Series 8- Computational Physics, MetroPolice Algorythm

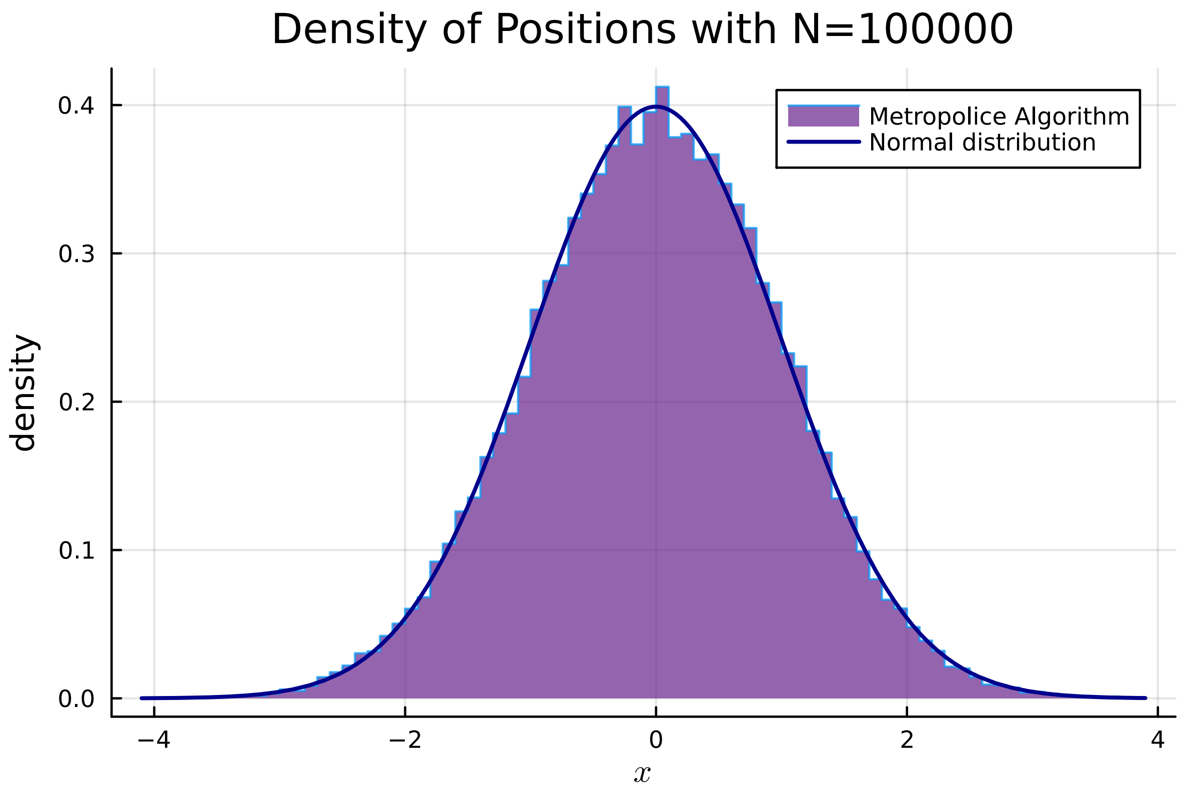
Zahra Akbari 98100612

8.1

In the first section of the question, implementing the Metropolice algorithm is asked. The code is meant to do that based on the explanations. In the code the Metropolice func gets a function to use as the desired distribution, a distance as the delta by which the new point is selected from starting point(step length) and a n variable which is the steps in which the points are generated and saved in PositionArray. The code also calculates acceptance rate and returns it witht the PositionArray.

