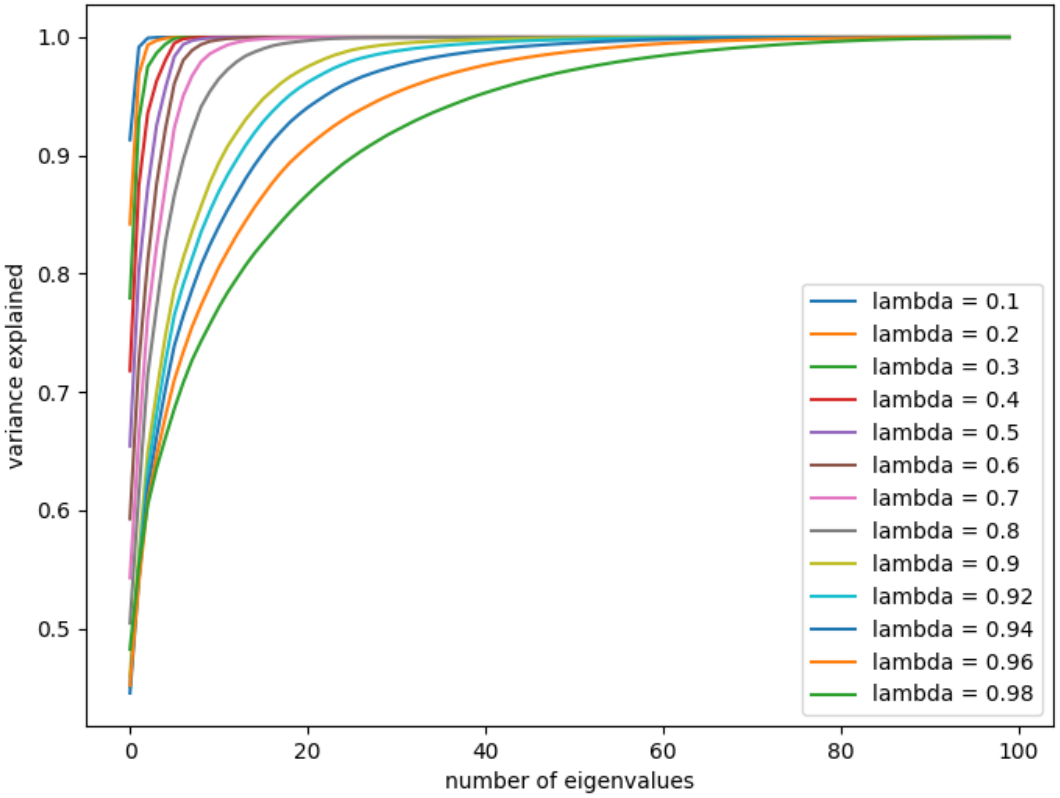
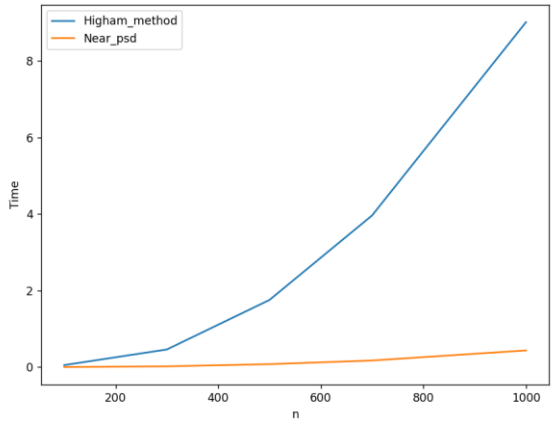
Problem1.

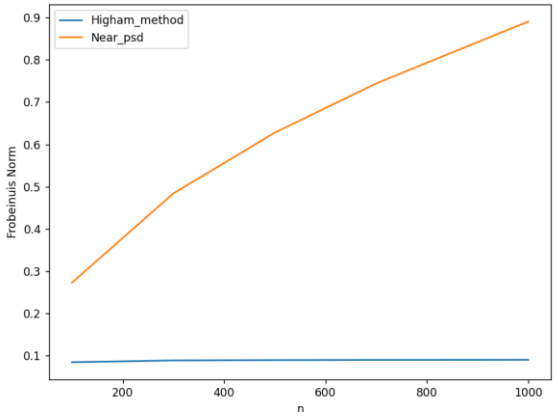


Lambda in this context represents the emphasis placed on recent information. A lower lambda means attaching greater importance to recent data due to the rapid decay of weights. As illustrated in our graph, a higher lambda results in a slower decay of weights, ensuring a more evenly distributed impact across the dataset. Lambda essentially determines the speed at which we discount past observations, and a higher lambda implies a more persistent influence of historical data.

