Homework 4

Maedeh Karkhane Yousefi

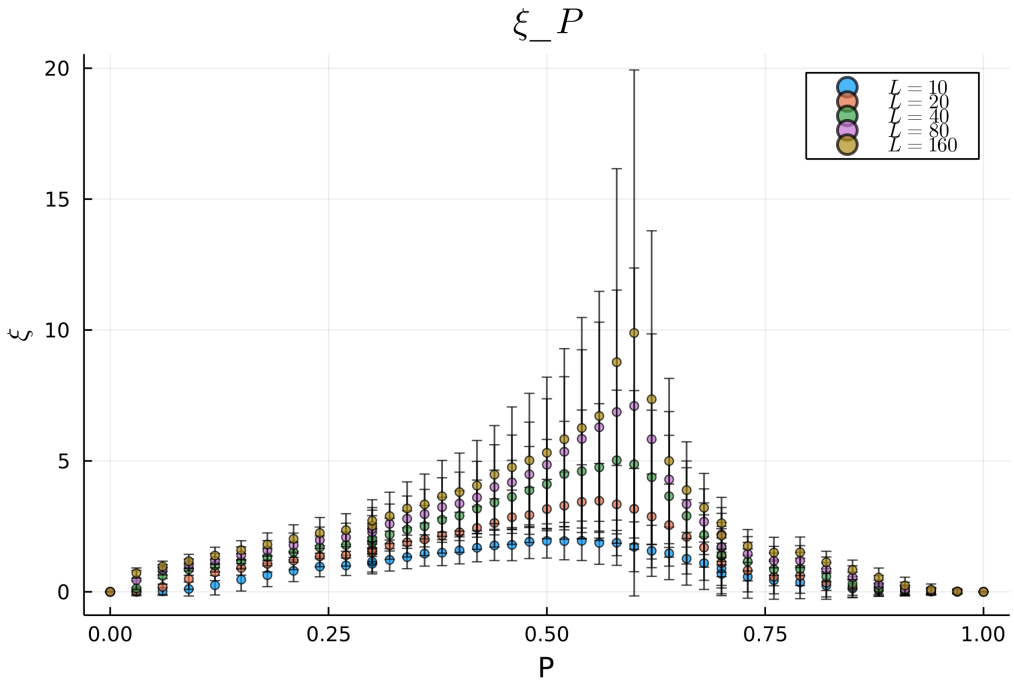
October 26, 2021

1. Exercise 4.5,4.6: Correlation Length

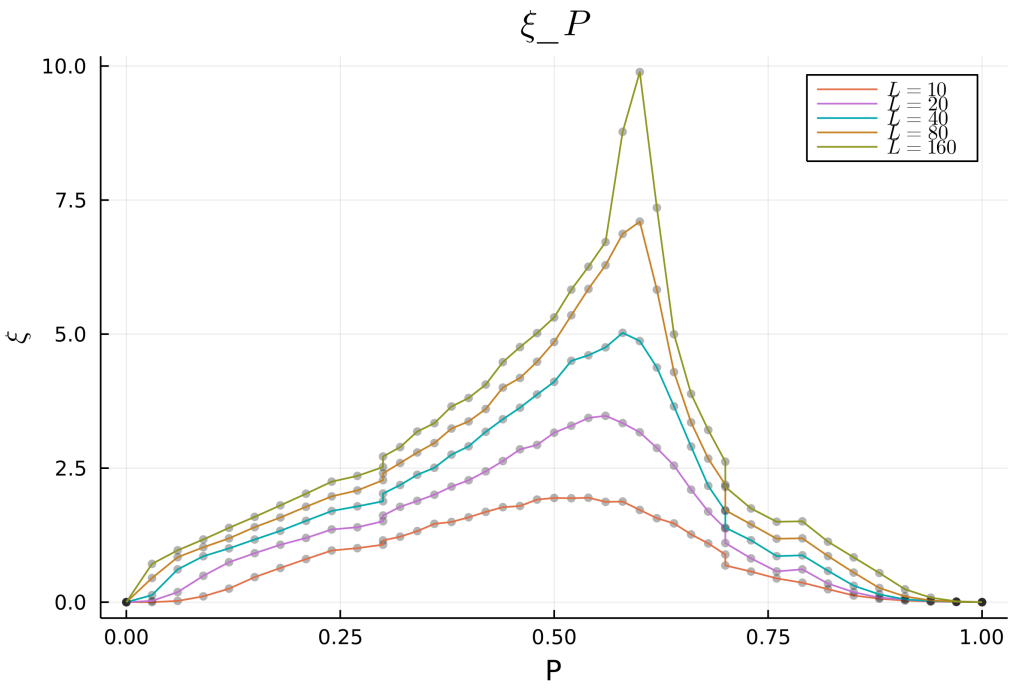
We are supposed to obtain the correlation length in each run for the previous code using the HK algorithm (exercise 4.4). I added the function *RadiusOfGyration*, which finds the maximum finite cluster and according to it's size and the related label, finds all the entries that are in this cluster, gets their indices, and their center of masses respectively. At last, the distance of all the entries from the center of mass is calculated, and the radius of Gyration is obtained accordingly. For each length in the list, we run the program for 1000 times, and the mean and STD of the Radius of Gyrations (ξ) is calculated.

