MCG 5138 Project

Name: Hao Li Student Number: 300128022

December 2020

1 Question 1

The code of this part could be seen in $Q1.ipynb$

Some part of the code could be referred to the following link. `https://github.com/stefantaylor/Parallel-Programming-Languages-and-Systems/blob/master/cw2/aquadPartA.c`

The main idea of the algorithm is as follows:

1. A worker cpu calculate the integral I1 and I2 by using Simpson rule and composite Simpson rule

$$I1 = h * (fleft + fmid * 4 + fright)/6 \tag{1.1}$$

$$I2 = h * (fleft + 4 * f((3 * left + right)/4) + fmid * 2 + 4 * f((left + 3 * right)/4) + fright)/12 \tag{1.2}$$

2. If the difference between I1 and I2 is smaller than the tolrance, it will return I2 to the farmer cpu. If not, it will divide the tomain by 2, and send the interval to farmer cpu. This worker cpu will be idle.

3. The farmer cpu pop a interval from the stack, and sends it to a worker cpu which is idle.

4. After receiving the sub-interval from the worker cpus, the farmer cpu will add them to the stack.

2 Question 2

The code of this part could be seen in $Q2.ipynb$

1)When, tolerance is $10^{-3}, howmanylevelsdoyouneedtoreachthetolerance?$

When $maxLevel = 10$, I can not get the desired tolerance by using my algorithm. I need at least 16 levels to reach the tolerance.

2) What is the resulting approximation and the estimated error?

When level is 16, the resulting approximation is 5.084225101442328.

The estimated error is $3.0175925736331215 * 10^{-5}$.

3) How does the estimated error compare to the real error?

The estimated error is $3.0175925736331215 * 10^{-5}$.

The real error is $3.1758753304078624 * 10^{-4}$.

The estimated error is smaller than the real error.

4) When $tolerance = 10^{-6}$, the number of levels is 91. The resulting approximation is 5.084542656189077. The estimated error is 3.867412674823703e-08. The real error is 3.278629012726242e-08. The plot of function evaluation points are shown in Figure 1.

