

AE3-422 High Performance Computing

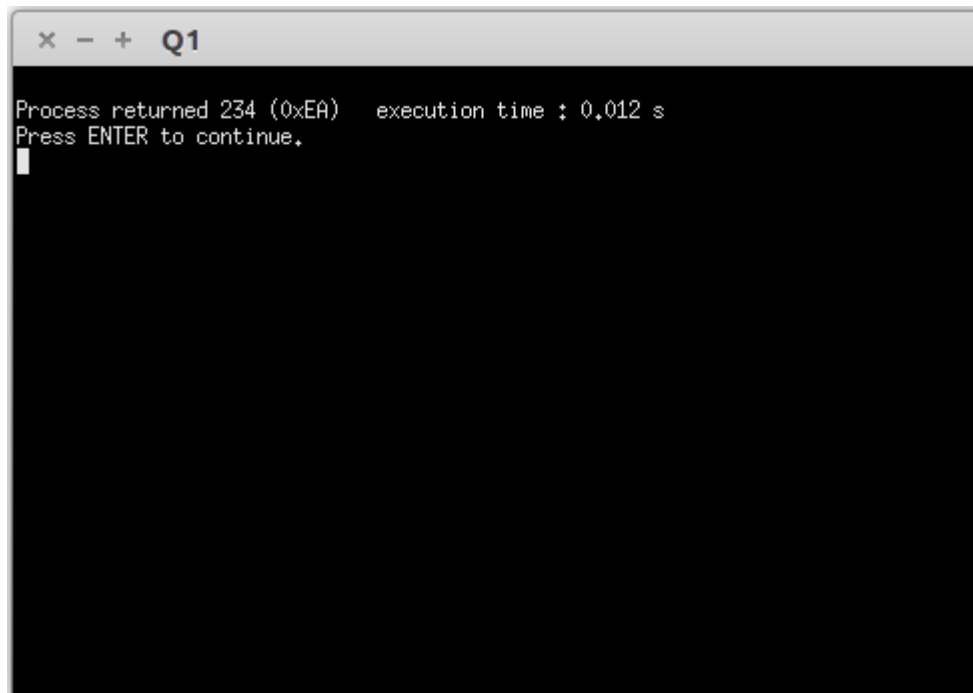
HONG YON KIM

CID: 00652661

Q1

The code is constructed using TriMatrix code from example 28 of C++ course [1]. The project is composed of three different files which are 'main.cpp', 'TriMatrix.cpp' and 'TriMatrix.h'. The class 'TriMatrix' only contains functions which are used to solve the heat equation. 'TriMatrix::TriMatrix' constructs diagonal matrix which is shown in [2] by taking v and the number of the discretisation of the domain (Ω). However, it does not store whole matrix but stores the diagonal part of the matrix in three arrays as the remaining part of the matrix have the value of 0. This function is also overloaded to output the value of the matrix at a specific position when two integers i (row) and j (column) are given as inputs. Finally 'TriMatrix::matrixMultiplication' calculates the next time step of vector U when the current time step U with boundary conditions are inputted.

Function 'TriMatrix::matrixMultiplication' is used with for loop in 'main.cpp' file. It turns out that the solution decays to less than 0.000001 after 234 iterations as shown in Figure 1.



```
x - + Q1
Process returned 234 (0xEA)   execution time : 0.012 s
Press ENTER to continue.
█
```

Figure 1: The Returned Integer After The Iteration

Q2

Based on the codes from Q1, C++ codes takes arguments (L the length of the domain, N_x the number of discretisation, T the target time for the solution, N_t the number of time steps, α the diffusivity coefficient) then outputs midpoint values ($U_{N_x/2}$) at each time step (t_n), the size of time step (dt), the size of discretised domain (dx) and Root Mean Square Error ($rmse$) in text files. $U_x = \sin\left(\frac{\pi x}{L}\right)$ is chosen for initial condition where the analytic solution of the heat equation is found to be $U_{x,t} = \sin\left(\frac{\pi x}{L}\right) \exp\left(-\frac{\alpha \pi^2}{L^2} t\right)$.

There are additional two python codes ('Q2.py' and 'Q2_2.py') that run C++ files then plot graphs using data in text files. 'Q2.py' code plots midpoint values against the corresponding time and 'Q2_2.py' plots $rmse$ against dt and dx from gathered data using randomly inputted arguments. 'text_file.read()' functions are used to read data, however, the numbers are stored in string format. Thus, 'yaml' module is used to convert string type data to float type data.