Problem2.



From here we can see that the Higham method takes more time to run than the near_psd method as the dimension of the matrix increases.

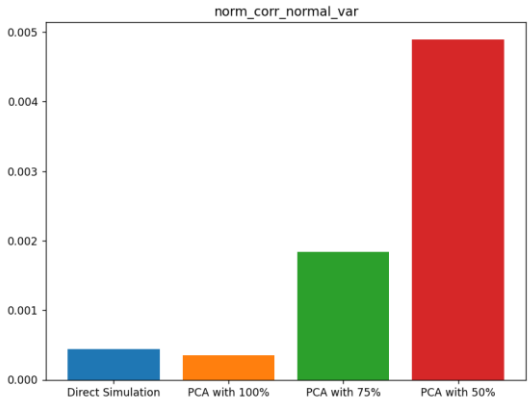


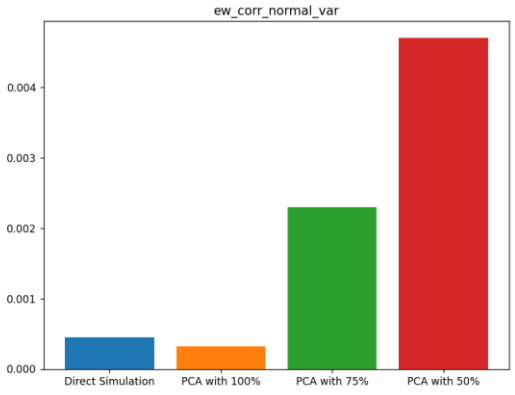
However, it's important to note that the error term in the Higham method remains constant as the dimensionality increases. Conversely, the error in the near_psd method tends to escalate as the dimensionality of the problem grows.

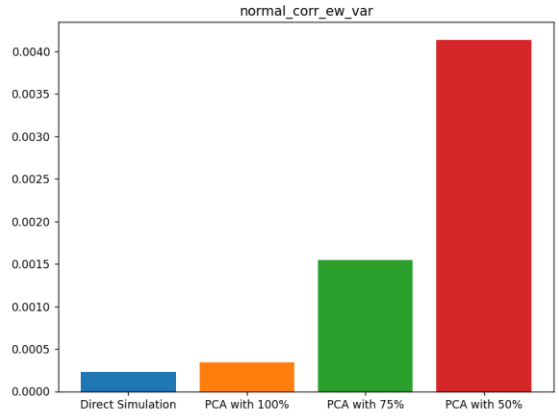
In essence, the Higham method offers higher accuracy, but at the cost of increased computational complexity and time. On the other hand, the near_psd method is faster to execute, but sacrifices some accuracy compared to the Higham method.

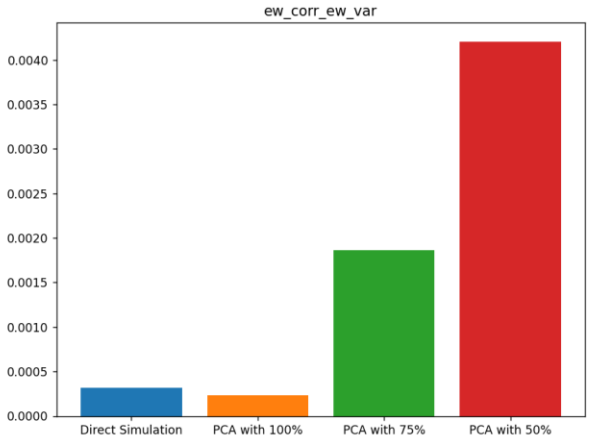
In my preference, I opt for the Higham method when dealing with smaller matrices or when utmost precision in computation is crucial. However, for larger matrices (exceeding 5000) where the need for precision is relatively lower, and computational speed is a priority, I find the near_psd method to be a more suitable choice.

