**CS 590 - Parallel and Distributed Computing**

**Assignment #5**

**LU Decompositin**

Submitted by: Zainab Altaweel

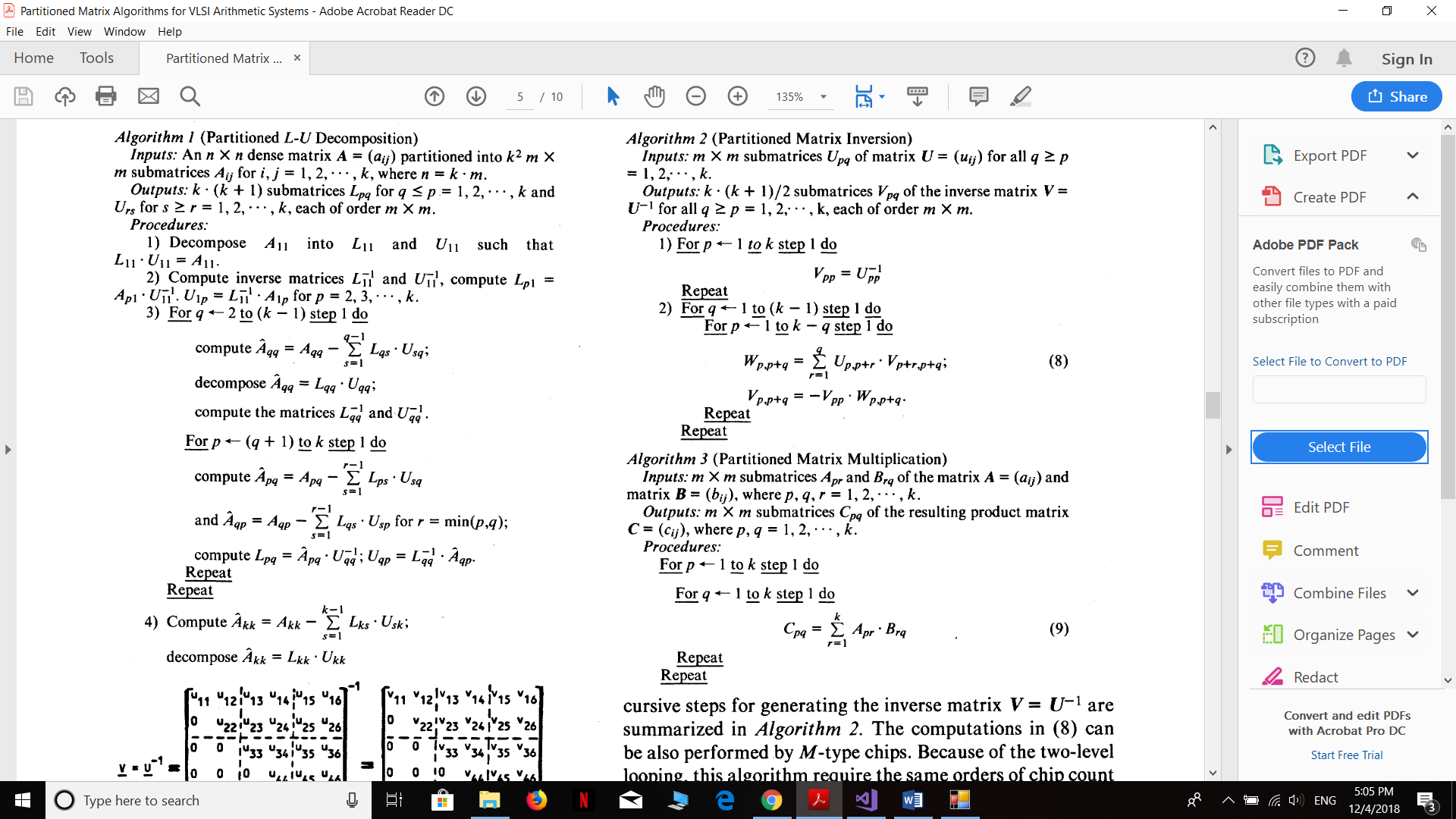
In order to I implemented block LU decomposition as sequential code, depending on the algorithm described by Kai Hwang and Yeng-Heng Cheng (Partitioned Matrix Algorithms for VLSI Arithmetic Systems).

The user can specify matrix size and block size as well, and the result was verified.

The user interface is shown in figure 1, two text boxes were added to enter the array and block size, a button was added to compute LU decomposition sequentially.

three methods were created, the first one is to process the first upper left block and the first row and column. The second method is to compute matrix diagonal, and the last one is to compute the other matrix blocks.

