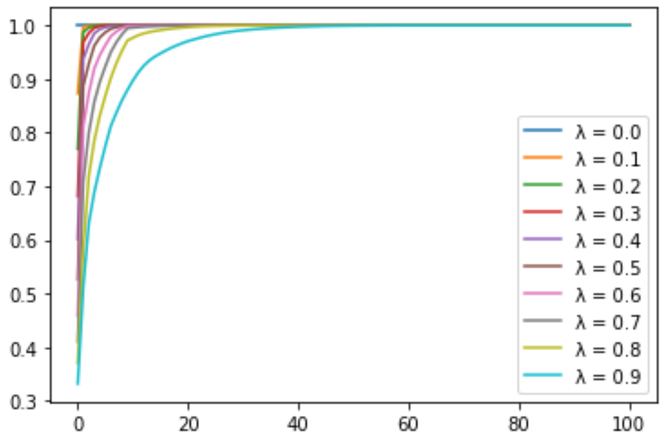
Problem 1

The plot of cumulative variance explained by each eigenvalue gives us insights into the relationship between λ and the covariance matrix used in the PCA analysis. The value of λ in an exponentially weighted covariance matrix determines the weight given to the historical returns.

A smaller value of λ gives more weight to recent returns and less weight to older returns. This results in a covariance matrix that more accurately reflects the current covariance structure of the returns. However, this also means that the covariance matrix is more sensitive to changes in the returns and may not be as stable over time. On the other hand, a larger λ gives more weight to older returns and less weight to recent returns. This results in a covariance matrix that is more stable over time, as it is less sensitive to changes in the returns. However, this also means that the covariance matrix may not accurately reflect the current covariance structure of the returns.



The plot shows that as λ increases, the cumulative variance explained by each eigenvalue approaches a stable value. This suggests that for a large enough λ, the covariance matrix becomes relatively stable and the eigenvalues of the PCA analysis become relatively constant. This means that the most important sources of risk in the returns are relatively constant over time, even though the covariance structure may be changing.

In conclusion, the choice of λ in an exponentially weighted covariance matrix is a trade-off between stability and accuracy. A smaller λ provides more accurate information about the current covariance structure of the returns, while a larger λ provides more stability over time.

Problem 2

Regarding the run time comparison between the two functions, as N increases, it's difficult to make a general statement without testing it for specific values of N. But in general, as N increases, both methods will likely take longer to run. The time complexity of Higham's method is O(N^3), which is relatively high compared to other methods, but the method is guaranteed to produce a PSD matrix if it converges. The time complexity of the near_psd() method is not specified, but it is likely to be faster for small values of N and slower for large values of N, compared to Higham's method.

```
Time for near_psd: 0.245650053024292
Time for highams: 3.5467357635498047
Frobenius norm difference (near_psd vs. highams): 37.57492421720327
```

In terms of the pros and cons of each method, Higham's method is guaranteed to produce a PSD matrix if it converges, but its time complexity is relatively high. The near_psd() method is likely to be faster for small values of N, but its time complexity is not specified and it may not be guaranteed to produce a PSD matrix. In general, we would use Higham's method if we need to ensure that the matrix is PSD and the computational time is not a concern, whereas we would use the near_psd() method if computational time is a concern and you can tolerate the matrix not being guaranteed to be PSD.

Problem 3

- **4 Different Covariance Matrices:**
  - ***covariance_matrix_1***

```
[[5.95185891e-09 1.26123779e-08 1.38105872e-08 ... 2.78324872e-08
  1.28160130e-08 1.21476180e-08]
 [1.26123779e-08 6.40928149e-08 3.85791502e-08 ... 3.91287404e-08
  1.17429246e-08 2.02571970e-08]
 [1.38105872e-08 3.85791502e-08 6.27769339e-08 ... 2.97894763e-08
  6.68260570e-09 1.69090213e-08]
 ...
 [2.78324872e-08 3.91287404e-08 2.97894763e-08 ... 5.14850739e-07
  1.07064660e-07 8.45471204e-08]
 [1.28160130e-08 1.17429246e-08 6.68260570e-09 ... 1.07064660e-07
  8.78106776e-08 2.62174160e-08]
 [1.21476180e-08 2.02571970e-08 1.69090213e-08 ... 8.45471204e-08
  2.62174160e-08 7.44218439e-08]]
```

  - ***covariance_matrix_2***

```
[[0.00431465 0.00900131 0.01065544 ... 0.01937871 0.00907114 0.00769928]
 [0.00900131 0.04503337 0.02930407 ... 0.02682161 0.00818279 0.01264022]
 [0.01065544 0.02930407 0.05154964 ... 0.02207507 0.00503409 0.01140628]
 ...
 [0.01937871 0.02682161 0.02207507 ... 0.34429787 0.07278392 0.05146807]
 [0.00907114 0.00818279 0.00503409 ... 0.07278392 0.06068382 0.01622427]
 [0.00769928 0.01264022 0.01140628 ... 0.05146807 0.01622427 0.04124066]]
```

