High Performance Scientific Computing

Report On

Homework 1

Submitted By

Keyur Borad

183109009

Indian Institute of Technology Bombay

Function given for integration:

$$I = \int_{-\pi/2}^{\pi/2} Cos(x)\, dx$$

The Exact answer of above integration is,

$$I = \int_{-\pi/2}^{\pi/2} Cos(x)\, dx \;=\; 2$$

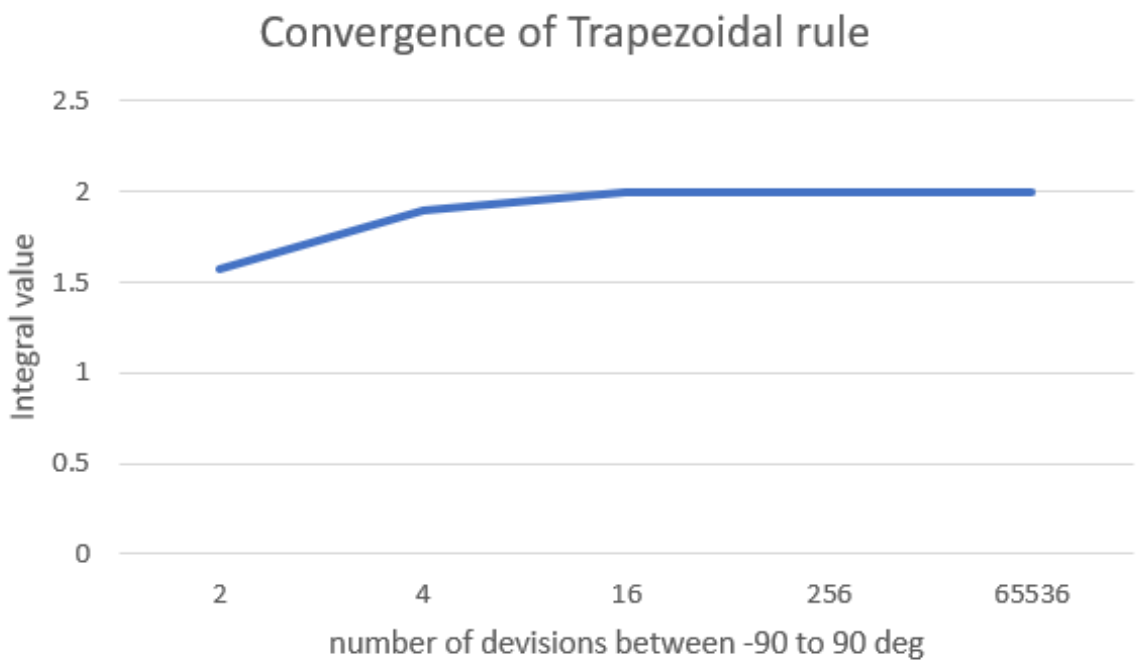
Convergence study

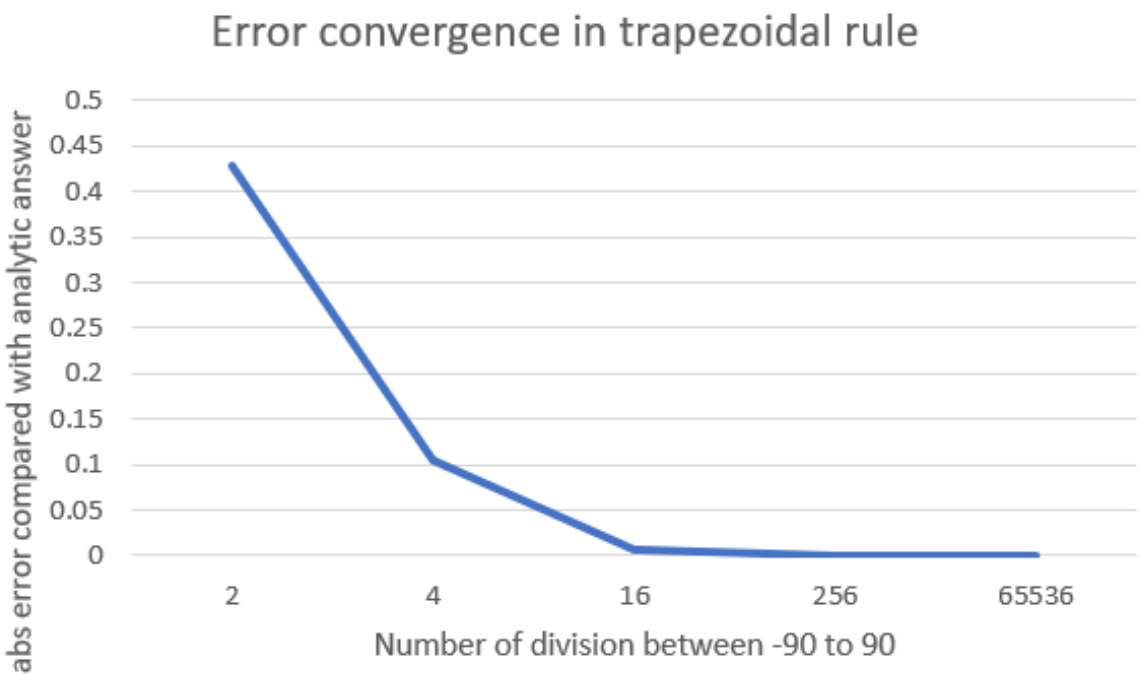
Convergence study is done for both Trapezoid method and Monte Carlo method of integration for different sample points