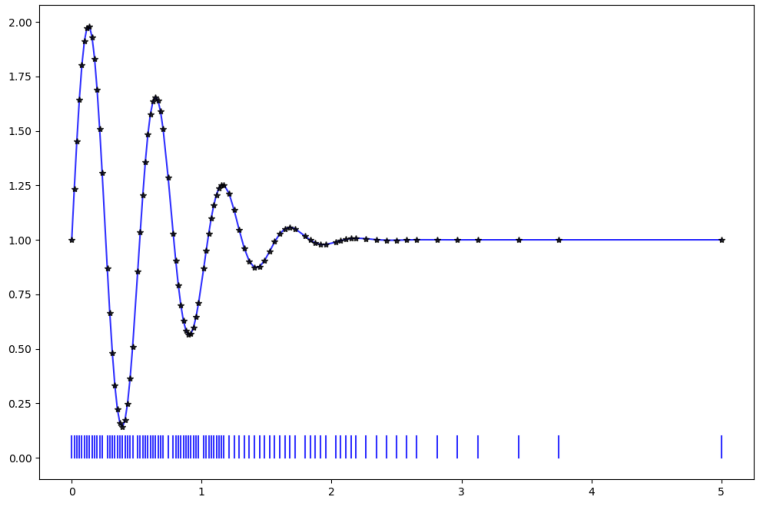


Figure 1: Adaptive integration

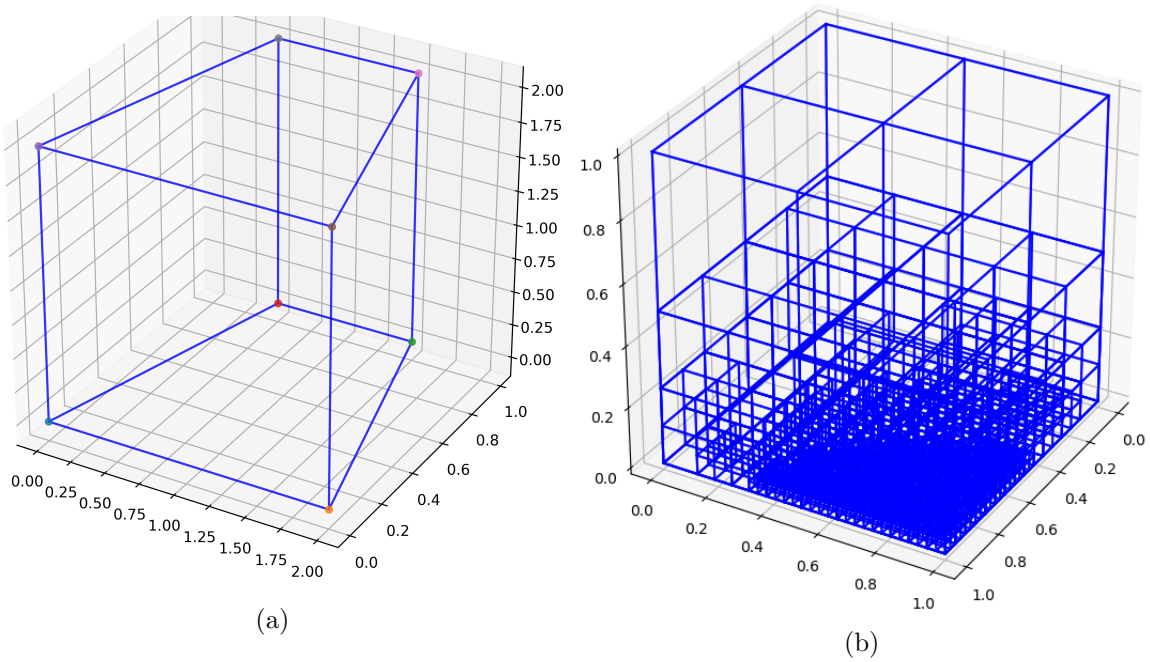
3 Question 3

The code of this question could be seen in **Q3.ipynb**.

Use the trilinear transformation method to map an arbitrary haxahedron into a referenced unitary cube. Then, use the parallelized 3-D adaptive Simpson algorithm to get the integral. The formula could be shown in the following. $\mathbf{J}$ is the Jacobian matrix. p, q, r are the coordinates in the unitary cube.

$$\begin{aligned}\Gamma &= \iiint_V f(x,y,z)dv = \int_{-1}^{0}\int_{-1}^{0}\int_{1}^{0} f(x(p,q,r),y(p,q,r),z(p,q,r))\frac{\partial(x,y,z)}{\partial(r,\xi,\eta)}\ dp\ dq\ dr \\ &= \int_{-1}^{0}\int_{-1}^{0}\int_{1}^{0} f(x(p,q,r),y(p,q,r),z(p,q,r))\ \det\mathbf{J}\ \ dp\ dq\ dr\end{aligned}$$

When $f(x,y,z) = x \cdot y \cdot z^{0.8} + sin(x) + sin(y)$, apply the above algorithm to calculate the integral. The domain of integration is defined by these eight vertices: $v000 = [0,0,0]$, $v100 = [2,0,0]$, $v110 = [1.5,1,0]$, $v010 = [0.5,1,0]$, $v001 = [0,0,2]$, $v101 = [2,0,2]$, $v111 = [1.5,1,2]$, $v011 = [0.5,1,2]$. It could be seen in Figure 2a. After transforming it into a unitary cube and applying adaptive Simpson algorithm, we can see the adaptive grids shown in Figure 2b.



(a) (b)

Figure 2: 2a shows the original arbitrary hexahedral. 2b shows the adaptive grids in the transformed unitary cube.

4 Question 4

In this question, I did not use the trilinear transformation Jacobian matrix mentioned in the discussion forum. Instead, I use the Jacobian transformation $\frac{\partial(x,y,z)}{\partial(r,\xi,\eta)}$ in equation (4.1) and (4.2) to transform the integral in x, y, z coordinates to r, ξ, η in cubed-sphere coordinates.