The program was written according to the following algorithm:



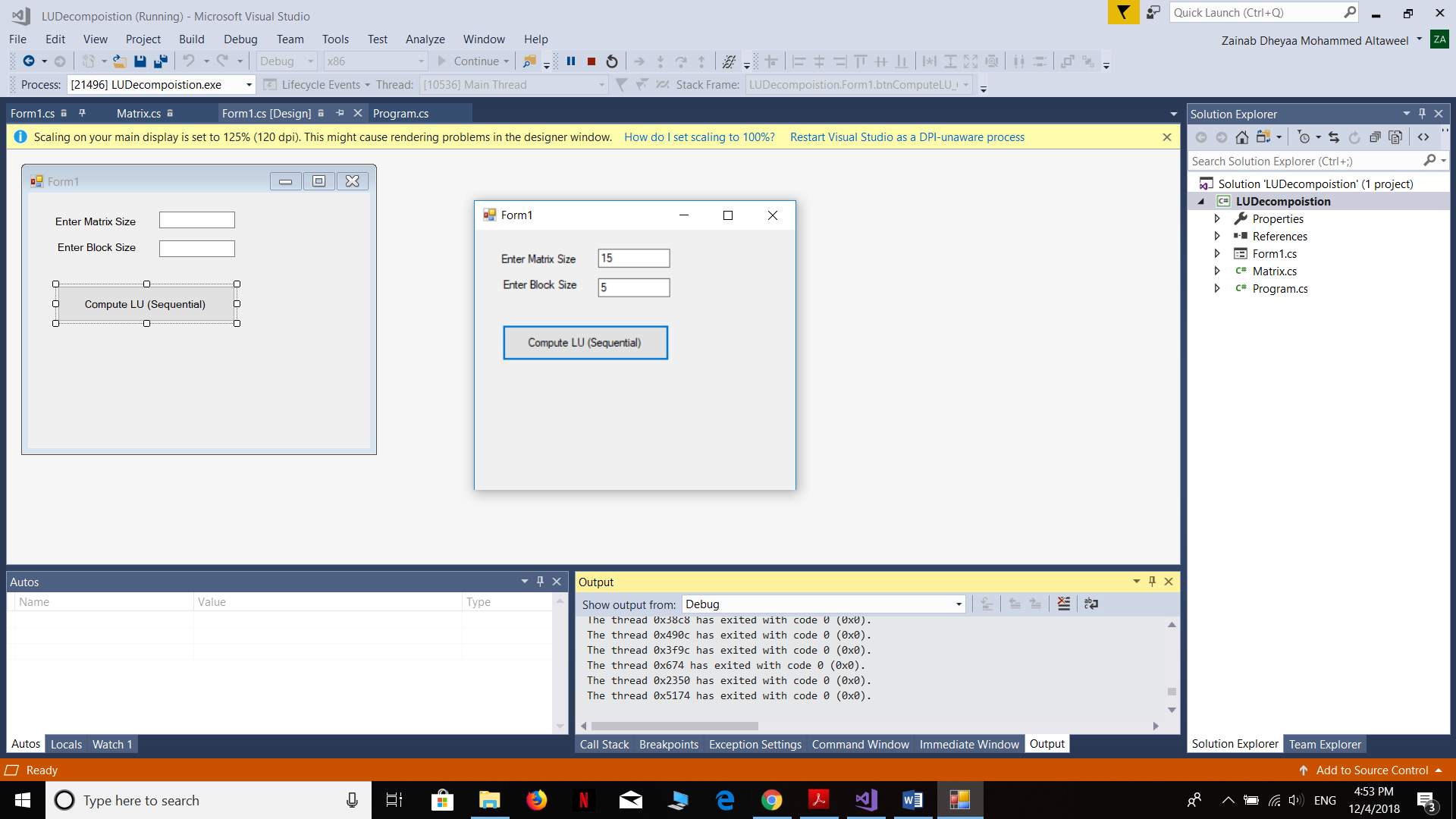


Figure 1 User interface

Then the program was executed and the time was calculated, this code takes 6003 ms as shown in figure 2. The results are shown in figure 3, 4 and 5.

