**Estimating Pi – Nafih Mohammed**

Background

The primary objective of this project was to find a method that would estimate the value of pi, and then compare this to other methods, finding the most computationally efficient method with the lowest error.

Algorithms

A graph with a red line and a blue line

Description automatically generated The first algorithm I used was the Monte Carlo method, which uses random sampling to approximate results. When estimating pi, this method randomly generates points within a known area and determines the fraction of random points that fall within the circular region. Different total numbers of samples were trialled, which can be seen in the graph below:

The next algorithm that was used was a root finding method. This was a method where a function is defined to have a root at the value to be estimated. Here, the function chosen was f(x) = sin(x), which has a root at pi. An interval of 3 – 4 was chosen as there exists a sign change in this range, which would halve each time a new sign change was detected. This function iterated until the estimated reached a preset tolerance of 1e-10.

The final algorithm used the Leibniz rule, which estimates using an infinite series of arctan, where the final value needed to be multiplied by 4 to estimate pi. It used a pre-defined number of terms, which affected its accuracy.

Findings:

The fastest of the three algorithm was the root finding algorithms, which estimated pi as 3.1416, with an error of 5.3081e-11, and took 0.0024363 seconds to compute. Next was the Leibniz algorithm, which estimated pi as 3.1416, with an error of 1e-05, and took 0.0028005 seconds to compute. Finally, the Monte Carlo algorithm with 1000000 points estimated pi as 3.1393, with an error of 9.1329e-06 and took 0.048438. Although the Monte Carlo and Leibniz algorithms are easy to implement and may suffice for simple estimations, for high precision they need of lot of trials to converge and are very inefficient Although the root finding algorithm requires a good initial guess, it is highly efficient and estimates to a very high precision, which is why I would recommend this algorithm in this instance.