4.1 a

Volume

When $1 \leq r \leq 10, -\frac{\pi}{4} \leq \xi \leq \frac{\pi}{4}, -\frac{\pi}{4} \leq \eta \leq \frac{\pi}{4}$, divide the sector into two cells, thus, left and right. The volume of the left and right cell are the same, and the volume of each hexahedral is **348.71**. The code of this part could be seen in **Q4aVolume.ipynb**. It apply the following formula to calculate the volume, taking the left cell for example.

$$V = \iiint_V dx\, dy\, dz \tag{4.1}$$

$$= \int_{-\frac{\pi}{4}}^{\frac{\pi}{4}} \int_{-\frac{\pi}{4}}^{0} \int_{1}^{10} \left| \frac{\partial(x,y,z)}{\partial(r,\xi,\eta)} \right| \, dr\, d\xi\, d\eta \tag{4.2}$$

The above equation transform a cubed-sphere sector to a 3-D cube. After that, use the parallelized 3-D adaptive algorithm to get the integral.The results could be seen in Figure 3:

```
############ Result of left cell ###############
volume of left:  348.71574361521783
errorEstimate:   0.001232734442718512
finestLevel:     1/4.0*(R2-R1)
############ Result of right cell ###############
volume of right: 348.7157436152179
errorEstimate:   0.0012327344427212356
finestLevel:     1/4.0*(R2-R1)
```

Figure 3: Volume of the left and right cells

Centroid

The code of this part could be seen in **Q4aCentroid.ipynb**. Use the following equations

to calculate the x, y, z coordinates of the centroid, taking the left cell for example.

$$\bar{x} = \frac{\iiint x \ dx\ dy\ dz}{V} = \frac{\int_{-\frac{\pi}{4}}^{\frac{\pi}{4}} \int_{-\frac{\pi}{4}}^{0} \int_{1}^{10} \left|\frac{\partial(x,y,z)}{\partial(r,\xi,\eta)}\right| x \ \ dr\ d\xi\ d\eta}{V}$$
$$\bar{y} = \frac{\iiint y \ dx\ dy\ dz}{V} = \frac{\int_{-\frac{\pi}{4}}^{\frac{\pi}{4}} \int_{-\frac{\pi}{4}}^{0} \int_{1}^{10} \left|\frac{\partial(x,y,z)}{\partial(r,\xi,\eta)}\right| y \ \ dr\ d\xi\ d\eta}{V}$$
$$\bar{z} = \frac{\iiint z \ dx\ dy\ dz}{V} = \frac{\int_{-\frac{\pi}{4}}^{\frac{\pi}{4}} \int_{-\frac{\pi}{4}}^{0} \int_{1}^{10} \left|\frac{\partial(x,y,z)}{\partial(r,\xi,\eta)}\right| z \ \ dr\ d\xi\ d\eta}{V}$$

Then, use the parallelized 3-D adaptive algorithm to get the integral. The results could be seen as follows:

```
############ Result of left cell ###############
The x coordinate of centroid =  6.23970592101232
The y coordinate of centroid =  -2.51032446900229
The z coordinate of centroid =  1.82252530441286e-16
errorEstimate of x:             -1.27077662640063e-5
errorEstimate of y:             -0.000431095578219720
finestLevel of x:               1/4.0*(R2-R1)
finestLevel of y:               1/1.0*(R2-R1)
############ Result of right cell ###############
The x coordinate of centroid =  6.23970592101232
The y coordinate of centroid =  2.51032446900229
The z coordinate of centroid =  -2.59762227295626e-16
errorEstimate of x:             -1.27077662638731e-5
errorEstimate of y:             0.000431095578219676
finestLevel of x:               1/4.0*(R2-R1)
finestLevel of y:               1/1.0*(R2-R1)
```

Figure 4: Coordinate of centroid of the left and right cells

4.2 b

When $1 \leq r \leq 10, -\frac{\pi}{4} \leq \xi \leq \frac{\pi}{4}, -\frac{\pi}{4} \leq \eta \leq \frac{\pi}{4}$, divide the sector into two cells, thus, left and right. The code of this part could be seen in **Q4b.ipynb**. It applies the parallelized 3-D adaptive algorithm to calculate the cell average values. It uses the Jacobian matrix to transform the x, y, z coordinates to r, ξ, η coordinates. The formula could be seen as follows, taking the left cell for example.

