WRITEUP

Wenyu Liang

January 23, 2023

1 UNIX commands description

```
iter = ${1:-800}
```

Makes the bash script accept positional argument. The first argument provided will be the number of data points for Figure1. This is also an optional argument, 800 data points will be generated if no arguments are provided.

```
./monte_carlo -n $iter > f1.dat
```

This command generates data point for Figure1

```
echo "0 0 1" > circle.dat
```

Generate the data for plotting the circle

```
awk '{if (( $5==0 )) print $0}' f1.dat > f1.dat0
awk '{if (( $5==1 )) print $0}' f1.dat > f1.dat1
```

Separates the data into f1.dat0 and f1.dat1 based on whether the data are in the circle. $0 means the entire row

```
gnuplot <<END
set terminal pdf
set output "Figure1.pdf"
set size square
plot [0:1][0:1] "f1.dat0" using 3:4 pt 7 ps 0.3 lc "red"
notitle, "f1.dat1" using 3:4 pt 7 ps 0.3 lc "blue" notitle,
"circle.dat" using 1:2:3 with circle lc "black" notitle
END
rm f1.dat* circle.dat
```

Use gnuplot to visualize the data. We make the figure a square PDF. [0:1][0:1] denotes the xrange and yrange. 3:4 means the 3rd and 4th columns, they are x and y coordinates. pt 7 and ps 0.3 set the line type and size. 2 sets of data points will be in red and blue. The circle will be plotted also. At the end, data will be deleted, remaining the Figure1.pdf

```
echo $(seq 65536) | tr ' ' '\n' > iter.dat
```

Generates 65536 data points with seq for figure2 and use tr to replace white space with newline to make the data a column instead of a row.

```
pi=$(echo "scale=5; 4*a(1)" | bc -l)
```

Calculates the actual π value, learned from https://stackoverflow.com/questions/23524661/how-can-i-calculate-pi-using-bash-command. scale is set depending on how accurate we want. bc used to do calculation.

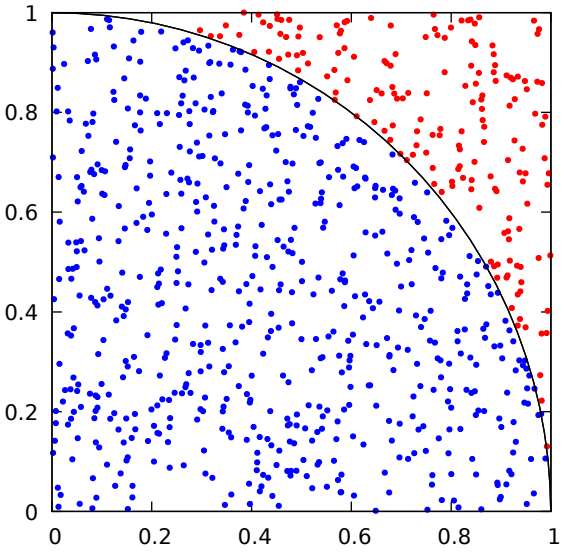
```
for i in $(seq 4)
    do
    ./monte_carlo -r $i -n 65536 | awk -v PI=$pi
    '{if (($2 != "Pi")) print $2-PI}' > temp.dat${i}
    done
paste iter.dat temp.dat* > err.dat
```

For 4 different seeds, we run monte_carlo to generate data. Use awk to get the difference between the estimated π and actual π, which will be write into a temporary data. When loop is finished, I use paste to combine these data into one file.

```
gnuplot <<END
set terminal pdf
set output "Figure2.pdf"
set title "Monte Carlo Error Estimation"
set grid xtics
set grid ytics
set ylabel "Error"
set xrange [1:65536]
set yrange [-1:1]
set ytics (-1,-0.5,0,0.5,1)
set logscale x 4
set xtics (1,4,16,64,256,1024,4096,16384)
plot "err.dat" using 1:2 with lines notitle, "" using
1:3 with lines notitle, "" using 1:4 with lines notitle,
"" using 1:5 with lines notitle
END
rm iter.dat temp.dat* err.dat
```

This figure will be x ranged from 1 to 65536 and y ranged from -1 to 1. Since the total number will be very large, x will be scaled by log_4. I also add tics to make the figure more readable. The first column will be the iteration number, and the rest will be estimated error from 4 seeds. Eventually, the temporary data will be deleted, Figure2.pdf remaining.

2 Output



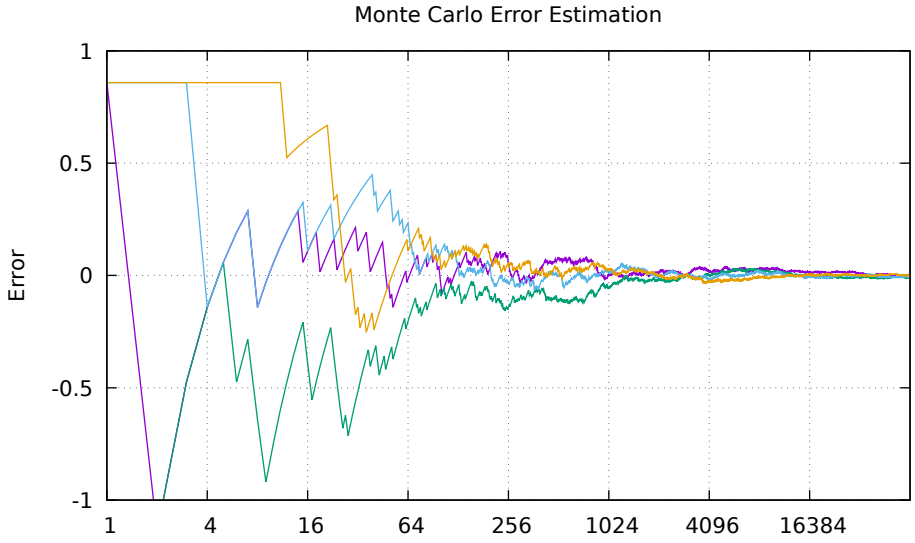


Figure1 has blue points in the circle and read points outside the circle. Figure2 have 4 lines in different color which represent 4 independent simulations. We can see that as the the number of data points become bigger, the error are closer to 0. When the number approaches 16384, 4 lines show the estimation has been very accurate.