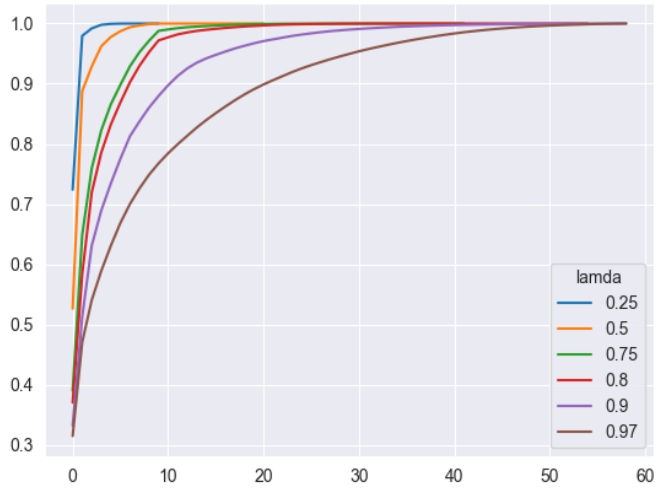
Problem1:

My process:

1. Generate a series of weights along the time axis, which has length equals to t (time length).
2. Use the weights generated and the weighted covariance formula to calculate the matrix.
3. Implement PCA on the weighted covariance matrix. That is remove the negative eigen values and the corresponding eigen vectors.
4. Calculate the Percent explained by each eigen values from biggest to smallest.
5. Use different λ to repeat the process.

Result:



Conclusion:

From the graph, we can see that as λ gets larger and closer to the optimal value of 0.97, the PCT curve gets smoother. It means that when λ is small, the covariance is nearly explained by only a few eigen values, while the large λ can make the pct explained increase evenly and steadily by more eigen values. It is because of how the exponential formula works: $w = (1-\lambda)\lambda^{i-1}$. If the λ is not big enough, the weights put on the first several recent time periods will be very large and shrinks quickly.

Problem2:

My process:

1. Implement functions for Cholesky, Rebonato and Jackel, and Higham respectively.
2. Generate non PSD correlation matrix
3. Fix the non PSD matrix use R&J and Higham.
4. Check eigen values of fixed matrix is now non negative
5. Calculate the Frobenius Norm for the difference between the fixed one and the original one.
6. Compare runtime and accuracy.

Result:

	Accuracy		Runtime	
Size	Near PSD	Higham	Near PSD	Higham
5	1.47E-10	1.02E-10	14.2 µs	3.66ms
50	0.03448	0.00619	343 µs	162 ms
100	0.07442	0.00716	1.29 ms	633 ms
500	0.39378	0.008037	43.5 ms	13.9s

Conclusion:

R&J Method:

Pro: The run time does not increase much as N increases

Con: The accuracy drops quickly as N increases

Higham:

Pro: Accuracy maintains as N increases

Con: The run time increases significantly as N increases.

Therefore, when choosing the methodology, it really depends on our needs. If we want to deal with a large matrix with tolerance of less accuracy, then we can use R&J Method. If we want more accurate result with tolerance of long run time, then we can use Higham.

Problem3:

My process:

1. Define four combinations of covariance matrix
2. Define the function to get PCA eigen values according to PCT explained
3. Define the PCA simulation function
4. For each PCT value, and for each covariance method, implement the simulation function and calculate the difference between the covariance of the simulated result and the input.

Result:

	cor + var	**cor + ewvar**	**ewcor+ ewvar**	**ewcor + var**
Direct	87.2ms	83.1ms	80.3ms	82ms
PCA 100%	41.4ms	38.8ms	50.3ms	45.7ms
PCA 75%	11.8ms	12.1ms	11.4ms	11.4ms
PCA 50%	6.64ms	6.14ms	5.19ms	5.01ms
	Accuracy			
Direct	4.72E-08	4.47E-08	4.39E-08	4.67E-08
PCA 100%	5.12E-08	4.93E-08	4.94E-08	5.11E-08
PCA 75%	2.68E-06	2.47E-06	2.45E-06	2.78E-06
PCA 50%	1.11E-05	1.03E-05	1.19E-05	1.25E-05

Conclusion:

The direct simulation generates the most accuracy, but takes the longest time compared to PCA. It is because the direct simulation does not reduce the dimension as PCA does. For PCA simulation, as PCT explained of PCA decreases, the run time is shortened while the accuracy drops at the same time. So the tradeoff is that we can not achieve a short run time and high accuracy at the same time. The more PCA is explained when simulating, the more accurate the simulated result will be, and the longer the run time will be.