Project 1: Monte Carlo Pi Calculation

Owen O'Reilly | 9/15/2024 | AMS-595

Summary

Set of scripts that use the Monte Carlo circle-in-square method to calculate pi to varying degrees of precision and accuracy. Script 1 provides pre-written tests of the efficiency and accuracy of the method; Script 2 provides a function that will generate a value for pi to a user-specified significant figure.

Script 1: MonteCarlo.m

Upon running, script will:

1) Use a For loop to calculate values of pi based on the Monte Carlo method described in the assignment.
    - Number of points sampled ranging from 10^1 to 10^8.
2) Produce a combined figure showing the progression of calculated pi values, deviation from the true value, and program runtime (efficiency) as a function of the growing sample size.
3) Use a While loop to generate pi values of increasing precision as measured by the number of significant digits using the same Monte Carlo method.
    - Specified precision is assumed to be achieved at the thirtieth repetition of the same value at the specified significant digit.

See Github for a PDF sample of the program's output.

Script 2: getMonteCarloPiVal.m

Provides a function that will calculate the value of pi to a specified precision, passed in as a parameter of the method.

1) User calls the function and provides a desired number of significant figures, as in getMonteCarloPiVal(*n*).
2) The program will begin generating random points and calculating a value of pi via the Monte Carlo method.

- Points generated will be plotted to a live-updating figure that color codes them based on whether they fall within the circle.
- A value for pi and running tally of points will be annotated on the figure, and will also update continuously.
- Precision of value will be determined in the same manner as in Script 1, with thirty consecutive repetitions at the significant digit.

See Github for PDF samples of the program's output. Second graph is the actual output.

Discussion

Accuracy of the calculated value showed very clear growth with the number of points used in the calculation. This can be seen in the graphs generated by the For loop, where the computed values of pi very quickly match up with the known value as the magnitude of points sampled increases.

Similarly, the number of points required to gain a significant digit of precision seems to increase tenfold in the trials of the while loop. One digit of precision takes tens of points, whereas four tends to take thousands. Interestingly, increased precision does not appear to guarantee accuracy; on many occasions the 3-sigfig product of the while loop is more accurate to pi than the 4-sigfig value.

With regard to the custom function, it is evident that plotting the points in real time significantly reduces the efficiency of the algorithm. The while loop in Script 1 generates 4000 points in ~10 seconds; the algorithm in the custom function produces less than a tenth of that in the same time, and also cannot guarantee the accuracy of its result. It is simply a good way to visualize the operation of the algorithm.