High Performance Scientific Computing

Homework 1

Report

Ayan Tanwar, 22b0931

Contents

1 Introduction

In this report, we studied the influence of the number of iterations and number of threads on the runtime of the program. Using OpenMP, we parallelized the serial version of the code, making it more robust, and applied some optimizations to reduce the time even further.

2 Convergence Study

Given function: $\cos(x)$
Integral limits: $\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$
Analytical integral value: 2

I plotted the errors vs. number of iterations in log scale (to capture even minute errors) and observed that, in the case of Trapezoidal Integration, the log(error) followed a decreasing linear trend. On the other hand, the log(error) for Monte Carlo integration was decreasing but not smooth, instead quite erratic, mainly due to the randomness inherent in the method. I performed this analysis for 10^6 iterations, as beyond that the error became too insignificant to count.

Below are the plots for the same:

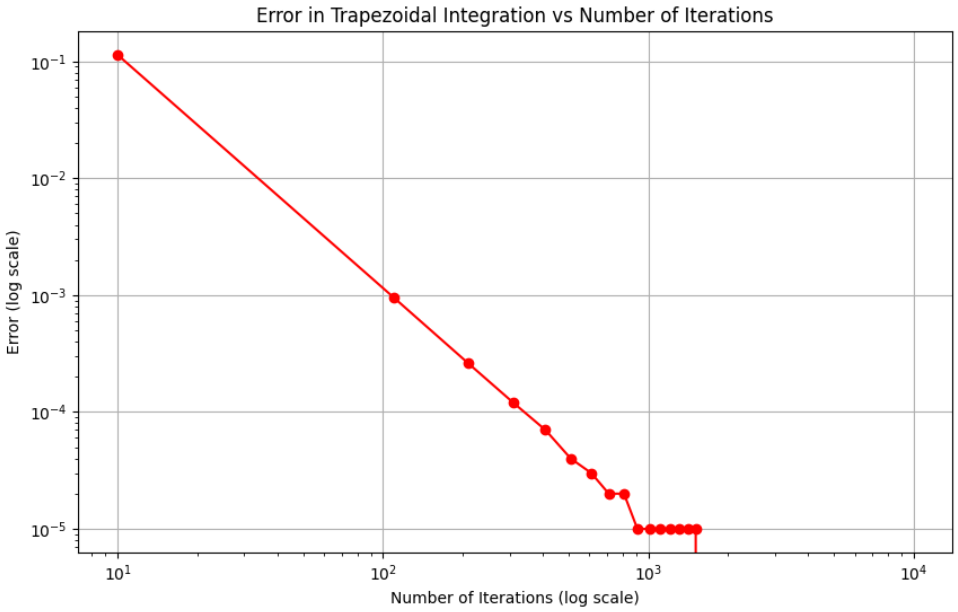

Figure 1: Error vs. Number of Iterations (Trapezoidal Integration)

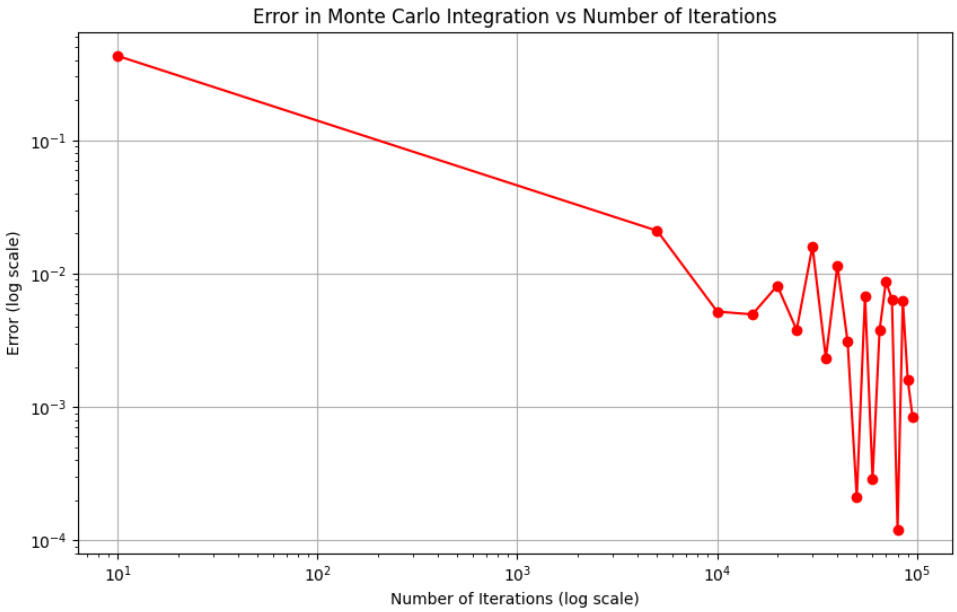

Figure 2: Error vs. Number of Iterations (Monte Carlo Integration)