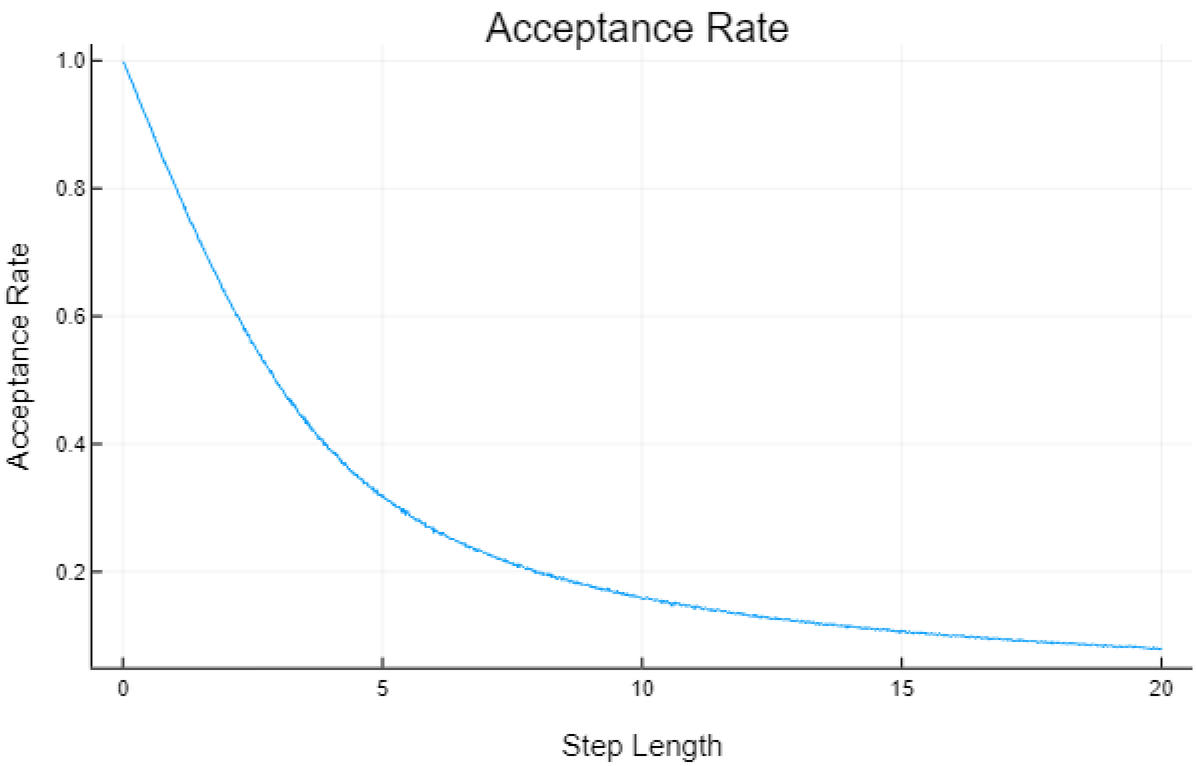
```
function Metropolis(func, distance, n)
    Position = 0.0
    count = 0
    PositionArray = zeros(n)
    for i in 1:n
        PositionArray[i] =
Position                                                     #
Saving the X poistions of ball moving throgh the domain.
        ProbableDestination = Position + distance *
rand(Uniform(-1,1))
        TransitionProbability =
func(ProbableDestination)/func(Position)              # Cheking
if the ball is going to move to a new position
        if rand() <= TransitionProbability
            Position = ProbableDestination
            count = count + 1
        end
    end
    AcceptenceRate =
count/n
 # Calculating acceptence Rate
    return PositionArray, AcceptenceRate
end
```

in the rest of the block the function for normal distribution is defined. Runing the code we get the following result:



Now we want to find the corresponding step length (distance variable) for each Acceptance Rate in {0.1, 0.2 ,0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}. For finding these values the Metropolice function is runned in a loop with distance variables from 0.01 to 2 with 0.01 steps. The results are plotted in an interactive plot so that we can find the correspoing values. The results are mentioned in the following table:

Acceptance Rate	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Distance or Step lengh	16.13	8.05	5.21	3.87	2.93	2.16	1.55	1.01	0.45

