Assignment 1 Writeup

Alex Yeh

January 19, 2023

1 Figure 2

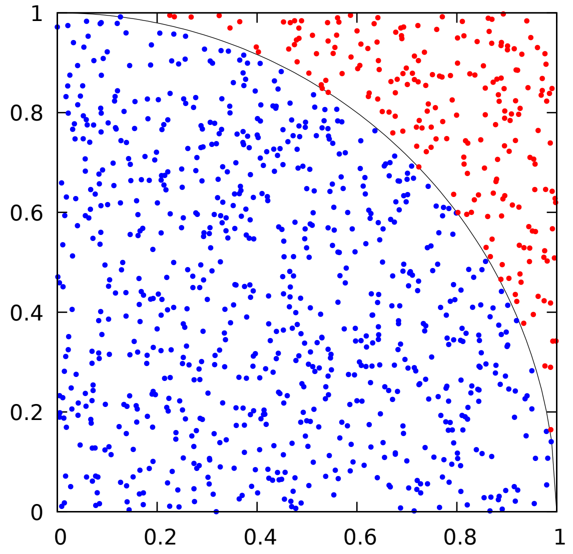
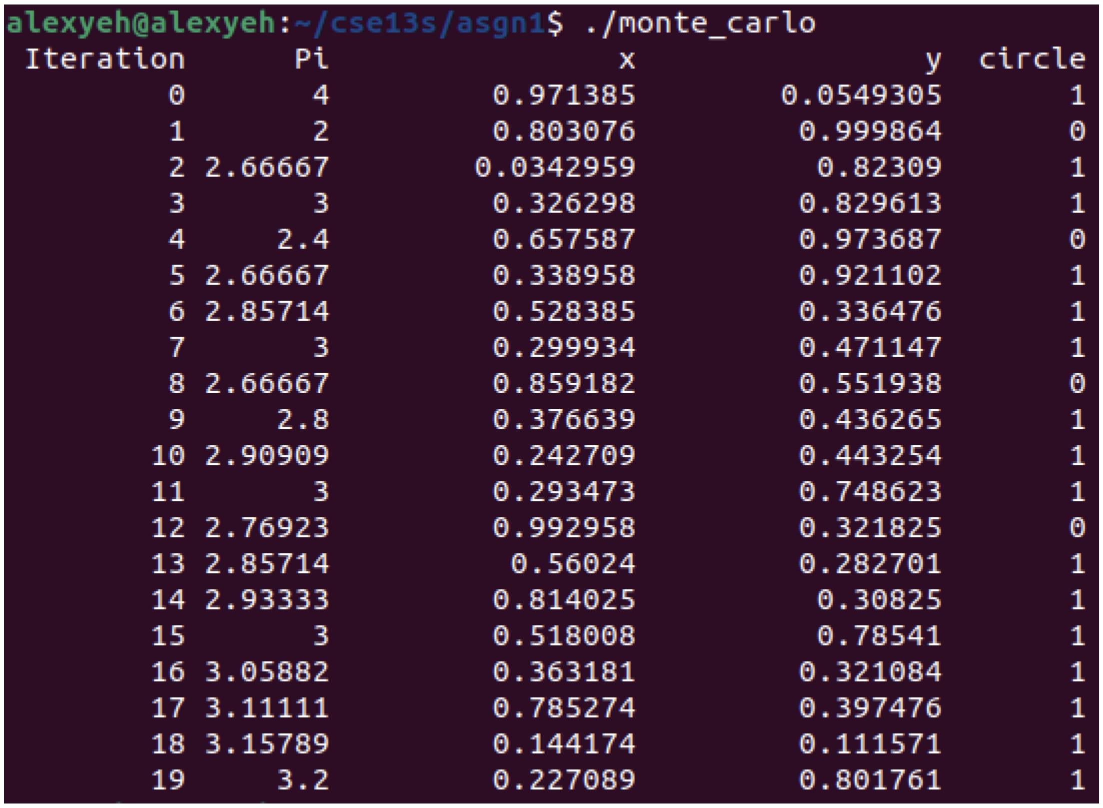


