problem 1

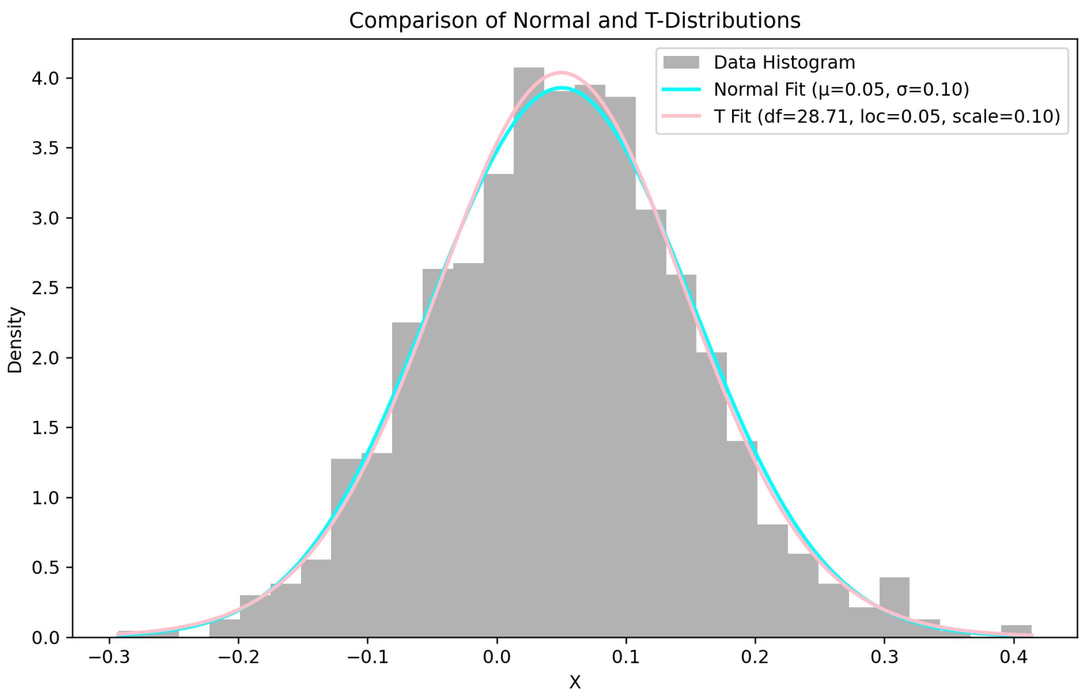
A
Mean: 0.05019795790476916
Variance: 0.010332476407479581
Skewness: 0.1204447119194402
Kurtosis: 0.2229270674503816
B

Normal distribution.

Based on the data's skewness (0.12) and kurtosis (0.22), the distribution is nearly symmetric with light tails, closely resembling a normal distribution. Thus, a normal distribution is a better choice for modeling this dataset as it aligns well with the observed characteristics.

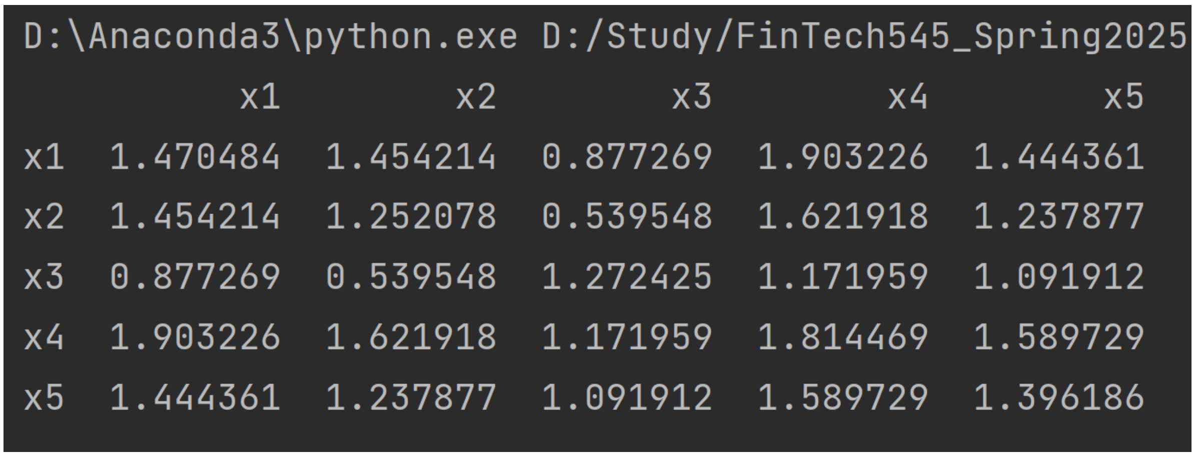
C

Both the normal and t-distributions were fitted to the data. The normal distribution closely matches the histogram with mean 0.05 and standard deviation 0.10. The t-distribution, with 28.71 degrees of freedom, also fits well but offers no significant advantage as the data lacks heavy tails.
This supports the earlier choice of the normal distribution.



problem2

A



```
D:\Anaconda3\python.exe D:/Study/FinTech545_Spring2025
          x1        x2        x3        x4        x5
x1  1.470484  1.454214  0.877269  1.903226  1.444361
x2  1.454214  1.252078  0.539548  1.621918  1.237877
x3  0.877269  0.539548  1.272425  1.171959  1.091912
x4  1.903226  1.621918  1.171959  1.814469  1.589729
x5  1.444361  1.237877  1.091912  1.589729  1.396186
```

