Gebze Technical University
Computer Engineering

CSE 244
2017 Spring

MID-TERM PROJECT REPORT

JAMES JOSHUA MSUYA
141044093

Part 1.
Efficiency.

In this project, as far as client server logic is concerned efficiency is a very important
This is a case study of a 6x6 matrix. So, in calculations involving matrices the general method which are very accurate are very inefficiency for large dimensional matrices. So, in general the approach was to optimize the code so that it can be as efficient and as accurate for any dimensional matrix. In calculating determinant rather than the brute force method which has a time complexity of $O(n^4)$ a Gaussian approach was used which has a time complexity of $O(n^2)$. So, it's closer to accurate but very efficient.

In generating random matrices, a random function of time in second was used. For successful calls of the matrix generator function if the time difference is almost negligible it ends up producing the same matrix. Because the idea of random is not random at all. It's a function of parameter which changes, so if no changes were noticed then the random stops being random and yields the same results.

For the inverse of a matrix the same concept was used more fast method was used and was tested for larger dimension. The only limitation to that method is that it involves division so any time a number which happen to be zero was replaces by a smaller number which is not zero. So in those cases it removes the possibility of having undefined matrix i.e matrix which has -inf or nan as its element. This for the case when you have a matrix which has a determinant not equal to zero but its quarter has a zero determinant. So, in those cases an approximate inverse was calculated. Also, the idea of generating quarter by quarter of the entire matrix was thought also and in terms of efficiency was not that much effective compared to finding an approximate inverse when some irregularity occurs.

Case study 6x6 matrix

5.000 4.000 2.000 3.000 7.000 1.000
5.000 1.000 4.000 3.000 2.000 3.000
4.000 6.000 1.000 9.000 5.000 9.000
5.000 3.000 4.000 9.000 2.000 6.000
2.000 4.000 7.000 7.000 5.000 4.000
3.000 7.000 5.000 2.000 9.000 1.000

<u>determinant -11200.000000</u>

The inverse of the original matrix

1.211 -0.421 -0.737 0.500 -9.667 3.167
-0.579 0.158 0.526 -0.000 3.000 -1.000
-1.368 0.737 0.789 -0.500 8.000 -2.500
0.279 -0.125 -0.048 -0.756 1.268 -0.537
-0.106 -0.125 0.260 0.024 -0.073 0.146
-0.019 0.250 -0.135 1.293 -1.878 0.756

The determinant -5.651147e-02

2D convoluted matrix with a 3x3 kernel matrix

5.000 4.000 2.000 3.000 7.000 1.000

5.000 1.000 4.000 3.000 2.000 3.000

4.000 6.000 1.000 9.000 5.000 9.000

5.000 3.000 4.000 9.000 2.000 6.000

2.000 4.000 7.000 7.000 5.000 4.000

3.000 7.000 5.000 2.000 9.000 1.000

determinant -11200.000000

In this project the concept of convolution was implemented using a kernel matrix with all elements 0 except the middle element. So successful sum of the neighbors product overlapping with its matrices elements resulted in the same value as the convoluted element at the mean time. The convolution method was of time Complexity $O(n^4)$. Because the basis of the project was to establish a stable server client and inter-process communication the more efficient method of calculating convolution of the matrix which requires some more advanced mathematical skills was not implemented.

Kernel matrix
0 0 0
0 1 0
0 0 0

Part 2.
Design

The design in this project included some global variables for the purpose of having a reach throughout the scope of the program. For the aim of releasing resources when the signal to kill is caught they were made to be available in global scope. Essentials.c and essentials.h containing all the common functions in the project. This project was compiled in standard 99 just to avoid not compiling in a more rigid system. In the server side the task of notifying the parent after some interval of time was delegated to the child process. It sends a signal to parent after a specified time and the parent handles all the signals that were coming to him. The parent process records all the signals that came to him and wait for the child process to notify him it’s time to handle them and send them matrices.

Conclusion.
The project keeps all the necessary logs and displays all the necessary information.