Figure 2 shows the points of Monte Carlo estimations. The blue points have distance less than or equal to 1 which is why they are in the square of the plot as well as the circle. The red points are in the square but not the circle. In order to get these points from the C program, monte_carlo.c, my bash script needed to get the third, fourth, and fifth column of data. The picture below shows the default result of running monte_carlo with no command options.



```
alexyeh@alexyeh:~/cse13s/asgn1$ ./monte_carlo
 Iteration       Pi                 x                y  circle
         0        4          0.971385        0.0549305       1
         1        2          0.803076         0.999864       0
         2  2.66667         0.0342959          0.82309       1
         3        3          0.326298         0.829613       1
         4      2.4          0.657587         0.973687       0
         5  2.66667          0.338958         0.921102       1
         6  2.85714          0.528385         0.336476       1
         7        3          0.299934         0.471147       1
         8  2.66667          0.859182         0.551938       0
         9      2.8          0.376639         0.436265       1
        10  2.90909          0.242709         0.443254       1
        11        3          0.293473         0.748623       1
        12  2.76923          0.992958         0.321825       0
        13  2.85714           0.56024         0.282701       1
        14  2.93333          0.814025          0.30825       1
        15        3          0.518008          0.78541       1
        16  3.05882          0.363181         0.321084       1
        17  3.11111          0.785274         0.397476       1
        18  3.15789          0.144174         0.111571       1
        19      3.2          0.227089         0.801761       1
```

As you can see, the x coordinate is the data in the third column, the y coordinate is the data in the fourth column, and the fifth columnn tells us if the point is in or out of circle by stating a 0 or 1. A 0 means the point is out of the circle and a 1 means the point is inside the circle. Because there are two colors of points, I needed to put each separate point in its own data file.

Since I needed the data from the third, fourth, and fifth column, I used the UNIX command “awk” to do so. “awk” allowed me to print only the data from the third, fourth, and fifth column. I did this two separate times with an if statement to check if the fifth column had a 0 or 1. If the fifth column had a 1, the data from the third and fourth column were transferred into a data file called “in_circle.dat” which contained all of the points inside the circle. Vice versa, If the fifth column had a 0, the data from the third and fourth column were transferred into a data file called “out_circle.dat” which contained all of the points outside the circle. I then was able to plot both data files on the same graph using gnuplot.

