Writeup Document - Assignment 1

Hsiang Yun Lu

January 23, 2023

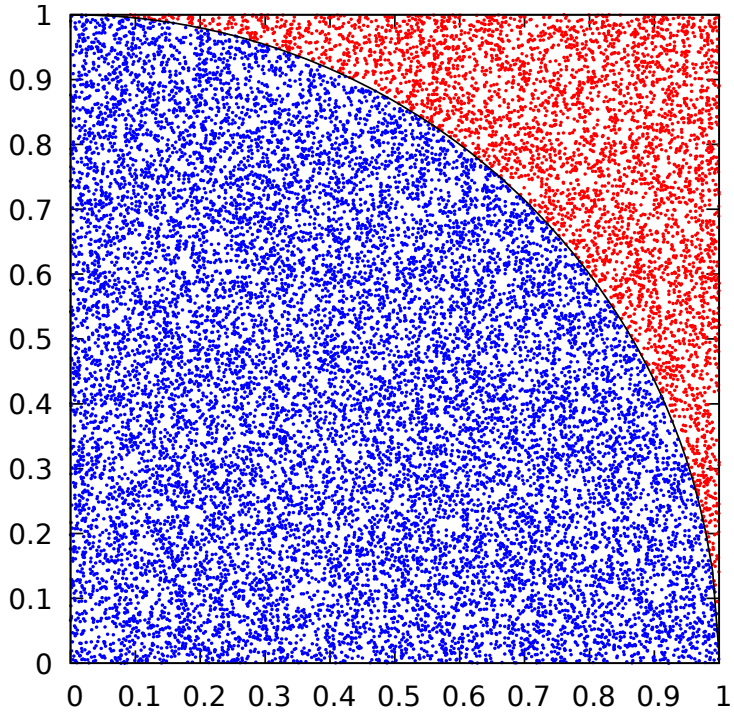
1 Monte Carlo Scattered Points for π Estimation

This plot demonstrates the uniformly scattered points in a square and inscribed quadrant. The points in the quadrant have a distance less than or equal to 1 and are blue. And the points that are a part of the square but not the circle are red. As iteration increases, which means more points, the estimated π value will get closer.

Bash script used to generate the plot:

```
./monte_carlo -n 20000 | tail -n +2 > monte_carlo.dat
awk '$5 == 1 {print $3, $4} ' monte_carlo.dat > blue_dots.dat
awk '$5 == 0 {print $3, $4} ' monte_carlo.dat > red_dots.dat

gnuplot <<END
    set terminal pdf
    set output "pi_dot.pdf"
    set size square
    f(x) = sqrt(1 - x**2)
    plot "blue_dots.dat" with dots title "" lt rgb "blue", \
        "red_dots.dat" with dots title "" lt rgb "red", \
        f(x) with lines title "" lt rgb "#000000"
END
```


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

2 Command Used and Discussion

- $./monte\_carlo -n$ 20000 | $tail -n +2$ > monte_carlo.dat
  - Run the monte_carlo C program and set the number of points to 20000 because more dots in the plot help demonstrate that the ratio of the number of points in the quadrant to the number of points in the square is the ratio of the area of the square to the area of a quadrant of a circle. The *tail* command with *-n +2* output starts with line 2, meaning trimming the header line. The output is redirected to a new file so that it can be separated into red and blue points.
- awk '$5 == 1 {$print$ $3, $4}' monte_carlo.dat > blue_dots.dat
  - The *awk* command goes through each line of the *monte_carlo.dat* file, uses the condition "the fifth column equals 1" to filter out only the coordinates of blue points and outputs as a new file. This approach is also applied to red points with the condition "the fifth column equals 0."
- gnuplot <<END
  - Sent the following here-document to gnuplot.
- set terminal pdf
  - To save the plots as pdf format.
- set output ''pi_dot.pdf''
  - Set output name.
- set size square
  - Set the output plot as square.
- $f(x) = sqrt(1 - x**2)$
  - Define the equation of the one-quarter circle.
- plot ''blue_dots.dat'' with dots title '''' lt rgb ''blue'',
  - To plot the blue dots with no title.
- f(x) with lines title '''' lt rgb ''000000''
  - To plot the one quarter circle with no title.

