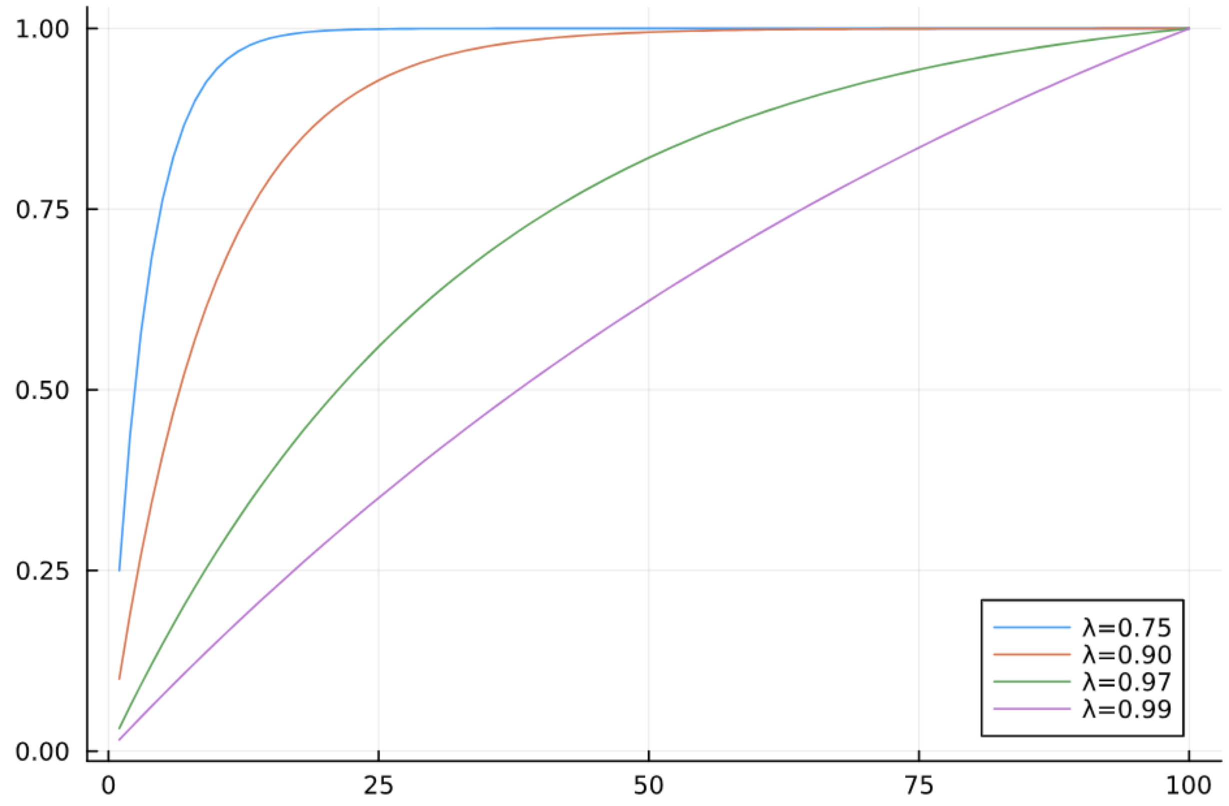
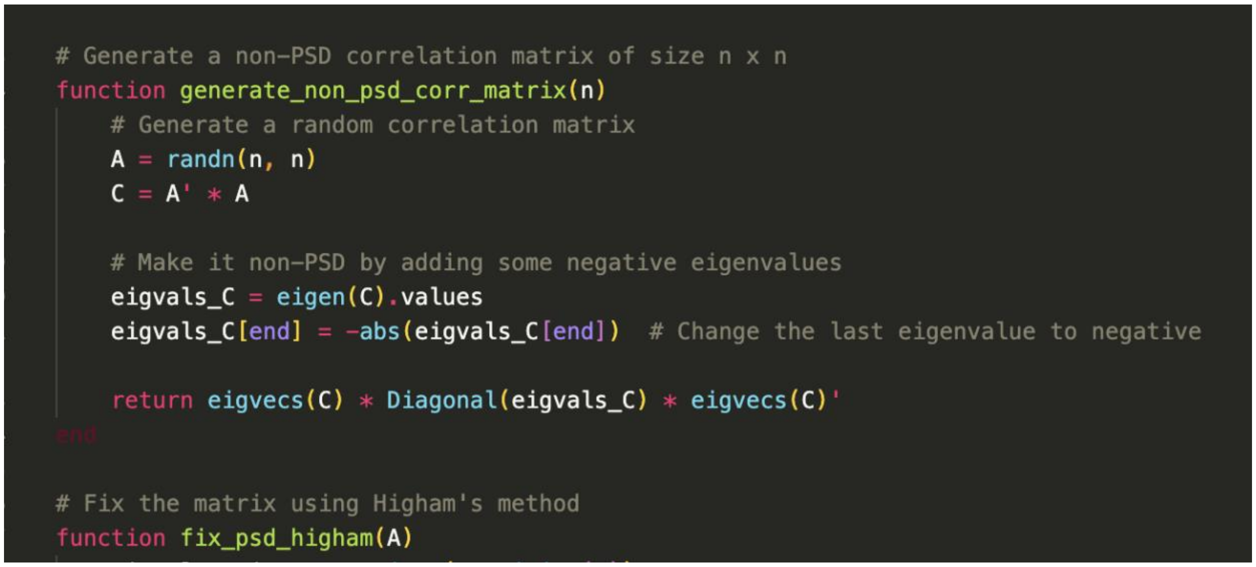
Problem 1

Using the code for generating Exponentially weighted covariance matrix and for PCA simulation with different lambda values, we could get the following graph. As lambda gets larger, the more eigen value are required for the explanation of variance. In addition, we could also find the for small lambda value, the weights are more heavily weighted for the most recent daily return.



