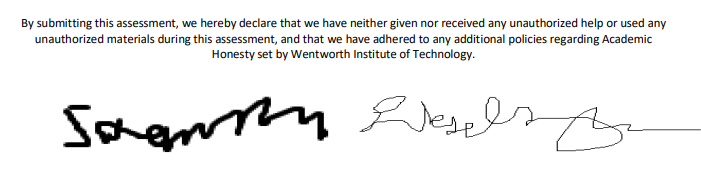
MPI Sample Analysis

Operating Systems Lab Exercise 3

Skyelar Craver & Steven Pitts

February 21, 2019

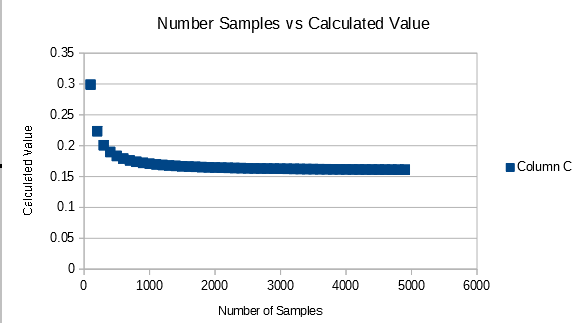


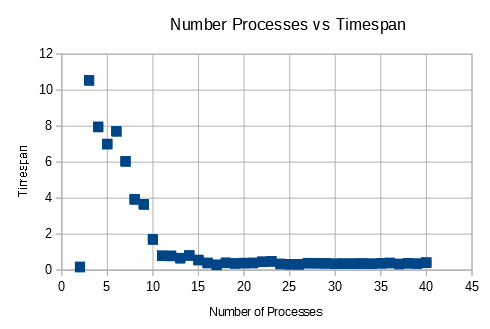
Part 1:

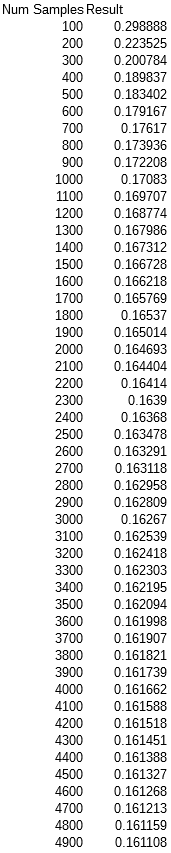
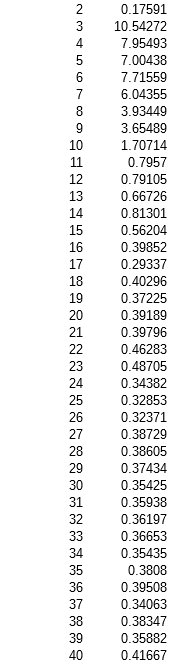
The goal of this experiment is to determine what optimal number of samples should be used when using the Monte Carlo Integration on a sequence of values. It will be assumed that the function f does not drastically change the answer, and it will be assumed that the Turing system in use is a reasonably accurate model for a generic 40-core host. The number of samples will be increased until a value is clearly being approached, and the lowest value of N that gives a decent representation of the true value will be chosen for future analysis.

Part 2:

The first necessary step is choosing a and b. For purposes of this experiment, it was decided that a will be 1 and b will be 100. The number of samples along with the value calculated from them is shown below:







Num Processors, Timespan

Given the approximation of 0.16, a value of 2000 was chosen to be a reasonable representation of the number of samples, but a value of 100000 samples was used to be safe and to get a reasonable timing diagram. Using that, we can now analyze execution time for different numbers of processes. The relation between number of processes and execution time is found above.

Part 3:

Given the asymptote seen around a value of 31 it can be gathered that increasing the number of processes above that will do either nothing or be detrimental to the execution time. Therefore, since the lowest execution time was found to occur at 31 processes, it can be determined that 31 is the optimal number of processes for the given program. Since the execution time for a single process was 10.54272, we can calculate that the calculated speedup with 31 processes would result in execution time of . However, the true execution time with 31 processes was 0.359, due to the fact that the master process must be synchronous, and the overhead associated with multiple processes. This overhead includes moving variables around between threads not on the same core (although this matters little in this program), and extra calculations/assignments that must be performed in order to set up the multiple processes. We are able to go above 40 processes since a processor core can run multiple processes in threads, but this is of little help to us, as it provides no improvement. Using batch processing, the program took 0.359 seconds to run, while without it took 10 seconds (over a minute counting wait time) to run. The results are very different, since batch processing allows for waiting for the computer to be ready with all cores needed, which can separate wait times from actual execution time.

Part 4:

Given the analysis of number of samples in part 2, the optimal number of samples was determined to be 100000, as it gives a close enough approximation of the integral without being so large that it would take a long period of time to process. Using the execution times mapped to the number of processes, the optimal processes count was determined to be 31, due to its lowest execution time.