Assignment 1 WRITEUP.pdf

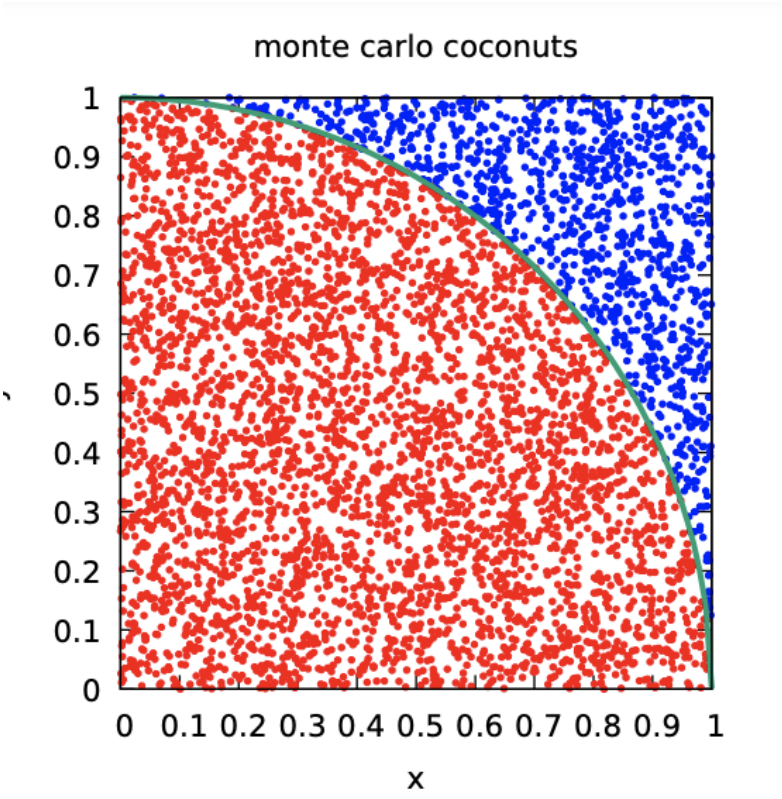
Julian Chop

January 23, 2023

1 Monte Carlo Coconuts

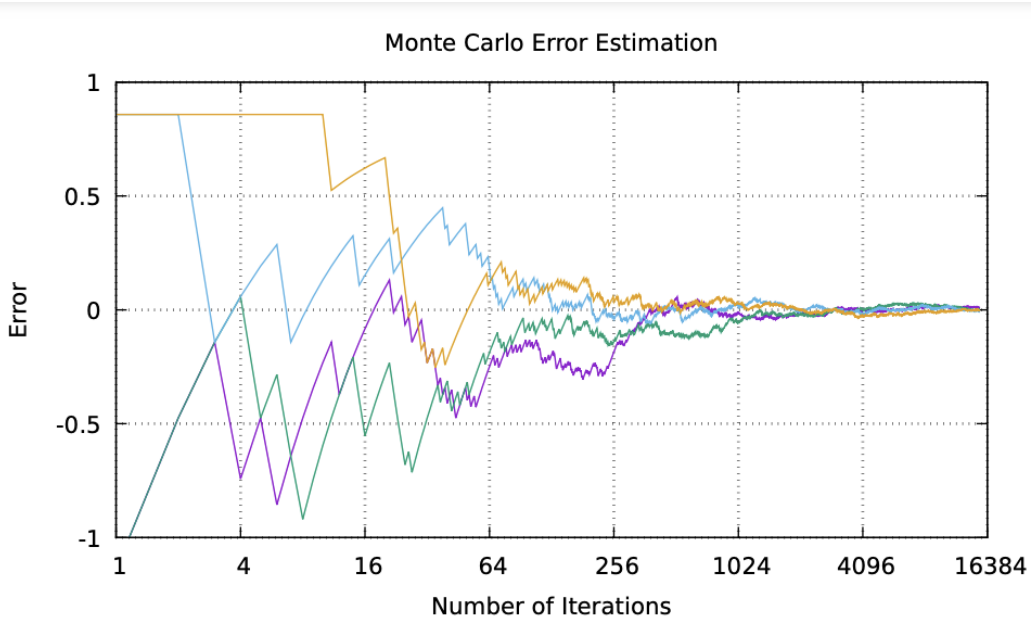
This first graph visually shows how the monte_carlo c program works. It randomly chooses a point in a 1-by-one square with a quarter circle with a radius of 1 inside it. Then it checks if that random point is within the quarter circle or not. Based on those points, it will estimate the value of pi. The longer you iterate over this process, the closer the program can estimate the value of pi. The x-axis of the graph is the x-coordinate, and the y-axis is the y-coordinate.

As you can see in the graph, the points that are within the circle are colored red, while the points outside the circle but within the square are colored blue. This is the result of the monte_carlo program being run 5000 times. The quarter circle is almost filled in. Having this many points gives a good estimate of the area of this circle. Having the area, the monte_carlo program can get a better estimate of pi. If you were to run the program even more times, you would get an even closer estimation.