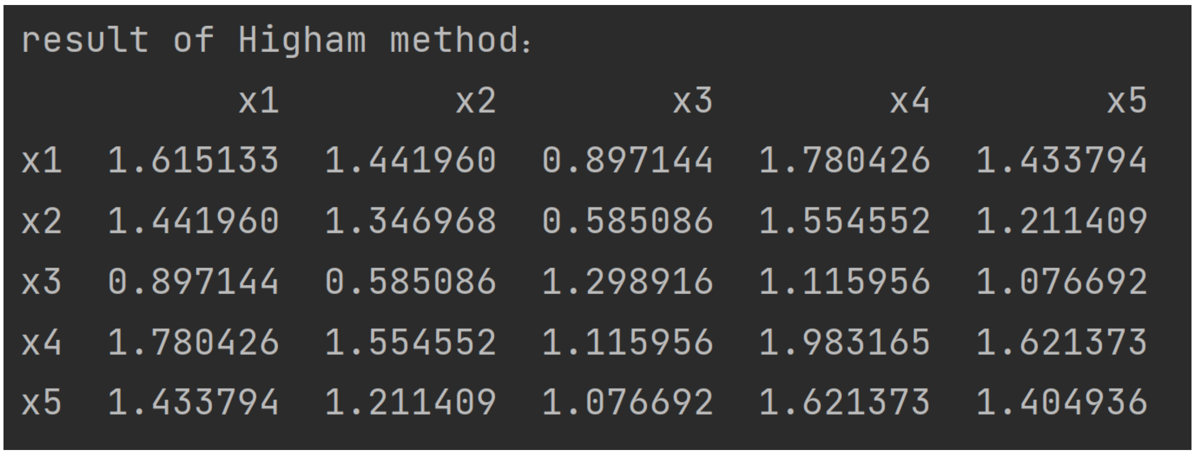
B

No, the covariance matrix is not at least positive semi-definite because it has negative eigenvalues.

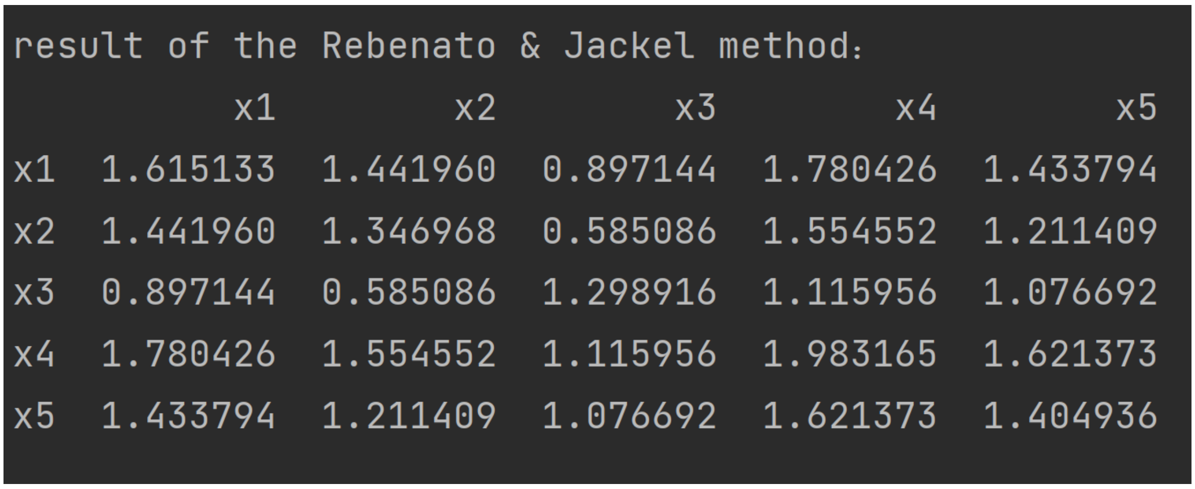
A matrix is positive semi-definite if all its eigenvalues are non-negative.

C

The two matrices using different methods share the same results.

