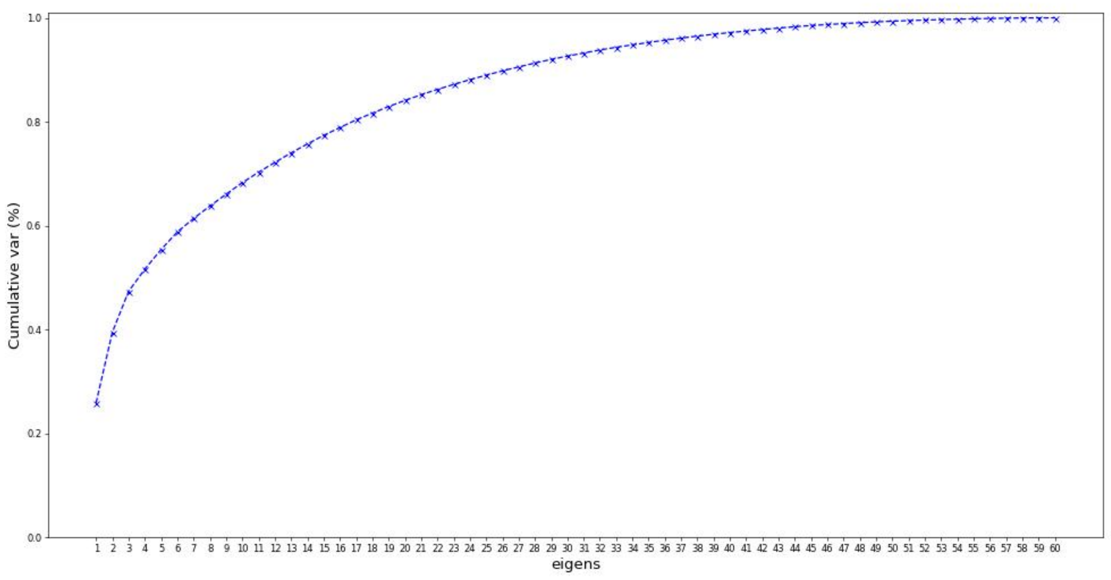
Problem 1

An function is created to calculate the exponentially weighted covariance matrix. The details of implementation can be seen in code file named `code/Problem1.ipynb`

Then, we apply Principal Component Analysis(PAC) to `DailyReturn` data. As expected, the projection dimension is not greater than 60. We plot the cumulative variance and find the curve eventually flattens out. So the previous eigenvalues are more representative of the data. Here is the curve plotted for cumulative variance.

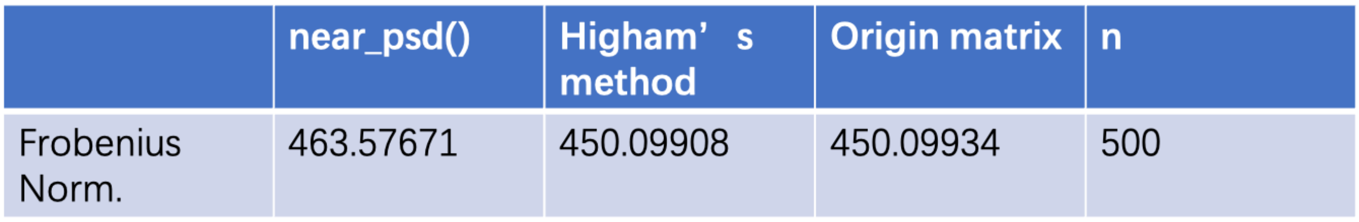


Lambda is the paramter to make exponential smoothing model. Risk Metrics research found that λ = 0. 97 was the best value, so we also set 0.97 as default value. Lambda can help us to combine time series returns, and we can focus on recent data or older data by adjusting lambda.

First, we implement near_psd() function and generate data function using python. And we check the generate function can genetate a non-psd correlation matrix as we want.

Second, We apply two approches to transform the non-psd correlation matrix to psd matrix. We check that eigen values are greater than zero. So it's confirmed the matrix is PSD now.

