Introduction

The program monte.f is a Fortran77 program that estimates the value of pi using the Monte Carlo method. Monte.f uses a multiplicative linear congruential generator to generate sets of random coordinates. Pi is then estimated by counting the proportion of random points that land inside a circle that is inscribed in square.

Method

1. *Random Number Generation.*

Monte.f uses a multiplicative linear congruential generator (MLCG) to generate 10 independent samples of 100,000 random x and y coordinates between 0 and 1. The MLCG follows the form:

$$x_{n+1} = a * x_n \, mod \, m,$$

Where the previous random number is used to generate the next random number. For this experiment, $a = 16807$ and $m = 2^{30} - 35$. To determine the seeds, or the first random number, x_0 and y_0, I used the intrinsic Fortran77 function rand(). While the quality of the rand() function is poor for generating random samples for Monte Carlo problems, it works well for generating seeds. For each independent sample, the same a and m are used, but the different random seeds for x and y are used to ensure that each sample is unique and is independent. I used a do loop to generate the rest of the random points needed.

2. *Estimating Pi*

I used the Monte Carlo method to estimate pi by generating random points inside a square centered at the origin. There is a circle of radius r =1 inscribed in the square and centered at the origin. Using the ratio between the area of a circle and the area of a circle, pi can be estimated by multiplying the number of points in the circle, c, by 4 and then dividing the product by number of points in the square, N, using the formula : $\pi \approx \frac{4c}{N}$. If a random point (x,y) fulfills the condition: $\sqrt{x^2 + y^2} = r < 1$, the point is considered inside the circle.

Results

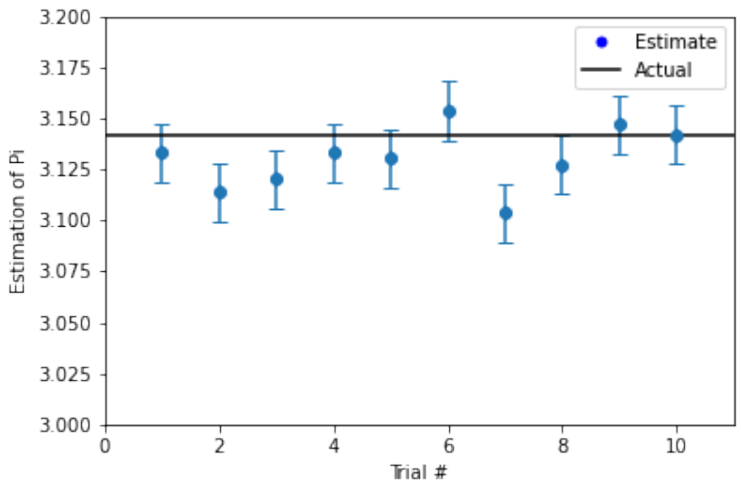

Fig 1. Comparison of estimates of pi using the Monte Carlo method to the actual value of pi using error bars.

The average value of the 10 sample means was 3.130344 with a standard deviation of .014535. Comparing the Monte Carlo estimate of 3.130344 to the actual value, 3.141592, the percent error is .358%.

Trial #	x-mean	y-mean
1	.50053	.500454
2	.50309	.502903
3	.50058	.503156
4	.50045	.500444
5	.50030	.500715
6	.50038	.500498
7	.50067	.503714
8	.50317	.500495
9	.50044	.500576
10	.50072	.500700

Table 1. Table of the mean x and y values generated by the MCLG for each sample. A mean value as close to .5 is desired.

Discussion

10 samples of 100,000 points were used to estimate pi. 10 samples of 100,000 points were used because it was sufficient enough to estimate pi within .5%. The MCLG was used because it is extremely fast, provided sufficient quality random number generation, and has a long period around 10^9. Table 1 shows the mean x and y values generated by the MCLG. The mean x and y values are close to .5 which means there is a uniform distribution of generated

numbers between 0 and 1. However, all the means are slightly above .5 when at least some of the mean x and y values should be slightly below .5 as well. This shows that the MLCG has some systematic bias and generates random numbers skewed towards 1. I set $a = 16807$ because it is a common parameter and it is a large prime number. I set $m = 2^{30} - 35$ because it is a very large prime number and allows for a large period. For MLCG's, the period is typically around m and so repeating random number cycles is not an issue.

The Monte Carlo estimation of 3.130344 is fairly accurate with a percent error of .358% and is precise with a standard deviation of .01435. The error bars of the sample means in Figure 1 have overlap and that along with a low standard deviation show the Monte Carlo estimation is precise.

Conclusions

The Monte Carlo estimate for pi was 3.130344 with a standard deviation of .01435. While the Monte Carlo estimation is both fairly accurate and precise, steps could be taken to more accurately estimate pi. Using more samples and increasing sample size would increase accuracy and precision, but also would be slower. Improvements to the MCLG such as using a higher m, combining multiple MCLGs, better generation of seeds would lead to better random number generation and therefore better estimation of pi.