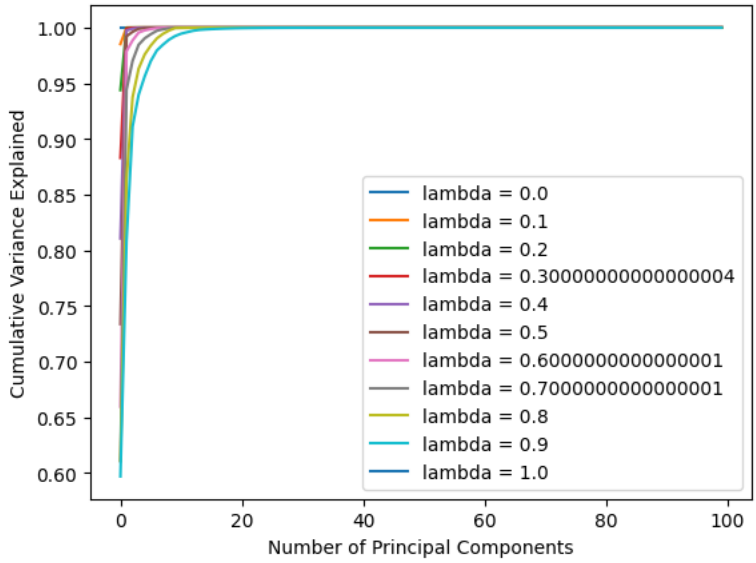
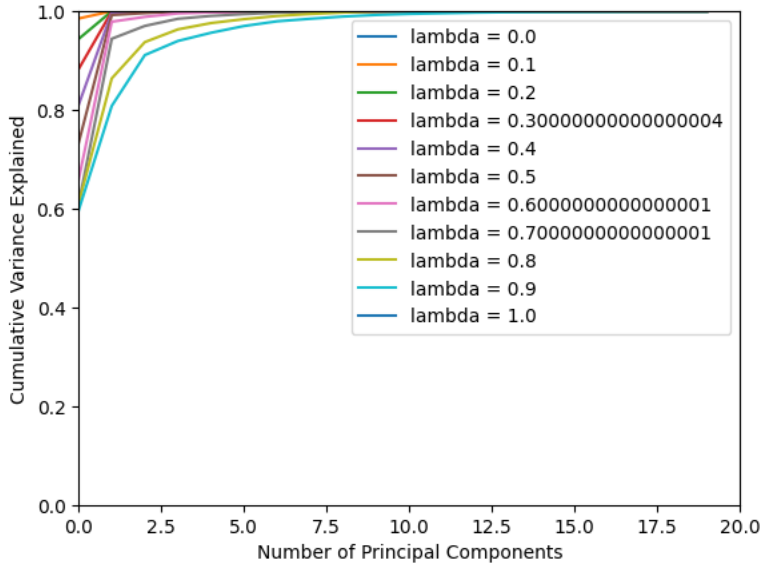
Problem 1:

Here is the plot for cumulative variance explained by each eigenvalue for each lambda chosen.



Here is a more zoomed in plot:



From above, the cumulative explained variance increases rapidly in the early number of principal components, it suggests that about 20 principal components can capture a significant amount of the variance in the Dailyreturn data. After 20 components, the cumulative variance explained are near 1. This support that 20 components are sufficient to capture almost all of the variance in the data. As for the increase in lambda, we can see as the lambda increases, the speed at which the cumulative variance explained increases becomes slower, in other words, more stable. That means the higher weight in recent observations are better than older observations in this case as more principal components are added.

Problem 2:

Chol_pd and near_psd Transferred code is in github.

Result:

Time taken by Higham's method: 0.1894989013671875 Time taken by near_PSD method: 0.07160186767578125

The run time of both functions will increase as N increases, as the matrix size is directly proportional to the amount of computation required. The near_PSD method will likely be faster than Higham's method for larger N, as it uses a simple eigenvalue decomposition and update. In contrast, Higham's method involves solving a series of linear systems iteratively.

Higham's method is easier to implement, I used much less time and codes to do that. Furthermore, the method always gave me PSD not just for close approximation. However, higham's method needs more time, like more than double time than near_PSD method. Which is computationally expensive.

near_PSD is more efficient and faster than Higham's method. But it is less accurate than Higham's method, in this case, I can calculate the frobenius norm of difference between original and Higham's correct is 0.06, but I can't get a exact number for near_PSD.

Problem 3

Result:

Frobenius norms: [0.00021132576943203615, 0.00021377547872296137, 0.00022518186702142286, 0.00022479425767135585, 0.00018550777180576598, 0.00018529920438740 37, 0.00018095297479934266, 0.00018618055255743 94]

Run times: [0.07173013687133789, 0.06245088577270508, 0.08223795890808105, 0.07356977462768555, 0.06804704666137695, 0.06780385971069336 0.08697509765625, 0.07811784744262695]

Here is the corresponding methods to the 8 simulations
1. Using cov_pearson and not using PCA (explained_variance=None)
2. Using cov_pearson and using PCA with explained_variance=0.50
3. Using cov_pearson and using PCA with explained_variance=0.75
4. Using cov_pearson and using PCA with explained_variance=0.100
5. Using cov_exponentially_weighted and not using PCA (explained_variance=None)
6. Using cov_exponentially_weighted and using PCA with explained_variance=0.50
7. Using cov_exponentially_weighted and using PCA with explained_variance=0.75

8. Using cov_exponentially_weighted and using PCA with explained_variance=0.100

Based on the results of the eight simulations, the tradeoff between accuracy and run time is somewhat dependent on the choice of the covariance matrix (cov_pearson or cov_exponentially_weighted) and the use of PCA.
For simulations using cov_pearson, using PCA with explained_variance values of 0.5, 0.75, and 1 resulted in a slightly higher Frobenius norm than not using PCA (explained_variance=None). This suggests that the accuracy is lower when PCA is used. However, the run times were slightly faster when PCA was used, with the quickest run time being 0.062 seconds for the simulation using PCA with explained_variance=0.5.
For simulations using cov_exponentially_weighted, PCA with explained_variance values of 0.5, 0.75, and 1 resulted in a slightly lower Frobenius norm than not using PCA (explained_variance=None) on average. This suggests that the accuracy is higher when PCA is used. The run times were also slightly faster when PCA was used, with the quickest run time being 0.067 seconds for the simulation using PCA with explained_variance=0.5.
In general, for both cov_pearson and cov_exponentially_weighted, using PCA resulted in a slightly faster run time but slightly decreased accuracy as measured by the Frobenius norm. The tradeoff between accuracy and run time is, therefore a delicate balance and depends on the specific needs and requirements of the task.