Assignment One: Pi Estimation

Rebecca Corley[†]

[1]University of North Georgia, Department of Physics and Astronomy

ABSTRACT

Monte Carlo simulations consist of a broad class of computational algorithms that use random sampling to calculate numerical results. In this project, we use Monte Carlo methods to estimate the value of Pi. Two experiments are generated considering a game of darts. In Experiment 1, we observe how the value of pi becomes more accurate as more darts are thrown at a board. In Experiment 2, we set a number of darts to be thrown in each game, then repeat that game 100000 times. Plots are generated for both experiments. Average Pi values and standard deviations are obtained and compared.

1. INTRODUCTION

Consider a game of darts. The darts are thrown at a square dart board. All darts thrown at the board are guaranteed to hit the board in some random (x,y) position, as seen in Fig. 1. On the board is a circle of radius r. To estimate the value of Pi, we first look at the ratio of darts that land in the area of the circle to total number of darts thrown, which will land within the area of the square. Mathematically this is shown as:

$$\frac{Area_{circle}}{Area_{square}} = \frac{\pi r^2}{(2r)^2} \tag{1}$$

Which then reduces to

$$\frac{Area_{circle}}{Area_{square}} = \frac{\pi}{4} \tag{2}$$

Therefore, the number of darts that land in the circle to the total number of darts thrown can be used to estimate Pi.

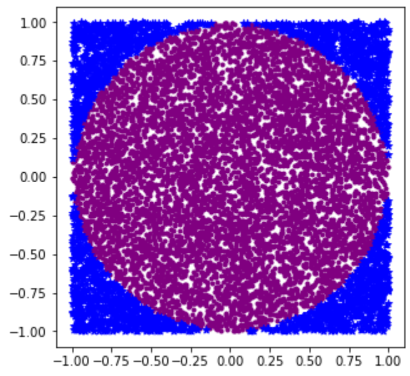
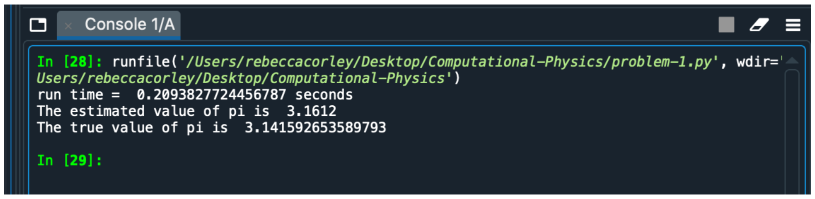


Figure 1. Plot of the dart board with throws generated by Monte Carlo methods. The purple dots indicate darts thrown at the board that land within a circle of radius 1 and the blue dots are darts that landed within the board, but outside the radius.

†: E-mail: rlcorl0436@ung.edu



```
In [28]: runfile('/Users/rebeccacorley/Desktop/Computational-Physics/problem-1.py', wdir='
Users/rebeccacorley/Desktop/Computational-Physics')
run time =  0.2093827724456787 seconds
The estimated value of pi is  3.1612
The true value of pi is  3.141592653589793

In [29]:
```

Figure 2. Output code estimating Pi and the true value of Pi from throwing darts at a dart board of radius 1.0.

1.1 Experiment 1

Here we observe how the value of Pi changes as the number of darts thrown at the dart board increases.
Fig. 3 shows plots for 100 darts thrown.
Fig. 4 shows plots for 1000 darts thrown.
Fig. 5 shows plots for 10000 darts thrown.