$$\bar{U} = \frac{1}{V} \int f(x,y,z) dv$$
$$\bar{U} = \frac{1}{V} \int_{-\frac{\pi}{4}}^{\frac{\pi}{4}} \int_{-\frac{\pi}{4}}^{0} \int_{1}^{10} f(x(r,\xi,\eta), y(r,\xi,\eta), z(r,\xi,\eta)) \left|\frac{\partial(x,y,z)}{\partial(r,\xi,\eta)}\right| \ \ dr\ d\xi\ d\eta$$

The cell average value of left cell is 2.95. The estimate error is -0.016. The cell average value of right cell is 3.79. The estimate error is -0.02. If the tolerance is too small (< 0.01), it will have more levels than the previous integral, and the simulation time is much longer (more than 10 minutes). The results could be seen In figure 5.

```
############ Result of left cell ###############
volume of left:  2.95019813355922
errorEstimate:   -0.0160691880731845
finestLevel:     1/8.0*(R2-R1)
############ Result of right cell ###############
volume of right: 3.78647505806275
errorEstimate:   -0.0207100749304543
finestLevel:     1/8.0*(R2-R1)
```

Figure 5: Cell average values of the left and right cells

5 Question 5

5.1 (a)

Use the build-in function **fminsearch** to find the minimum value. The code of this part could be seen in **Q5a.m** when $N = 5$.

The global minimum for $N = 3, 5, 7$ using 5 initial guess are shown in the table 1, 2 and 3.

Table 1: The global minimum value of Rosenbrock function ($N = 3$)

Initial Guess	Number of Iterations	Global Minimum
0.5, 0.5, 0.5	112	5.56e-10
0.8, 0.8, 0.8	73	9.58e-10
1.2, 1.2, 1.2	108	8.82e-10
1.3, 1.3, 1.3	113	6.13e-10
1.5, 1.5, 1.5	125	1.79e-09

Table 2: The global minimum value of Rosenbrock function ($N = 5$)

Initial Guess	Number of Iterations	Global Minimum
0.5, 0.5, 0.5, 0.5, 0.5	325	3.98e-09
0.8, 0.8, 0.8, 0.8, 0.8	220	2.23e-09
1.2, 1.2, 1.2, 1.2, 1.2	235	2.28e-09
1.3, 1.3, 1.3, 1.3, 1.3	232	1.22e-08
1.5, 1.5, 1.5, 1.5, 1.5	381	1.36e-08

Table 3: The global minimum value of Rosenbrock function ($N = 7$)

Initial Guess	Number of Iterations	Global Minimum
0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5	767	2.67e-09
0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8	500	1.78e-09
1.2, 1.2, 1.2, 1.2, 1.2, 1.2, 1.2	504	3.84e-09
1.3, 1.3, 1.3, 1.3, 1.3, 1.3, 1.3	623	2.05e-09
1.5, 1.5, 1.5, 1.5, 1.5, 1.5, 1.5	633	2.49e-09

The local minimum for $N = 3, 5, 7$ using 5 initial guess are shown in the table 4, 5 and 6.

Table 4: The local minimum value of Rosenbrock function ($N = 3$)

Initial Guess	Number of Iterations	Local Minimum
-0.5, 0.5, 0.5	191	7.35e-10
-0.8, 0.8, 0.8	213	8.24e-10
-1.2, 1.2, 1.2	224	1.25e-09
-1.3, 1.3, 1.3	243	3.37e-09
-1.5, 1.5, 1.5	256	2.04e-10

Table 5: The local minimum value of Rosenbrock function ($N = 5$)

Initial Guess	Number of Iterations	Local Minimum
-0.5, 0.5, 0.5, 0.5, 0.5	326	3.9308
-0.8, 0.8, 0.8, 0.8, 0.8	201	3.9308
-1.2, 1.2, 1.2, 1.2, 1.2	278	3.9308
-1.3, 1.3, 1.3, 1.3, 1.3	284	3.9308
-1.5, 1.5, 1.5, 1.5, 1.5	400	3.9308

Table 6: The local minimum value of Rosenbrock function ($N = 7$)

Initial Guess	Number of Iterations	Local Minimum
-0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5	682	3.9836
-0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8	445	3.9836
-1.2, 1.2, 1.2, 1.2, 1.2, 1.2, 1.2	494	3.9836
-1.3, 1.3, 1.3, 1.3, 1.3, 1.3, 1.3	710	3.9836
-1.5, 1.5, 1.5, 1.5, 1.5, 1.5, 1.5	613	3.9836

To get the local minimum, the Rosenbrock function value versus the number of iterations when $N = 3, 5, 7$ could be seen in Figure 6, 7, 8.