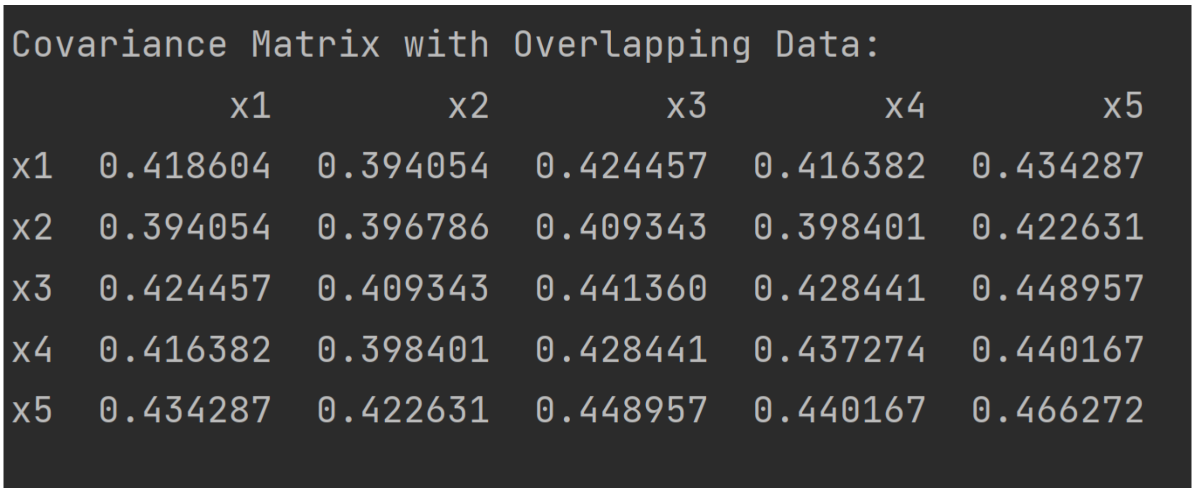


```
result of Higham method:
          x1        x2        x3        x4        x5
x1  1.615133  1.441960  0.897144  1.780426  1.433794
x2  1.441960  1.346968  0.585086  1.554552  1.211409
x3  0.897144  0.585086  1.298916  1.115956  1.076692
x4  1.780426  1.554552  1.115956  1.983165  1.621373
x5  1.433794  1.211409  1.076692  1.621373  1.404936
```



```
result of the Rebenato & Jackel method:
          x1        x2        x3        x4        x5
x1  1.615133  1.441960  0.897144  1.780426  1.433794
x2  1.441960  1.346968  0.585086  1.554552  1.211409
x3  0.897144  0.585086  1.298916  1.115956  1.076692
x4  1.780426  1.554552  1.115956  1.983165  1.621373
x5  1.433794  1.211409  1.076692  1.621373  1.404936
```

D



```
Covariance Matrix with Overlapping Data:
          x1        x2        x3        x4        x5
x1  0.418604  0.394054  0.424457  0.416382  0.434287
x2  0.394054  0.396786  0.409343  0.398401  0.422631
x3  0.424457  0.409343  0.441360  0.428441  0.448957
x4  0.416382  0.398401  0.428441  0.437274  0.440167
x5  0.434287  0.422631  0.448957  0.440167  0.466272
```

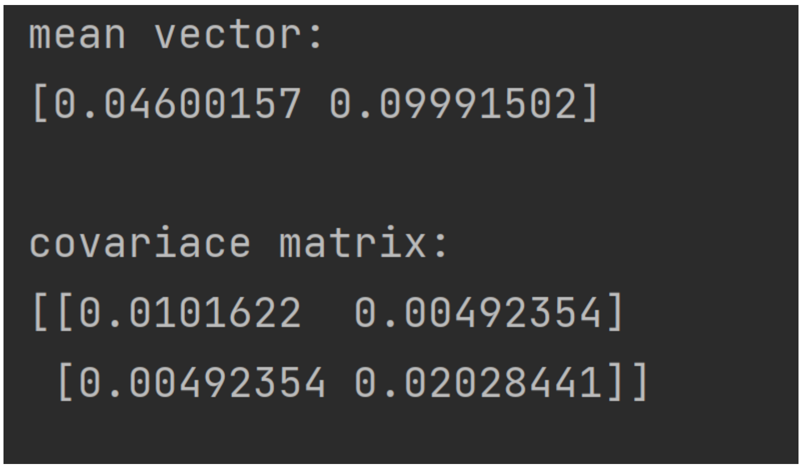
E

Matrix in question C ensures mathematical validity through adjusting negative eigenvalues, so it has higher variances and covariances.

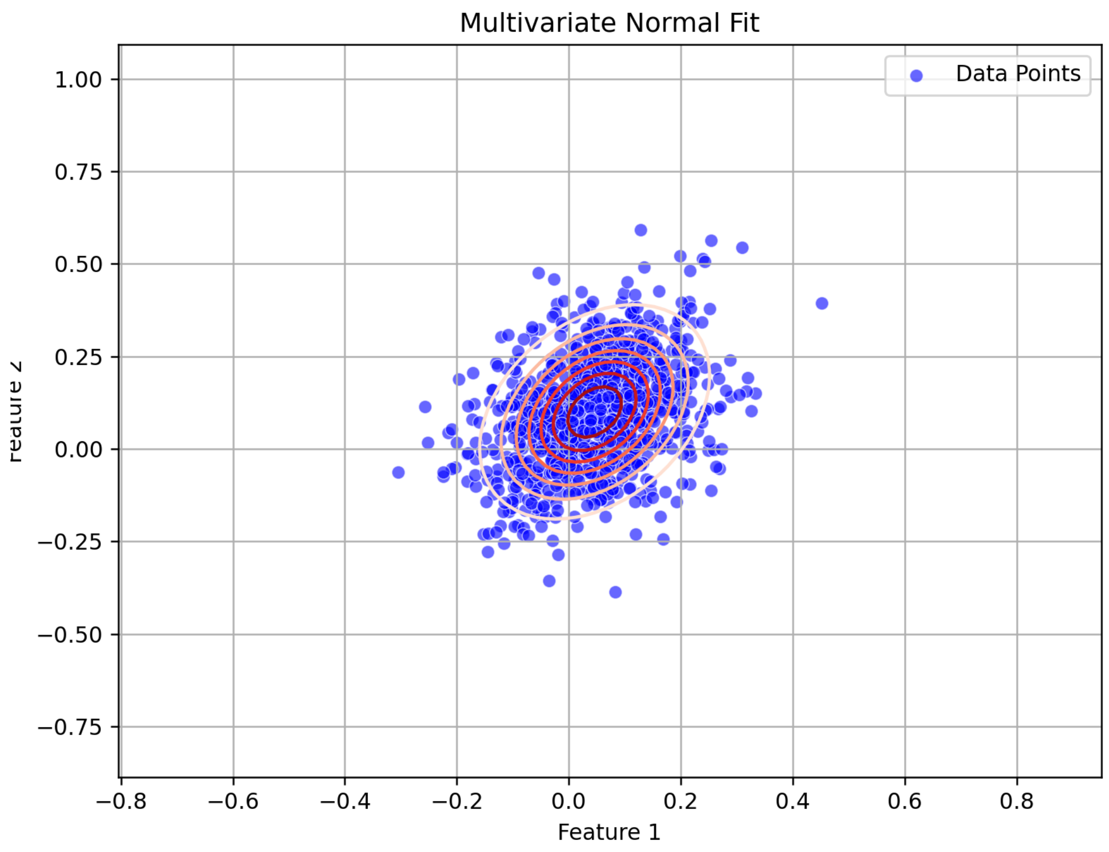
Matrix in question D is calculated based on overlapping data from the original dataset, reflecting true data relationships but with smaller values. So it has lower variances and covariances.