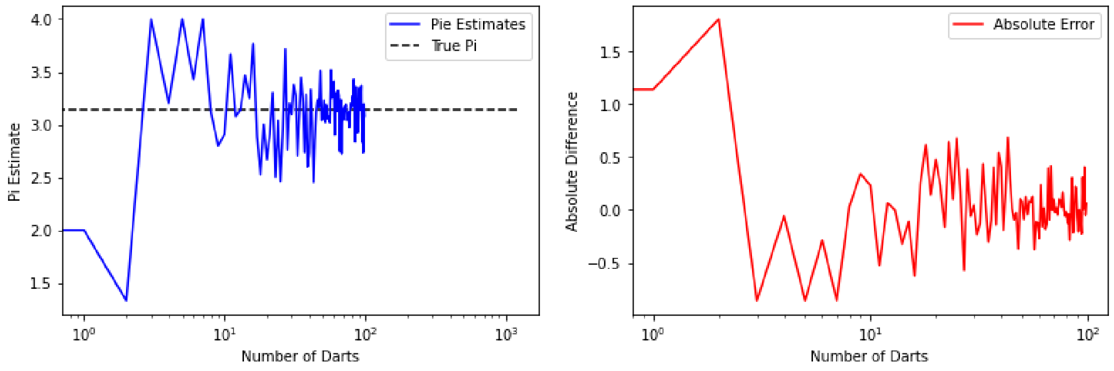


Figure 3. Left: Estimated value of Pi plotted as a function of the number of darts thrown (on a log scale) for 100 darts. The blue line indicates the estimated value of Pi and the black line represents the true value of Pi. Right: The difference to the true value of Pi as a function of number of darts thrown. Absolute error is given by the red line.

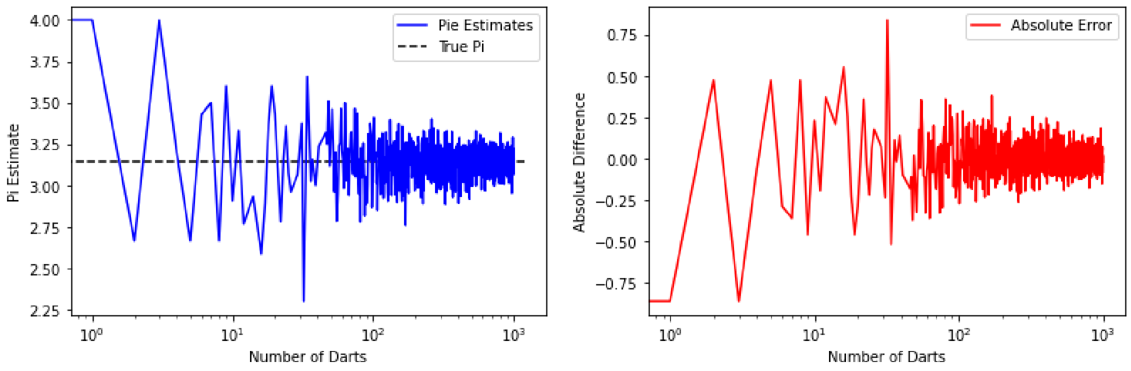


Figure 4. Left: Estimated value of Pi plotted as a function of the number of darts thrown (on a log scale) for 1000 darts. The blue line indicates the estimated value of Pi and the black line represents the true value of Pi. Right: The difference to the true value of Pi as a function of number of darts thrown. Absolute error is given by the red line.

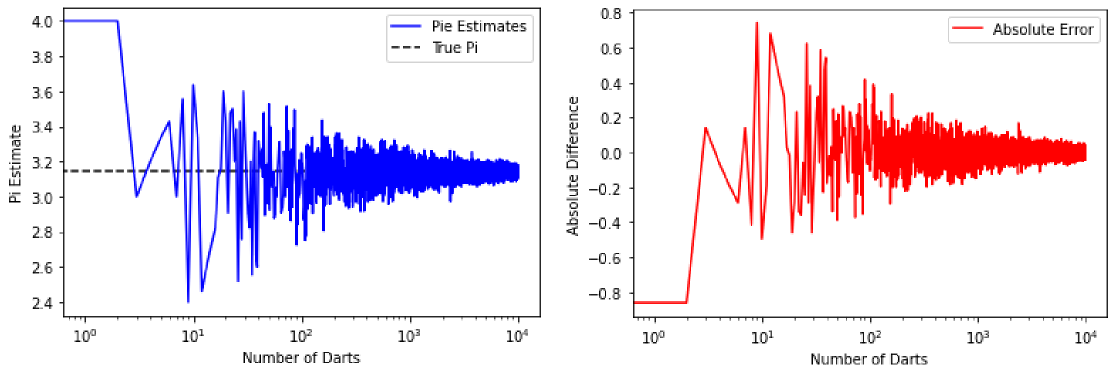


Figure 5. Left: Estimated value of Pi plotted as a function of the number of darts thrown (on a log scale) for 10000 darts. The blue line indicates the estimated value of Pi and the black line represents the true value of Pi. Right: The difference to the true value of Pi as a function of number of darts thrown. Absolute error is given by the red line.