In order to get the amounts ν and $P_c(\infty)$, I used the package *LsqFit* and the *curve_fit* method subsequently. Putting the equation($|p_c(L) - P_c(\infty)|^{-\nu} \sim L$) as the *model*. The critical values of probability for each length were resulted from the achieved data, and the preassumed values for ν and $P_c(\infty)$ are 0.59 and 1.3, respectively.

results from *curve_fit* are: $\nu \sim 0.61$, $P_c(\infty) \sim 1.02$



(a) The scatter ξ over probability figure (error bars included).



(b) The plotξ over probability.

Figure 1: The correlation lengths over probability figures of exercise 4.5. probability: $0 \leq p \leq 0.25$ ($steps = 0.03$),$0.25 \leq p \leq 0.75$ ($steps = 0.02$), $0.75 \leq p \leq 1$ ($steps = 0.03$) , Length List= [10,20,40,80,160], Run Number=1000

2. Exercise 4.7: Cluster Growth Algorithm

The overall explanation for this code is that, the program loops through each entry of the matrix, finding that entry which is empty (*zero*) and has at least one *ON* neighbor; After adding this entry to the list of entries that have the same characteristic, each member of the list are sent to the function *on_or_block*, becoming either *ON (represented with 1)* or *Blocked (represented with -1)*, by a given probability. The operation stops when the cluster is blocked either by the boundaries or the blocked entries.

The correlation lengths and the size of the clusters for the given probabilities, averaged over 100 runs are saved in the *"ClusterGrowth_S_xi.jld"* file.

