CSCI 2020 - SOFTWARE SYSTEMS DESIGN AND INTEGRATION

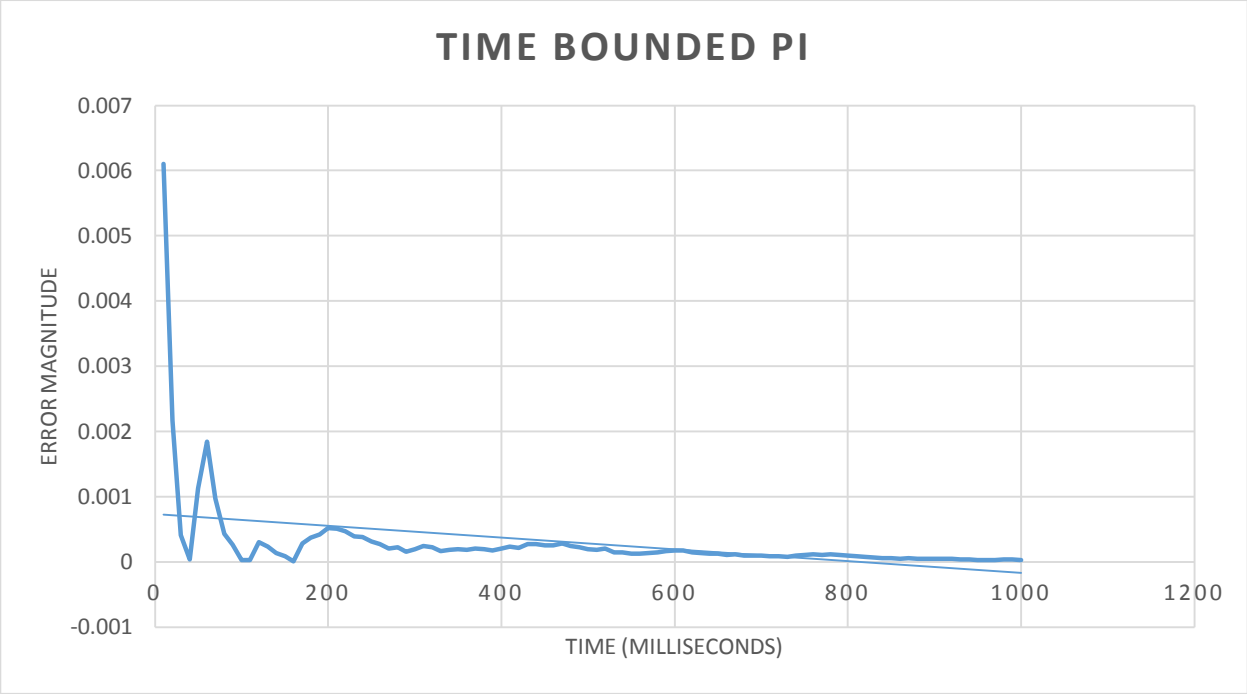
ASSIGNMENT 4 REPORT

March 29th, 2015

Luisa Rojas Garcia

The objective of this assignment was to calculate an estimated value of pi using both a non-threaded and a non-threaded time-bounded simulation. In order to achieve this, the Monte Carlo program was implemented; it's important to point out that this program is based on the fact that, in a unit square, the probability of a point generated at random fallen inside the unit circle is given by $\frac{\pi}{4}$.