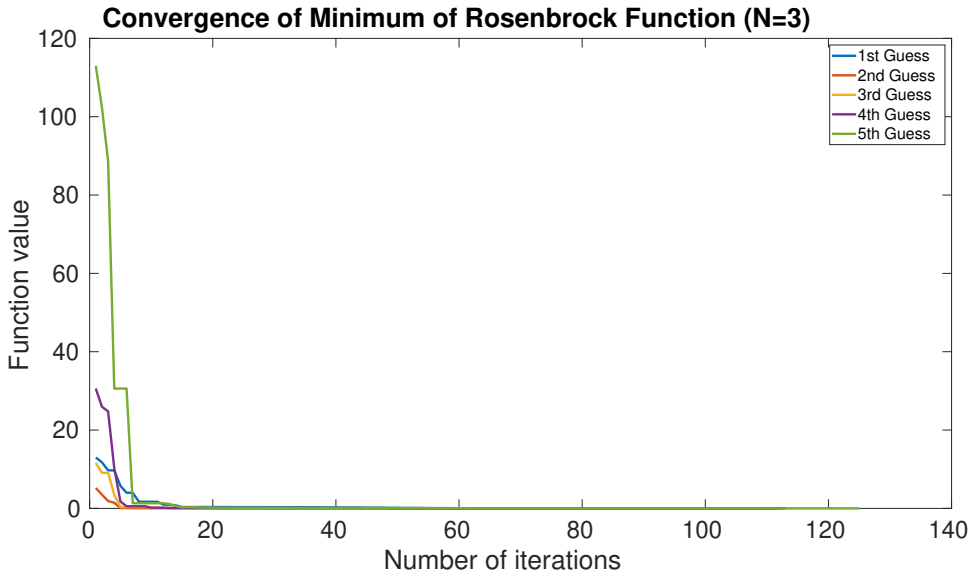


Figure 6: Rosenbrock function value versus the number of iterations (N=3)

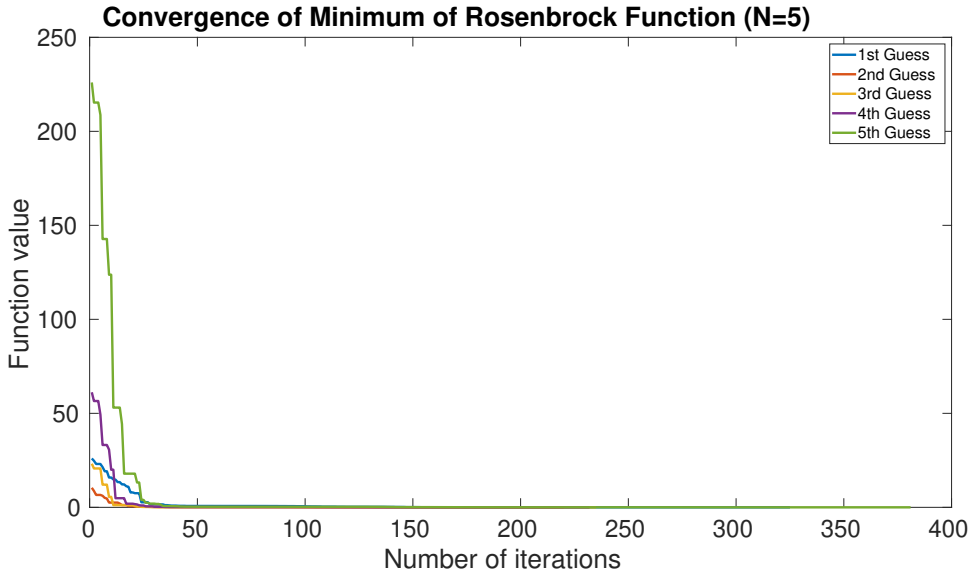


Figure 7: Rosenbrock function value versus the number of iterations (N=5)