problem3

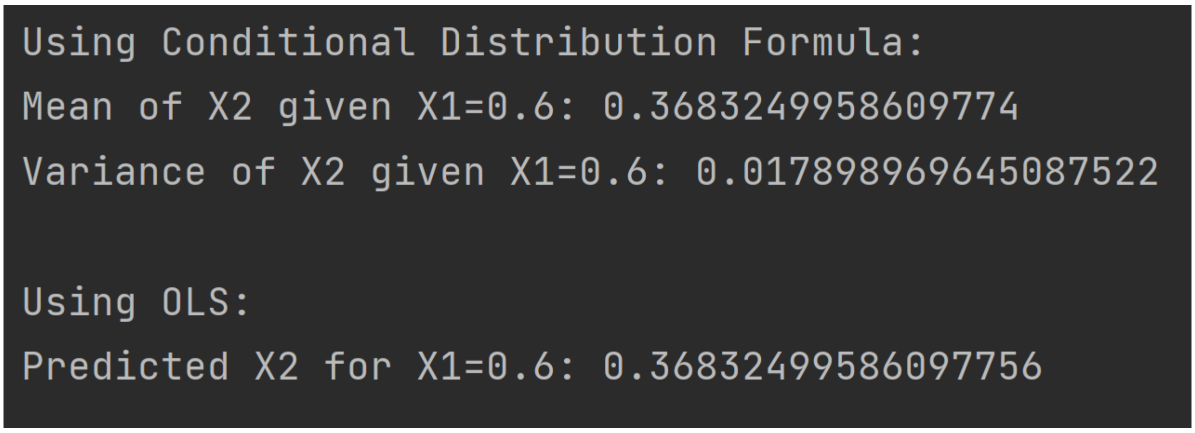
A



```
mean vector:
[0.04600157 0.09991502]

covariace matrix:
[[0.0101622  0.00492354]
 [0.00492354 0.02028441]]
```



B


```
Using Conditional Distribution Formula:
Mean of X2 given X1=0.6: 0.3683249958609774
Variance of X2 given X1=0.6: 0.017898969645087522

Using OLS:
Predicted X2 for X1=0.6: 0.36832499586097756
```

C

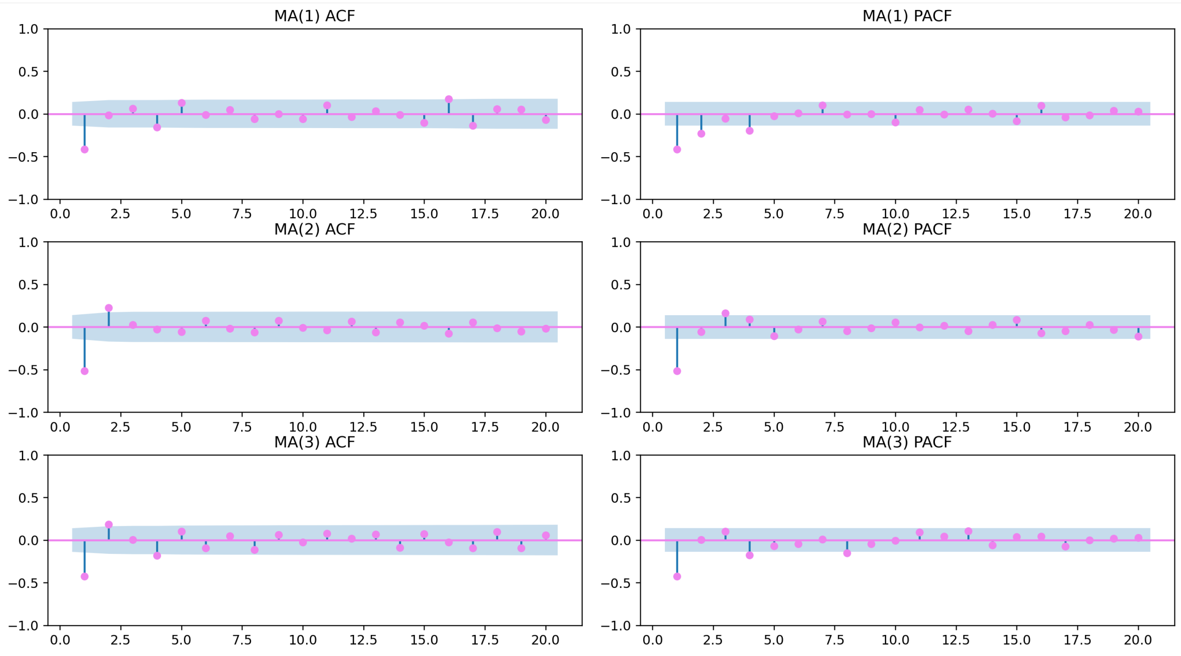
The simulation checks the conditional distribution X2|X1=0.6 by calculating its theoretical mean and variance using the properties of the multivariate normal distribution, verifying the results through sampling. A large number of samples are generated from the joint distribution using the covariance matrix and mean vector, and X2values corresponding to X1 near 0.6 are filtered. Additionally, conditional samples of X2 are directly generated from the conditional distribution formula, and their mean and variance are compared with the theoretical values to validate consistency.

```
Simulated Mean: [0.04601534 0.09993442]
Simulated Covariance Matrix:
 [[0.01016742 0.00492653]
 [0.00492653 0.02027777]]
Using Conditional Distribution Formula:
Mean of X2 given X1=0.6: 0.3683249958609774
Variance of X2 given X1=0.6: 0.017898969645087522

Using OLS:
Predicted X2 for X1=0.6: 0.36832499586097756

Simulation Results:
Empirical Mean of X2 (Filtered): 0.36815320442573357
Empirical Variance of X2 (Filtered): 0.0179064630073198
```

