Stephen Sun

Dr. Kerry Veenstra

CSE 13S

22 January 2023

# Assignment 1: Getting Acquainted with Unix and C Writeup

## Introduction:

In this assignment, the goal was to program a shell file, titled ‘plot.sh’, that when run, will create two plots similar to Figures 2 and 3 in the assignment pdf (shown below).

**Figure 2:**

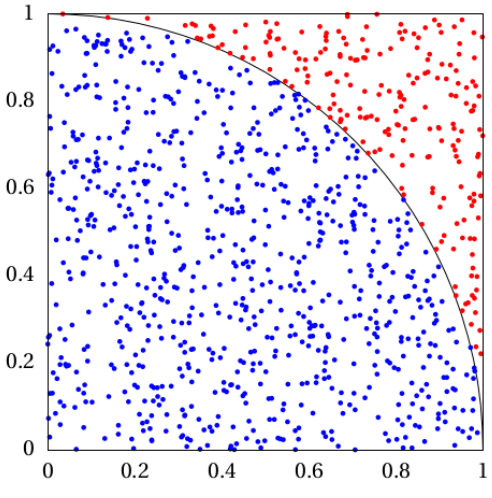
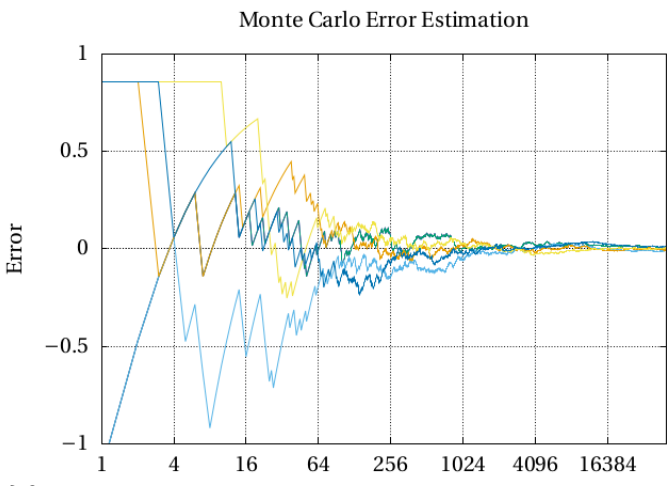



**Figure 3:**



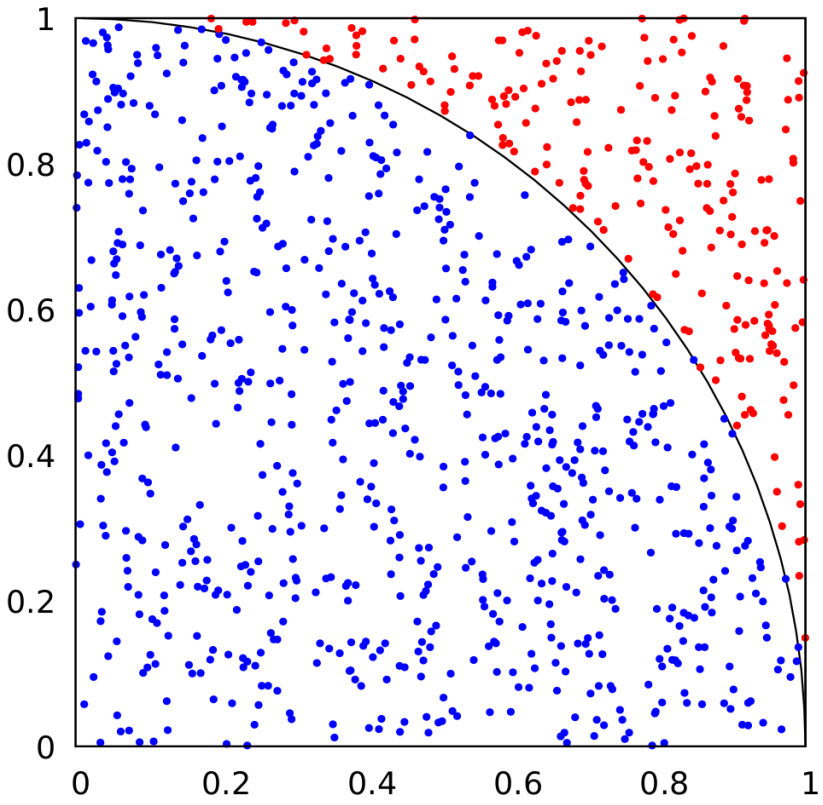

I will be discussing and breaking the commands used in my plot.sh file, as well as displaying the plots that I generated with my shell script. I will start off with the ‘make’ command. The command was used because I was provided a Makefile for this assignment that already had some code to compile the monte_carlo.c program so I just took advantage of that by using the ‘make’ command in my code. There was also a preset in there to clear previously created executable and

object files so the ‘make’ command could make use of that. ‘make format’ was the last section of the Makefile I was provided with. ‘make format’ ended up being the first line of code I wrote in my shell file. This line of code will format any .c or .h files it finds in the directory. In this case, it formatted monte_carlo.c