1.1.1 Experiment 1: Methods/Code

The code for Experiment 1 is shown in Fig. 6. The structure of the code is as follows: First import required libraries and define any variables and define empty arrays to later be used in the code.
The for loop is the meat and potatoes of this code. It will loop through the number of iterations defined in "n_iterations". Within the for loop is a while loop, which will continue to execute so long as a condition is true. Here, the dart count is iterating through, increasing by one, creating random values of (x,y) pairs. The if statement says that for every one of those pairs generated, if the sum of $x^2 + y^2$ is less than or equal to the radius 1.0, then that counts as a dart hit inside the circle. The last two lines within the for loop append, or adds the values in the statements to previously defined empty arrays.
The following lines calculate the average and standard deviations of the lists of pies, then plots are created of the estimated values and the absolute difference of Pi.
The output of the code for Experiment 1 can be seen in Fig. 9

1.1.2 Experiment 1: Conclusions

Figures 3, 4, and 5 show plots for 100, 1000, and 10000 darts thrown, respectively. Looking at the left plots, the blue lines represent the estimated values of Pi as a function of the number of darts thrown. The black dotted line plots the true value of Pi. As the dart throws increase, the estimate of Pi becomes more accurate, which makes the blue line narrow in closer to the true value. The red lines in the right plots are the absolute errors in each trial. The absolute error plots are calculated by taking the absolute value difference between the actual Pi value and the estimated Pi value. This value is calculated for each Pi estimate in the list. The absolute error plot appears as a flipped version of the Pi estimate (blue) plot.
For 100000 darts thrown, Experiment 1 was incredibly computationally expensive. After about an hour of run-time, the code was killed. However, for an order of magnitude less, 10000 darts thrown, the average of the Pies was: 3.141435095439712 and the standard deviation was: 0.051471179213560016 (Fig. 9).

1.2 Experiment 2

Here the number of dart throws is set to 10000 and the experiment is repeated 100,000 times. We find the average value of Pi and the standard deviation of each of the experiment trials.

1.2.1 Experiment 2: Methods/Code

For Experiment 2, the code was structured slightly different than Experiment 1.
First the necessary libraries were imported. Variables and empty arrays were declared and defined.
The most important part of this code was the nested for loop. The outer loop iterates through the number of defined experiments. The inner for loop iterates through the number of darts thrown in each experiment. The inner for loop is set up similar to Experiment 1, in that it randomly generates (x,y) values to then send to a

previously defined empty array such that the values pass the criteria requirements.
The average and standard deviation are then calculated to be output. Lastly, a histogram is generated.
The out put code for Experiment 2 can be seen in Fig. **??**

1.2.2 Experiment 2: Conclusions

In Fig. 10 a distribution of 10000 darts thrown with that experiment repeated 100000 times is represented. The vertical black like is the true value of Pi. The distribution is mostly Gaussian, having most Pi estimates closer to the true value of Pi. I find it quite interesting that although the number of darts thrown and the number of iterations through the experiment is high, looking at the histogram, the experimental estimated value of Pi is not exactly in line withe the true value of Pi.
The average value of the Pies was 3.141550044 and the standard deviation was 0.01645168260624638 (Fig. **??**).

1.3 General Conclusions

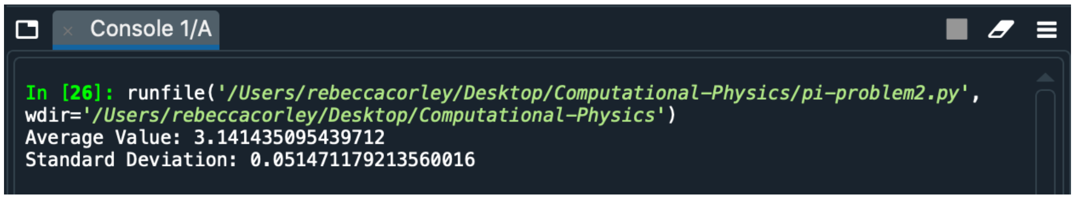
I found that the code for Experiment 1 took significantly longer to run for the same number of iterations. In fact, the simulation for Experiment 1 took so long to run for 100000 darts thrown (over an hour), I was unable to obtain the average Pi and standard deviation. I am wondering if the duration of the run-time is due to the structure of my code. After consulting with my group members, it seems that their codes are more efficient. Perhaps some further optimizations to my code could be made in the future.
The average and standard deviation from Experiment 1 (for 10000 darts) were 3.141435095439712 and 0.051471179213560016, respectively. The average and standard deviation from Experiment 2 were 3.141550044 and 0.01645168260624638, respectively. The true value of Pi is 3.141592653589793. Although the two experiments were run with an order of magnitude difference, the average value of Pi and the standard deviation are not much different. Experiment 2, with the higher number of darts, is only accurate to one more decimal place than Experiment 1. To obtain the most accurate Pi, much more significant statistics must be obtained.