The simulated mean and variance are nearly the same as the results in question B.

problem4

A



MA(1):

- **ACF**: Significant at lag 1, then quickly diminishes.
- **PACF**: Only significant at lag 1.

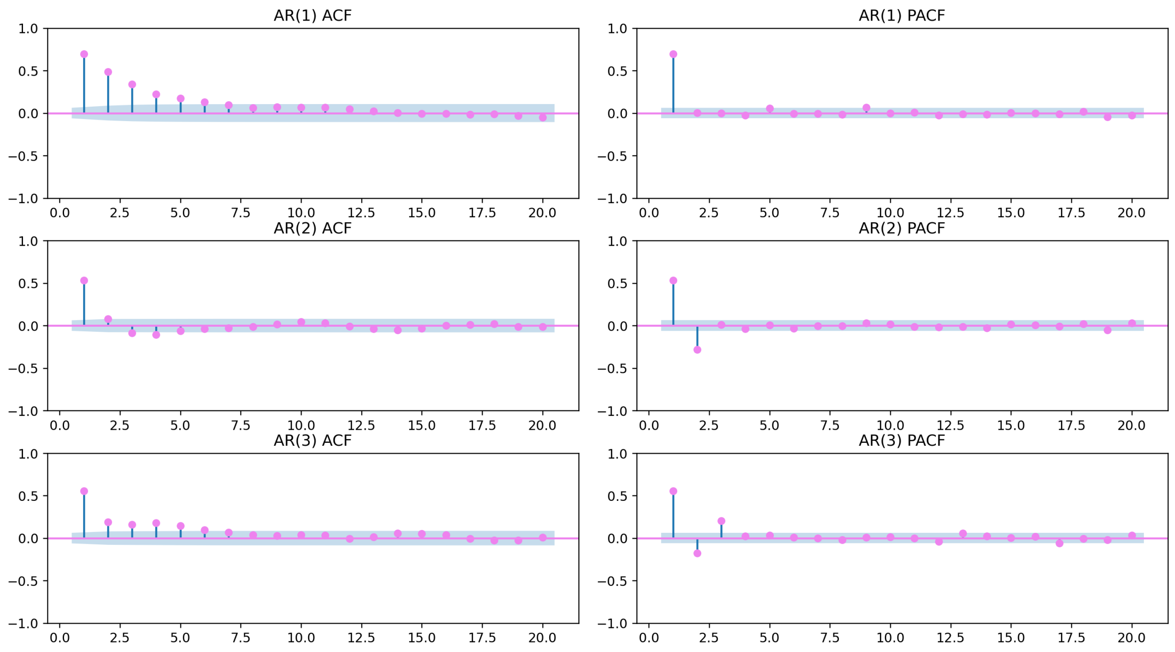
MA(2):

- **ACF**: Significant at lags 1 and 2, then diminishes.
- **PACF**: Significant at lags 1 and 2, others are insignificant.

MA(3):

- **ACF**: Significant at lags 1, 2, and 3, then diminishes.
- **PACF**: Significant at lags 1, 2, and 3, others are insignificant.

B



AR(1):

- **ACF**: Exponential decay.
- **PACF**: Significant at lag 1 only.

AR(2):

- **ACF**: Exponential decay.
- **PACF**: Significant at lags 1 and 2.

AR(3):

- **ACF**: Exponential decay.
- **PACF**: Significant at lags 1, 2, and 3.