2 Figure 3

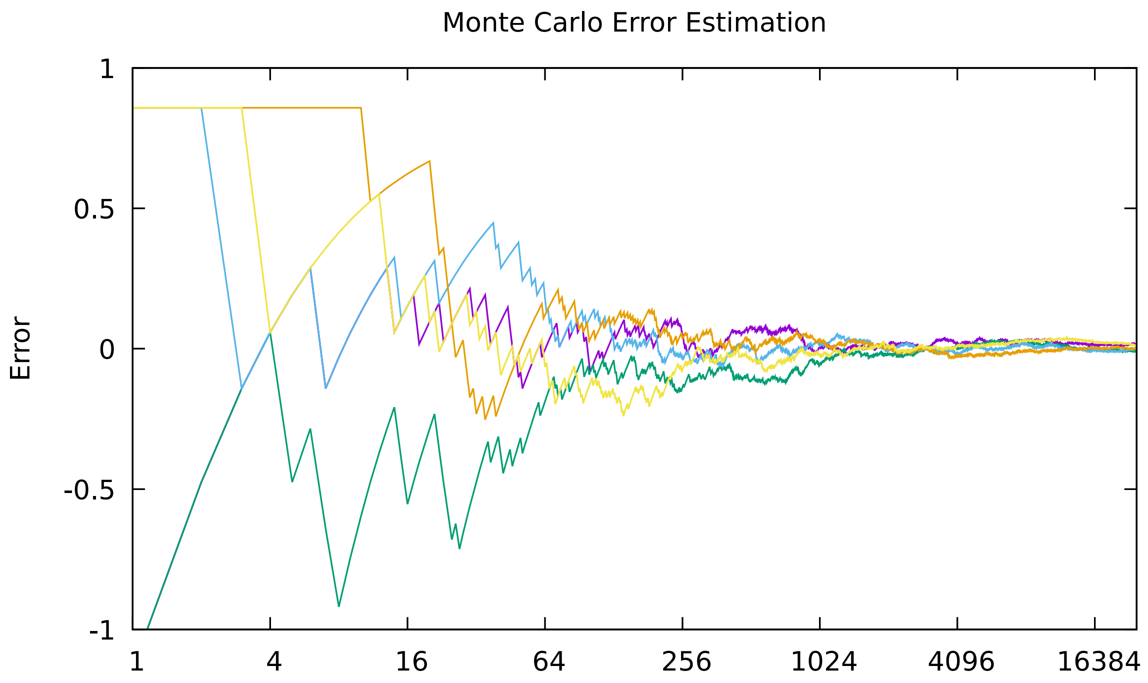
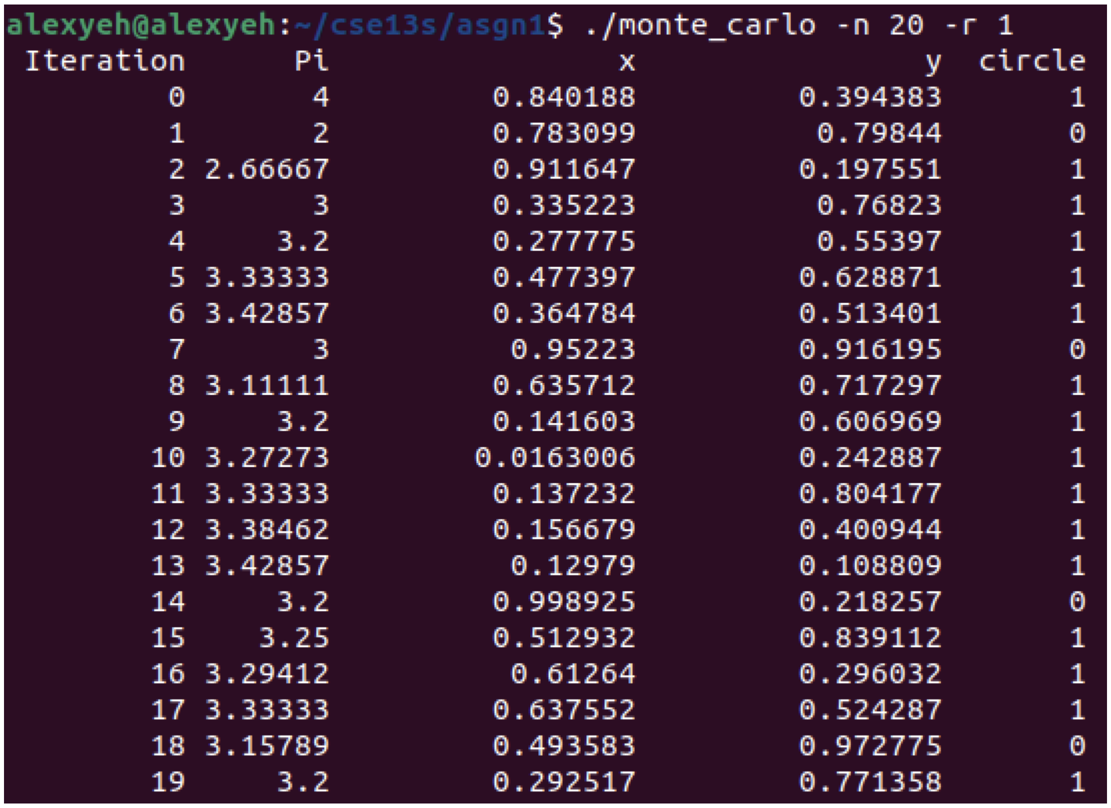