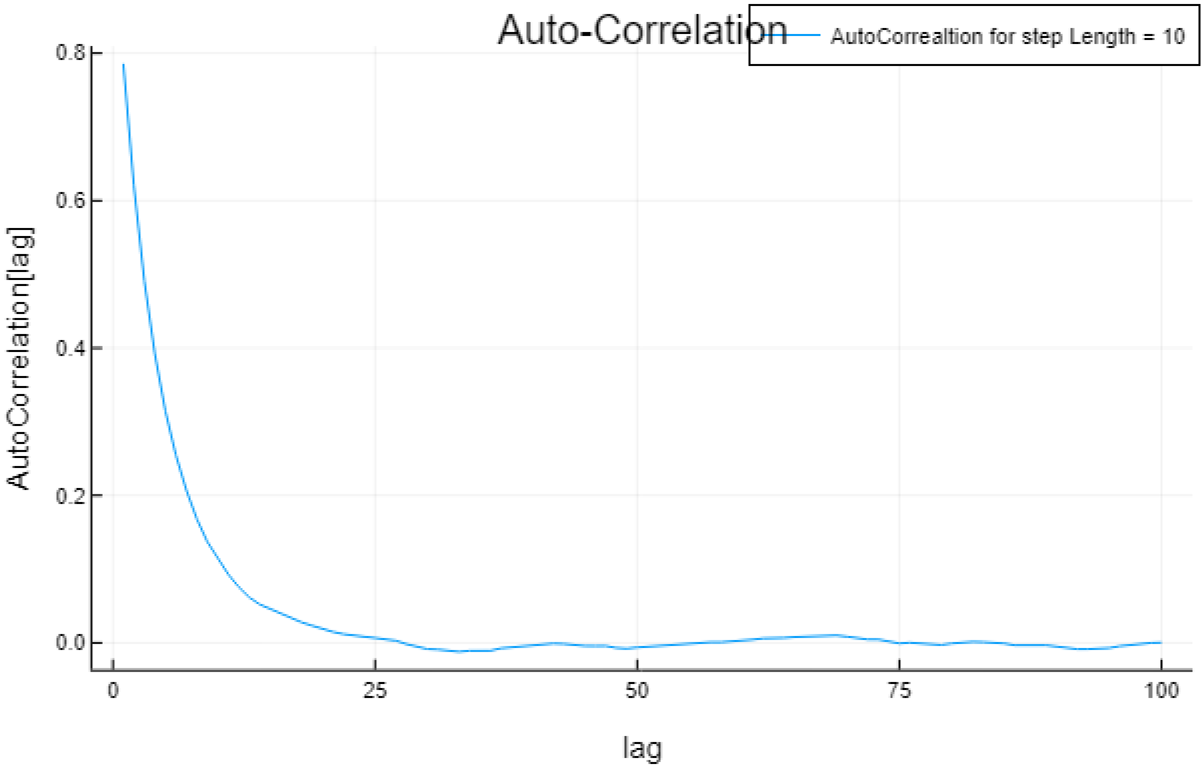


For the last part we are going to calculate auto correlation for the mentioned acceptance rates(Step Length). For doing so the following formula is used implemented:

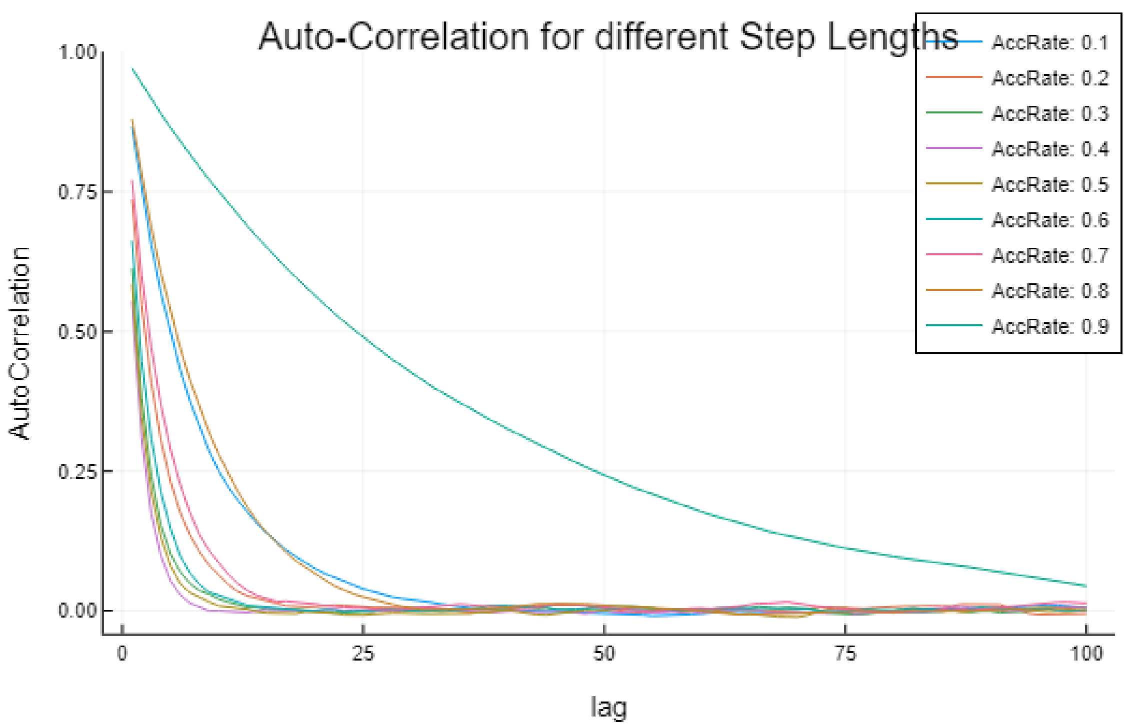
$$\frac{\langle x_i x_{i+j} >_i - \langle x_i \rangle_i < x_{i+j} \rangle_i}{\sigma^2}$$