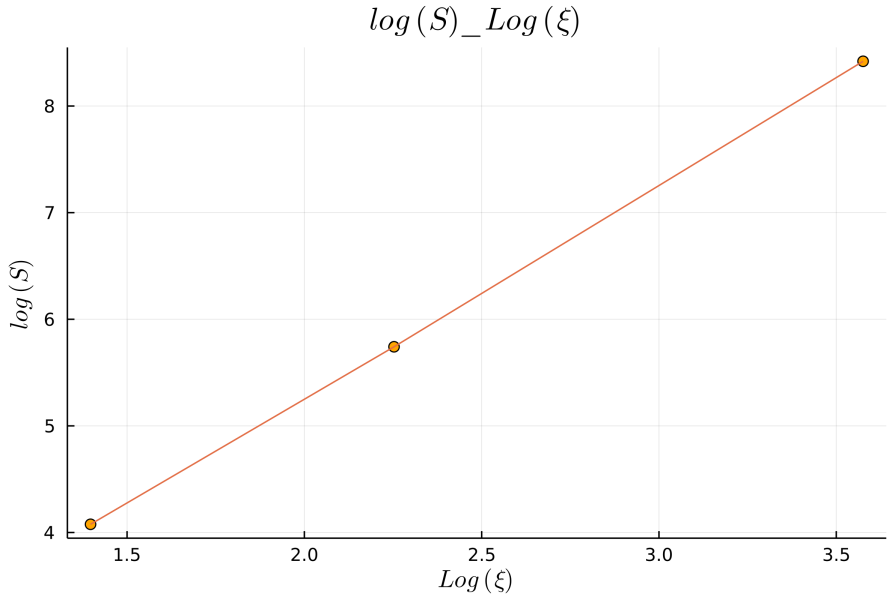


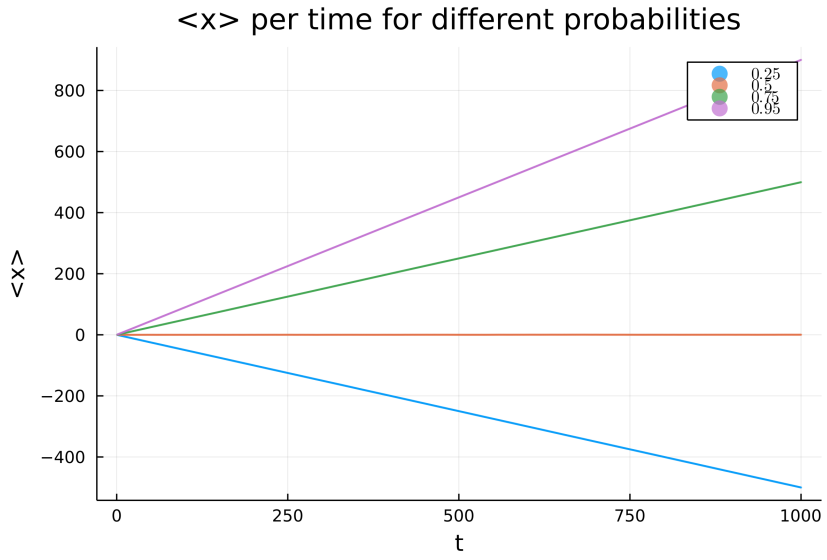
Figure 2: The log(S) over log(ξ) plot for exercise 4.2.the list of probability=[0.5, 0.55,0.59], Run Number=100.As it's clear from the figure, a line can be plotted passing through the points.

3. Exercise 4.1

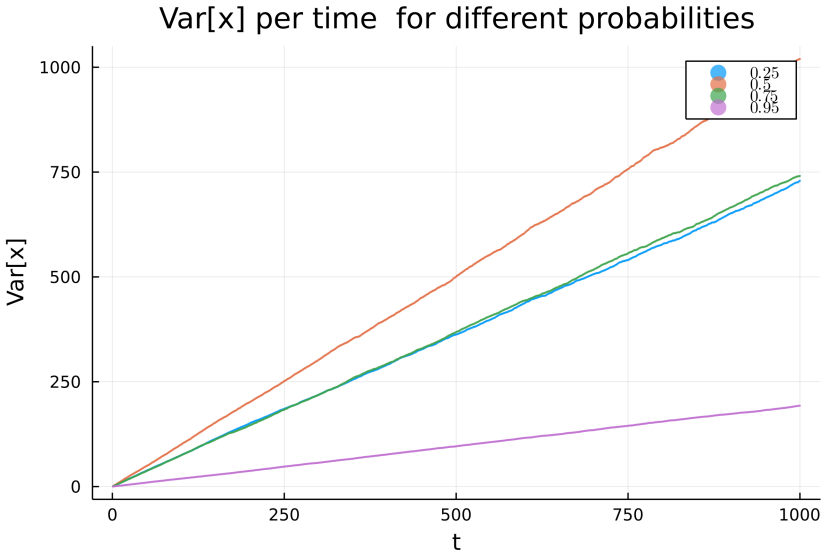
$\langle x^2(t)\rangle = \langle (x(t-\tau)+al)^2\rangle = \langle x^2(t-\tau) + 2alx(t-\tau) + a^2l^2\rangle$
$= \langle x^2(t-\tau)\rangle + 2l\langle a\rangle\langle x(t-\tau)\rangle + l^2\langle a^2\rangle = \langle x^2(t-\tau)\rangle + 2l(p-q)\langle x(t-\tau)\rangle + l^2(p+q)$
$= \langle x^2(t-\tau)\rangle + 2l(p-q)\left(\frac{l}{\tau}(p-q)(t-\tau)\right) + l^2(p+q)$
$= l^2\frac{t}{\tau}\left((p-q)^2\left(\frac{t}{\tau}+1\right) - 2(p-q)^2 + (p+q)\right)$
$\sigma^2(t) = \langle x^2(t)\rangle - \langle x(t)\rangle^2$
$= l^2\frac{t}{\tau}\left((p-q)^2\left(\frac{t}{\tau}+1\right) - 2(p-q)^2 + (p+q) - (p-q)^2\frac{t}{\tau}\right)$
$= l^2\frac{t}{\tau}(p - p^2 + q - q^2 + 2pq) = l^2\frac{t}{\tau}(pq + qp + 2pq)$
$= \frac{4l^2}{\tau}pqt$