Problem3

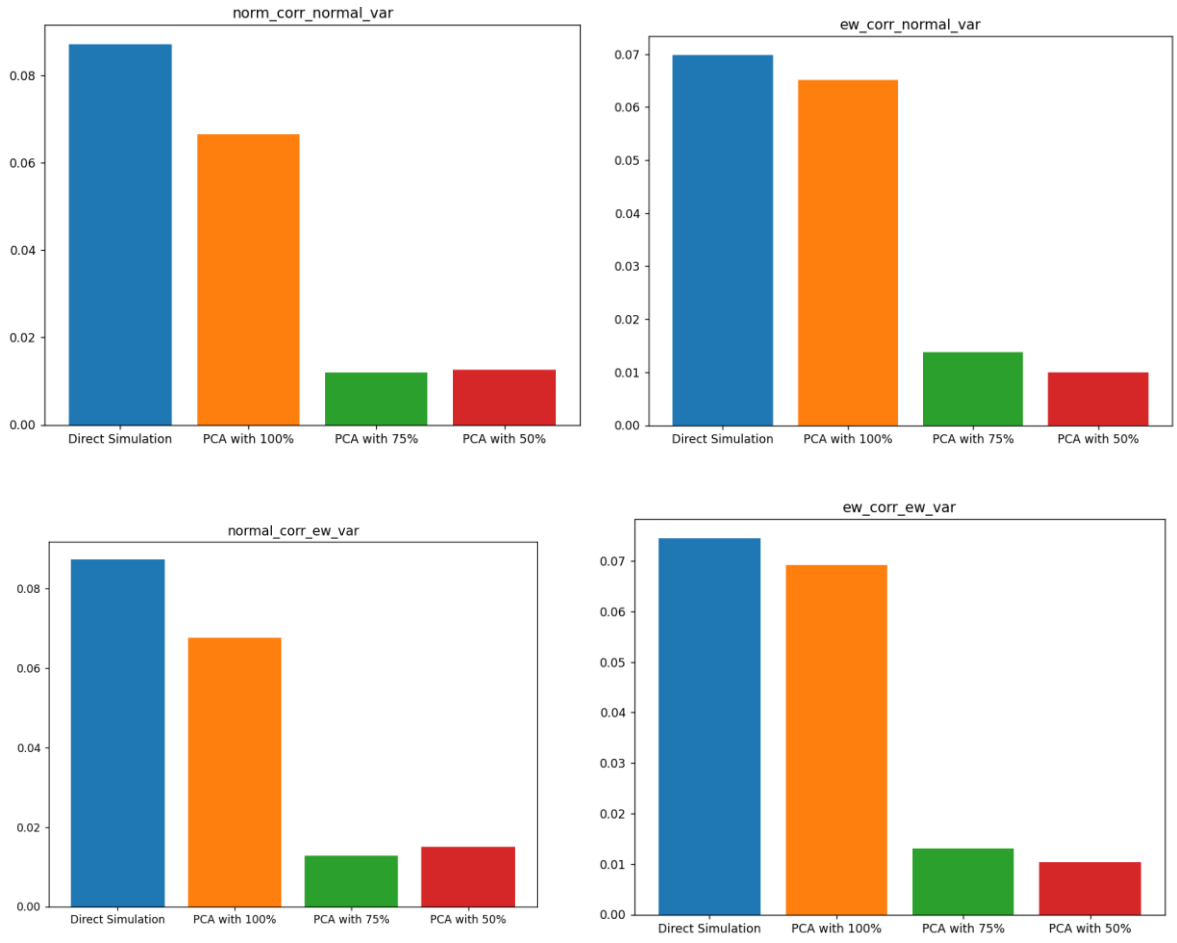








As observed, the error term, measured using the Frobenius norm, is minimal for both direct simulation and PCA when retaining 100% of the features. However, as the PCA retention percentage decreases, leaving more features behind, the error term proportionally increases. This occurs because a lower PCA retention percentage implies a reduced representation of the original data, resulting in a higher error compared to using the complete set of features.



However, when examining the computational time, it's evident that it decreases as the PCA retention percentage decreases. The underlying reason is straightforward: as the matrix's dimensionality decreases due to a lower PCA retention percentage, the computational time naturally reduces.

This scenario exemplifies the tradeoff between accuracy and computational efficiency. The lower the desired accuracy, the less time is needed for computation. Interestingly, even a PCA retention of 100% can contribute to reduced computational time, as it involves discarding features with eigenvalues deemed insignificant.