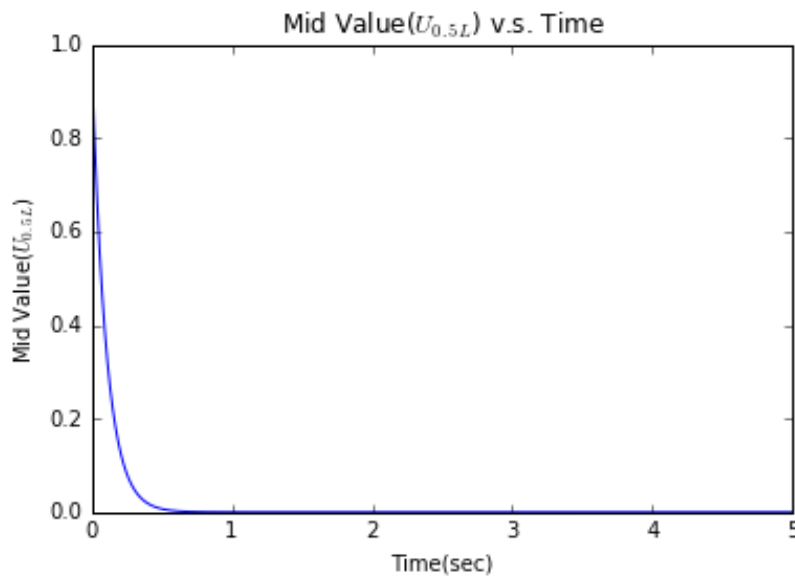


Figure 2: Graph Plotted using 'Q2.py'

From the Figure above, it can be seen that the midpoint value decays exponentially as time increases. Python codes. 'Subprocess.call' function successfully compiles then runs C++ codes, however, there were difficulty in finding a method to input the arguments to C++ codes while running. Also, using the 'Subprocess.Popen(cmd arguments, stdin=Subprocess.PIPE)' function to invoke the C++ program seems to be successful but using the program.communicate() to inject the inputs calculated within the Python program has difficulties due to type compatibility issues. (The code base is commented out for compilation purposes)

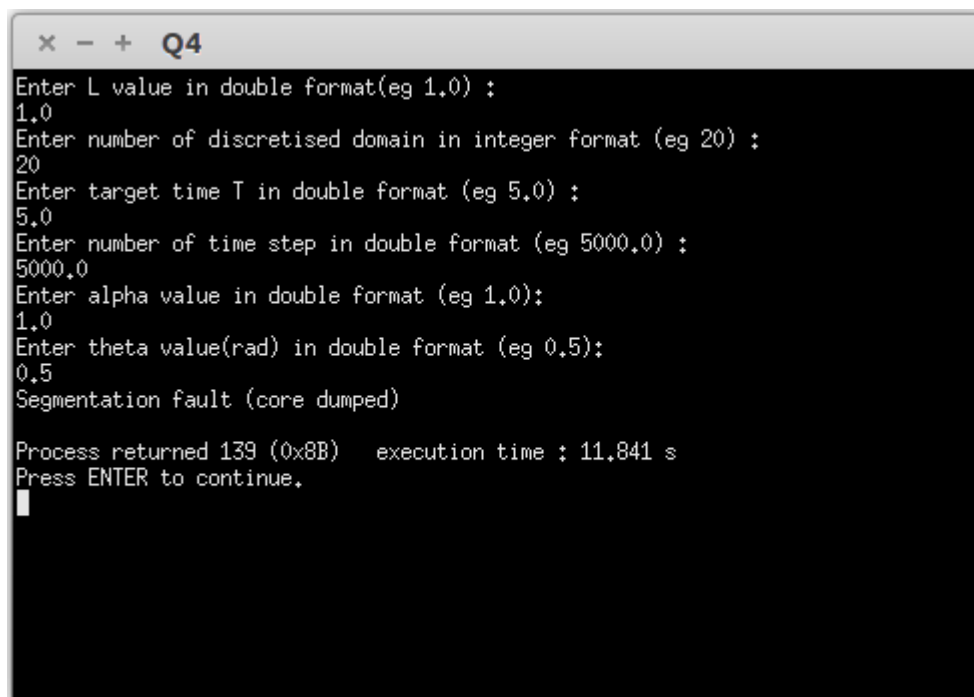
Q3

The codes are manipulated to solve the heat equation by using different time integration technique. 'TriMatrix::Trimatrix' function in 'TriMatrix.cpp' file now takes additional argument θ then generates two matrices $(I - \theta \nu L)$ and $(I + (1 - \theta) \nu L)$ on each side of the equation $(I - \theta \nu L)U^{k+1} = (I + (1 - \theta) \nu L)U^k$ in the same method used in previous questions. However, back substitution is performed in 'TriMatrix::matrixMultiplication' function in order to calculate the next time step solution.