1.4 Acknowledgments

I would like to thank my group members for support through this project.

```
#Computational Physics: 23-08-2021 Problem 1
#python libraries used for this code
import numpy as np
import matplotlib.pyplot as plt
n_iterations = 10000 #declare number of iterations
pies_list = [] #define empty array for pies
diff_list = [] #define empty array for the absolute error
#for loop to loop through defined number of iterations
for i in range(n_iterations):
    darts = 0 #initalize dart throws to zero
    circle = 0 #initalize darts inside circle to zero
    square = 0 #initalize darts inisde square to zero
   #while loop (executes a set of statements as long as a condition is true)
    while darts <= i:
        x = np.random.uniform() #random number generator
        y = np.random.uniform() #random number generator
        #if statement adding darts to circle list if condition is met (within radius of circle)
        if (x*x + y*y) <= 1.0:
            circle += 1.0
        darts += 1 #adds number to dart count
    pies_list.append(4*(float(circle)/float(darts))) #calculates pi value and adds
to pies_list array
   #abs error is difference between exp. value and true value
    diff_list.append(np.pi- abs(4*(float(circle)/float(darts)))) #calculates
the absolute error and adds to diff_list array
#plot commands for pie estimates
plt.plot(np.arange(0, n_iterations, 1), pies_list, color="blue",
label="Pie Estimates")
plt.hlines(y=np.pi, xmin=-1.0, xmax=1200.0, linestyles="--",
color="black", label="True Pi")
plt.legend()
plt.xscale("log")
plt.xlabel("Number of Darts")
plt.ylabel("Pi Estimate")
plt.show()
#plot commands for absolute difference of pi
plt.plot(np.arange(0, n_iterations, 1), diff_list, color="red",
label="Absolute Error")
plt.legend()
plt.xscale("log")
plt.xlabel("Number of Darts")
plt.ylabel("Absolute Difference")
plt.show()
```

Figure 6. Code used for Experiment 1. This code implements a Monte Carlo random number generator to observe how the estimate of Pi changes as the number of darts thrown is adjusted. The code calculates and plots the average value of Pi and standard deviation of all darts thrown.

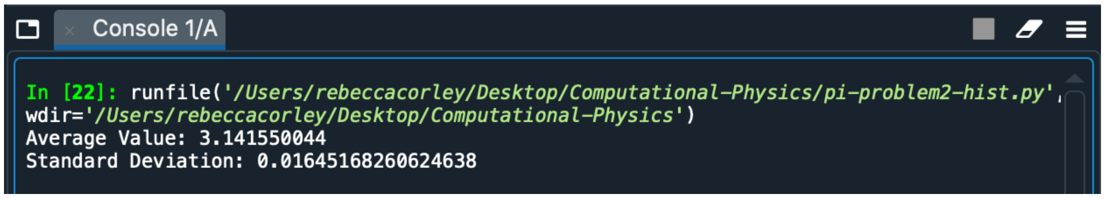


```
In [26]: runfile('/Users/rebeccacorley/Desktop/Computational-Physics/pi-problem2.py',
wdir='/Users/rebeccacorley/Desktop/Computational-Physics')
Average Value: 3.141435095439712
Standard Deviation: 0.051471179213560016
```