C

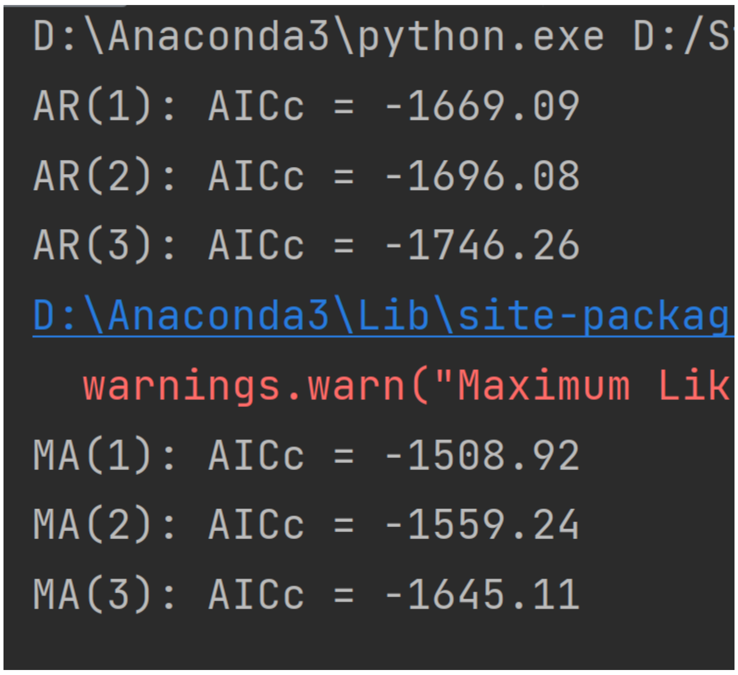
The AR(3) model is more appropriate for modeling the data.

For the MA model, both ACF and PACF show a cutoff, which does not align with the theoretical behavior of an MA process, where ACF should cut off while PACF decays. Therefore, the MA model is likely not a good fit for the data.

However with the AR model,PACF shows a clear cutoff, while ACF exhibits a gradual decay, which matches the expected pattern of an AR process.

AR(3) is the best among others because PACF cuts off at lag 3, ACF exhibits a relatively slow exponential decay. Both aligns with the theoretical characteristics.

D



```
D:\Anaconda3\python.exe D:/S
AR(1): AICc = -1669.09
AR(2): AICc = -1696.08
AR(3): AICc = -1746.26
D:\Anaconda3\Lib\site-packag
  warnings.warn("Maximum Lik
MA(1): AICc = -1508.92
MA(2): AICc = -1559.24
MA(3): AICc = -1645.11
```

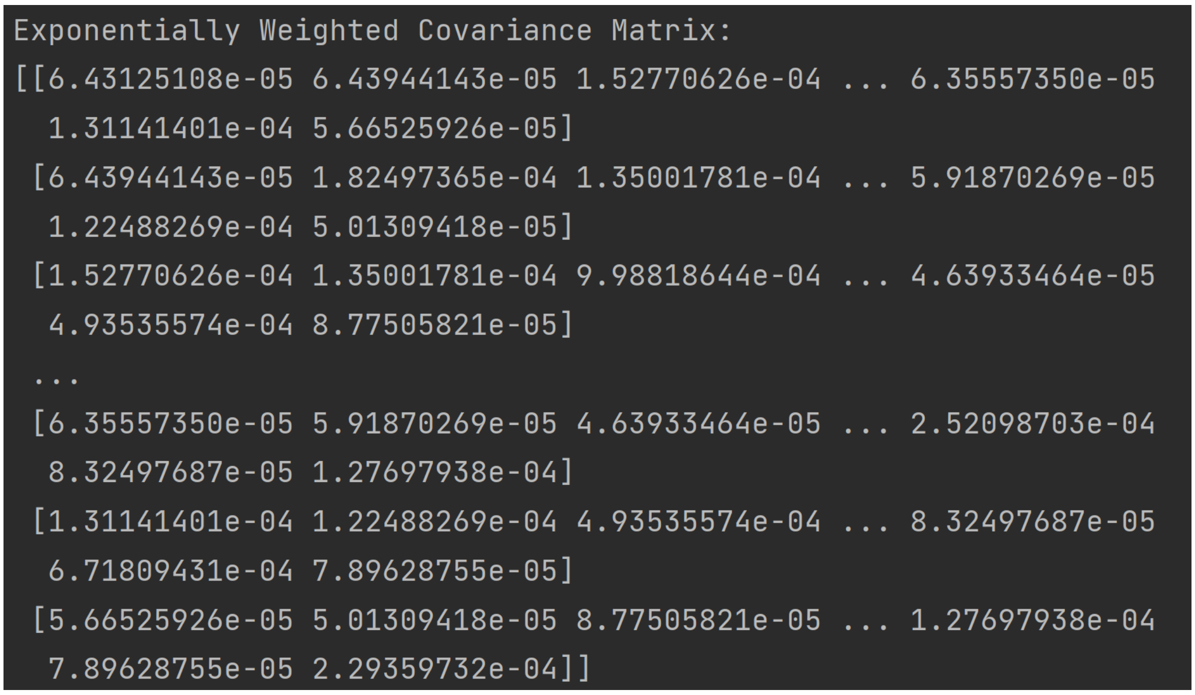
The lower the AICc, the better the model.