1. Trapezoidal Rule

For convergence study of Trapezoidal method ranges -90 to 90 is equally spaced divided

Equi-spaced divisions	2	4	16	256	65536
Integral	1.570796	1.896119	1.994376	1.999886	1.999948





2. Monte-Carlo Method

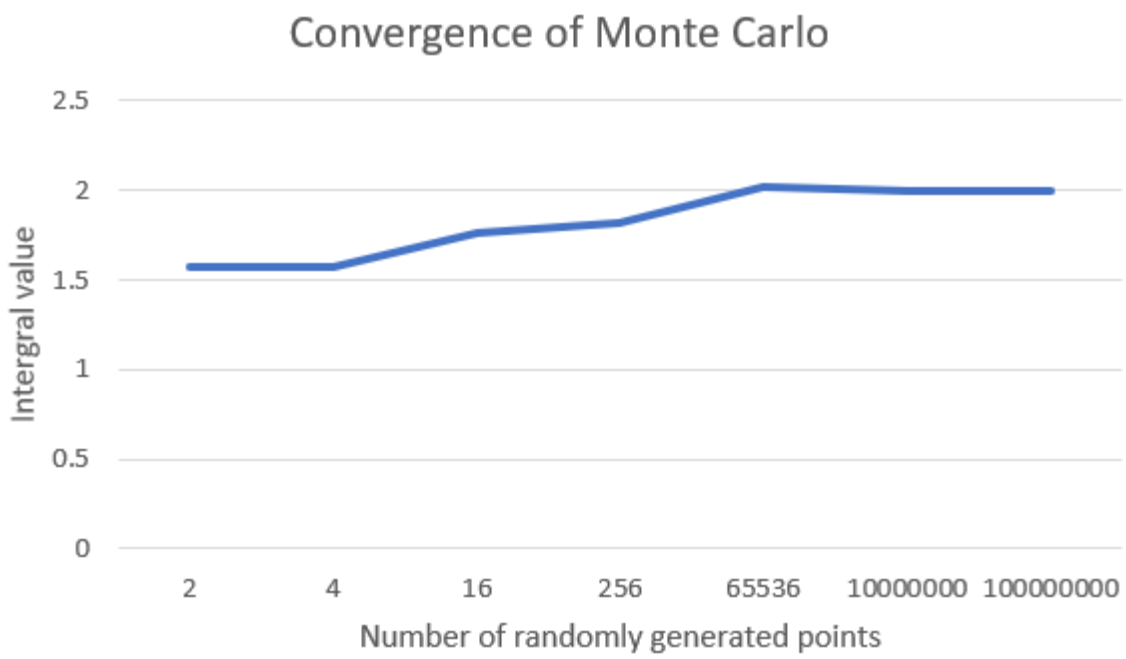
The following table shows the relation between number of sampling points used for calculation of integration and error in integration value calculated with respect to exact value. The error calculated as,

