1 Question 1

1.1 Rejection Sampling

- This method is commonly used for quantum measurements.[5]
- Can some times require the generation of throw away random numbers before generating the values to be used.
- In the case of normalized distributions is very nice to work with. Other wise need to scale the generator function [7]
- In the case here can lead to many rejection
- Can reduce the rejection rate by using a gaussian envelope In this case. [1]

1.2 Markov Chain Monte Carlo

- Follows a random walk
- Could get stuck in a region of $f(x)$[6]
- May need a burnin time if initial guess is bad[3]
- No need to calculate the norm
- Need to choose an appropriate step generator to insure all domain is sampled.

1.3 Inverse Transform Sampling

- Need to create an inverse of the function
- Best to do that numerically
- Would need very small Δ as continuous
- Memory usage high to store array for inverse of p(x)
- In the given example each probability will have to x values
- Need flip coin to determine if grater then or less then μ
- Is non-continues [4][2]

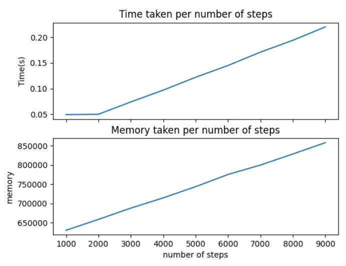
2 Question 2

The code was written in python and implemented the Markov Chain Monte Carlo methods. The code can be found in the file mha.py. I choses the Metropolis–Hastings algorithm as it does not need the calculation of the norm saving some time. Also this method does not have any need to make a choice when generating a sample as each x has a unique p. Directions for the code can be found in the readme.

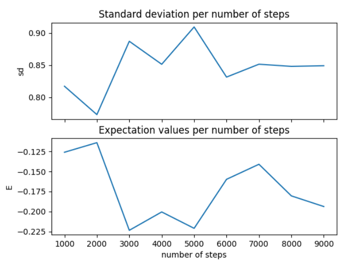
3 Question 3

Some results from for the Metropolis–Hastings algorithm are show below in the figures 1, 2, 3, and 4. The time and memory usage is shown for each set of parameter in 1(a), 2(a), 2(a), and 4(a). In these figures it is shown that time and the memory scale linearly with the number of steps regardless of the choice for parameters.

In figures 1(b), 2(b), 2(b), and 4(b), the expectation value and standard deviation are shown. In the case where μ was less then α the expectation value and standard deviation where close to the chosen parameters for p(x). When μ was taken to be larger then α, as was the case in 2(b), and 4(b), the method failed to converge in the vicinity of given parameters. In this case a different method would likely give better results.



(a)



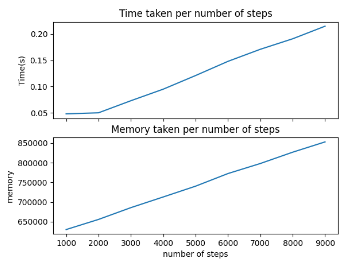
(b)

Figure 1: Fig (a) shows the memory and time taken for $\mu = 0$, $\omega = 1$, $\alpha = 1$, and $\beta = 0.5$. (b) show the expectation value and standard deviation for the same values.

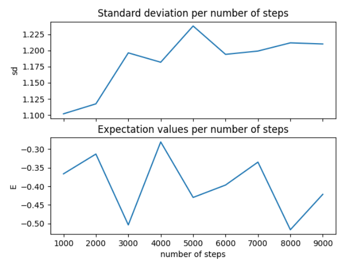
In figures 5 and 6 the results are shown for when the method was allowed to run for longer, a larger number of steps. Again the growth of the time taken and memory usage remained linear as seen in figures 5(a) and 6(a). Figures 5(b) and 6(b) show that the expectation value and the standard deviation do converge with increased steps. Once again the error is greater in the case that μ is grater then α.

4 Question 4

In the case of having accesses specialized hardware, I would make changes to the way p(x) is calculated. The calculation of p(x) would be moved onto the

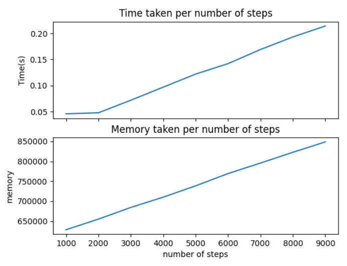


(a)

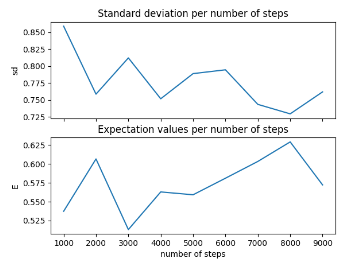


(b)

Figure 2: Fig (a) shows the memory and time taken for $\mu = 0$, $\omega = 1.5$, $\alpha = 1$, and $\beta = 0.5$. (b) show the expectation value and standard deviation for the same values.



