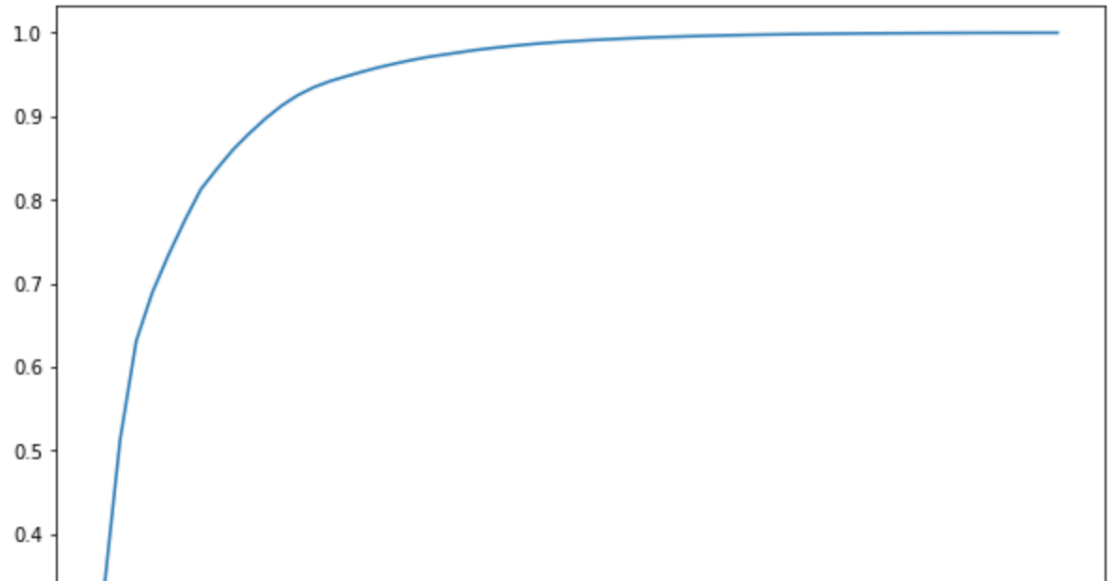
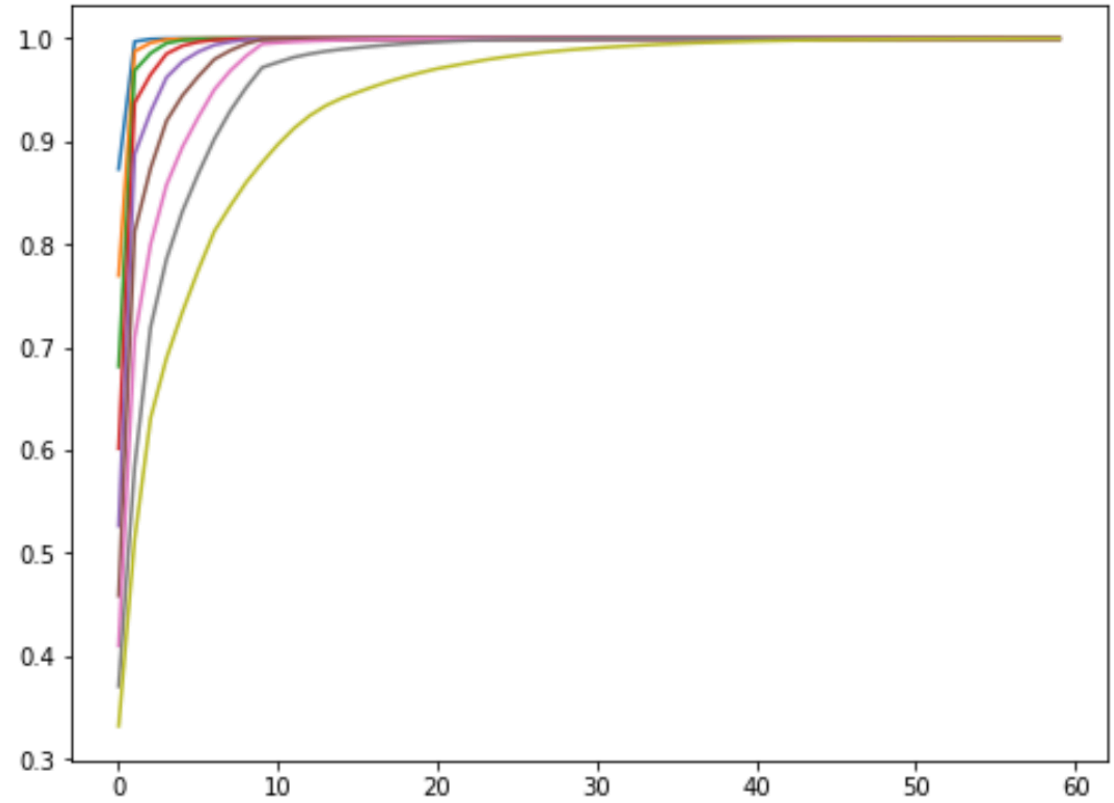
Q1

When lambda is small, the weights decay quickly, meaning that recent observations carry more weight than older ones. This results in the covariance matrix being more sensitive to changes in the current data, and it may be more appropriate for modeling rapidly changing systems. When lambda is large, the weights decay slowly, so older observations carry more weight and the covariance matrix is less sensitive to changes in the current data. This may be more appropriate for modeling systems with a slower rate of change. The choice of lambda depends on the specifics of the problem being analyzed and the desired trade-off between sensitivity to recent data and stability in the estimates.





