Name: Patel Aayush
Enroll no. 183109002
Email id: 183109002@iitb.ac.in

Homework Assignment 1
High Performance Computing, ME 667

2021-03-13

1 First Exercise: Integration using Trapezoidal method

To integrate $\cos(theta)$ from $-\pi/2$ to $\pi/2$ using trapezoidal method the language used is C++ and compilation done using g++ gnu compiler on Intel i4 processor.

1.1 Convergence study

To quantify and finding optimal number of grid points, code was run at different number of grid points(N) starting from 10 to 10^7. For all simulations, code was run serially a.k.a using one thread. Below graph quantifies the relative error of approximate trapezoidal integral with respect to definite integral using different grid points. It can be seen that at 10^7 the relative error is too small and it can be taken as correct value. In addition data is provided in the table 1.1

Sr No.	no. of grid points	integral	abs relative error(%)
1	10	1.97965	1.017458
2	100	1.999832	0.0083
3	1000	1.9999983	8.24e-5
4	10000	1.999999	8.22e-7
5	100000	2	8.22e-9
6	1000000	2	7.89e-11
7	10000000	2	8.51e-12

Table 1.1 grid convergence data

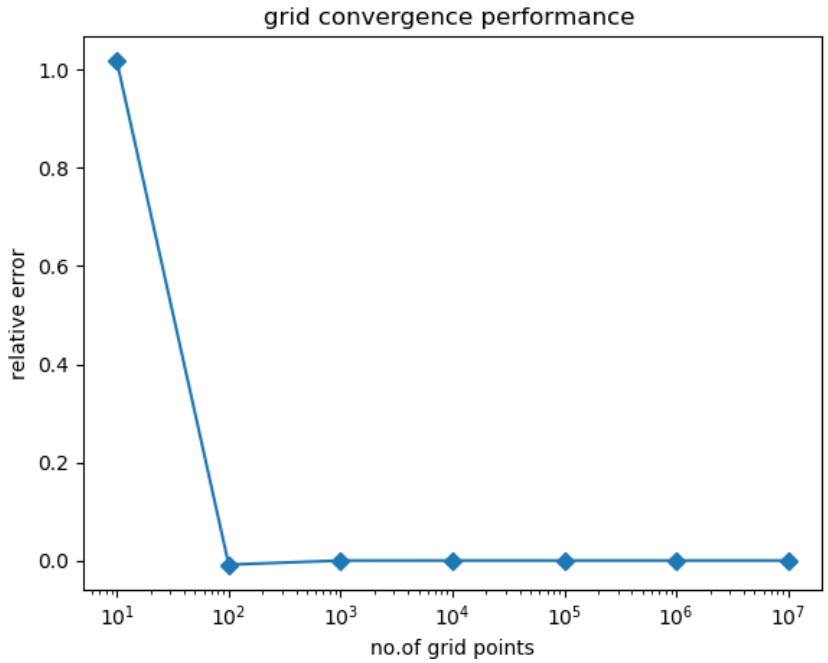

figure 1.1

1.2 Performance study using OpenMp

The code is written on shared memory parallelisms. It uses the library to include functionality of thread parallelisation.The performance is quantified using 1,2,4,6 & 8 number of threads using 10^7 grid points.It is evident that after 4 threads the speedup is not significant due to current capability of cpu. Below graph shows the time performance averaged over five cycles using different threads. Additionally data is provided in tabulated format.

Sr No.	no. of threads	time average of five cycle in sec.	speed up
1	1	0.232	1
2	2	0.1216	1.90
3	4	0.1088	2.13
4	6	0.1174	1.97
5	8	0.1084	2.14

Table 1.2 computation performance using shared memory parallelism

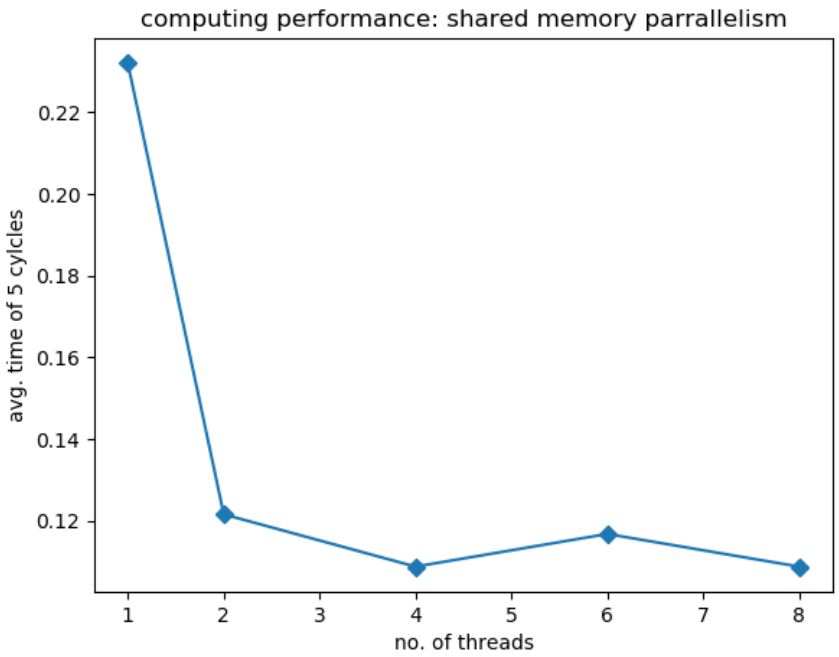

figure 1.2

2 Second Exercise: Integration using Monte Carlo method

To integrate $\cos(theta)$ from $-\pi/2$ to $\pi/2$ using Monte Carlo method the language used is C++ and compilation done using g++, gnu compiler on Intel i4 processor.

