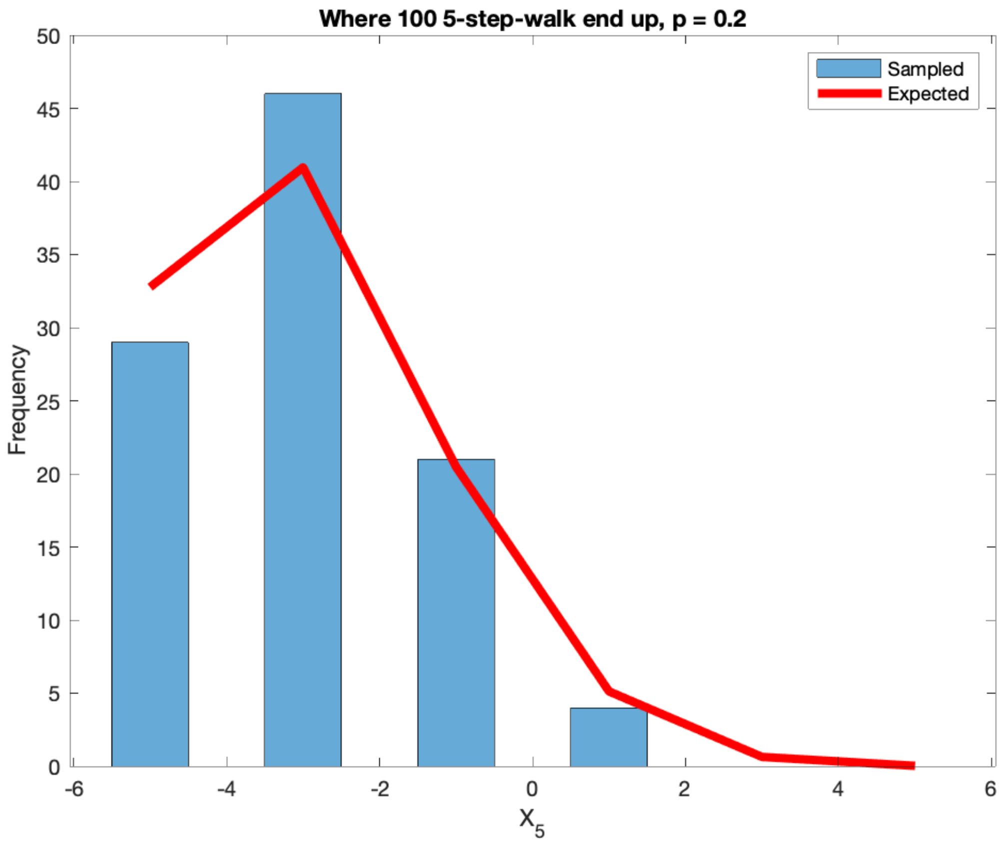
This one fits so much neater than (iii) thanks to the increased amount of walkers (and look better than the 5-step-walk thanks to the variability of destinations).

(b)

(i)