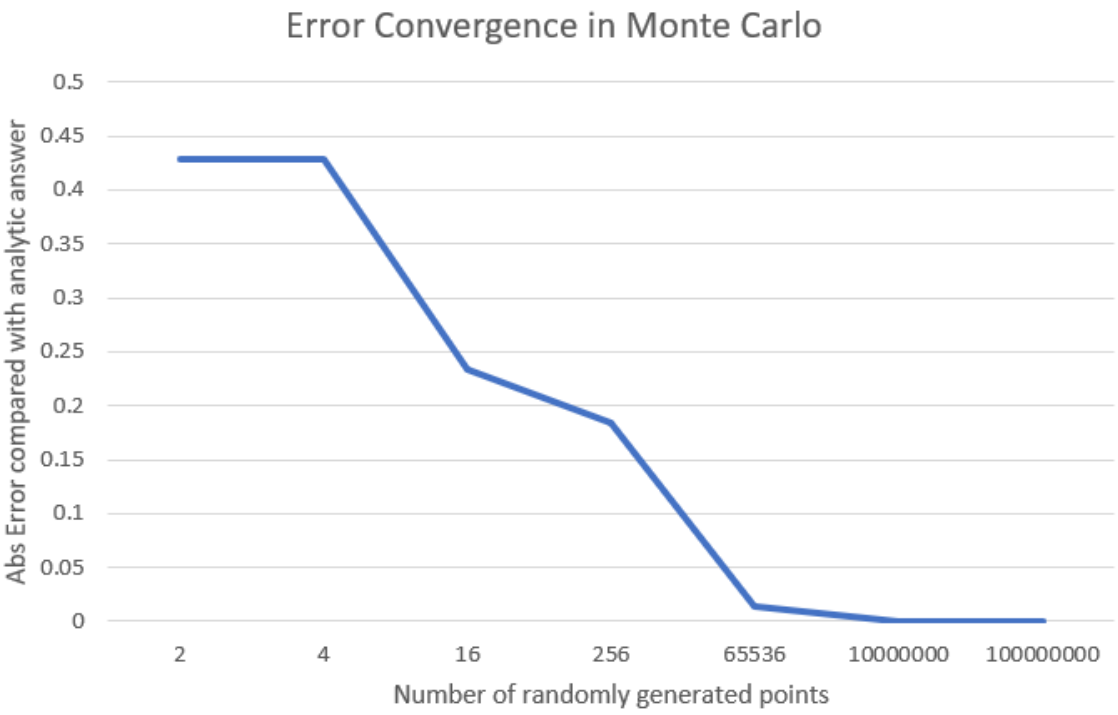
$$Error = abs(Exact\ integration\ calculated\ analytically - Integration\ calculated\ Numerically)$$

The error is calculated by taking average of error for five iterations.

No. Sample Points	2	4	16	256	65536	10^7	10^8
Integral	1.570796	1.570796	1.767146	1.816233	2.01354	1.999664	1.999925
Error	0.429204	0.429204	0.232854	0.183767	0.01354	0.000336	7.5E-05

For better visualization graph of the Error vs Sampling points is plotted.


Convergence of Monte Carlo
Intergral value
2.5
2
1.5
1
0.5
0
2
4
16
256
65536
10000000
100000000
Number of randomly generated points


Error Convergence in Monte Carlo
Abs Error compared with analytic answer
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
2
4
16
256
65536
10000000
100000000
Number of randomly generated points