4. Exercise 4.2: Random Walk

I consider the dynamics a matrix, which one dimension represents the time steps and the other probabilities. In other words, each column is one walker's changing positions to left and right in a total time of 1000. The entries of the matrix are filled with the given probabilities with 1 (right) and with -1 (left) at once at the beginning. At the end using *cumsum*, the matrix of positions at each time, is returned to the a for loop, to repeat for 10000 runs. The position mean and variance per time is listed in a list for arbitrary probabilities, and the figures are plotted.



(a)

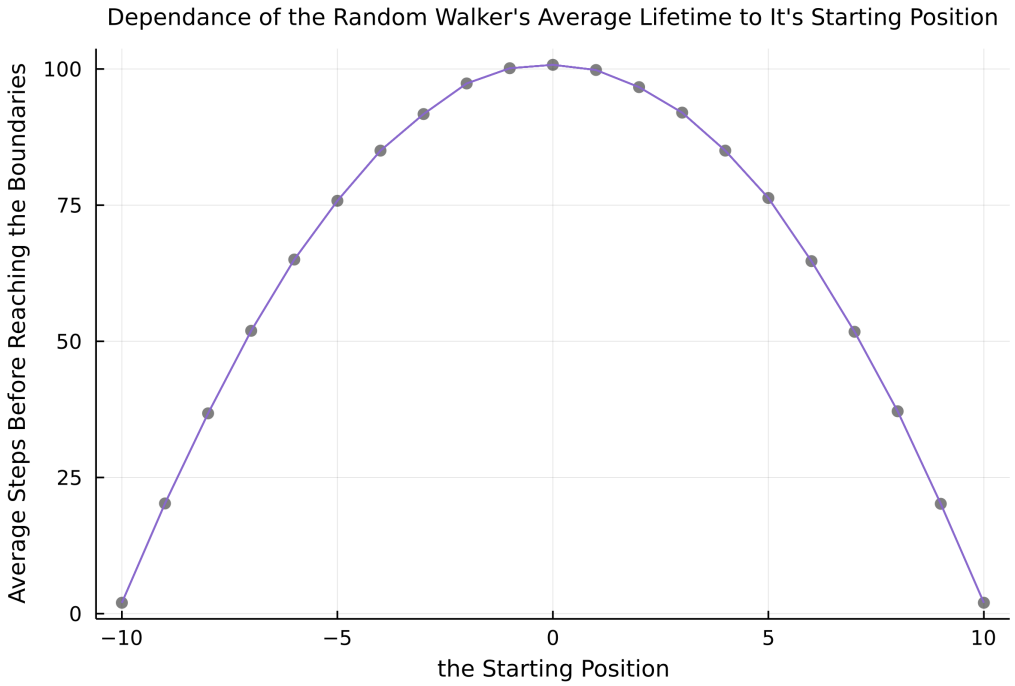


(b)

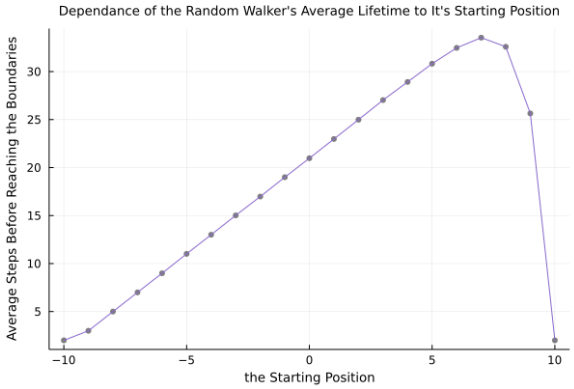
Figure 3: $\langle x \rangle$ and variance(x) over time. Total time=1000, Run Number=1000, list of probabilities =[0.25,0.5,0.75,0.95]