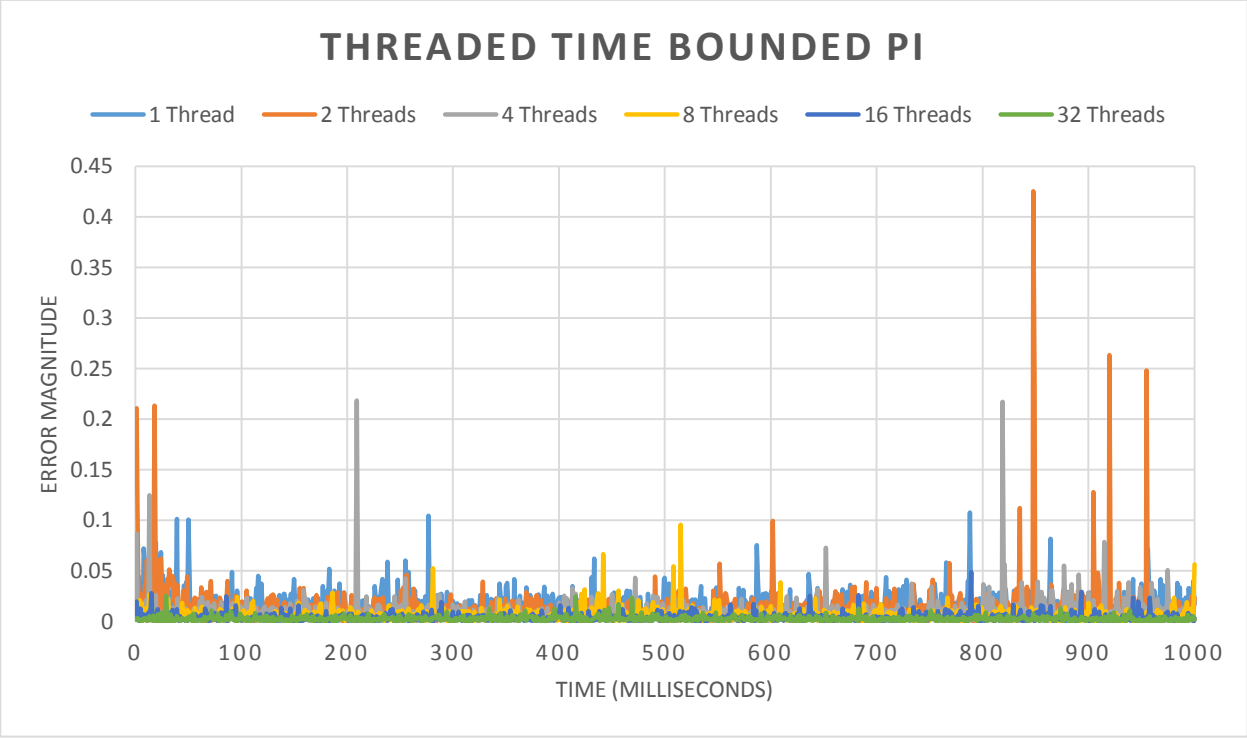
For the first part of the assignment, a TimeBoundedPi class was created. This class implemented the code provided and, additionally, it took optional command line inputs, so if the user does not give any input, the default value for the execution time would just be 100 milliseconds. The graph of this program's performance is shown below:



For this graph I decided it was best to also add trend line to see the pattern more clearly. Now, the decrease in the error magnitude (and therefore, the estimation quality) decreases as the time the program has to run increases.

The second part was slightly trickier, since the use of threads was needed. The input for this was not only the execution time but also the number of threads wanted (notice that these are set as optional as well), but the output was the same as part 1.

The way this class works is: First, the number of arguments are checked and, if needed, the default values are used. Then, a results array is created for all the threads to store their results in and an ArrayList to store the actual thread objects. Once this is set, the program runs all the threads using the start() function and then waits for them to be finished using the join() function. Once this is all done, all the individual results in the results array are added up and the estimated value of pi is finally calculated. The graph for the second part of this assignment is shown below:



In this graph, the trend is a bit more difficult to identify due to the amount of data plotted and the various peaks on the graph.

From what I can see, it seems that the test using 4-6 threads showed the lowest error magnitude and therefore, the best pi approximation. This means, then, that this number of threads is the optimum quantity for this kind of calculation.

As for the peaks on the graph, the cause is not clear, but I assume it must have to be something related to the usage of the laptop at the time of the tests.