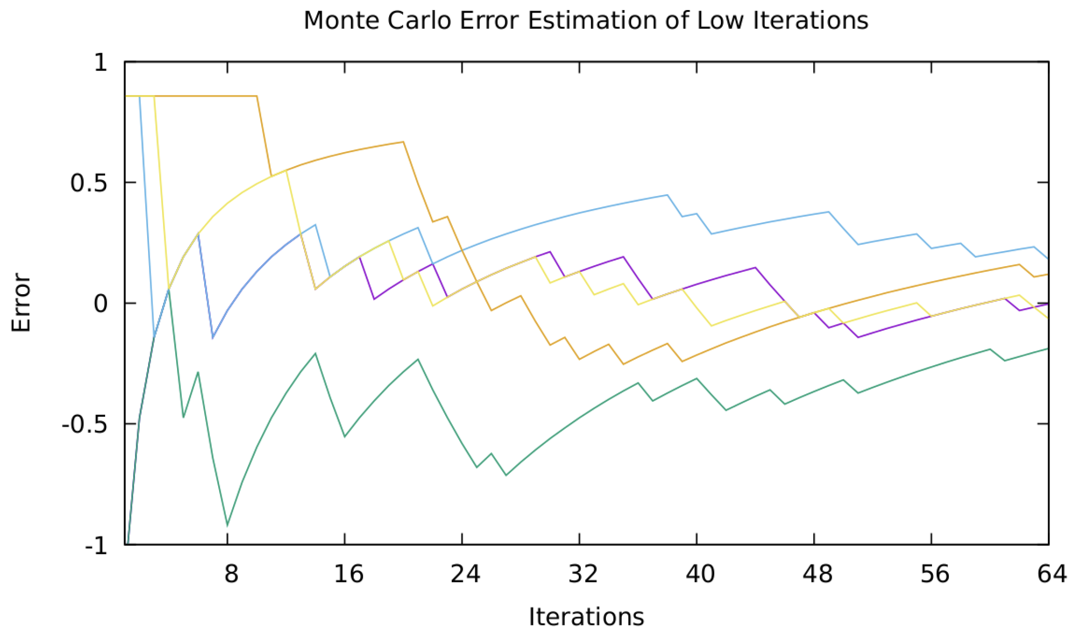
Figure 3 shows the Monte Carlo error estimations for different seeds. As you can see, the value of the difference between the estimated pi and actual pi gets closer to zero as we increase iterations. The different colors represent different seeds for the random number generator.



```
alexyeh@alexyeh:~/cse13s/asgn1$ ./monte_carlo -n 20 -r 1
 Iteration       Pi                 x                 y  circle
         0        4          0.840188          0.394383       1
         1        2          0.783099           0.79844       0
         2  2.66667          0.911647          0.197551       1
         3        3          0.335223           0.76823       1
         4      3.2          0.277775           0.55397       1
         5  3.33333          0.477397          0.628871       1
         6  3.42857          0.364784          0.513401       1
         7        3           0.95223          0.916195       0
         8  3.11111          0.635712          0.717297       1
         9      3.2          0.141603          0.606969       1
        10  3.27273         0.0163006          0.242887       1
        11  3.33333          0.137232          0.804177       1
        12  3.38462          0.156679          0.400944       1
        13  3.42857           0.12979          0.108809       1
        14      3.2          0.998925          0.218257       0
        15     3.25          0.512932          0.839112       1
        16  3.29412           0.61264          0.296032       1
        17  3.33333          0.637552          0.524287       1
        18  3.15789          0.493583          0.972775       0
        19      3.2          0.292517          0.771358       1
```