Time Study

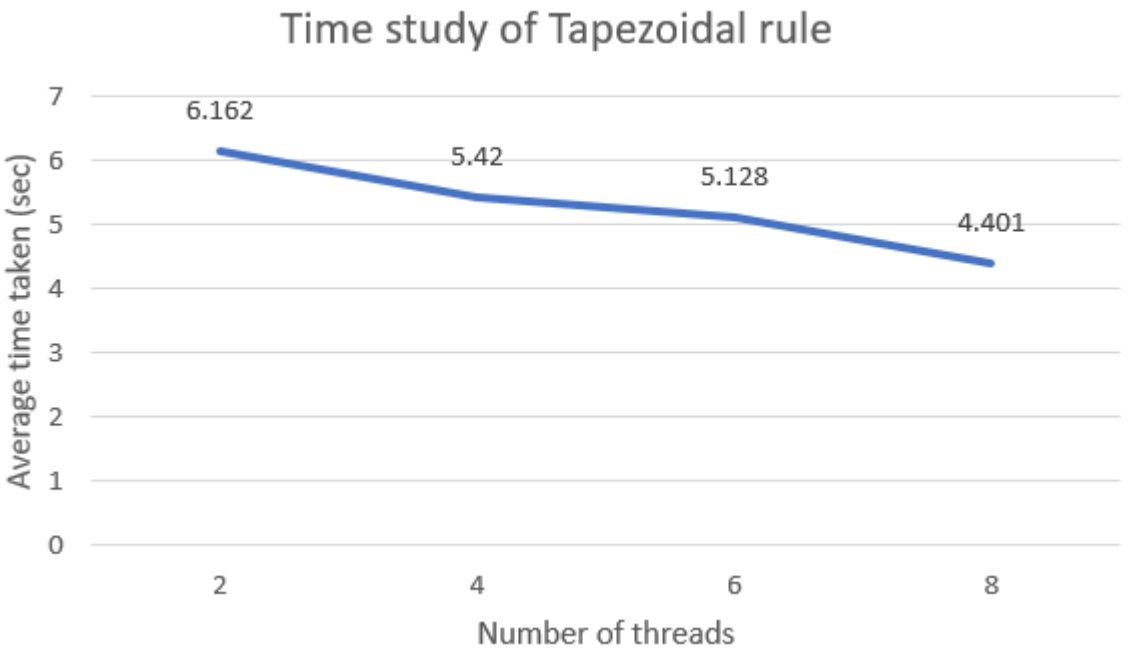
The time study is done for parallel code. The serial code is parallelized with 2, 4, 6 and 8 number of threads.

1. Trapezoidal Rule

The following table shows the relation between number of threads used for parallelization and corresponding time taken for calculation of integration. The time is calculated by taking average of time for five iterations. The time is corresponding to $\mathbf{5 \times 10^8}$ number of equispaced divisions.

```
keyur_ubuntu@Keyurs-PAVILION:/mnt/e/Shell_script/HPSC/HW1$ gcc -o m -fopenmp trapezoidal.c -lm
keyur_ubuntu@Keyurs-PAVILION:/mnt/e/Shell_script/HPSC/HW1$ time ./m

Integration of cos from -pi/2 to +pi/2 is : 1.999949 , with 2 threads takes at avg time of  6.162500 secs

Integration of cos from -pi/2 to +pi/2 is : 1.999949 , with 4 threads takes at avg time of  5.420312 secs

Integration of cos from -pi/2 to +pi/2 is : 1.999949 , with 6 threads takes at avg time of  5.128646 secs

Integration of cos from -pi/2 to +pi/2 is : 1.999949 , with 8 threads takes at avg time of  4.401953 secs

real    1m54.205s
user    8m19.969s
sys     0m0.000s
keyur_ubuntu@Keyurs-PAVILION:/mnt/e/Shell_script/HPSC/HW1$
```