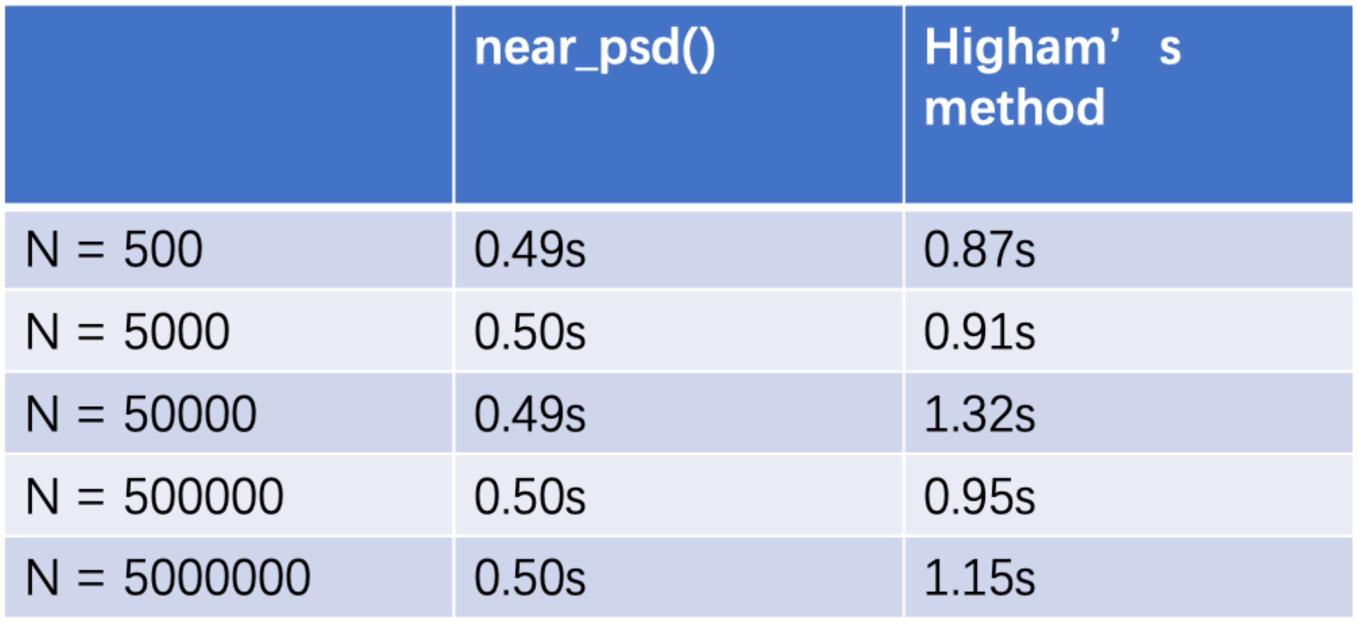
here is Forbenius norm comparing both results:



	near_psd()	Higham’ s method	Origin matrix	n
Frobenius Norm.	463.57671	450.09908	450.09934	500

We can find that the result of Higham's method is more close to origin matrix on Forbenius norm.

here is the run time comparing both results while N is increaseing:



	near_psd()	Higham’ s method
N = 500	0.49s	0.87s
N = 5000	0.50s	0.91s
N = 50000	0.49s	1.32s
N = 500000	0.50s	0.95s
N = 5000000	0.50s	1.15s

We can find the near_psd() method is faster and simpler.

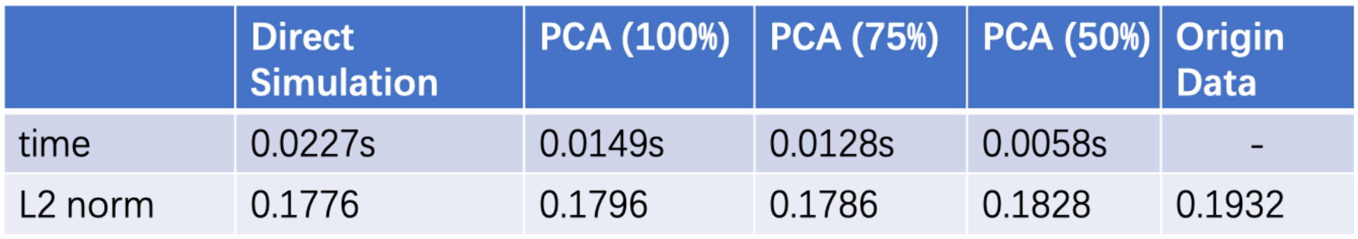
To summary, near_psd is not bad algorithm in time consuming, but it's not near to origin matrix. So we should choose the appropriate method to deal with the current problem.

First, we implement a multivariate normal simulation by python, which is similar with the numpy API `np.random.multivariate_normal`. We have check that it function as expected. More details can be seen in `code/Problem3.ipynb`.

Then, we generate standard Pearson correlation and EW covariance, and combine these to form 4 different covariance matrices.

Last, we simulate 25000 draws from each covariance matrix. The we compare L2 norm and runtime for each simualtion.

Here is the run time and L2 norm of four simulations:



	Direct Simulation	PCA (100%)	PCA (75%)	PCA (50%)	Origin Data
time	0.0227s	0.0149s	0.0128s	0.0058s	-
L2 norm	0.1776	0.1796	0.1786	0.1828	0.1932

We find that PCA with higher percent explained will consume more time. So when considering PCA appraoch, we can cutoff some component to run faster. And the result of L2 norm shows that PCA approach can represent the data better because its L2 norm is more close to origin data than direct simulation.