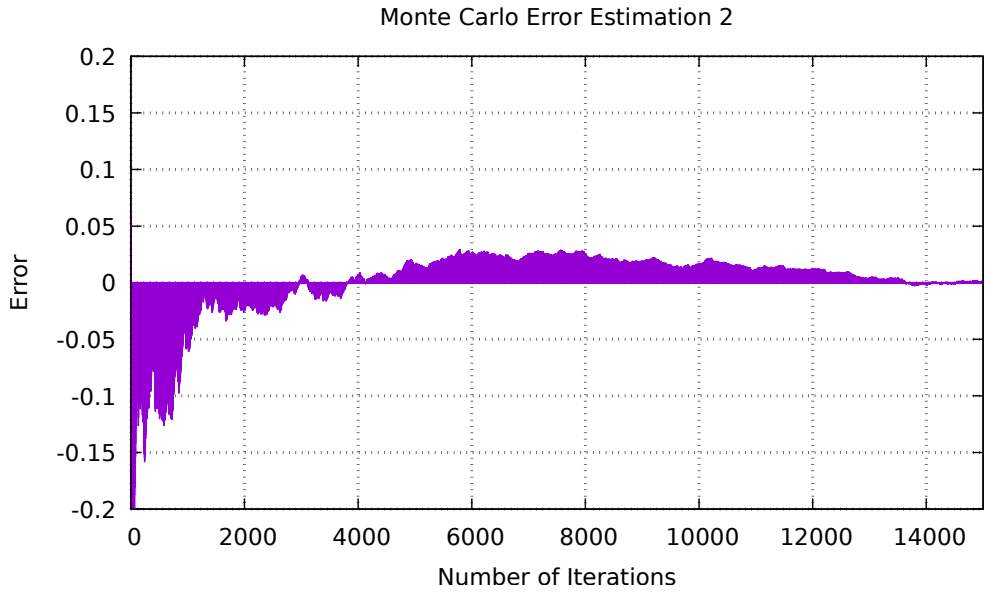
2 Monte Carlo Error Estimation

This graph shows the difference between the monte_carlo program's estimation of pi and the value of pi(Note that for the value of pi, I used 3.14159265359). The x-axis of the graph is the number of iterations of monte_carlo run, and the y-axis is the difference between the current estimation of pi and the value of pi. There are four lines on the graph, each one representing a running of monte_carlo, each with a different seed randomizer. The longer the programs are iterated, the difference between their estimated values of pi and pi gets smaller and smaller. This is shown in the graph. In the first few iterations, the line graphs are pretty far away from 0. But as more iterations are run, you can see the lines get closer and closer to 0.



3 Monte Carlo Error Estimation 2

This graph is basically the same as my second graph in that it shows the estimation error of the monte_carlo program based on its current iteration. But instead of using a line to show how the estimation gets closer to pi, I used a bar graph. I did this to show a different visualization of the estimation error. Because I used a bar graph, you can see the shaded areas that show the amount of error in the estimation. Then as more iterations are run, you see the shaded areas get smaller and smaller showing how the error gets smaller.



4 UNIX Commands Used

Make Command

make clean %% make monte_carlo

This command comes from the Makefile we have in the directory. 'make clean clears your directory of the monte_carlo executable and any .o files. Then, 'make monte_carlo' creates a monte_carlo executable.

Tail command

Because the monte_carlo program has words in the first row of output which shows the reader what each column represents, gnuplot cannot read it. To fix this, I used the tail command in junction with pipes to remove the first row of the mont_carlo output. The line of text below shows how I use the tail command using pipes. '*tail -n +2*' removes the first row of the monte_carlo programs's output.

$$./monte\_carlo -n 15000 | tail -n +2 | awk' \{print \$1 "" \$2 - 3.14159265359\} > /tmp/graph2L1.dat \quad (1)$$

Awk command

The Awk command can do many things to edit the outputs of data. The way I used it was to find the difference between the monte_carlo program's estimation of pi and the value of pi(in this case, 3.1415926359). In the code above, I wrote '*awk '{print $1" "$2 - 3.14159265359}*'. This command outputs the first column and and the different between the second column and pi. Now I can use this data for my error estimation graphs!

Pipes The '—' in between each of the different commands are Pipes. They basically take the output of the command and feed it to the command next in line. So after all my commands are run, the output goes into the graph2L1.dat file.

5 Conclusion

This lab got me familiarized with bash scripting, manipulating data in files, running C programs in UNIX, and gnuplot. It was a great introduction to what is to come in future assignments. Though I don't think I wrote an optimized bash script, it got the job done and introduced me to something I had never seen before. I think the hardest challenge in this assignment was just understanding how the components of the assignment come together. The Makefile, plot.sh, monte_carlo.c, README.md, DESIGN.pdf, and WRITEME.pdf. Making sure you're on top of it all and making sure everything is working was pretty stressful. In the end, assignment 1 was a good way of getting me Acquainted with UNIX and C.

6 Citations

- I went to Office Hours and Tutoring sessions to get ideas on how I might implement the data manipulation in my bash script. I ended up using tail commands I learned from the tutor Ben Grant.
- I watched a Youtube video on the basics of awk commands so I can use them in my code as well. Here is the link to the video I watched: `https://www.youtube.com/watch?v=9YOZmI-zWok&t=599s`
- I also used an equation given on discord by professor Long that makes a quarter circle. I used this to make my Monte Carlo Coconut graph. The equation was: [x=0:1] sqrt(1-x**2)
- For learning how to use gnuplot, I used to gnpuplot documentation and their website. Here is a link to their website: `http://www.gnuplot.info/`