Assignment Report #2
Monte-Carlo estimation of the volume of an n-dimensional ℓ^p sphere

Student Name: Dias Suleimenov (ID: 202158836)
Course: PHYS 421 Parallel Computing

Submitted: October 21, 2025

Software packages used:

g++ 13.2.0	Main code
OpenMP 4.5	Parallelization
Matplotlib 3.8.0	Plotting
Jupyter Notebook	Data analysis

AI tools used:

Github Copilot	Coding assistance
Gemini 2.5 Pro	Help in conceptual understanding and refining of writing

Abstract

The performance of parallelized and serial Monte-Carlo algorigthm for estimating volume of n-dimensional ℓ^p-sphere was made. Maximum 3.22 times speedup of parallelized algorigthm was measured. Perfomative equivalence of static and dynamic scheduling was observed.

1 Introduction

A n-dimensional ball in the ℓ^p space is defined as (where $p > 0$):

$$|x_1|^p + |x_2|^p + \ldots + |x_n|^p \leq R^p.$$

By means of analysis it can be shown that the volume of n-dimensional ball in ℓ^p space with radius R is equal to:

$$V_n^p(R) = \frac{[2\Gamma(1+1/p)]^n}{\Gamma(1+n/p)} R^n.$$

However, one might estimate the volume of ℓ^p-sphere using the Monte-Carlo method. By emdedding sphere in hypercube of the same diameter. And then number of hits h of uniformly

randomly distrubuted points in hypercube can be calculated. And using that, volume can be estimated as:

$$V_n^p(R) \approx \frac{h}{N}(2R)^n.$$

The standard error of such Monte-Carlo approaches behaves like $\mathcal{O}(1/\sqrt{N})$.

This reports investigates performance of parallelized Monte-Carlo method compared to the single-threaded single threaded implementation. Relationtship between time performance, error and number of threads and trials is measured and discussed.

2 Methodology

Serialized and parallelized Monte-Carlo method for estimating the volume of ℓ^p sphere was implemented using OpenMP 4.5 and C++11. To guarantee thread-safety instances of random generators were run inside the threads. By `omp_get_wtime()` from OpemMP library, time performance of methods was measured. The scheduling and chunk size and number threads was defined by envirnoment variables. $\mathrm{T}_T hestandarderror fortheestimatedvolumewasderivedwithinaprobabilisticf$ sphere and a "failure" otherwise. Consequently, the total count of successful trials over N samples follows a Binomial distribution. Estimated proportion of hits $\hat{p}$ have the standard deviation of $\sqrt{\frac{p(1-p)}{N}}$. The estimated volume then equals $V_{\text{box}} \cdot \hat{p}$, where $V_{\text{box}} = (2R)^n$. Putting altogether it can be derived that standard deviation of volume is $\sqrt{\frac{V_{\text{box}}V - V^2}{N}}$.