Using the same structure from 'Q2_2.py', 'Q3.py' code is constructed to plot *rmse* against *dt* and *dx*. However, also the issue previously mentioned occurred in the code.

Q4

LAPACK and BLAS libraries are used to compute the same process in Q3. In 'TriMatrix::TriMatix', 'dgetrf' function is used for LU decomposition of the left hand side of the matrix for faster calculation. Full matrices are constructed in order to use the libraries as there have been difficulties of finding banded matrix array format. 'TriMatrix::matrixMultiplication' function is used with while loop to carry out the solution at the targeted time. After using 'dgemv' to calculate right hand side of the equation $((I + (1 - \theta) \nu L)U^k)$, the solution of each time step is meant to be obtained by 'dgtrs' function.



```
Q4
Enter L value in double format(eg 1.0) :
1.0
Enter number of discretised domain in integer format (eg 20) :
20
Enter target time T in double format (eg 5.0) :
5.0
Enter number of time step in double format (eg 5000.0) :
5000.0
Enter alpha value in double format (eg 1.0):
1.0
Enter theta value(rad) in double format (eg 0.5):
0.5
Segmentation fault (core dumped)

Process returned 139 (0xBB)   execution time : 11.841 s
Press ENTER to continue.
```

Figure 3: Error from Terminal, Q4

The code successfully compiles, however, error 'Segmentation fault (core dumped)' message returns as shown in Figure 3. Debugging the code have been difficult, nonetheless, it is thought that the error is due to the wrong usage of pointer.

Q5

Parallel computing technique is used to solve the same equation in Q3 and Q4. The exactly same method of Q4, BLAS and LAPACK library, is used to perform the calculations after number of attempt to use SCALAPACK library. Additionally to the codes from Q4, the domain is broken down to half, as shown in Figure 4, to process the calculation in two processors. Furthermore, some values (e.g. u_{n-1} and u_{n+1} of Figure 4) needed to be exchanged between the processors as the solutions of each time step (u_i^{n+1}) depend on data of previous time step (u_{i-1}^n , u_i^n and u_{i+1}^n) as shown in right side of Figure 4. Therefore, all functions in 'TriMatrix.cpp' file are designed to calculate solutions using two processors while communicating effectively in 'main.cpp'.

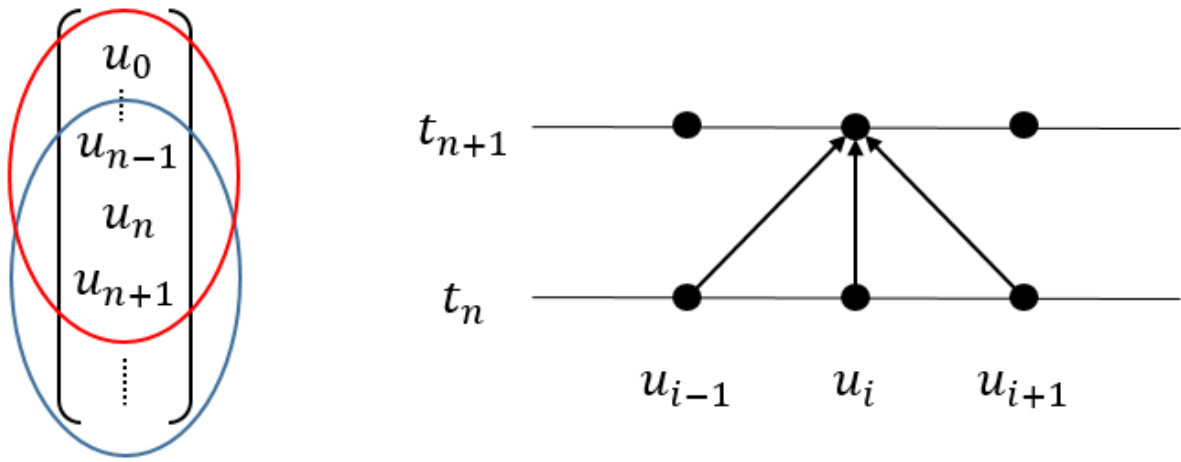


Figure 4: Separated Domain and Dependence to the Previous Time Step of the Solution

However, 'Segmentation fault (11)' error appears when the compiled code is ran. It has been failed to debug the code but the wrong usage of pointer is suspected for the reason of the error. It is thought that the performance is going to be faster compared to the codes from Q4 and Q5 once the codes are debugged.

Reference

- [1] Chris Cantwell, Exercises Part 2, Imperial College London, 2016
- [2] Chris Cantwell and Omar Rodrigo Bacarreza Nogales, AE3-422 High-performance Computing, Imperial College London, 2016