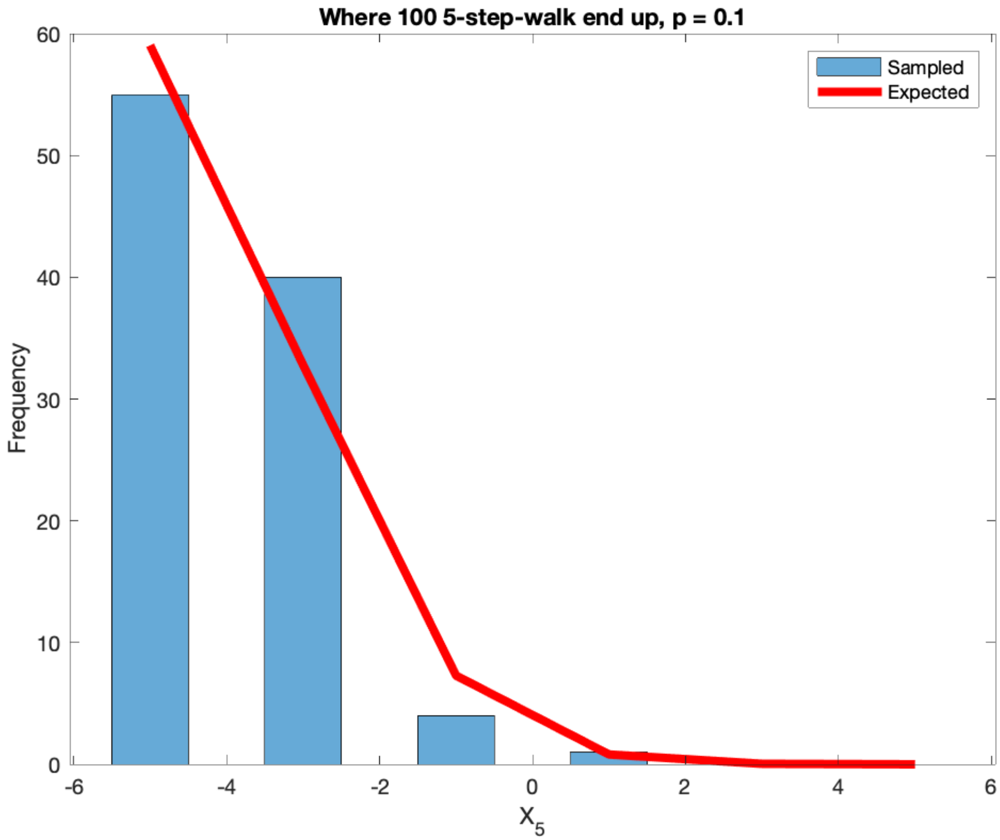


(ii)



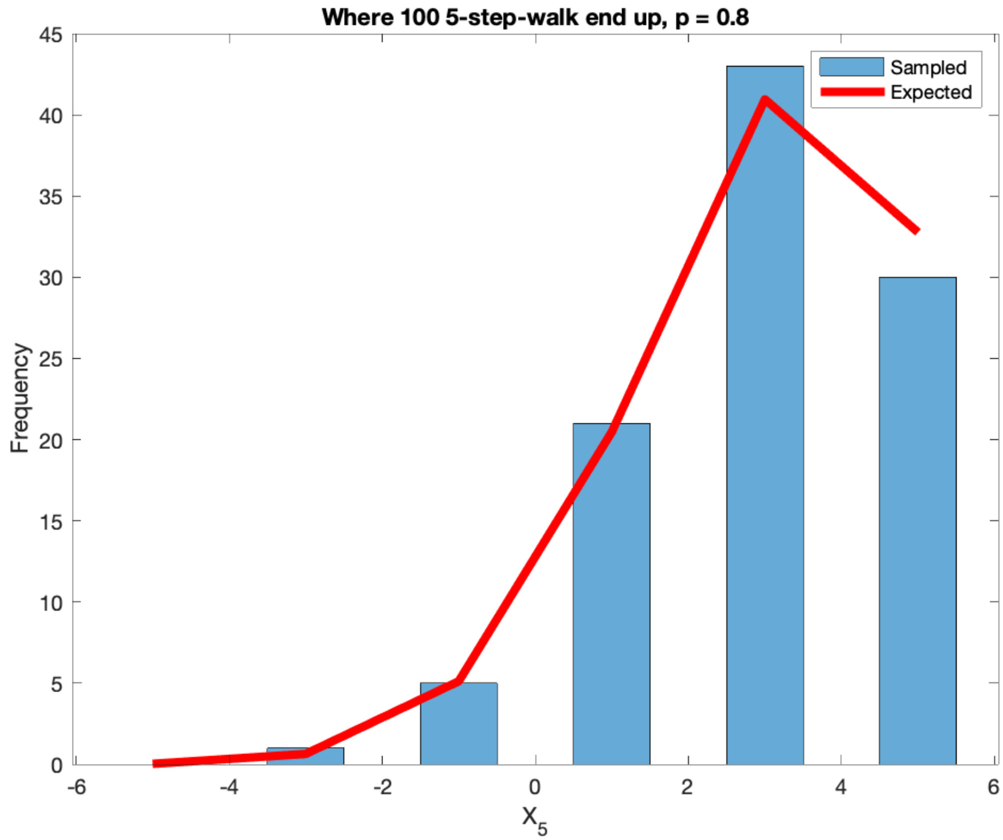
So when $p = 0.2$, X_5 tends to end up at lower values, which is expected as X now tends to take more negative steps in its random walk.

What if we decrease p down to 0.1? I predict X_5 distribution to be even more positively skewed.



And yes, we get what I predicted above.

And what if we boost p up to 0.8? The opposite effect would take place.



Question 5.

(a)

See my file: 5a.ipynb. It's pretty neat.

(b)

Steps	recursive	simplesampling	sd error	How far off?	Error margin
1	4	3.999996	0.001999999	4E-06	1E-06
2	12	12.001744	1.73179901	-0.001744	0.000145333
3	36	36.00224	3.96859168	-0.00224	6.22222E-05
4	100	100.029952	7.806666926	-0.029952	0.00029952
5	284	284.127232	14.32798022	-0.127232	0.000448
6	780	781.692928	25.14979731	-1.692928	0.002170421
7	2172	2172.35456	43.40877871	-0.35456	0.000163241
8	5916	5924.716544	73.41047929	-8.716544	0.001473385
9	16268	16152.52685	123.1148106	115.473152	0.007098178
10	44100	43972.03456	205.2512297	127.96544	0.002901711

And no, unfortunately I could not use my function to estimate all the way up to 50.