3 Results

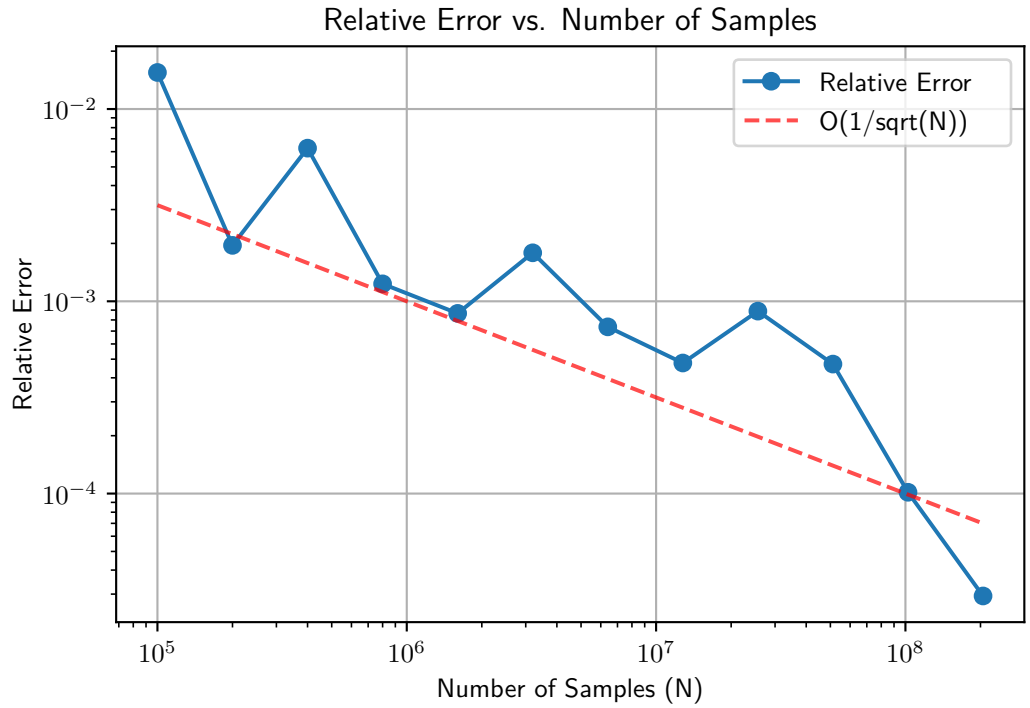

Figure 1: Relative error versus the number of samples for sphere with dimension $n = 10$, norm $p = 4$ and radius $R = 1$, with number of samples varying $N = 10^5, 2 \times 10^5, \ldots, 4.096 \times 10^8$.

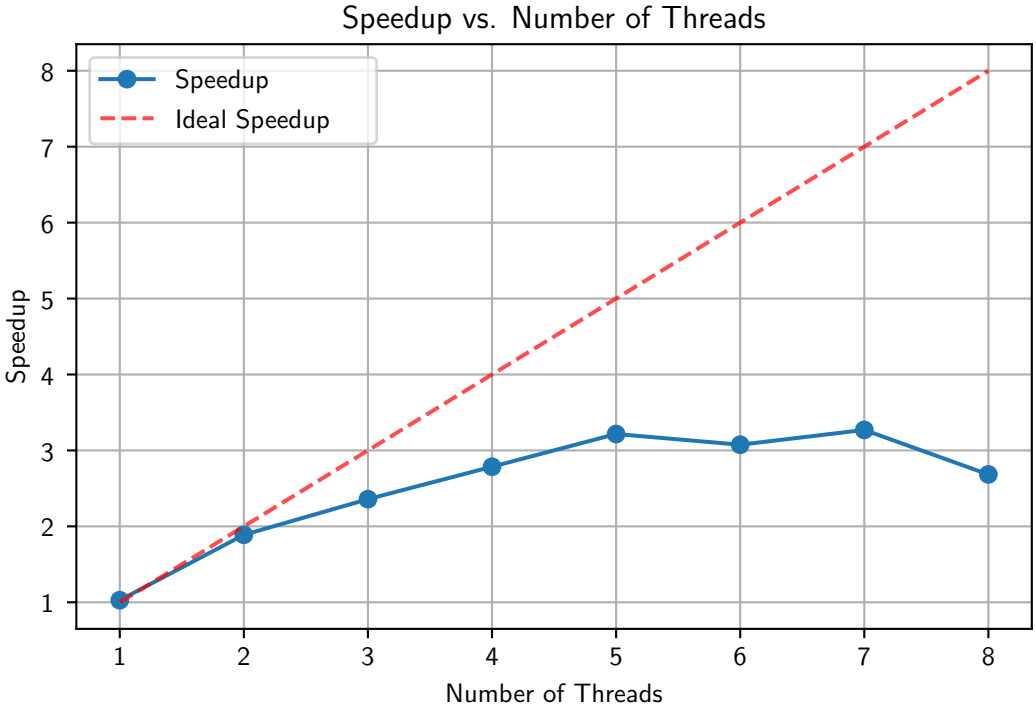

Figure 2: The speedup factor versus the number of threads for sphere with dimension $n = 10$, norm $p = 4$ and radius $R = 1$, number of samples $N = 5 \times 10^6$ and number of threads varying from 1 to 8.

Number Threads	Serial Time	Parallel Time	Speedup	Efficiency
1	1.393237	1.356019	1.027446	1.027446
2	1.393237	0.737777	1.888426	0.944213
3	1.393237	0.590847	2.358034	0.786011
4	1.393237	0.500212	2.785290	0.696323
5	1.393237	0.433204	3.216118	0.643224
6	1.393237	0.453095	3.074934	0.512489
7	1.393237	0.426079	3.269898	0.467128
8	1.393237	0.519195	2.683455	0.335432

Table 1: The speedup factor and efficiency in relation to number of threads for sphere with dimension $n = 10$, norm $p = 4$ and radius $R = 1$, number of samples $N = 5 \times 10^6$