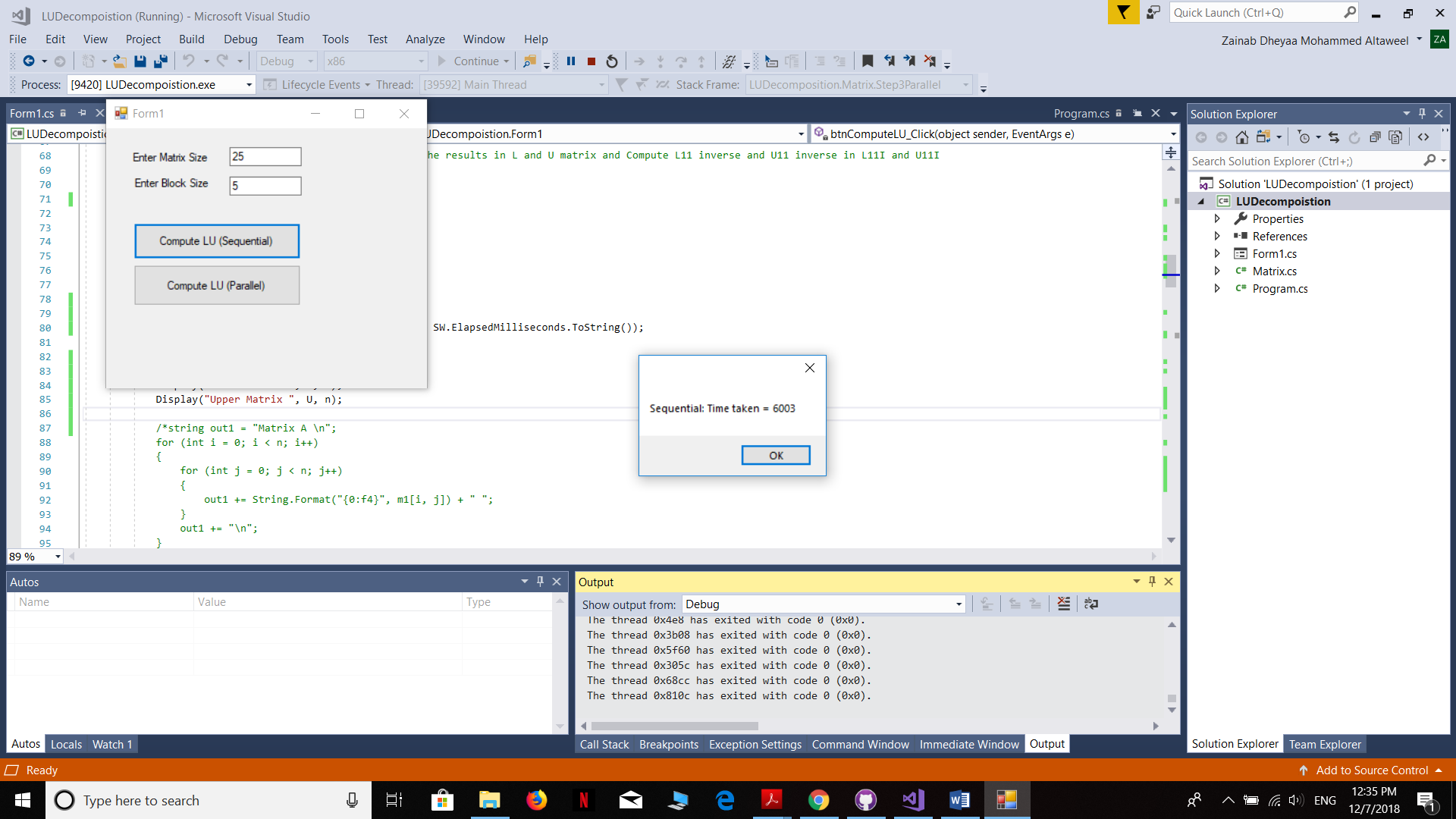


Figure 2 Time taken to execute sequential LU Decomposition

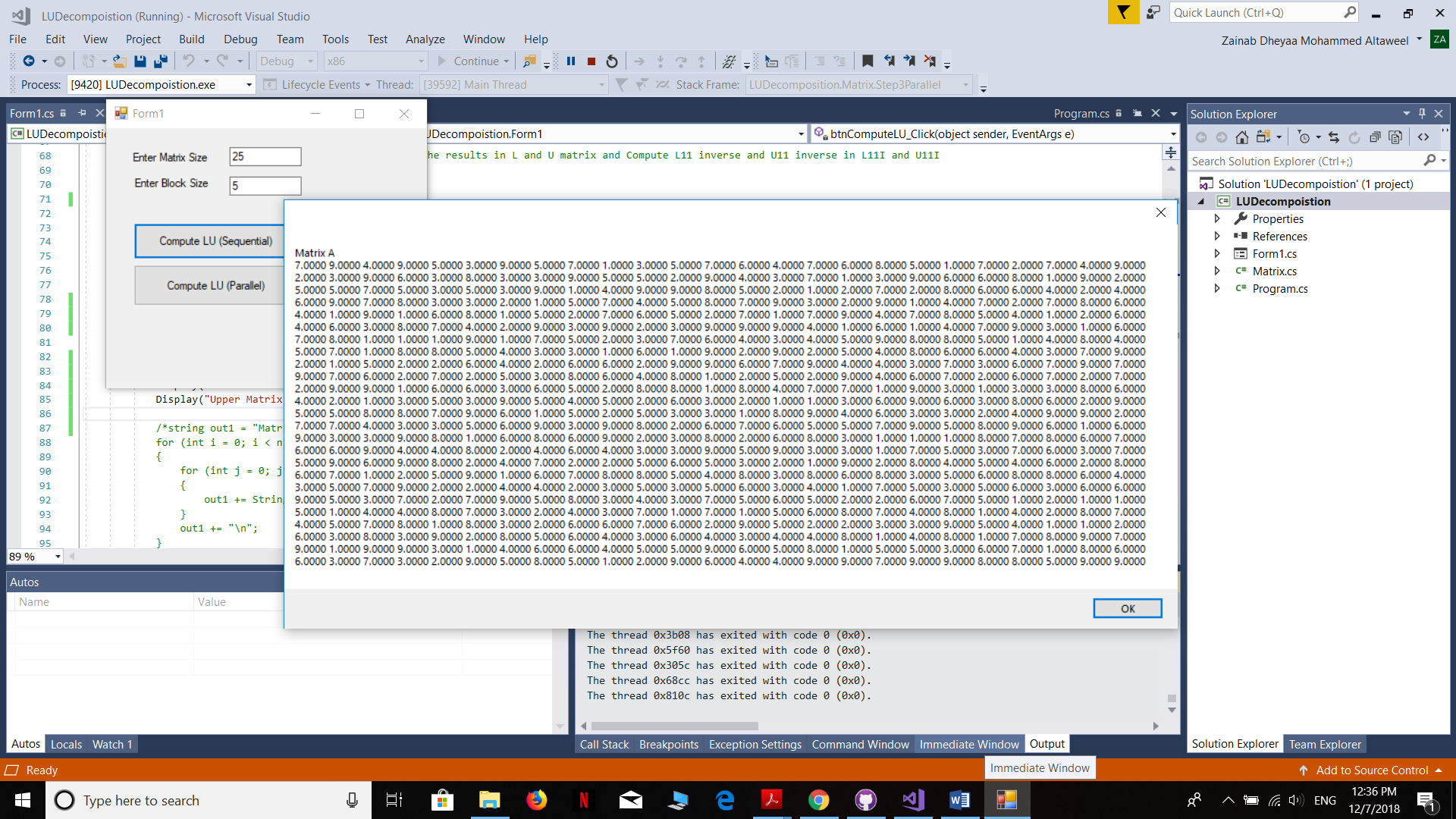


Figure 3 The original matrix

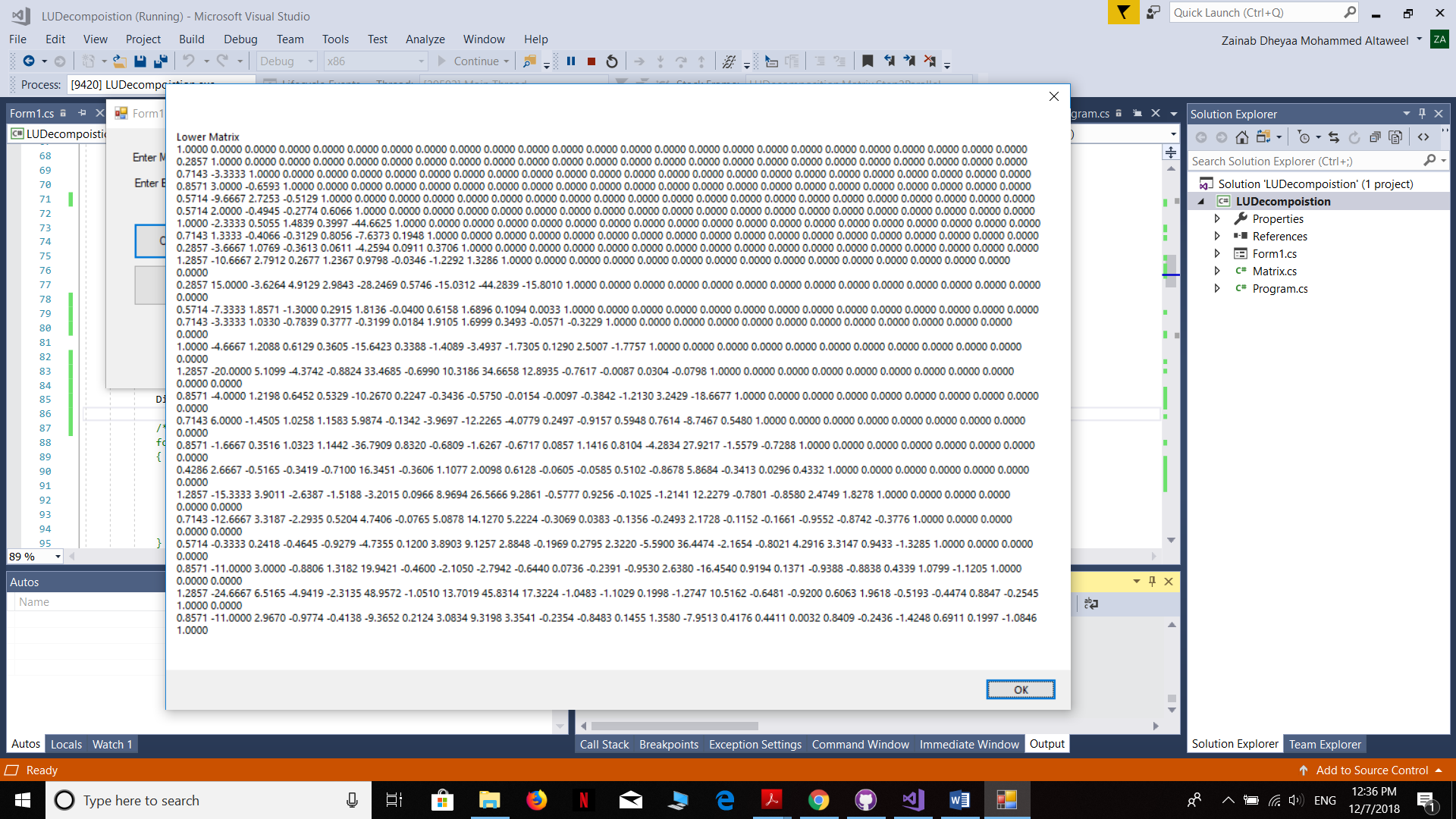


Figure 4 The lower composition matrix

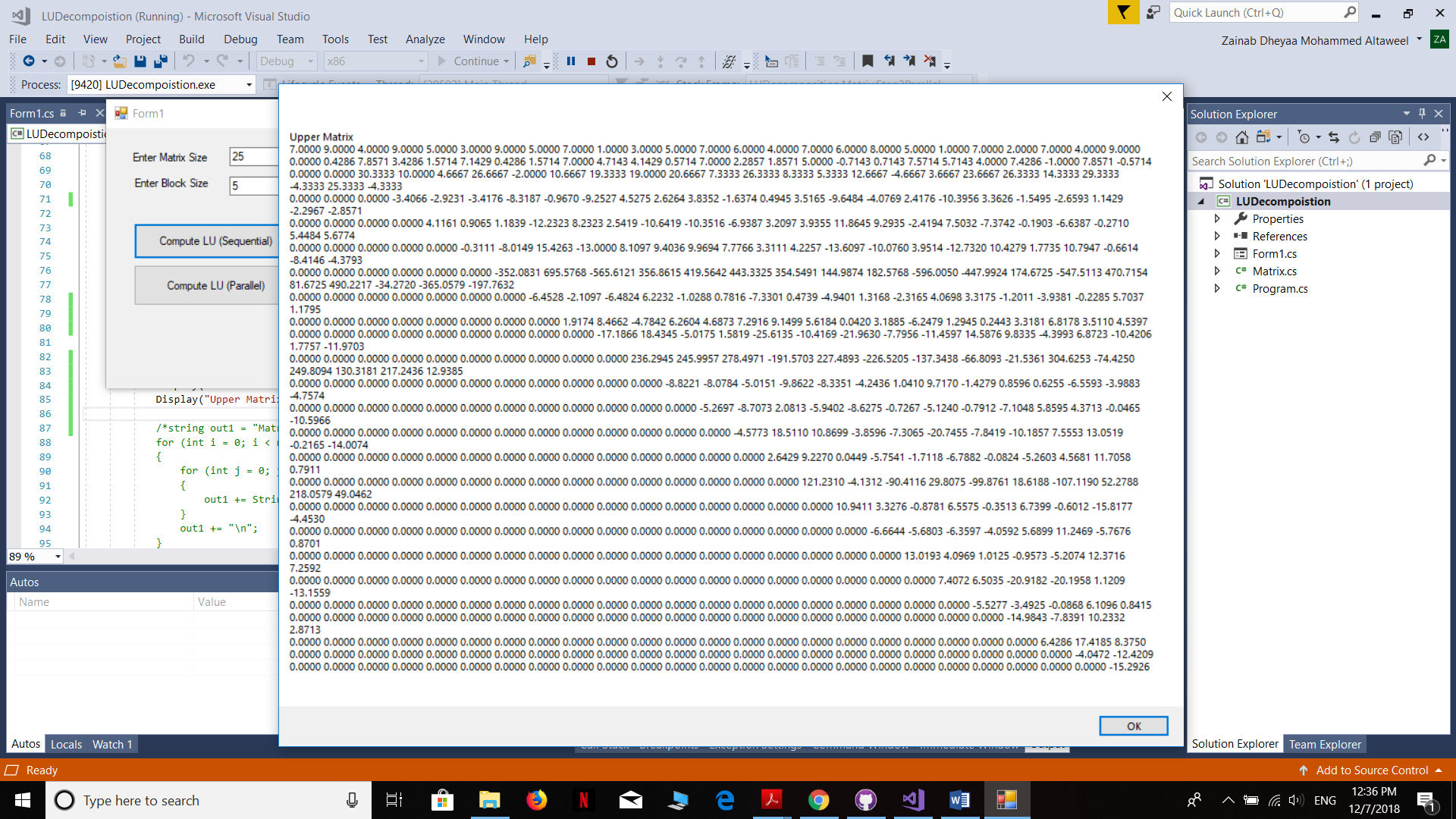


Figure 5 The lower composition of the matrix

In order to parallelize the code, TPL was used. A new button was added, which is used to trigger the parallelized code. The first parallelization technique used is Parallel.For which is used to process the blocks in the first row and the blocks in the first column in parallel, since they depend only on their data and the inverse lower and upper composition of the first block. Additionally, Parallel.Invoke was used to process both the first row and column together.

Parallelization was also implemented to process the blocks in rows and columns after processing the diagonal blocks. The time was calculated and it was 21141 ms as shown in figure 6.

Since I needed to store the blocks matrix in a list so that each block is stored in a different variable, in order to enable parallelization, so this process takes time.