(c)

Steps	real	Rosenbluth	How far off?	Error margin
1.00	4.00	3.999996	0.00	1E-06
2.00	12.00	11.999628	0.00	3.1001E-05
3.00	36.00	36.025156	-0.03	0.00069829
4.00	100.00	100.066072	-0.07	0.000660284
5.00	284.00	281.74848	2.25	0.007991241
6.00	780.00	783.057876	-3.06	0.003905045
7.00	2,172.00	2184.248384	-12.25	0.005607597
8.00	5,916.00	6096.489348	-180.49	0.029605456
9.00	16,268.00	16943.3635	-675.36	0.039860061
10.00	44,100.00	47239.45302	-3,139.45	0.066458285

Yes, I can estimate the number of 50-step walk: It's something like $2.79 * 10^{22}$.

(d)

Steps	Real	recursiveSAW	simplesamplingSAW	simplesampling sd error	How far off (simple)?	Error margin
1	4	4	4.00	0.00	0.00	0.00
2	12	12	12.00	1.73	0.00	0.00
3	36	36	36.00	3.97	0.00	0.00
4	100	100	100.03	7.81	-0.03	0.00
5	284	284	284.13	14.33	-0.13	0.00
6	780	780	781.69	25.15	-1.69	0.00
7	2,172	2,172	2,172.35	43.41	-0.35	0.00
8	5,916	5,916	5,924.72	73.41	-8.72	0.00
9	16,268	16,268	16,152.53	123.11	115.47	0.01
10	44,100	44,100	43,972.03	205.25	127.97	0.00
11	120,292	120,292	120,271.67	341.79	20.33	0.00
12	324,932	324,932	323,666.05	563.40	1,265.95	0.00
13	881,500	881,500	876,508.87	930.09	4,991.13	0.01
14	2,374,444	2,374,444	2,370,285.08	1,532.76	4,158.92	0.00
15	6,416,596	6,416,596	6,326,486.83	2,507.83	90,109.17	0.01
16	17,245,332	17,245,332	16,690,242.91	4,077.42	555,089.09	0.03
17	46,466,676	46,466,676	47,210,280.52	6,861.53	-743,604.52	0.02
18	124,658,732	124,658,732	125,138,167.14	11,176.33	-479,435.14	0.00

19	335,116,620	335,116,620	324,630,808.10	18,006.87	10,485,811.90	0.03
20	897,697,164	897,697,164	924,689,278.96	30,395.91	-26,992,114.96	0.03
21	2,408,806,028	2,408,806,028	2,476,100,185.75	49,746.42	-67,294,157.75	0.03
22	6,444,560,484		6,157,265,115.55	78,454.51	287,295,368.45	0.04
23	17,266,613,812		16,818,129,858.46	129,669.23	448,483,953.54	0.03
24	46,146,397,316		52,072,870,691.47	228,173.70	-5,926,473,375.47	0.13
25	123,481,354,908		120,471,290,032.16	347,071.17	3,010,064,875.84	0.02
26	329,712,786,220		360,287,970,189.64	600,215.92	-30,575,183,969.64	0.09
27	881,317,491,628		774,619,135,907.73	880,105.58	106,698,355,720.28	0.12
28			2,738,188,573,441.26	1,654,715.84		
29			4,611,686,018,427.38	2,147,466.47		
30			17,293,822,569,102.70	4,158,553.01		
31			55,340,232,221,128.60	7,439,056.94		
32			166,020,696,663,385.00	12,884,843.91		
33			368,934,881,474,191.00	19,207,629.65		
34			590,295,810,358,705.00	24,295,979.70		
35			2,361,183,241,434,820.00	48,591,959.41		
36			4,722,366,482,869,640.00	68,719,442.38		
37			18,889,465,931,478,500.00	137,438,884.75		

38			75,557,863,725,914,300.00	274,877,769.51		
39						
40						

Steps	Real	RosenbluthSamplingSAW	How far off(Rosenbluth)?	Error margin
1	4	3.999998	0.00	5E-07
2	12	12.003218	0.00	0.000268167
3	36	35.995886	0.00	0.000114278
4	100	99.966624	0.03	0.00033376
5	284	281.900692	2.10	0.00739193
6	780	782.615578	-2.62	0.003353305
7	2,172	2187.682588	-15.68	0.007220344
8	5,916	6093.521942	-177.52	0.03000709
9	16,268	16954.41307	-686.41	0.042194066
10	44,100	47341.97794	-3,241.98	0.073514239
11	120,292	131822.2115	-11,530.21	0.095851856
12	324,932	367130.0842	-42,198.08	0.129867431
13	881,500	1027204.754	-145,704.75	0.165291837
14	2,374,444	2866903.41	-492,459.41	0.207399884
15	6,416,596	8027988.336	-1,611,392.34	0.251128844
16	17,245,332	22263406.52	-5,018,074.52	0.290981613
17	46,466,676	62205792.21	-15,739,116.21	0.338718358
18	124,658,732	173554383.4	-48,895,651.40	0.392236072