(a)



(b)

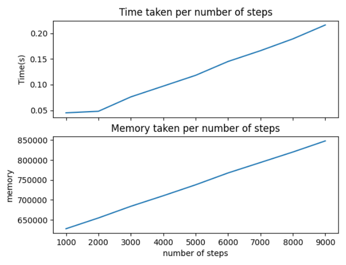
Figure 3: Fig (a) shows the memory and time taken for $\mu = 1$, $\omega = 1$, $\alpha = 1$, and $\beta = 0.5$. (b) show the expectation value and standard deviation for the same values.

accelerator. I would set it up so all the calculations concerning the probability would stay on the accelerator. In the case of multiple processor I would set up a walker on each processor to run independently and then merge the results of the walker to get better accuracy at the end.

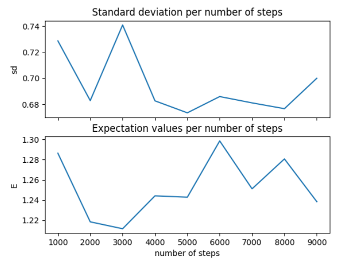
References

[1] Rel Guzman. Sampling: Rejection Sampling Explained . `https://relguzman.blogspot.com/2018/04/rejection-sampling-explained.html`, 2018.

[2] Kristin Kuter. 4.1: Probability Density Functions (PDFs) and Cumulative Distribution Functions (CDFs) for Continuous Random Variables. `https://stats.libretexts.org/Courses/Saint_Mary%27s_College_`

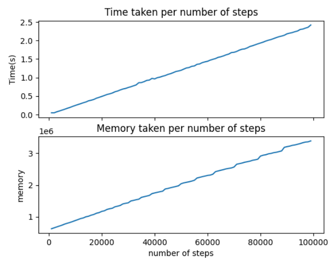


(a)

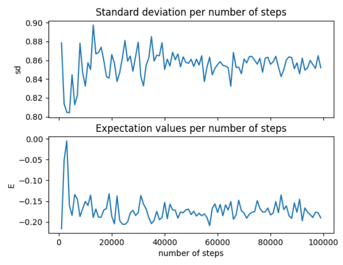


(b)

Figure 4: Fig (a) shows the memory and time taken for $\mu = 2$, $\omega = 1$, $\alpha = 1$, and $\beta = 0.5$. (b) show the expectation value and standard deviation for the same values.



(a)



(b)

Figure 5: Fig (a) shows the memory and time taken for $\mu = 0$, $\omega = 1$, $\alpha = 1$, and $\beta = 0.5$. (b) show the expectation value and standard deviation for the same values. Both of the above figures show results for a large number of steps.

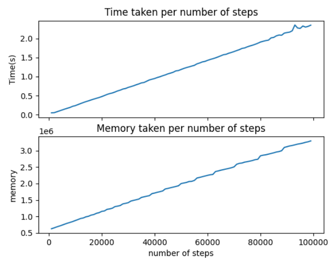
`Notre_Dame/MATH_345__-_Probability_(Kuter)/4%3A_Continuous_Random_Variables/4.1%3A_Probability_Density_Functions_(PDFs)_and_Cumulative_Distribution_Functions_(CDFs)_for_Continuous_Random_Variables`, 2024.

[3] Danielle Navarroe. The Metropolis-Hastings algorithm. `https://blog.djnavarro.net/posts/2023-04-12_metropolis-hastings/`, 2023.

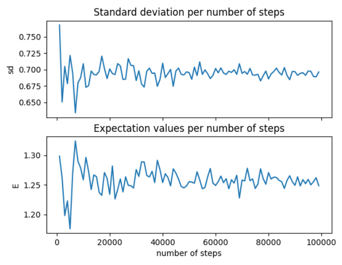
[4] H. Pishro-Nik. Introduction to probability, statistics, and random processes. `https://www.probabilitycourse.com`, 2014.

[5] A. Rivas and S. F. Huelga. *Open Quantum Systems: an introduction.* Springer, 2011.

[6] Christian P. Robert. The metropolis-hastings algorithm, 2016.



(a)



(b)

Figure 6: Fig (a) shows the memory and time taken for $\mu = 2$, $\omega = 1$, $\alpha = 1$, and $\beta = 0.5$. (b) show the expectation value and standard deviation for the same values. Both of the above figures show results for a large number of steps.

[7] Jake Tae. Rejection Sampling . `https://jaketae.github.io/study/rejection-sampling/`, 2021.