Problem 2.

This code will:



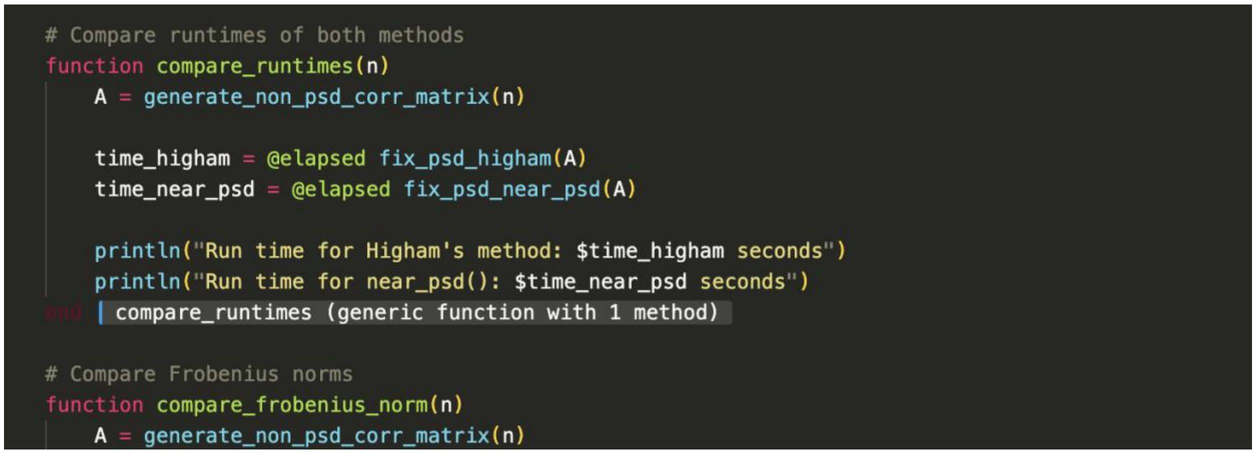
```julia
# Generate a non-PSD correlation matrix of size n x n
function generate_non_psd_corr_matrix(n)
    # Generate a random correlation matrix
    A = randn(n, n)
    C = A' * A

    # Make it non-PSD by adding some negative eigenvalues
    eigvals_C = eigen(C).values
    eigvals_C[end] = -abs(eigvals_C[end])  # Change the last eigenvalue to negative

    return eigvecs(C) * Diagonal(eigvals_C) * eigvecs(C)'
end

# Fix the matrix using Higham's method
function fix_psd_higham(A)
```

Generate a non-positive definite correlation matrix of size 500x500.
Fix it using both Higham's method and near_psd().
Compare the runtimes and Frobenius norms for both methods.



```julia
# Compare runtimes of both methods
function compare_runtimes(n)
    A = generate_non_psd_corr_matrix(n)

    time_higham = @elapsed fix_psd_higham(A)
    time_near_psd = @elapsed fix_psd_near_psd(A)

    println("Run time for Higham's method: $time_higham seconds")
    println("Run time for near_psd(): $time_near_psd seconds")
end | compare_runtimes (generic function with 1 method)

# Compare Frobenius norms
function compare_frobenius_norm(n)
    A = generate_non_psd_corr_matrix(n)
```

I run the code in a Julia environment to observe the results for the specified matrix size (n=500). And I explore how runtime changes with matrix size (N), through adjusting the value of n and run the corresponding functions accordingly. According to the result, we could find that, in general, the Forbenius Norm of the Higham is always quite close to 0 as the sample size increase, while the other Forbenius norm is increase as sample size increase. The run time for the higham method is slower, especially as sample size is larger than 500, while for the near psd, the runtime is stable as sample size get larger, and it is faster than the higham as sample size gets larger.

Problem 3

For this problem, we generate a random covariance matrix. Calculate the correlation matrix and variance vector using standard Pearson correlation/variance and calculate the correlation matrix and variance vector using exponentially weighted correlation/variance with λ = 0.97.

For the accuracy comparison:

The direct method and PC with 100% explain how the lowest and highest accuracy and as the percentage explained decreases PC become less accurate. As percent explains decreases, less features are incorporated into the PC model and thus less accurate. According to the result, we could find that PCA with 50% explanation is the most inaccurate, which is about 2*10^(-5). As the explained percentage gets larger, the accuracy is increased, and the direct simulation is the most accurate one with lowest error, which is quite close to 0.

For the run time:

The direct calculation cost the most time for simulation, while the that PCA with 50% explanation is the most fast one. More specificially, the tun time for the direct calculation is about 0.17 s, and PCA with 100% cost about 0.052 s, PCA with 90% cost about 0.041, and PCA with 50% explanation cost about 0 s.

As the percentage of the explained becomes larger, the operation process time gets smaller. So there is a trade off between the accuracy and run time of the code.