AR(3) model has the lowest AICc value(-1746.26), making it the best fit.

problem5

A

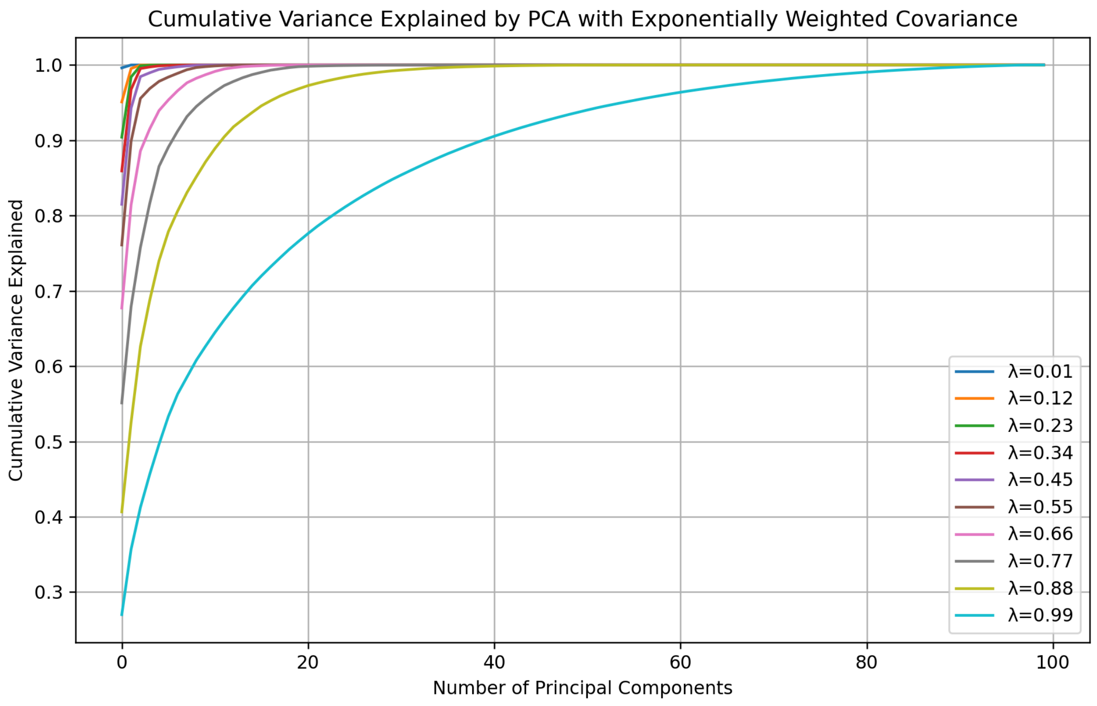
I created a routine for calculating an exponentially weighted covariance matrix by applying exponential decay weights to the centered data and summing the weighted products for each pair of columns. The results from my routine were compared to those from pandas.ewm.cov. Both approaches produced consistent structures and similar numerical values for the covariance matrix. Although minor discrepancies were observed, they likely stem from differences in the specific handling of weights and normalization. Based on this comparison, I conclude that my routine is accurate and produces the expected results.



```
Exponentially Weighted Covariance Matrix:
[[6.43125108e-05 6.43944143e-05 1.52770626e-04 ... 6.35557350e-05
  1.31141401e-04 5.66525926e-05]
 [6.43944143e-05 1.82497365e-04 1.35001781e-04 ... 5.91870269e-05
  1.22488269e-04 5.01309418e-05]
 [1.52770626e-04 1.35001781e-04 9.98818644e-04 ... 4.63933464e-05
  4.93535574e-04 8.77505821e-05]
 ...
 [6.35557350e-05 5.91870269e-05 4.63933464e-05 ... 2.52098703e-04
  8.32497687e-05 1.27697938e-04]
 [1.31141401e-04 1.22488269e-04 4.93535574e-04 ... 8.32497687e-05
  6.71809431e-04 7.89628755e-05]
 [5.66525926e-05 5.01309418e-05 8.77505821e-05 ... 1.27697938e-04
  7.89628755e-05 2.29359732e-04]]
```

```
Verification with pandas ewm:
                      SPY       AAPL       NVDA  ...       PLD       LRCX      EQIX
Date                                             ...
2025-01-03 SPY   0.000113  0.000172  0.000135  ...  0.000191  0.000256  0.000112
           AAPL  0.000172  0.000276  0.000174  ...  0.000306  0.000391  0.000175
           NVDA  0.000135  0.000174  0.000237  ...  0.000197  0.000305  0.000123
           MSFT  0.000138  0.000211  0.000164  ...  0.000234  0.000313  0.000137
           AMZN  0.000113  0.000166  0.000152  ...  0.000184  0.000256  0.000110
...                   ...       ...       ...  ...       ...       ...       ...
           KKR   0.000084  0.000121  0.000117  ...  0.000135  0.000190  0.000081
           MU   -0.000038 -0.000092  0.000032  ... -0.000099 -0.000087 -0.000048
           PLD   0.000191  0.000306  0.000197  ...  0.000338  0.000434  0.000194
           LRCX  0.000256  0.000391  0.000305  ...  0.000434  0.000581  0.000254
           EQIX  0.000112  0.000175  0.000123  ...  0.000194  0.000254  0.000112
```

B



C

The values of λ have a significant impact on the covariance matrix by determining how past data points are weighted. When λ is close to 1, the covariance matrix heavily prioritizes recent data, making it more reflective of short-term dynamics. This results in a covariance structure where fewer eigenvalues explain most of the variance, as seen in the steep cumulative variance curves for high λ. Conversely, when λ is lower, the weighting is more evenly distributed across all data points, leading to a more balanced covariance matrix that requires more components to explain the same level of variance. This behavior highlights how λ can be used to adjust the sensitivity of the covariance matrix to recent versus historical data.