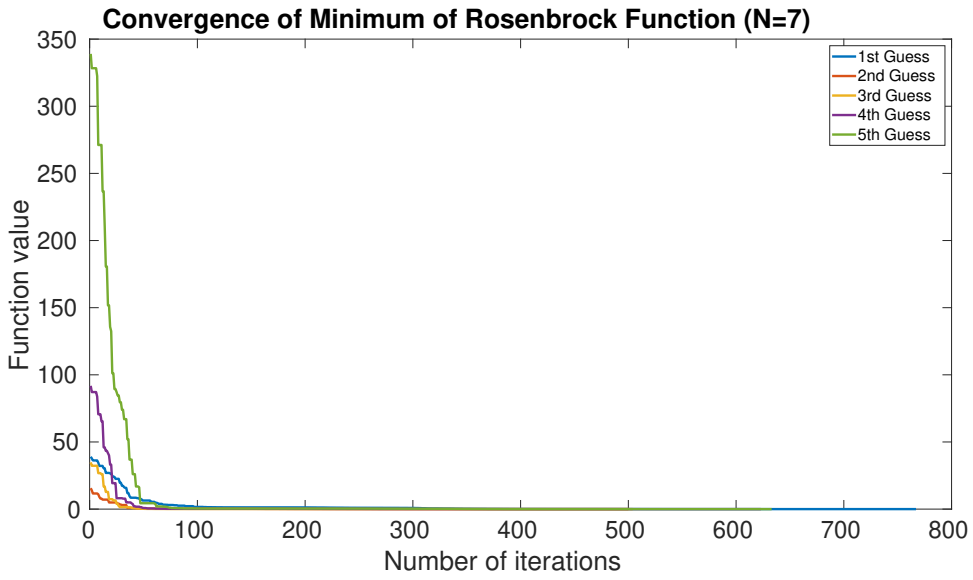


Figure 8: Rosenbrock function value versus the number of iterations (N=7)

5.2 (b)

The code of this part could be seen in **Q5b.m**. When $N = 50$, use the build-in function **fminunc**, and set the 'Algorithm' as 'quasi-newton', 'UseParallel' as true and 'SpecifyObjectiveGradient' as true. The initial guess is $\hat{x} = (5, 5, 5, ...5)$. It takes 144 iterations to converge. The minimum value is $3.81e-06$. If it does not plot the value of function, the simulation time is about 0.17 s, which is sufficiently small.

The objective function value versus the number of iterations are shown in Figure 9.

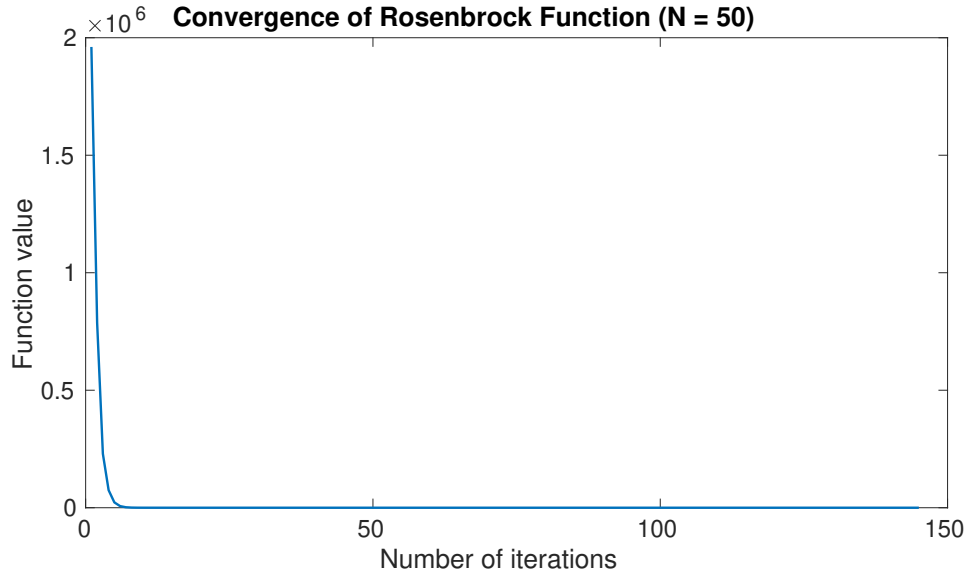


Figure 9: Rosenbrock function value versus the number of iterations (N=50)