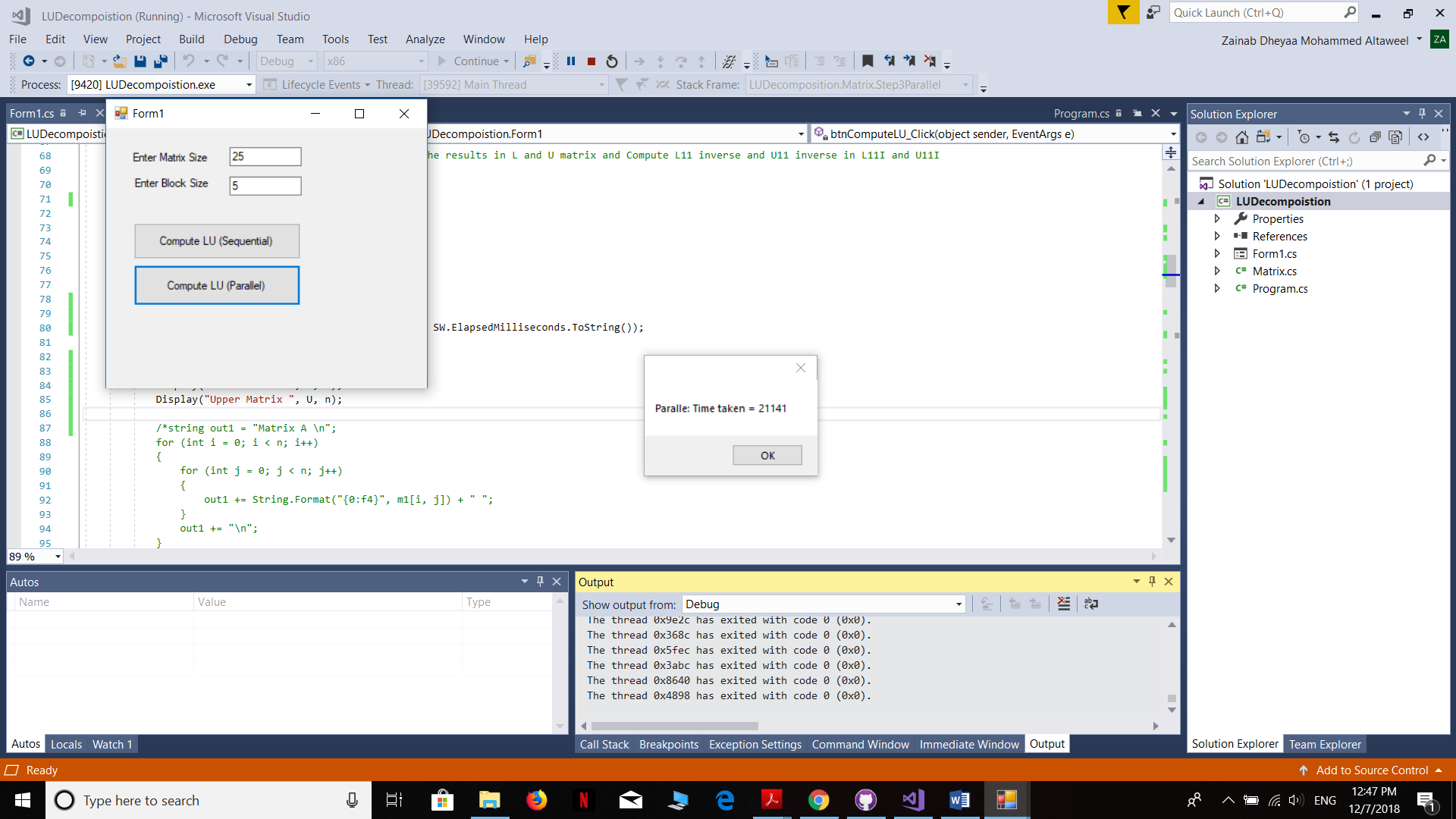


Figure 6 Time taken to execute parallel LU Decomposition

To implement the code as asynchronous algorithm, a 6\*6 matrix was created which consists of 9 blocks each one of size 2\*2. An array of semaphores was created to signal the processing in each block, once the first block is done it will signal the second, third fourth and fifth blocks to start processing, since they depend on the data acquired from the first block, and so on for the next blocks. This will reduce the execution time as shown in figures below, the time required for the computation using semaphore is 258 ms (figure 7) while the time required for sequential computation is 663 ms (figure 8).