2.1 Convergence study

To quantify and finding optimal number of grid points, code was run at different number of grid points(N) starting from 10 to 10^7. For all simulations, code was run serially a.k.a using one thread. Below graph quantifies the relative error of approximated integral with respect to definite integral using different grid points. It can be seen that at 10^7 the relative error is too small and it can be taken as correct. In addition data is provided in the table 2.1

Sr No.	no. of grid points	integral	Relative error(%)
1	10	2.51	25.667
2	100	1.82212	-8.8938
3	1000	2.02318	1.1592
4	10000	1.99334	-0.3397
5	100000	2.0064	-0.04019
6	1000000	1.9992	-0.0400
7	10000000	2.0012	0.06366

Table 2.1 grid convergence data

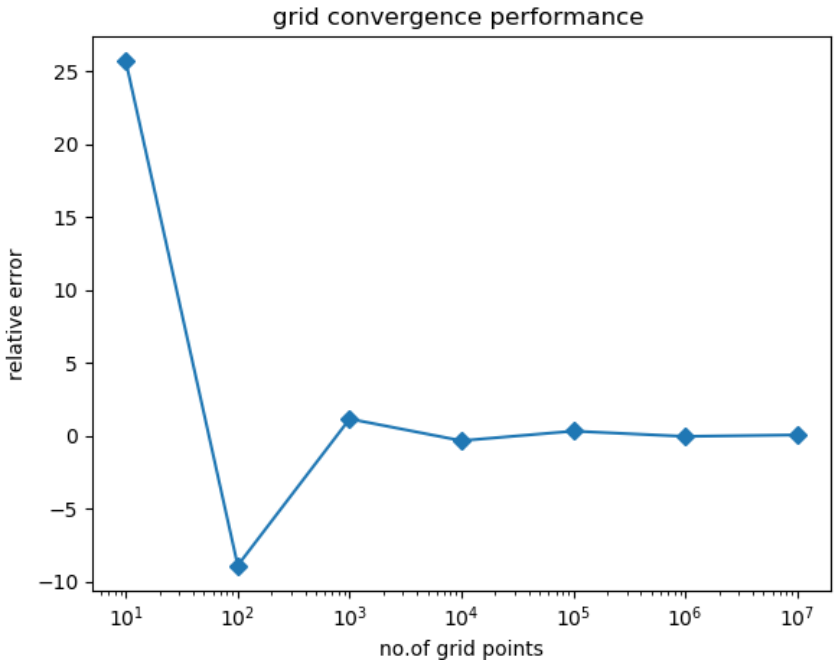

figure 2.1

2.2 Performance study using OpenMp

The code is written on shared memory parallelisms. It uses the omp library to include functionality of thread parallelisation.The performance is quantified using 1,2,4,6 & 8 number of threads using 10^7 grid points.It is evident that after 4 threads the speedup is not significant due to current capability of cpu. Below graph shows the time performance averaged over five cycles using different threads. Additionally data is provided in tabulated format.

Sr No.	no. of threads	time average five cycle in sec.	speed up
1	1	1.8912	1
2	2	1.8426	1.02
3	4	1.63575	1.15
4	6	1.2508	1.5
5	8	1.29	1.46

Table 2.2 computation performance using shared memory parallelism

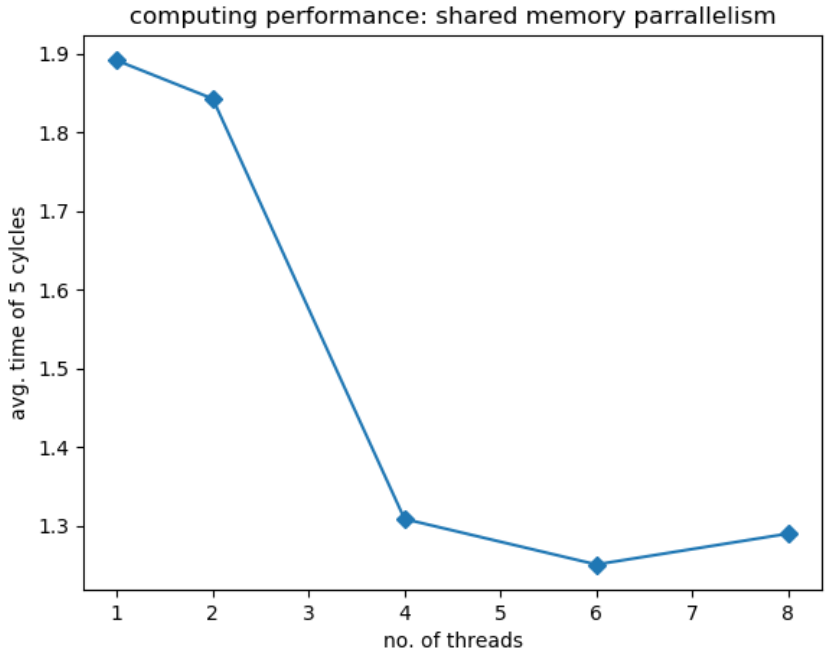

figure 2.2