19	335,116,620	478849735	-143,733,115.04	0.428904765
20	897,697,164	1348434111	-450,736,946.96	0.502103566
21	2,408,806,028	3784385724	-1,375,579,696.39	0.571062875
22	6,444,560,484	10541480559	-4,096,920,074.89	0.63571753
23	17,266,613,812	29053915036	-11,787,301,223.76	0.682664323
24	46,146,397,316	80869994058	-34,723,596,741.74	0.752466038
25	123,481,354,908	2.26436E+11	-102,954,867,196.21	0.833768525
26	329,712,786,220	6.31987E+11	-302,274,208,205.70	0.916780364
27	881,317,491,628	1.78404E+12	-902,723,567,181.70	1.024288722
28		4.96851E+12		
29		1.3529E+13		
30		3.83212E+13		
31		1.08812E+14		
32		2.89639E+14		
33		8.17682E+14		
34		2.30778E+15		
35		6.55353E+15		
36		1.76E+16		
37		4.66E+16		

38		1.23E+17		
39		4.37E+17		
40		9.70E+17		
41		2.91E+18		
42		9.20E+18		
43		2.20E+19		
44		4.77E+19		
45		2.00E+20		
46		4.33E+20		
47		1.33E+21		
48		3.37E+21		
49		1.12E+22		
50		3.14E+22		
51		6.28E+22		
52		1.88E+23		
53		4.69E+23		
54		1.06E+24		
55		5.28E+24		
56		3.81E+25		

57		3.04E+25		
58		1.32E+26		
59		9.80E+26		
60		3.76E+26		
61		2.01E+27		
62		2.86E+27		
63		3.67E+28		
64		1.49E+28		
65		5.57E+28		
66		1.85E+29		
67		7.89E+29		
68		1.49E+30		
69		2.19E+31		
70		1.74E+31		
71		7.75E+30		
72		4.68E+31		
73		5.04E+31		
74		6.61E+32		
75		8.54E+31		

76		5.14E+33		
77		3.14E+33		
78		2.99E+34		
79		3.47E+33		
80		9.85E+34		
81		1.89E+35		
82		5.02E+35		
83		1.38E+37		
84		2.74E+36		
85		2.12E+36		
86		1.00E+36		
87		3.68E+37		
88		1.46E+39		
89		3.10E+39		
90		2.71E+39		
91		6.93E+38		
92		3.21E+40		
93		1.24E+41		
94		4.06E+38		

95		4.20E+40		
96		3.10E+40		
97		1.50E+42		
98		4.53E+42		
99		2.02E+45		
100		2.00E+42		

(e)

1st algorithm: The recursive algorithm try to enumerate every single possible path of a walk at a certain number of steps.

My algorithm check North, West, South, East for an empty space of 'False' and let the path walks there, marking places it has been as 'True'.

- Strengths:

+ 100% correct.

- Weakness:

+ Quickly gets very slow, even at 20-step walk.

+ Complexity quickly gets out of hand, even at 22-step walk.

2nd algorithm: The Simple Sampling algorithm generate random walks by randomly picking a direction to go (out of 4 in a 2d lattice). It essentially generate millions of simple random walk of length N, and then calculate probability that a random walk is self-avoiding. Extrapolate to 4^N possible cases to estimate the number of self-avoiding walks.

My algorithm randomly check a space to walk to. If the space was walked on before, eliminate walk and try again. Walk ends when no. of steps required is reached.

- Strength:

+ Capable of producing estimates for up to 40-step walk.

+ Relatively accurate with low standard error.

+ Relatively fast: takes 73 seconds to estimate 1-step walk to 38-step walk, from 1,000,000 random walks each.

- Weakness:

+ Can't go further than 40-step walk.

3rd algorithm: Rosenbluth sampling only chooses free edges to branch out. This introduces bias as more compact walks with fewer choices available are overly represented. This bias is corrected by adding weights to walks. Weights are decided by "atmosphere":

the number of possibilities in growing the current configuration.

My algorithm only walks randomly into unoccupied spaces.

- Strength:

+ Capable of producing estimates for up to 100-step walk.

- Weakness:

+ Relatively slow compare to simple sampling: takes 353 second to estimate 1-step walk to 38-step walk, from 1,000,000 random walks each.

+ Error gets out of hand above 12-step walk, hence this is a poor estimator.