## Figure 2:

When recreating the Figure 2 plot, I used the command ‘./monte_carlo -n 1000’ so that the monte_carlo program will generate 1000 iterations. I chose 1000 after a bit of testing as that was the number that seemed to produce a plot that looked similar to the example given to us. I then used Unix file redirection to transfer the output in a .dat file. The purpose of this is that I can feed the .dat file to gnuplot to generate the graph. Next, I used ‘set terminal pdf’ and ‘set output’. This was because I wanted to have my plot be produced in a pdf file upon completion. The next 9 lines of code were mostly for cosmetic purposes in an attempt to match the sample plot I was given. The first of these commands is ‘set palette defined (0 "red", 1 "blue")’. This was simply to make the points outside of the circle red and the points inside blue to match the sample plot. I ended up getting the code for this after browsing through the gnuplot demos, specifically the ‘pm3d colors’ demo. I then generated the circle for the plot using the command ‘set object 1 circle at 0,0 size 1 back clip’. This generates a circle of radius 1 and centered at (0,0). The ‘back’ in this command will put it behind all the data points and ‘clip’ will prevent the circle from crossing the axes of the plot. ‘set xrange [0:1]’ and ‘set yrange [0:1]’ was used in order to control the size of the axes to only show that much of the graph. ‘set xtics scale 0’ and ‘set ytics scale 0’ was to get rid of the tics on the axes to match the appearance of the example plot. Then, I used ‘set size square’ to make the plot square, ‘set pointsize 0.25’ to make the size of the plotted

points smaller, and ‘unset colorbox’ to get rid of the color box that appears when you use the ‘palette’ command. Again, the purpose of those previous 3 commands were to make the graph appear like Figure 2 shown above. Finally, we are now actually able to run the command to plot the graph, which is ‘plot "/tmp/monte_carlo.dat" using 3:4:5 with points pt 7 palette title ""’. "/tmp/monte_carlo.dat" is datafile that I created when I ran the executable at the beginning of the section. This command will load the contents of that data file into gnuplot. The ‘3:4:5’ indicates which columns of the datafile will be used for the plot, in this case, column 3 will be the x-coordinate, column 4 being the y-coordinate, and column 5 used to determine the point’s color. ‘with points pt 7’ tells gnuplot to only use points for this graph and to use point type 7, which resembles the one used in the example graph. ‘palette’ tells gnuplot to use the color palette we set earlier, and finally, ‘title “”’ will create an empty title, as otherwise, a title will be generated in the top left of the graph. In the end, a plot that looks like this should be generated:



I would say that is a pretty faithful recreation of Figure 2, so I'm pretty happy with that. Now, I will use the command 'reset' to reset my plot and then I can move on to Figure 3.