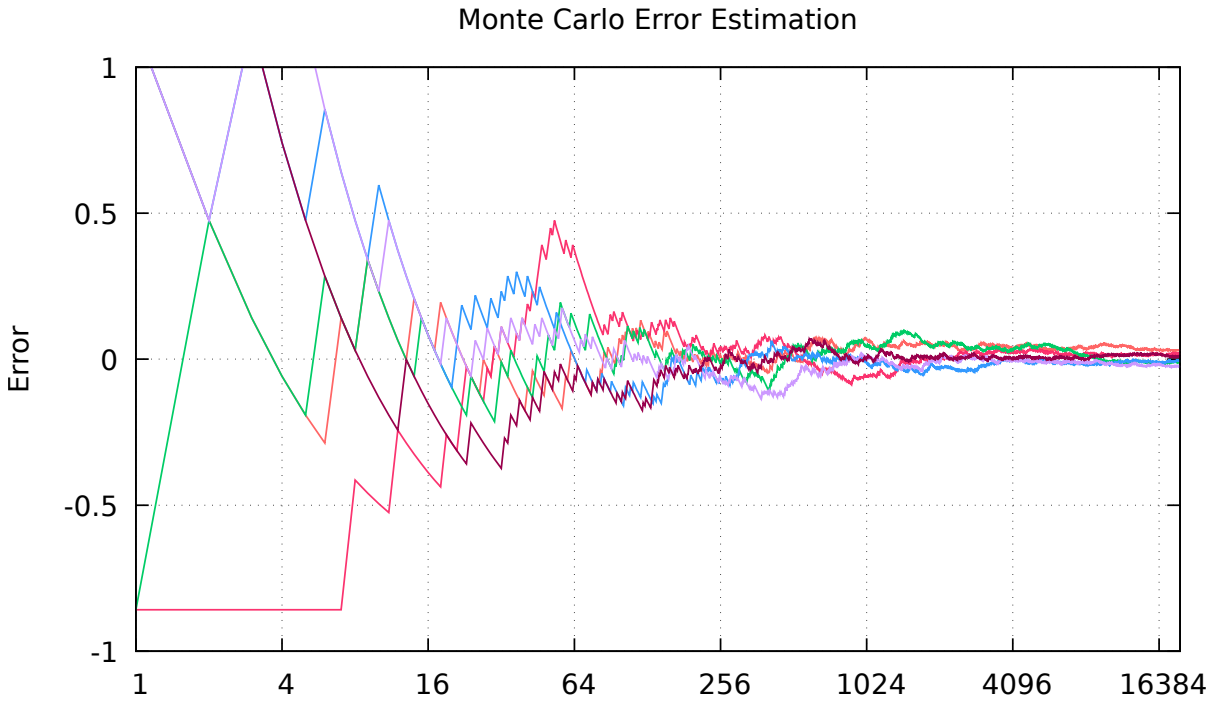
3 Monte Carlo Error Estimation

This plot shows the difference between the estimated π and "true" π as iteration increases under different seeds for the random number generator. As iteration increases, the error between the estimated π and "true" π becomes smaller and smaller, and approaches zero.

Bash script used to generate the plot:

```
for i in {0..5}
    do
            num=$RANDOM
            ./monte_carlo -n 20000 -r $num | tail -n +2 \
            | awk ' {print $1, ('$pi'-$2)} ' > mc_err_${i}.dat
    done

gnuplot <<END
    set terminal pdf
    set output "mc_error.pdf"
    set yrange [-1:1]
    set xrange [1:20000]
    set logscale x 2
    set title "Monte Carlo Error Estimation"
    set ylabel "Error"
    set grid xtics ytics
    plot "mc_err_0.dat" with lines title "" lt rgb "#FF6666", \
        "mc_err_1.dat" with lines title "" lt rgb "#FFB2666", \
        "mc_err_2.dat" with lines title "" lt rgb "#00CC66", \
        "mc_err_3.dat" with lines title "" lt rgb "#3399FF", \
        "mc_err_4.dat" with lines title "" lt rgb "#CC99FF", \
        "mc_err_5.dat" with lines title "" lt rgb "#99004C"
END
```


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

4 Command Used and Discussion

- `pi=‘echo ‘‘4*a(1)’’ | bc -l‘`
  - The `bc` command with the option `-l` is used for calculation with the standard math library. In the math library, $a(x)$ is for the arctangent of x, and the returned arctangent is in radians. Since the arctangent of 1 is $\pi/4$ radians, the expression $4 * a(1)$ gives the value of π. Therefore, the value of π can be printed in a shell script by passing the expression $4 * a(1)$ as standard input to `bc`.
- `for i in {0..5}; do num=$RANDOM; ./`$monte\_carlo -n$ `20000 -r $`$num|tail -n+$`2 | awk ’ print $1, (’$pi’-$2) ’ > mc_err_$i.dat; done`
  - I use a for loop from 0 to 5 to implement six estimation results using six different random seeds. In each implementation, a random seed is set as *num* and passed to the seed option of `monte_carlo` program as a deterministic random starting point. The option of the number of points for estimation is set to 20000 to get enough iteration to show the change of the errors. The *tail* command with *-n +2* output starts with line 2, meaning trimming the header line. The output is piped to *awk*. The *awk* command goes through each line of the previous output, and prints the first column and the calculation of error (the defined π minus the estimated π in the second column). The final output is redirected to a new file named by each seed implementation.
- `gnuplot <<END`
  - Sent the following here-document to `gnuplot`.
- `set terminal pdf`
  - To save the plots as pdf format.
- `set output ‘‘mc_error.pdf’’`
  - Set output name.
- `set yrange [-1:1]`
  - Set the limit of *y* axis to be in the range [-1:1].
- `set xrange [1:20000]`
  - Set the limit of *x* axis to be in the range [1:20000] because the `monte_carlo` iteration number is 20000.
- `set logscale x 2`

  - This command set the x axis to logscale base 2 as the original sample plot, making it easier to see the error change.

- `set title ‘‘Monte Carlo Error Estimation’’`
  - Set the plot title.
- `set ylabel ‘‘Error’’`
  - Set the label of y axis to explain the plot.
- `set grid xtics ytics`
  - Set the gridlines so that it's easier to tell the scale of error changes.
- `plot ‘‘mc_err_0.dat’’ with lines title ‘‘’’ lt rgb ‘‘FF6666’’,`
  - To plot the output of six files as lines with different colors and no titles.

5 Conclusions

In this assignment, I learned how to write a bash script and use `gnuplot` to produce sufficient plots to demonstrate mathematical concepts. I also gained more insight into the Monte Carlo method and the estimation of π. Some commands definitely became handy to me, including `awk`, `head`, `tail`, `for` loop, and file redirection. I also practiced explaining my thoughts and logic more thoroughly by writing and editing my design and writeup documents.