No. of threads	2	4	6	8
Time(sec)	6.162	5.4203	5.128	4.4019



Plot of Time vs Number of threads Trapezoidal Method

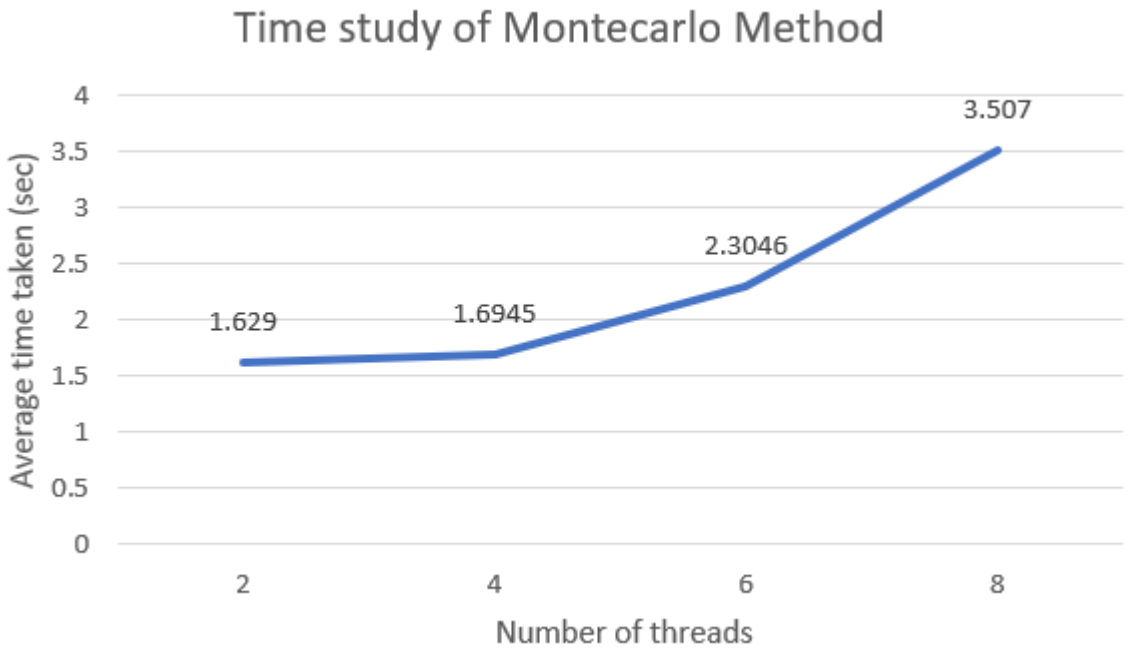
2. Monte Carlo Method

For time study of Monte Carlo method same number of sample points and threads are considered as that of Trapezoidal method.

The following table shows the relation between number of threads used for parallelization and corresponding time taken for calculation of integration. The time is calculated by taking average of time for five iterations. The time is corresponding to 10^7 number of sampling points.

```
keyur_ubuntu@Keyurs-PAVILION:/mnt/e/Shell_script/HPSC/HW1$ gcc -o m -fopenmp monte_carlo_omp.c -lm
keyur_ubuntu@Keyurs-PAVILION:/mnt/e/Shell_script/HPSC/HW1$ time ./m

Intergral value of cos from -90 to 90: 2.000087, with 2 threads
Average time taken for 5 times executing the code: 1.629687 sec
************************************************************

Intergral value of cos from -90 to 90: 2.000083, with 4 threads
Average time taken for 5 times executing the code: 1.694531 sec
************************************************************

Intergral value of cos from -90 to 90: 2.000059, with 6 threads
Average time taken for 5 times executing the code: 2.304688 sec
************************************************************

Intergral value of cos from -90 to 90: 1.999772, with 8 threads
Average time taken for 5 times executing the code: 3.507031 sec
************************************************************

real    0m51.254s
user    1m37.531s
sys     2m42.172s
keyur_ubuntu@Keyurs-PAVILION:/mnt/e/Shell_script/HPSC/HW1$
```

No. of threads	2	4	6	8
Time(sec)	1.6296	1.6945	2.3046	3.50703



Plot of Time vs Number of threads Monte Carlo Method

Conclusion

1. Both trapezoidal and montecarlo method converges to the analytical solution as the number divisions(for trapezoidal) and random sample points(for montecarlo) are increased to higher order.
2. It is been noticed that trapezoidal method converges faster than that of Montecarlo method
3. Talking about the **time study**, It is observed in **Trapezoidal rule** code as the number of threads are increased from 2 to 8 consequently **time taken** to run the code is also **reduces**. This has been possible only with the help of openmp library. Clauses used are “Private” and “reduction”.
4. **Time study in Montecarlo** method showed some adverse effect that is as the number of threads are increased from 2 to 8 , the **time taken** for the code to run is also **increases.** And this might be because of the use of the random number generator in c, where it seeds the time and generates the random number. So basically this might force the program to run serially as 2 iterations cannot have the same random number because of the same time stamp.