Computational Physics: Problem Set 7

Mohammad Ghoraishi
99100788

April 25 2023

1 8.1 Metropolis Algorithm

In this problem we try to create a random generator with a Gaussian distribution using the Metropolis algorithm. First we decide on the initial values where I have chosen $\sigma = 1$ so the Gaussian distribution is as follows:

$$p(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

1.1 randRanged() function

This function was also used in the previous problem set and it just takes a lower and upper boundary and generates a random floating point number in that range.

1.2 metropolisRand() function

This function applies the metropolis algorithm and generates a random number. It takes the desired distribution function, the starting point,the step length and the number of random numbers we want to generate as input, then it generates an array of random numbers with the distribution of our desired function and the acceptance rate that was defined in the textbook. We test our function by generating a large amount of numbers, drawing the histogram and fitting the theoretical curve on top.

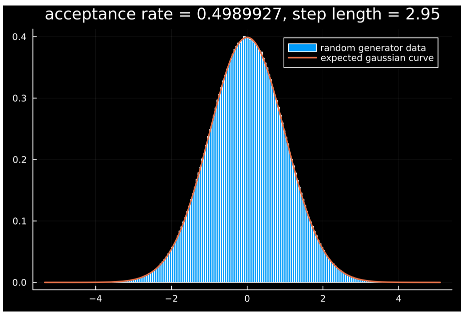

Figure 1: Distribution of the metropolisRand() random generator.

1.3 Acceptance Rate

Our metropolisRand() function also gives us the acceptance rate for every number generation. So we generate numbers using our random generator and play with the step length until it gives us and acceptable acceptance rate. We do that by generating 10 acceptance rates per step length, and modifying the step length until there are around 5 acceptance rates above and 5 below the desired amount and so we will reach an error of about 0.005 on the acceptance rating. The results are as follows:

acceptance rate	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
step length	15.9	7.95	5.32	3.89	2.95	2.2	1.58	1.02	0.5

1.4 correlation() function

This function takes an array of random numbers and j as input, then returns $c(j)$ defined in the textbook which is calculated using the formula below:

$$c(j) = \frac{< x_i x_{i+j} >_i - < x_i >_i < x_{i+j} >_i}{< x^2 > - < x >^2}$$

1.5 Correlation Length

We want to fit the curve $c(j) = e^{-\frac{j}{\xi}}$ to our data so we can plot $\log(c(j))$ with regards to j and the resulting slope will be $-\frac{1}{\xi}$ which we can use to calculate the correlation length. We generate a small set of $c(j)$ because it approaches zero extremely quickly and that data is not useful. We then fit a line through the data using the Polynomials.fit() function and calculate the correlation lengths: We can also test these results by plotting $c(j)$ based on j and plotting our

acceptance rate	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
correlation length	7.26	3.67	2.36	1.51	2.16	2.82	4.16	8.00	27.6

theoretical curve on top. The results for acceptance rate of 0.5 are as follows:

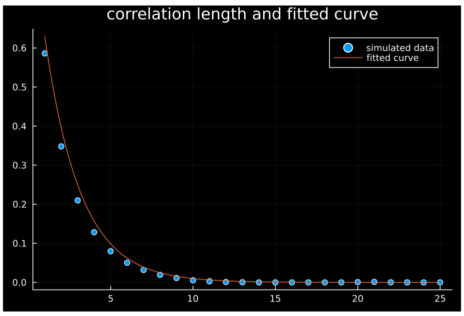

Figure 2: Testing the correlation length and the fitted curve for acceptance rate of 0.5