The formula is implemented in a for loop in the AutoCorrelation function:

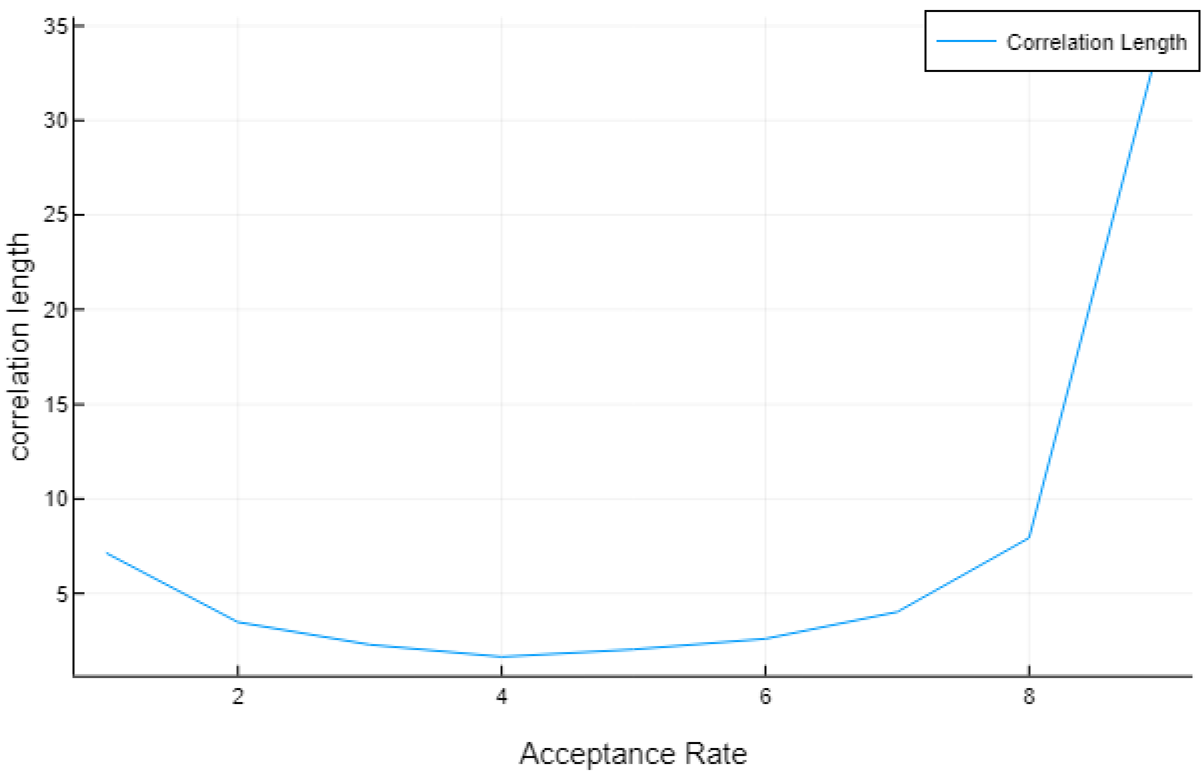
```
function AutoCorrelation(array,n)
    AutoCorr = zeros(n)
    @simd for j in  1:n
        AutoCorr[j] = (sum((array[1:(n-j)] .- mean(array)) .*
(array[(j+1):(n)] .- mean(array))))/ (N*var(array))        #
Using Auto-Correlation Formula mentioned in report
    end
    return AutoCorr
end
```



Now the same code is looped over for the corresponding Step length we found in the previous part resulting in:



By using least square method we fit a line to logarithm of the AutoCorrelationlists to find correlation legth for each Acceptance Rate.



By using the data from interactive Plot we get the values:

Acceptance Rate	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Correlation Length	7.14	3.48	2.29	1.65	2.04	2.61	4.02	7.94	34.48