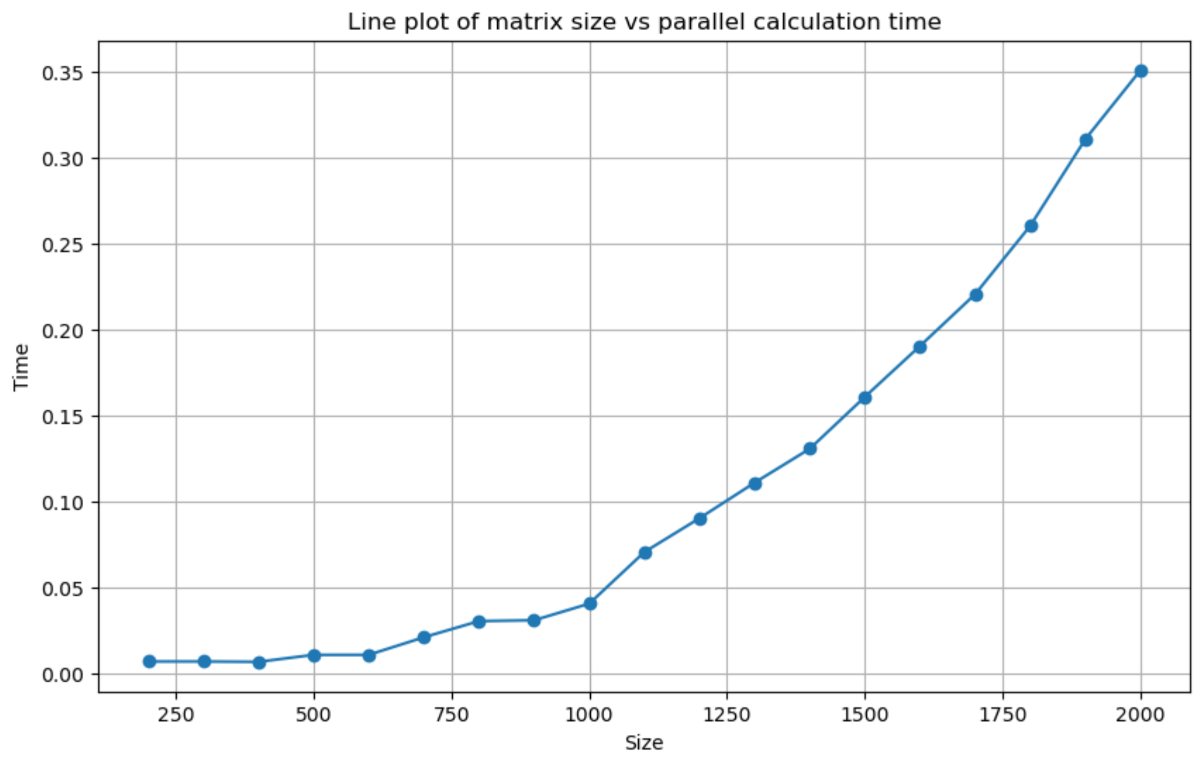
Analysis

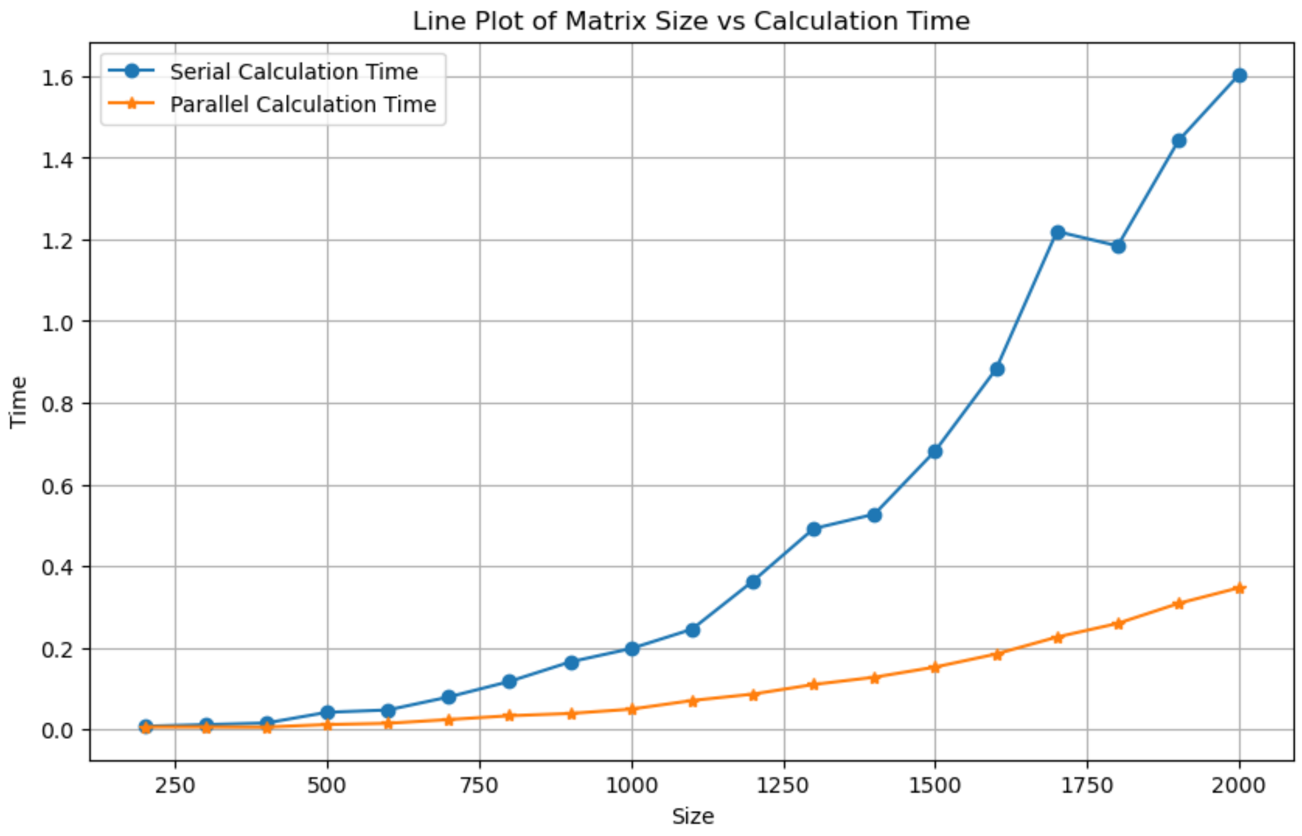
The dependence of the execution time of the program on the matrix size n



Examining the graph, we can easily see that as the matrix size *n*, grows, so does the time required for computation. This trend is expected since larger matrices demand more processing power. We can also see that the increase in time is non-linear, resembling a curve, possibly polynomial or exponential, as *n* increases linearly.

However, the graph does not provide a comprehensive view of the efficiency of the parallel method. To explore of the performance further, I will discuss the efficiency in relation to the serial method.

The speedup over a serial counterpart of the program



Looking at the graph, we can clearly see that parallel computation has a substantial advantage, particularly when it comes to larger matrices. As the size of the matrix increases, the line for serial computation begins to rise steeply, in contrast to the more gradual ascent of the parallel computation line. This marked difference demonstrates the superior efficiency of parallel processing for handling complex tasks involving large datasets. In fact, all but matrix n=100 had a speedup in performance.

The reason for the parallel method having a more consistent increase in time lies in its approach to splitting the matrix into smaller sections, which are then processed concurrently. This effectively utilises the multi-core design of contemporary processors, distributing the computational load across several threads. In contrast, the serial method, which handles the entire matrix in a singular sequence, finds itself increasingly challenged by larger matrices, leading to a much sharper increase in computation time.

It is worth noting from the values in the data_q2.csv that the increase in performance was a not straight line, with some matrix sizes having a greater speedup than the next matrix size, such as n=1900 vs n=2000, implying some sizes fit better with the hardware of my processor. That being said there was still a great improvement with the parallel method.

This graph provides a good illustration of the strengths of parallel computing, especially for demanding computational tasks. It demonstrates that as the size of the data grows, parallel processing can significantly mitigate the impact on computation time, making it a more efficient choice for intensive calculations.