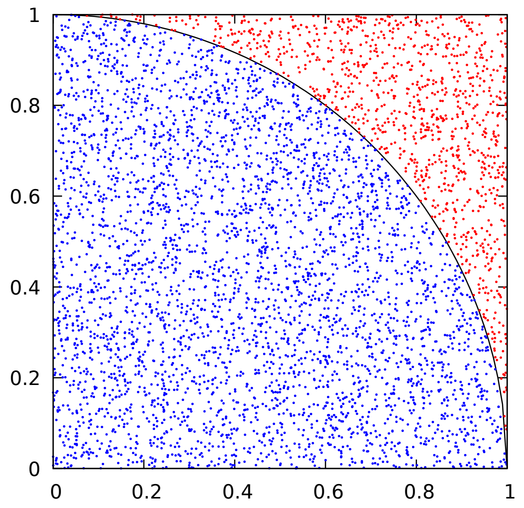
Monte Carlo Writeup

Angela Shen

January 2023

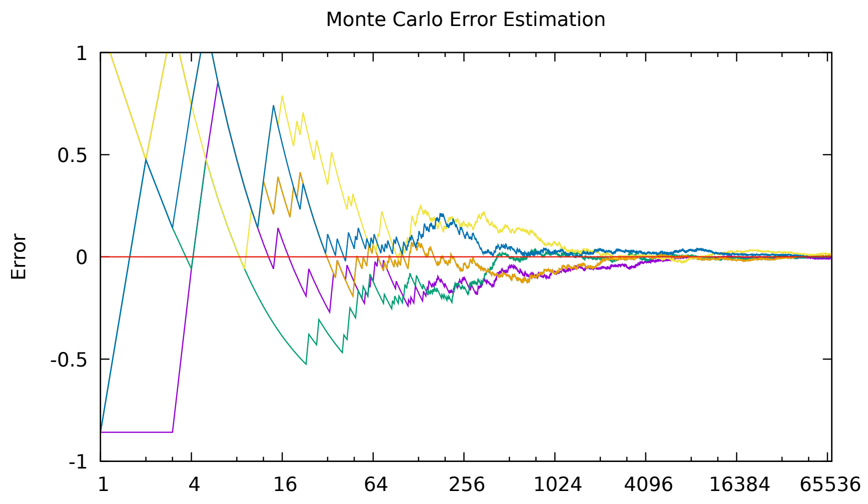
1 Visual Representation of Monte Carlo



This first graph creates a visual representation of the Monte Carlo experiment, using dots rather than coconuts. Blue dots represent "coconuts" that fall inside of the circle and red dots represent ones that fall outside of the circle, creating a simplified idea of what a Monte Carlo experiment might look like.

To generate the data used in the graph, I used 5000 iterations because I thought it would be a sufficient amount of dots to represent the simulation without being too cluttered. To ensure that the headers of each column would be excluded, I used tail. The necessary columns for plotting this graph are columns 3, 4, and 5 which contain the x, y, and circle boolean values respectively. Since the dots outside and inside of the circle are different colors, I separated the output into two .dat files using an "if" statement in awk to print the third and fourth columns depending on the value in column 5. This created one .dat file consisting of data points inside the circle (where column 5 is equal to 1) and another consisting of data points outside the circle (where column 5 is equal to 0). Through the process of creating this graph, I learned about bash scripting, gnuplot, and conditional statements in bash.

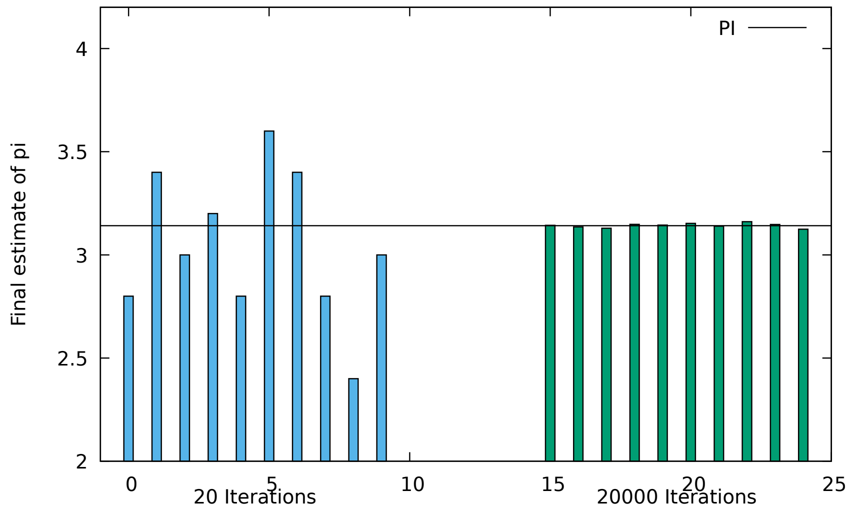
2 Error Estimation Graph



This graph is a line plot rather than a graph with dots, showing the general trend of the estimation of pi for multiple simulations of Monte Carlo. The error values of each line generally starts at around 1 or -1 and as the x value (number of iterations) increases, the error value approaches 0. This demonstrates that as more iterations of Monte Carlo are performed, the estimate of pi generally becomes more accurate.

To generate the data used in the graph, I used 70000 iterations of Monte Carlo because this plot is less concerned about clutter (in relation to the previous graph) and because a greater amount of iterations can better demonstrate the trend of the data. I also used tail and awk for this graph, though I did not use any conditionals for this graph. Instead, I used awk to subtract pi from each value in column 2. I repeated this 5 times, using sleep 1 in between each simulation to ensure the seed would be different, creating 5 different .dat files to plot. I learned about more uses for awk by creating this graph.

3 Histogram Graph



This graph uses a bar graph to demonstrate a general trend similar to the one shown in the previous graph. It compares the final result (estimation of pi) for 10 simulations of Monte Carlo with 20 iterations and

10 simulations of Monte Carlo with 20000 iterations. A line representing the value of pi is also depicted, to show how close an individual simulation was to pi. In general, the estimates for the simulations with 20000 iterations should be closer to the value of pi. The simulations with more iterations are also generally more consistent with each other compared to the simulations with less iterations, which give more inconsistent results.

This graph uses multiple simulations of pi, which are concatenated into a single file for plotting. First, I created 10 simulations of Monte Carlo with 20 iterations each. Rather than using sleep 1, I used the bash $RANDOM variable, because I felt extending the runtime by 18 seconds would be excessive, and I accept the risk of accidentally generating the same data sets. I used tail and awk to record the last row and column 2 of each simulation, putting each simulation into separate files and then concatenating them. I repeated this process with 10 simulations of 20000 iterations. This process taught me more bash features like cat and $RANDOM. I was also able to learn about gnuplot histograms.

4 Credits

For code that was not created by me:

On Tuesday (1/17/23) I had a tutor session with Jennie Lin where she showed me how to use tail and the -q/-n flags to filter the headers out of the data.

On Friday (1/20/23) I had a tutor session with Audrey Ostrom where she explained how to hardcode pi and use awk to subtract pi from the values in column 2 of the Monte Carlo output.

On Saturday (1/21/23) Professor Long posted code using sleep 1 to generate unique Monte Carlo data sets for the Error Estimation graph.