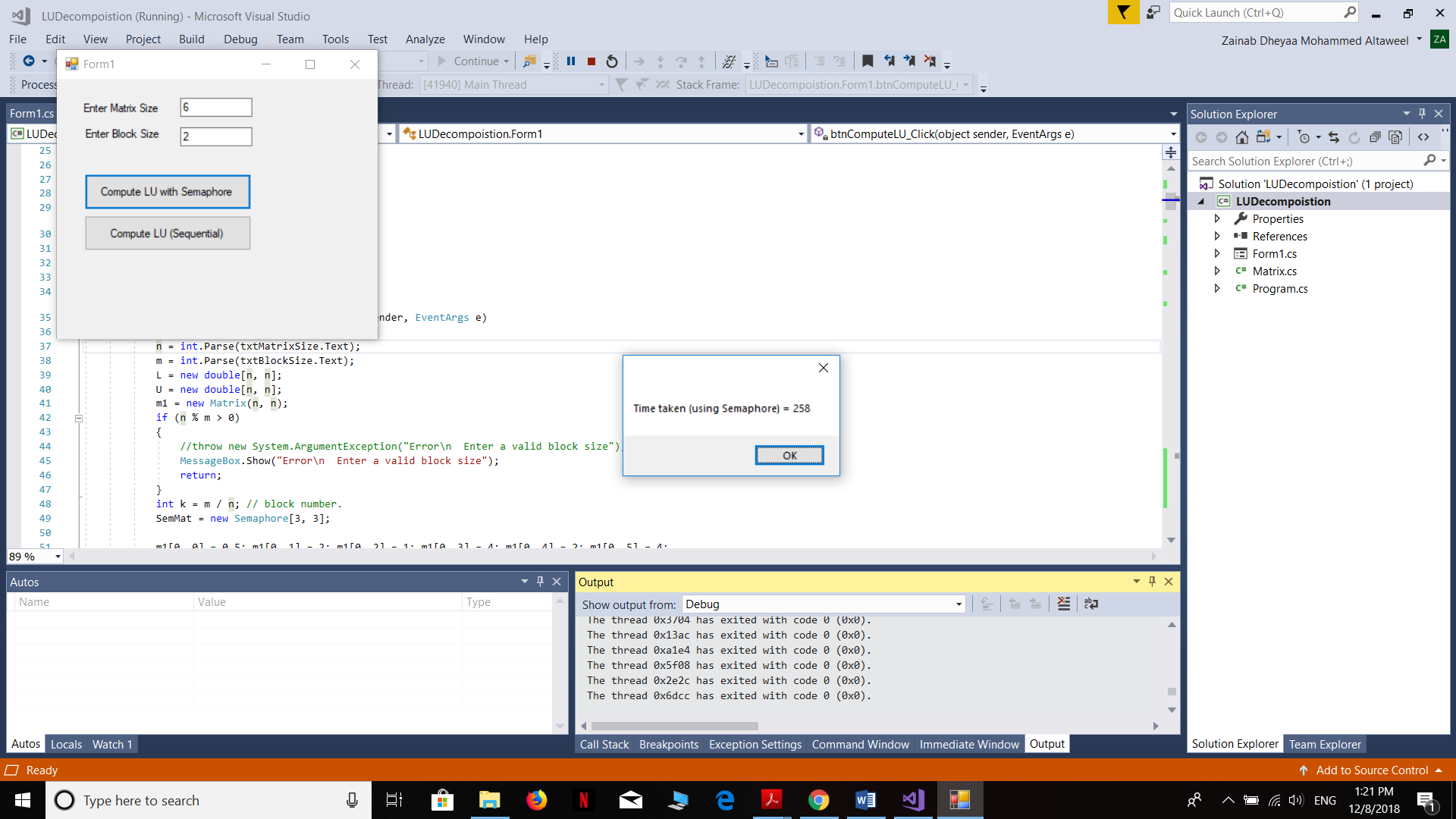


Figure 7 time taken for LU Decomposition using Semaphore

