This project was good practice to learn how to parallelize matrix algorithms. The most important part is the realization of the fact that it is not necessary to parallelize the complete algorithm everytime. It is important to identify the parts of algorithm that require most computation time and the parallelization will be worth the overhead. For example, in this project the part involving reducing the reduction of matrix to upper triangular was the most expensive so the overall time performance improved significantly by parallelzing that part. Whereas the other parts of the algorithm like back-substitution require the value from previous step to go ahead which means it is inherently sequential in that regard. However, it can be parallelized within the calculation of a single row but that requires cumbersome implementation and significant overhead and does not lead to significant performance gains.

Another important observation from this project is that the factors that limit the parallel speedup. One such factor is hardware limitation, with same implementation and number of threads the run time on dual core machine increases substantially as compared to quad core machine when the input size grows bigger. Second factor is overheads introduced by threading, for example one such overhead in this case after each step of forward elimination you have to wait for all the threads to finish so you can perform next pivoting step so it is very important that load is roughly equally balanced among all the threads to avoid bottle-necks. Another such overhead is maintaining the integrity of shared memory, for example in this implementation every time one thread is working with current_row variable (to decide which row to work on) it has to lock that resource so that other threads don't update it while it's working on it.

Overall, threadingdoes indeed give significant performance gains as shown in the plots and run-time results but it requires more careful implementation from developers and also hardware support to enhance those gains.