5. Exercise 4.3:Random Walk with Traps

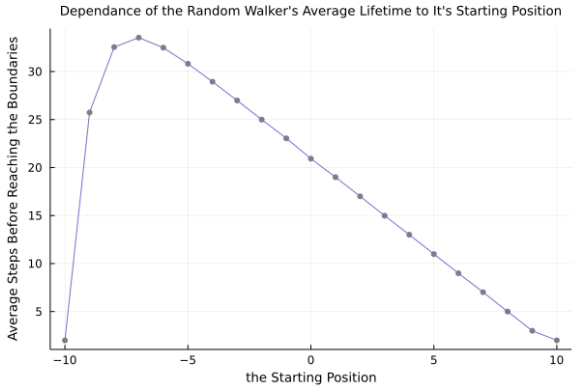
I considered each walker's movements being ordered in a one dimensional array. This time in order to stop the the random walker when reaching the boundaries, we cannot fill the entries at once! It should be done step by step. After each step, the cumulative sum is being calculated instantly to check whether we have reached the positions -10 or 10 yet. If not, the loop continues, but if yes, then the resulting array is returned to a loop to repeat the process for 100000 times for each starting position.



(a)



(b)



(c)

Figure 4: Average steps before falling into the traps over the starting position for (a) p=0.5, (b) p=0.25 and (c) p=0.75 for 100000 walkers. Starting position in range form -10 to 10.

6. Exercise 4.4: Random Walk with Traps(Deterministic Algorithm)

According to this algorithm we should produce the table mentioned in the textbook. So, I created a list, consisting of the probabilities in each step. In every step, a *for loop* goes through each entry of the *R list*, which initially is set to be zero. Every value of the previous step's probabilities is divided by 2 and added with the previous and the next entries' values in this step's list.

The first and the last entries in each step add with the first and the last step's entries, respectively. These entries are the traps.

The process goes on until the sum of the traps in each step (the death probability) become a number near to 1, which I assumed to be 0.99999999.

The mean life-span is set to zero initially and adds with the sum of the life-probabilities in every step.

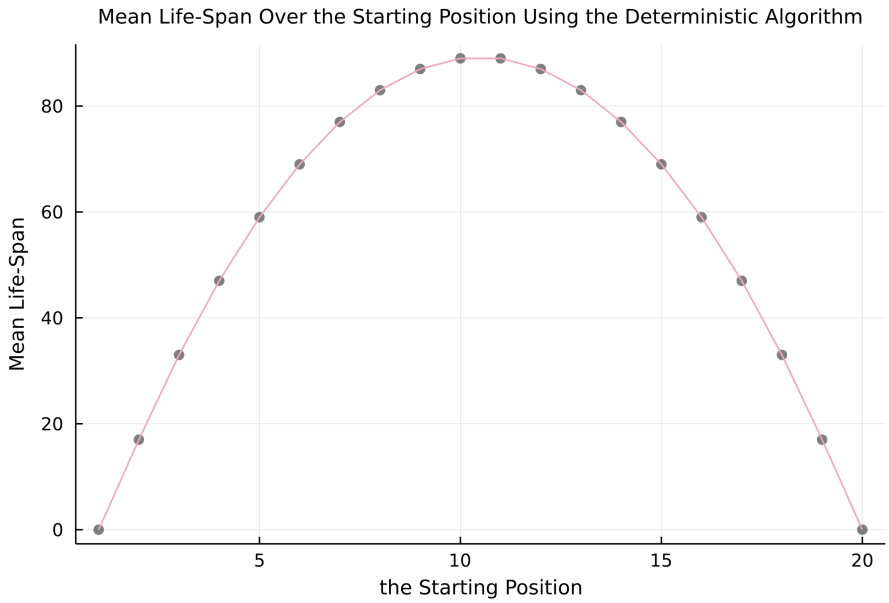


Figure 5: The mean life_ span of each walker over the starting positions ranging from 1 to 20. The run-time is significantly less than the previous algorithm's run-time, and the results are both the same, obviously.