Figure 7. Output code estimating the averages of Pi and the standard deviation of Pi for 10000 dart throws in Experiment 1.

```python
#python libraries used for this code
import numpy as np
import matplotlib.pyplot as plt
import random
import statistics
n_iterations = 100000 #define number of points in histogram
n_darts = 10000 #define number of darts thrown
circle = 0 #initialize circle counts
square = 0 #initialize square counts
hist_data = [] #initalize empty array for histogram

#big loop
for i in range(n_iterations):

    circle = 0
    square = 0
    #random generator loop
    for j in range(n_darts):

        x = random.random() #normalized from (0,1) as floats
        y = random.random()

        if (((x*x) + (y*y)) <= (1.0)):
            circle += 1.0
            square += 1.0
        else:
            square += 1.0

    hist_data.append(4*(float(circle)/float(square)))

average = statistics.mean(hist_data)
stdev = statistics.stdev(hist_data)

plt.hist(hist_data, 50)
plt.vlines(np.pi, ymin=0, ymax=8000, linestyles="--", label="True pi",
color="black")
plt.legend()
plt.ylabel('Probability')
plt.xlabel('Pi Extimate')
plt.show()

print("Average Value: " +str(average))
print("Standard Deviation: " +str(stdev))
```

Figure 8. Code for Experiment 2. This code implements a Monte Carlo random number generator to calculate the average number and standard deviation of Pi for 100000 games of darts for 10000 darts thrown in each game. The data is then plotted in a histogram.



```
In [22]: runfile('/Users/rebeccacorley/Desktop/Computational-Physics/pi-problem2-hist.py',
wdir='/Users/rebeccacorley/Desktop/Computational-Physics')
Average Value: 3.141550044
Standard Deviation: 0.01645168260624638
```

Figure 9. Output code estimating the averages of Pi and the standard deviation of Pi for 10000 dart throws in 100000 games for Experiment 2.

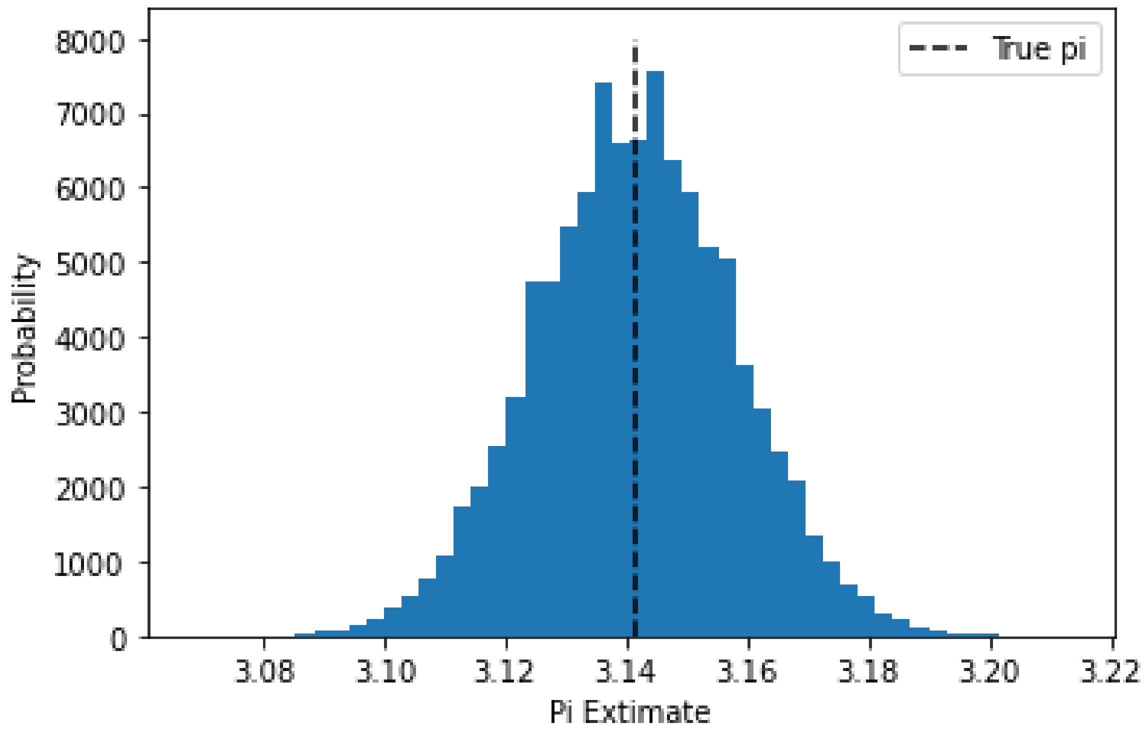


Figure 10. Histogram representing 10000 darts thrown and 100000 experiments conducted. The black like is the true value of Pi.