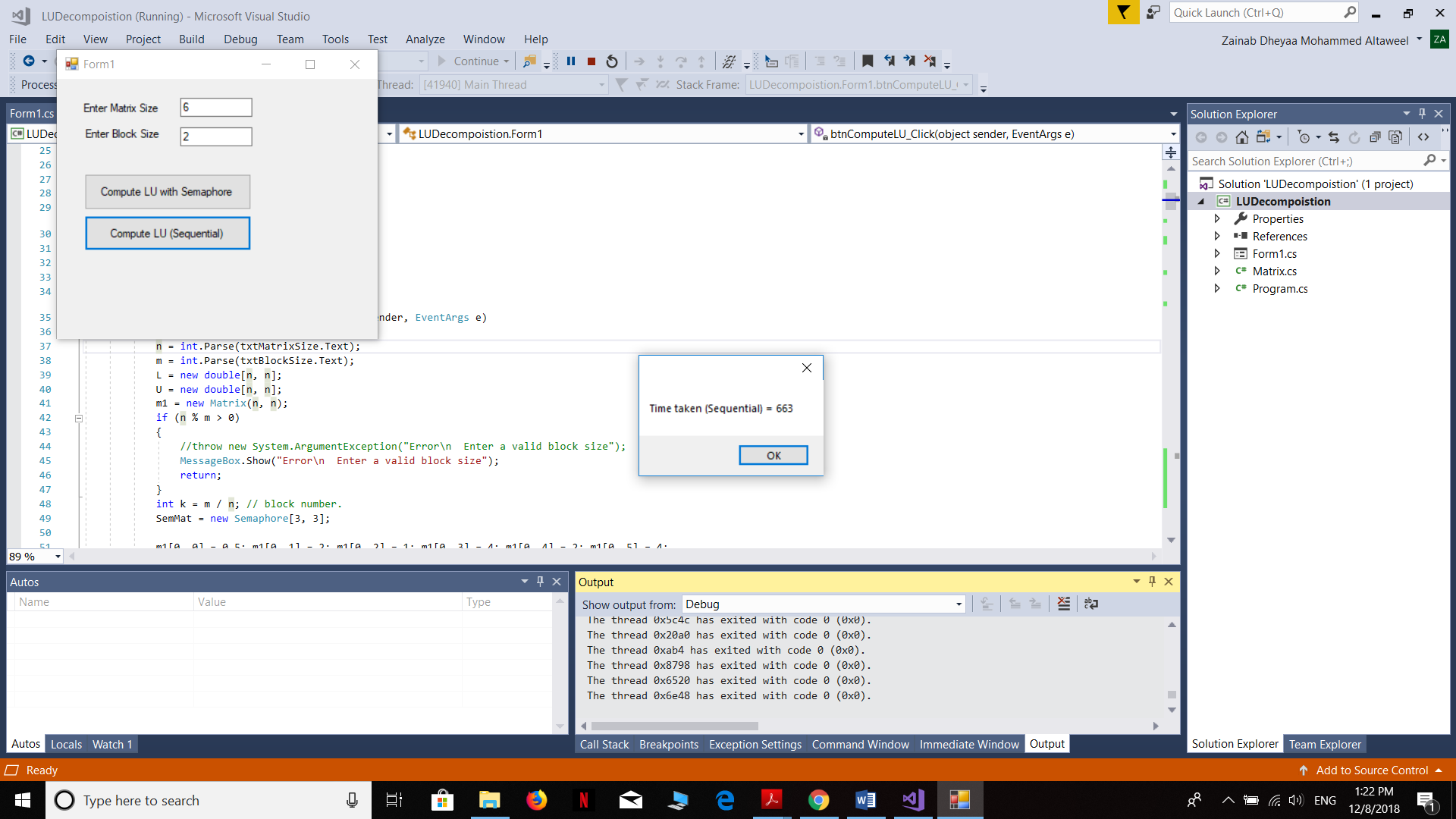


Figure 7 time taken for LU Decomposition using Semaphore