3 Timing Study

I ran the code with each thread configuration 5 times on large number of iterations to obtain some distinguishable results and averaged the runtimes to get the average runtime for each thread count, for both the Monte Carlo method and the Trapezoidal method of integration. The Trapezoidal method took less time on average compared to the Monte Carlo method, mainly due to the overhead of generating random numbers repeatedly. I also observed that as the number of threads increased, the runtime decreased for both methods. Below are the runtimes for both methods.

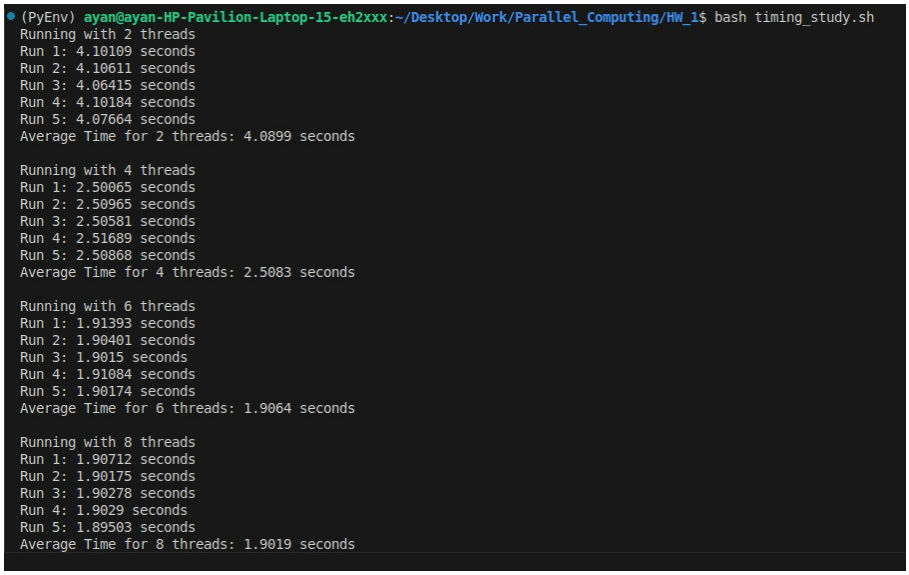

Figure 3: Timing Study (iterations : 10^9) (Trapezoidal Integration)

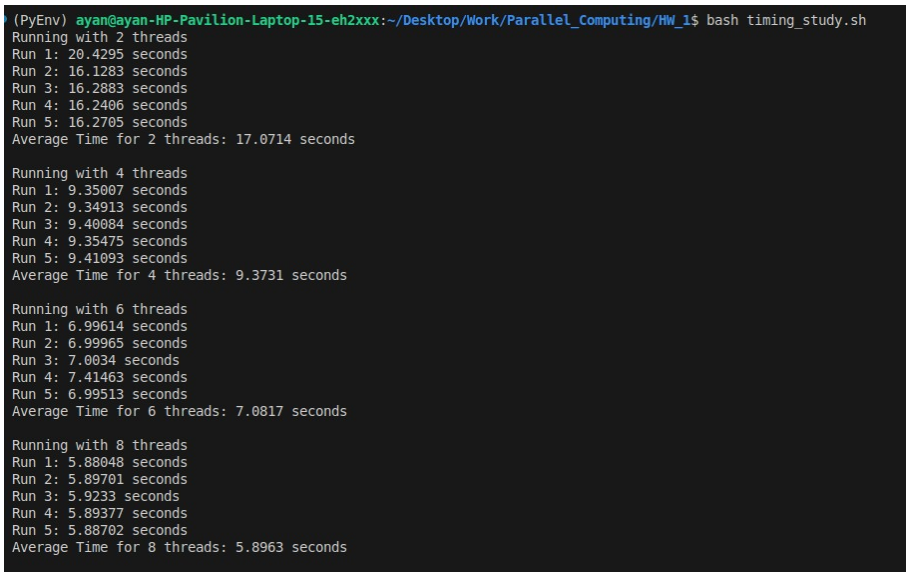

Figure 4: Timing Study (iterations : 10^8) (Monte Carlo Integration)