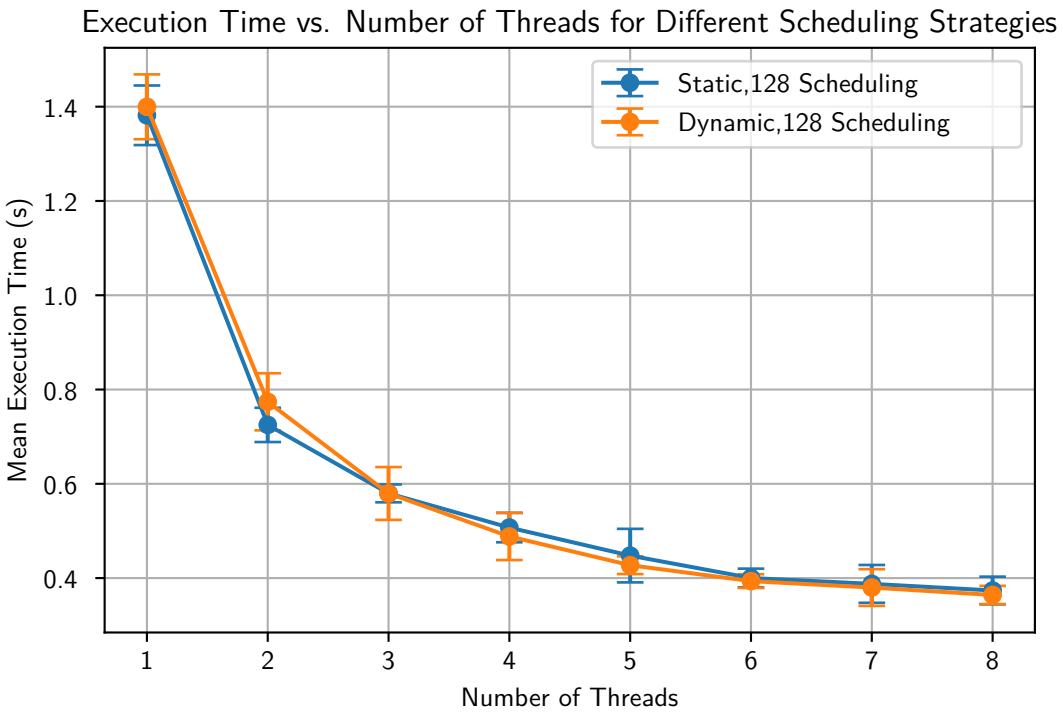

Figure 3: The comparison of performance of parallelization under static and dynamic scheduling strategy for sphere with dimension $n = 10$, norm $p = 4$ and radius $R = 1$, number of samples $N = 5 \times 10^6$ and number of threads varying from 1 to 8.

Dimensions	Monte-Carlo Volume	Analytical Volume	Error	Estimated Error
2	3.142532	3.141593	0.000939	0.000734
3	4.189376	4.188790	0.000586	0.001787
4	4.929168	4.934802	0.005634	0.003304
5	5.257600	5.263789	0.006189	0.005303
6	5.146240	5.167713	0.021473	0.007783
7	4.747392	4.724766	0.022626	0.010818
8	4.069632	4.058712	0.010920	0.014320
9	3.334656	3.298509	0.036147	0.018419
10	2.554880	2.550164	0.004716	0.022846

Table 2: The comparison between the absolute error and estimated error of volume obtained analytically versus by Monte-Carlo method, for the sphere of radius $R = 1$, norm $p = 2$, number of sample $N = 10^5$ and dimensions varying from 2 to 10.

4 Discussion

The results from the Figure **??** confirms that the absolute error of the Monte-Carlo estimation of volume decreases as $\mathcal{O}\left(\frac{1}{\sqrt{N}}\right)$.

The Table **??** confirms the formula derived at Methodology section. The error grows approximately the same as the formula and almost all errors in bounds of theoretical error prediction.

While the parallelization indeed gives significant performance boost, the speedup does not scale linearly. After parallelization on 5 threads the speedup is plateauing. Parallelization achieves the highest speedup of 3.22 times compared to the single-threaded version. The most efficient scaling, in terms of performance gain per thread, was observed with two and three threads.

A comparison between OpenMP scheduling strategies showed that static and dynamic schedules yielded nearly identical runtimes. This result is expected, since the Monte-Carlo algorigthm does not create disbalanced utilization of cores, since each thread have equal job. Through empirical testing it was found that parallelization of used Monte-Carlo algorigthm is mostly indifferent to used chunk size.

5 Conclusion

The parallelized implementation of the Monte-Carlo algorithm successfully accelerated the volume estimation, achieving a maximum speedup of 3.22 times over the serial version. It was also determined that both static and dynamic scheduling policies are equally effective due to the balanced nature of the computational workload. The $\mathcal{O}\left(\frac{1}{\sqrt{N}}\right)$ asymptotic decline of the Monte Carlo method's absolute error was empirically confirmed.