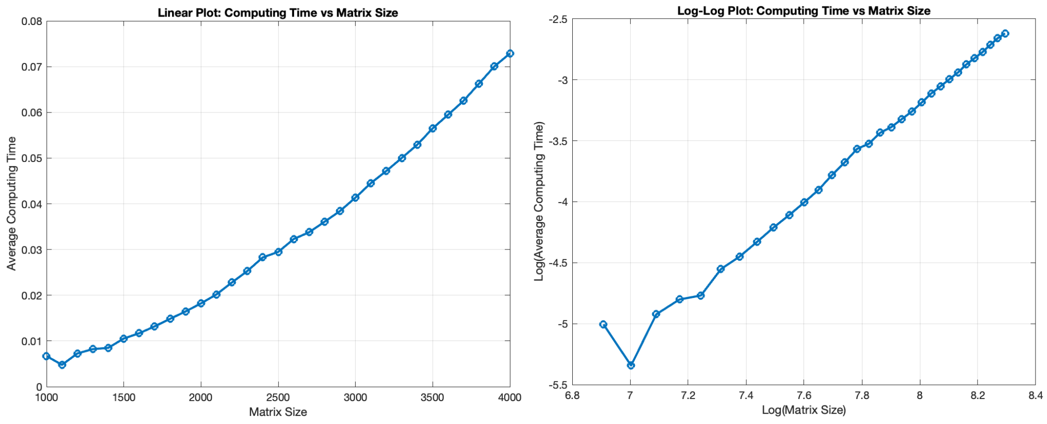
Computing *Assignment2*

a. To describe the relationship between n & t i.e. the array size and time required to find the inverse of the matrix, I plotted the graph of t v/s n and determined it's slope using the graph of $\log t$ v/s $\log n$
I used the Log scale graph because-

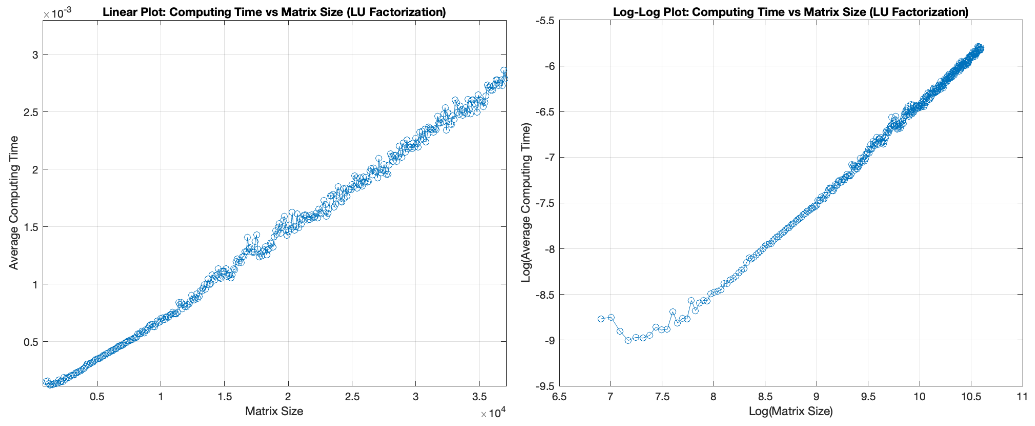
$$y = mx + c(\text{equation of line}) \Rightarrow \log y = m \log x + c \Rightarrow \log y = \log x^m + c \Rightarrow \log \frac{y}{x^m} = c \Rightarrow 10^c = \frac{y}{x^m} \Rightarrow y = 10^c x^m \Rightarrow y = dx^m$$

This analysis demonstrates that the slope of the logarithmic scale plot offers a reliable approximation of the polynomial degree depicted in our original graph.



In MATLAB, I found that the slope of the graph is ≈ 2 which provides an insight onto the relationship between the array size & time taken to find the inverse of the matrix. Getting a slope of 2 suggests that it takes $O(n^2)$ operations to find the inverse of the sparse tridiagnol matrix. This is happening because MATLAB's inverse method method uses LU decomposition which takes $O(n^2)$ steps to solve the system due to which, our slope is coming out to be 2.

b. I plotted the graph of t v/s n and determined it's slope using the graphs of $\log t$ v/s $\log n$.



In MATLAB, I found that the slope of the graph is ≈ 1 which provides an insight onto the relationship between the array size & the time taken to decompose the matrix. Getting a slope of 1 suggests that it takes $O(n)$ operations to decompose the sparse tri-diagnol matrix. Also, decomposing a sparse tridiagnol matrix takes $O(n)$ steps because MATLAB makes sure to use efficient algorithms when the matrix is sparse tridiagnol otherwise, it may take considerably more steps. We may ignore the first few values since they are not inline with the graph.

c. To assess whether the solutions derived from two different methods align, we generate two sets of solutions labeled as x_inv and x_lu respectively signifying solutions obtained from inverse and LU decomposition. Their equality should result in x_inv - x_lu = 0. However, in computational practice, we notice an error of order $1e - 10$ which is not very big that we doubt their equality. This is happening because, the LU decomposition method involves $O(n)$ operations, whereas inverse method requires $O(n^2)$ operations at most. The additional operations involved in the inverse method causes **Roundoff errors** which leads to small errors.