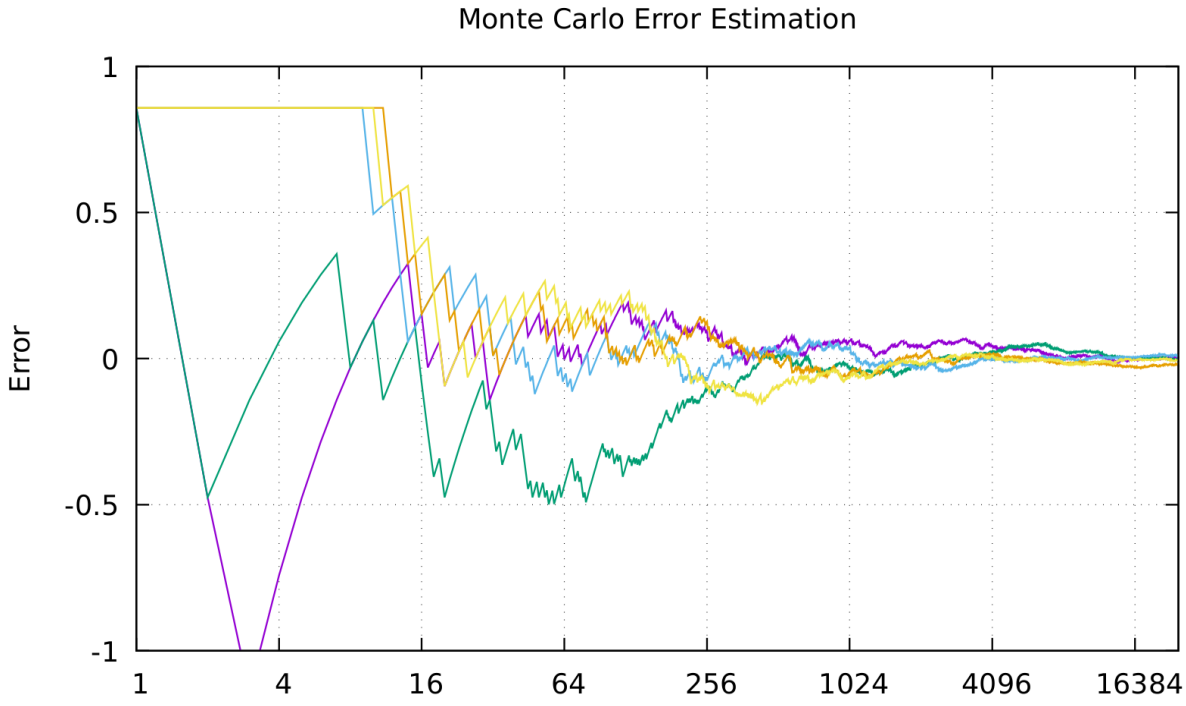
## Figure 3:

To start out for Figure 3, I will have to repeat a few of the same commands for Figure 2. I again start off by using './monte_carlo -n 25000' and redirecting the output into a .dat file. Note that the number of iterations is now up 25,000. The reason for that is because the Figure 3 graph given clearly goes beyond 16,384 iterations and I thought 25,000 was the next 'nice' number that was bigger. As figure 3 has 5 different lines, that means I had to repeat this process 5 times. However, this caused a problem as the data files were identical and only one distinct line ended up appearing. This was because the pseudorandom method to generate points in the monte_carlo.c program appears to be influenced by time, so not enough time has passed for new data to be produced. This was remedied by putting the command 'sleep 1' before generating a new .dat file. 'sleep 1' will put the program to sleep for 1 second, giving monte_carlo.c time to generate a new data set. I ended up choosing 1 second as it appeared to be the shortest amount of time I could get away with while producing 5 unique data sets. Now, moving into the code for gnuplot, I again started out by using 'set terminal pdf' and 'set output' just like for Figure 2. However, this time I set a title and a label for the y-axis using 'set title <title_name>' and 'set ylabel <label_name>'. This was to match the Figure 3 given to us. The next 5 lines were also for cosmetic purposes in order to match the appearance of the example Figure 3. 'set logscale x 4' was used to use a logarithmic scale for the x-axis, with the scale increasing by a factor of 4. 'set yrange [-1:1]' was used for the same reason as it was used for in Figure 2. 'set xtics in scale 1,0' is to make only the major tics appear on the x-axis, deleting the minor tics. Finally, 'set grid xtics

back' and 'set grid ytics back' was to create gridlines on the graph running parallel to the x- and y-axes, having them start at the tic marks for both axes. Now, I can actually run the command to plot the graph, which is:

```
plot "/tmp/monte_carlo1.dat" using 1:(column(2)-pi) with lines title "",\
"/tmp/monte_carlo2.dat" using 1:(column(2)-pi) with lines title "",\
"/tmp/monte_carlo3.dat" using 1:(column(2)-pi) with lines title "",\
"/tmp/monte_carlo4.dat" using 1:(column(2)-pi) with lines title "",\
"/tmp/monte_carlo5.dat" using 1:(column(2)-pi) with lines title ""
```

Doing it this way will ensure all 5 data sets will be plotted on the same graph. Aside from the fact that 5 different data sets have to be plotted, this is mostly the same command as the Figure 2 graph, though I will point out 2 differences. 'using 1:(column(2)-pi)' tells the gnuplot to plot the values found in column 1 of the .dat file as the x-value and the values found in column 2 subtracted by pi as the y-value. Note that 'pi' is a predefined number and gnuplot, so there was no need to assign a value to a variable. The other difference is that I plot 'with lines'. This basically tells gnuplot to connect the data points with a line and not to place a point at a data point. This code will create this graph below before terminating as the end of the shell file is reached.


Monte Carlo Error Estimation
Error
1
0.5
0
-0.5
-1
1
4
16
64
256
1024
4096
16384