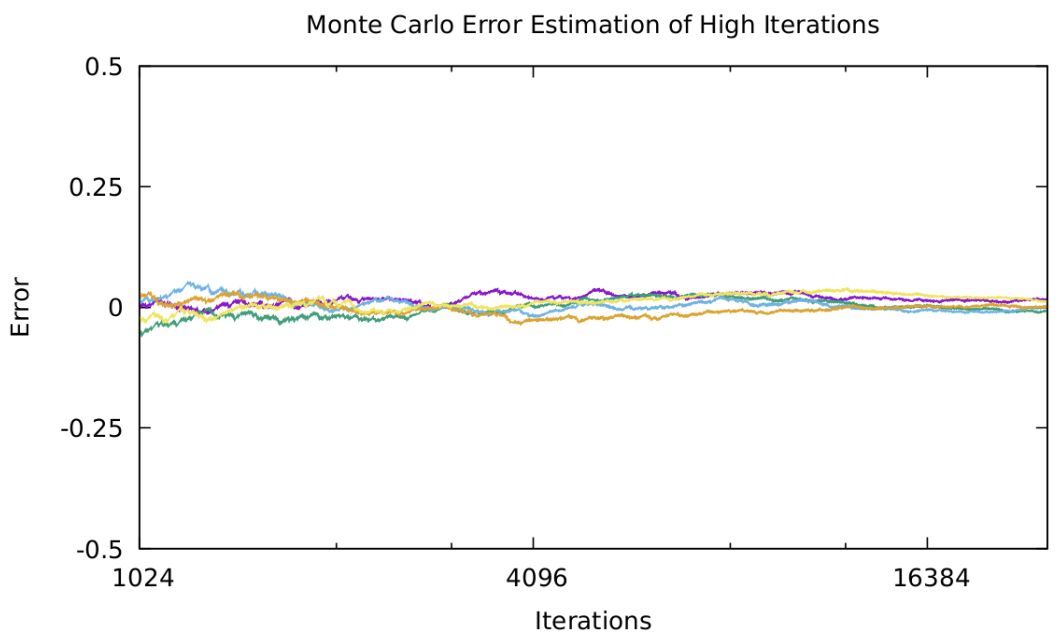
The picture above shows the data from monte_carlo.c with 20 iterations and a random seed of 1. For my bash script to plot figure 3, I needed the data from the first and second column. The first column is the iteration number and the second column is the estimation of pi. Like figure 2, I used the UNIX command “awk” to get the data from columns 1 and 2. Before I used “awk”, however, I used another Unix command called “tail”. Since the first row of data is the header, I used “tail” to only get the data values and not the header because the header is not a numerical value. When using “awk” I printed the first column and the second column - pi, which I defined earlier in my bash script, and put the data into separate data files using a for loop. The for loop allowed me to get data from 5 different seeds which allowed me to plot 5 different seeds using gnuplot.

3 First Extra Plot of Low Iterations



For my first extra plot, I decided to plot a small amount of low iterations to show that the error estimation starts off far from 0. The x axis shows the number of iterations and the y axis shows the erorr estimation. Like figure 3, I used “awk” to print the values to a data file and “tail” to only get the numerical values. I then was able to plot the values in the data file using gnuplot. As we can see from the plot, a low number of iterations results in a error estimation far from 0 but starts getting closer as the number of iterations increases.

4 Second Extra Plot of High Iterations



For my second extra plot, I decided to plot a large number of high iterations to show that the error estimation becomes very close to 0 as the amount of iterations increases. The x axis shows the number of iterations and the y axis shows the error estimation. Like figure 3 and my first extra plot, I used “awk” to print the values to a data file and “tail” to only get the numerical values. I then was able to plot the values in the data file using gnuplot. As we can see from the plot, a high number of iterations results in a error estimation very close to 0.