  - ***covariance_matrix_3***

```
[[5.95185891e-09 1.38248778e-08 1.49647835e-08 ... 2.77320535e-08
  1.15998245e-08 1.21739185e-08]
 [1.38248778e-08 6.40928149e-08 4.37312009e-08 ... 4.76289069e-08
  1.16582779e-08 2.11107397e-08]
 [1.49647835e-08 4.37312009e-08 6.27769339e-08 ... 3.12574725e-08
  7.89451798e-09 1.74233739e-08]
 ...
 [2.77320535e-08 4.76289069e-08 3.12574725e-08 ... 5.14850739e-07
  1.19451920e-07 8.92330468e-08]
 [1.15998245e-08 1.16582779e-08 7.89451798e-09 ... 1.19451920e-07
  8.78106776e-08 2.45546666e-08]
 [1.21739185e-08 2.11107397e-08 1.74233739e-08 ... 8.92330468e-08
  2.45546666e-08 7.44218439e-08]]
```

- ***covariance_matrix_4***

```
[[0.00431465 0.00986666 0.01154595 ... 0.01930878 0.00821033 0.00771595]
 [0.00986666 0.04503337 0.03321748 ... 0.03264823 0.00812381 0.01317282]
 [0.01154595 0.03321748 0.05154964 ... 0.02316291 0.00594704 0.01175324]
 ...
 [0.01930878 0.03264823 0.02316291 ... 0.34429787 0.08120493 0.05432063]
 [0.00821033 0.00812381 0.00594704 ... 0.08120493 0.06068382 0.0151953 ]
 [0.00771595 0.01317282 0.01175324 ... 0.05432063 0.0151953  0.04124066]]
```

- **The run times for each simulation:**

```
Direct simulation (covariance matrix 1) time: 0.3353116512298584
Direct simulation (covariance matrix 2) time: 0.1626298427581787
Direct simulation (covariance matrix 3) time: 0.30080628395080566
Direct simulation (covariance matrix 4) time: 0.18924903869628906
PCA simulation (100% explained variance, covariance matrix 1) time: 0.17568278312683105
PCA simulation (100% explained variance, covariance matrix 2) time: 0.009835004806518555
PCA simulation (100% explained variance, covariance matrix 3) time: 0.002969980239868164
PCA simulation (100% explained variance, covariance matrix 4) time: 0.003757953643798828
PCA simulation (75% explained variance, covariance matrix 1) time: 0.0035500526428222656
PCA simulation (75% explained variance, covariance matrix 2) time: 0.006932258605957031
PCA simulation (75% explained variance, covariance matrix 3) time: 0.004830837249755859
PCA simulation (75% explained variance, covariance matrix 4) time: 0.0032842159271240234
PCA simulation (50% explained variance, covariance matrix 1) time: 0.0022363662719726562
PCA simulation (50% explained variance, covariance matrix 2) time: 0.0021910667419433594
PCA simulation (50% explained variance, covariance matrix 3) time: 0.002429962158203125
PCA simulation (50% explained variance, covariance matrix 4) time: 0.0019600391387939453
Frobenius norm (direct simulation, covariance matrix 1): 1.0714085391847087e-07
Frobenius norm (direct simulation, covariance matrix 2): 0.0732773719281461
Frobenius norm (direct simulation, covariance matrix 3): 1.3207847295009985e-07
Frobenius norm (direct simulation, covariance matrix 4): 0.09405983093784905
Frobenius norm (PCA simulation, 100% explained variance, covariance matrix 1): 0.011961307852746865
Frobenius norm (PCA simulation, 100% explained variance, covariance matrix 2): 5.322875303433184
Frobenius norm (PCA simulation, 100% explained variance, covariance matrix 3): 0.011960869595632738
Frobenius norm (PCA simulation, 100% explained variance, covariance matrix 4): 5.697557838036493
Frobenius norm (PCA simulation, 75% explained variance, covariance matrix 1): 0.0118630223543325
Frobenius norm (PCA simulation, 75% explained variance, covariance matrix 2): 5.32287506146283
Frobenius norm (PCA simulation, 75% explained variance, covariance matrix 3): 0.011876296730622241
Frobenius norm (PCA simulation, 75% explained variance, covariance matrix 4): 5.697557601217375
Frobenius norm (PCA simulation, 50% explained variance, covariance matrix 1): 0.011780342999561688
Frobenius norm (PCA simulation, 50% explained variance, covariance matrix 2): 5.3228749756066875
Frobenius norm (PCA simulation, 50% explained variance, covariance matrix 3): 0.0117765267328611
Frobenius norm (PCA simulation, 50% explained variance, covariance matrix 4): 5.697557602683435
```

- **The trade-offs between time to run and accuracy:**

The trade-offs between time to run and accuracy in the multivariate normal simulation are often a balance between computational efficiency and the quality of the simulation results. Direct simulation is generally faster than PCA-based simulation, but it may not always accurately capture the complex relationships between variables that are present in the original covariance matrix. On the other hand, PCA-based simulation can be more computationally intensive, but it can better preserve the relationships between variables and result in more accurate simulations. The trade-off between time and accuracy also depends on the amount of explained variance desired. PCA-based simulation with a high explained variance will result in more accurate simulations but will take longer to run, while PCA-based simulation with a lower explained variance will be faster but may not be as accurate.