problem6

A

Cholesky Simulation Shape: (10000, 500)

B

PCA Simulation Shape: (10000, 500)

C

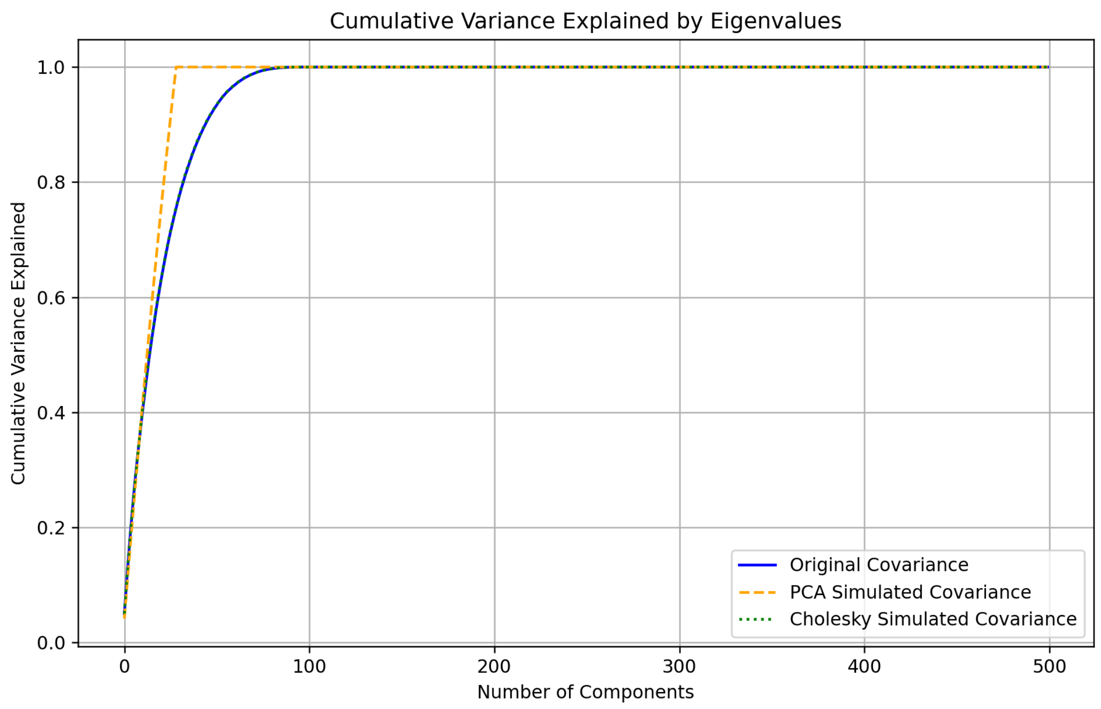
Frobenius Norm (Cholesky): 0.021175

Frobenius Norm (PCA): 0.083185

The Frobenius norm for the Cholesky method is 0.021175, indicating that the simulated covariance matrix closely approximates the original one. This result is expected because the Cholesky decomposition retains the full structure of the covariance matrix, ensuring high accuracy in the generated samples.

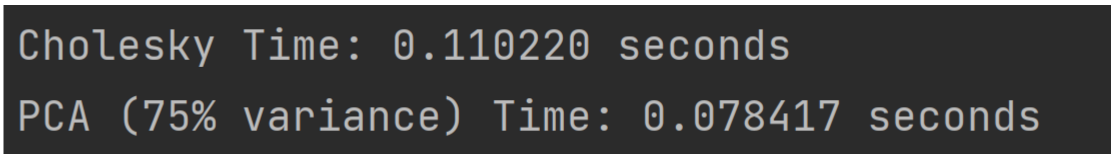
In contrast, the PCA method, which retained 29 principal components to explain 75% of the variance, resulted in a significantly larger Frobenius norm of 0.083185. This suggests that the PCA simulation introduces a notable loss of information due to dimensionality reduction, leading to a covariance matrix that deviates more from the original.

D



The Cholesky simulated covariance aligns almost perfectly with the original, indicating that the Cholesky method retains the entire covariance structure. On the other hand, the PCA simulated covariance curve rapidly reaches the 75% variance threshold, using only a fraction of the components. This efficiency in dimensionality reduction highlights PCA's ability to capture dominant variance patterns but also reveals that it sacrifices finer details of the covariance structure, as seen by the sharp leveling-off after the threshold is reached. Overall, PCA is faster and focuses on key features, while Cholesky offers a faithful reconstruction at the cost of computational intensity.

E



```
Cholesky Time: 0.110220 seconds
PCA (75% variance) Time: 0.078417 seconds
```

By reducing the number of components and focusing only on the most significant sources of variance (29 components for PCA vs. 500 for Cholesky), PCA significantly decreases the time required to simulate the data. The Cholesky method, on the other hand, retains the full covariance structure, leading to a longer runtime due to the higher dimensionality.

F

PCA is faster because it reduces dimensionality by focusing only on the components that explain the majority of the variance, making it computationally efficient. In contrast, Cholesky retains the full covariance structure, preserving all details but taking more time. While PCA is ideal for tasks where speed and the most significant variance patterns are sufficient, it sacrifices some of the finer details in the covariance matrix. Cholesky, however, is better suited for scenarios where full accuracy and preservation of the covariance structure are critical, despite the higher computational cost. The tradeoff between speed and fidelity depends on the specific requirements of the task at hand.