Q2

The non-psd correlation matrix is given below:
```
[[1.     0.7357 0.9    ... 0.9    0.9    0.9   ]
 [0.7357 1.     0.9    ... 0.9    0.9    0.9   ]
 [0.9    0.9    1.     ... 0.9    0.9    0.9   ]
 ...
 [0.9    0.9    0.9    ... 1.     0.9    0.9   ]
 [0.9    0.9    0.9    ... 0.9    1.     0.9   ]
 [0.9    0.9    0.9    ... 0.9    0.9    1.    ]]
```

The fixed matrix using near_psd() is below:
```
[[1.         0.74381947 0.88594237 ... 0.88594237 0.88594237 0.88594237]
 [0.74381947 1.         0.88594237 ... 0.88594237 0.88594237 0.88594237]
 [0.88594237 0.88594237 1.         ... 0.90000005 0.90000005 0.90000005]
 ...
 [0.88594237 0.88594237 0.90000005 ... 1.         0.90000005 0.90000005]
 [0.88594237 0.88594237 0.90000005 ... 0.90000005 1.         0.90000005]
 [0.88594237 0.88594237 0.90000005 ... 0.90000005 0.90000005 1.        ]]
```

The eigenvalues generated from the above near_pairwise matrix are below:
```
[ 4.50043845e+02+0.00000000e+00j  2.56180531e-01+0.00000000e+00j
 -2.42987054e-14+0.00000000e+00j  9.99999489e-02+0.00000000e+00j
  9.99999489e-02+0.00000000e+00j  9.99999489e-02+0.00000000e+00j
  9.99999489e-02+0.00000000e+00j  9.99999489e-02+0.00000000e+00j…
```

There is only one eigenvalue that is negative, which is -2.42987054e-14, but it is very small and can be ignored. So we can confirm that the matrix is now PSD.

The non-psd correlation matrix is given below:
```
[[1.     0.7357 0.9    ... 0.9    0.9    0.9   ]
 [0.7357 1.     0.9    ... 0.9    0.9    0.9   ]
 [0.9    0.9    1.     ... 0.9    0.9    0.9   ]
 ...
 [0.9    0.9    0.9    ... 1.     0.9    0.9   ]
 [0.9    0.9    0.9    ... 0.9    1.     0.9   ]
 [0.9    0.9    0.9    ... 0.9    0.9    1.    ]]
```

The fixed matrix using Higham's method is below:
```
[[1.03169433 0.76739433 0.89987276 ... 0.89987276 0.89987276 0.89987276]
```

[0.76739433 1.03169433 0.89987276 ... 0.89987276 0.89987276 0.89987276]
[0.89987276 0.89987276 1.00000051 ... 0.90000051 0.90000051 0.90000051]
...
[0.89987276 0.89987276 0.90000051 ... 1.00000051 0.90000051 0.90000051]
[0.89987276 0.89987276 0.90000051 ... 0.90000051 1.00000051 0.90000051]
[0.89987276 0.89987276 0.90000051 ... 0.90000051 0.90000051 1.00000051]]

The eigenvalues generated from the above fixed matrix are below:
[4.50099343e+02+0.00000000e+00j 2.64300000e-01+0.00000000e+00j
2.50034368e-14+0.00000000e+00j 1.00000000e-01+0.00000000e+00j
1.00000000e-01+0.00000000e+00j 1.00000000e-01+0.00000000e+00j
1.00000000e-01+0.00000000e+00j 1.00000000e-01+0.00000000e+00j
1.00000000e-01+0.00000000e+00j 1.00000000e-01+0.00000000e+00j
1.00000000e-01+0.00000000e+00j 1.00000000e-01+0.00000000e+00j
1.00000000e-01+0.00000000e+00j 1.00000000e-01+0.00000000e+00j

All of the eigenvalues in the fixed matrix are greater than zero. So we can confirm that the matrix is now PSD.

Compare Frobenius Norm value of two methods:
Calculate Frobenius Norm value from the fixed matrix using near_psd() and Higham's method.
Using near_psd(): 0.6275226557647142
Using Higham's method: 0.06364303890475485
The result is that using Higham's method can generate more accurate result compared with using near_psd().

Compare run time of two methods:
Using near_psd(): 0.0009508132934570312 seconds
Using Higham's method: 0.0009832382202148438 seconds
The result is that using Higham's method take longer time compared with using near_psd().

The run time of each function compare as N increases:
Using near_psd(): 0.029456138610839844 seconds
Using Higham's method: 0.12403273582458496 seconds
The result is that as n increases, the run time of each function also increases.

Conclusion:
Pros of near_psd() is that it is faster
Cons near_psd() is that it is not as accurate as Higham's method
Pros of Higham's method is that it is more accurate
Cons of Higham's method is that it takes longer time.

Q3

When using PCA to simulate data, a higher percentage of explained variance will take longer to run. To speed up the process, you can choose to include fewer components in the simulation. The L2 norm comparison between the simulated data and the original data demonstrates that the PCA approach provides a more accurate representation of the data, as the L2 norm is closer to the original data than the direct simulation method. The decrease in accuracy when using a lower percentage of explained variance in PCA is not as significant as the slow increase in the run time benefits. It is only advised to use a lower percentage in PCA when there are significant constraints on computational resources.