Report for Lab 1

Introduction:

This lab focus at enhancing our understanding of Open_MP. In this lab assignment, the heated_plate.cpp code, which simulates the temperature change of a heated plate, has been modified. Open_MP was used to add the parallel feature into the code, especially on the for loops of the code. The code was then tested with different number of threads. For detailed documentation of this program and the method to build execution file, see README.pdf.

Methodology:

This modified program was runned three times with the thread number to be 1, 2 and 4, and tolerance set to be 0.001. The program was runned on linux virtual machine installed on personal computer with 2 intel cpu cores. The virtual machine contain 2 processors and can run maximum 2 threads simultaneously(the threads number is limited by the setting of virtual machine).

Results:

Running the program for number of threads to be 1, 2 and 4 generates the following result in Table 1:

Number of Threads:	1	2	4
Elapsed wall clock times(s):	1.9358	1.28991	1.73073

Table 1: the elapsed wall clock time with number of threads to be 1, 2 and 4.

Discussion:

From this results, we can see that increasing the thread number from 1 to 2 can significantly reduce the elapsed wall clock times, together with the fact that the program in both case can produce accurate result(see README.pdf), we can concluded that the modification of the code has successfully made the code paralleled. The parallel of the code can indeed enhance the performance when there are two treads avaliable. But doubled the thread numbers cannot reduce the execution time to half, since the program also contains parts which are not supposed to be paralleled and the threads synchronization also consumed some of the computational resource.

Increasing the thread number from 2 to 4, however, will not reduce the elapsed wall clock time of program execution, because the virtual machine can only support 2 threads to be running in processor simultaneously. The generating and synchronizing of extra threads will be resource consuming in this case and slow down the program. The same situations were found in other exercises of this lab(which is stored inside code&documentation_for_exercise).