1. 1.
   Mean: 0.050198
   By formula:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Variance: 0.010332
By formula:

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$$

Skewness:0.120626
By formula:

$$S = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^3}{\sigma^3}$$

Kurtosis: 0.23007
By formula:

$$K = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^4}{\sigma^4}$$

B. I will choose Normal-Distribution, since skweness and kurtosis is just above 0. This means that there's no obvious bias.

C. We compute the AICC for both distributions and end up with

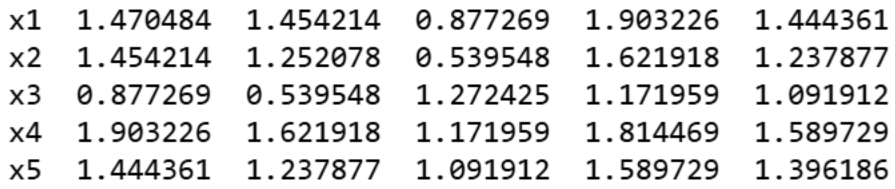
Normal-Distribution: -1731.574692728183

T-Distribution: -1731.394272534012

Since Normal-Distriibution is less than T-Distribution, so choose Normal-Distribution is good.

2. A. By formula

$$\mathrm{Cov}(X_i, X_j) = \frac{1}{n-1}\sum_{k=1}^{n}(X_{ki} - \mu_i)(X_{kj} - \mu_j)$$



```
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. For this question, we have to calculate the eigenvalues of the matrix, if all the eigenvalues is larger or equal to 0, then it is a least positive semi-difinite. After we run our python code, we get the eigenvalues: -0.31024286 -0.13323183 0.02797828 0.83443367 6.78670573. There are few values less than 0, so not least positive semi-difinite.

C. Higham’s Nearest PSD Method

a. we symmetrization matrix

$$B = \frac{A + A^T}{2}$$

b. then we get eigen-decomposition

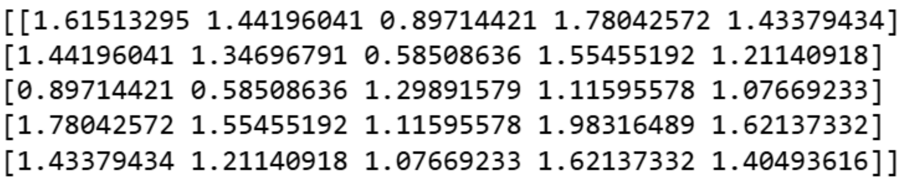
$$B = Q \Lambda Q^T$$

c. We replace every negative eigenvalue to 0, keep the positive value or 0.

d. We modify the matrix

$$A_{\text{PSD}} = Q \Lambda_{\text{modified}} Q^T$$

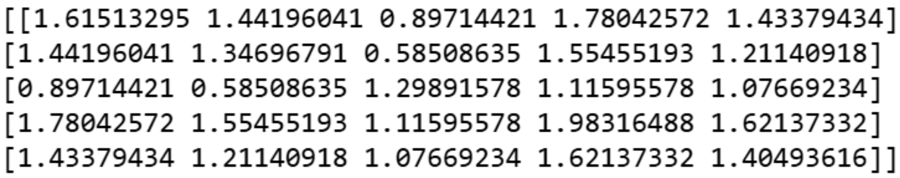
e. Then we will get matrix



```
[[1.61513295 1.44196041 0.89714421 1.78042572 1.43379434]
 [1.44196041 1.34696791 0.58508636 1.55455192 1.21140918]
 [0.89714421 0.58508636 1.29891579 1.11595578 1.07669233]
 [1.78042572 1.55455192 1.11595578 1.98316489 1.62137332]
 [1.43379434 1.21140918 1.07669233 1.62137332 1.40493616]]
```

Rebenato and Jackel

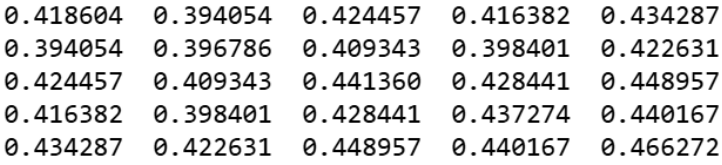
We simply replace every negative eigen-value to 0.



```
[[1.61513295 1.44196041 0.89714421 1.78042572 1.43379434]
 [1.44196041 1.34696791 0.58508635 1.55455193 1.21140918]
 [0.89714421 0.58508635 1.29891578 1.11595578 1.07669234]
 [1.78042572 1.55455193 1.11595578 1.98316488 1.62137332]
 [1.43379434 1.21140918 1.07669234 1.62137332 1.40493616]]
```

D. If there is missing data in that row, we use dropna() to delete that row, only remain those raws with no missing data.

Then we use the same formula in part A



```
0.418604  0.394054  0.424457  0.416382  0.434287
0.394054  0.396786  0.409343  0.398401  0.422631
0.424457  0.409343  0.441360  0.428441  0.448957
0.416382  0.398401  0.428441  0.437274  0.440167
0.434287  0.422631  0.448957  0.440167  0.466272
```

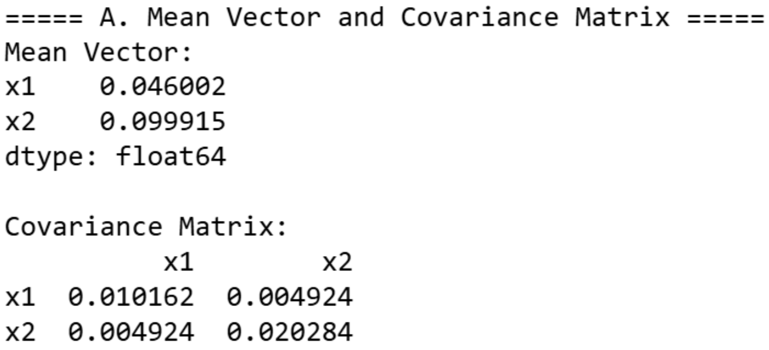
E.

1. Option C utilizes more data than option D, resulting in higher accuracy.
2. Higham's method demonstrates better convergence and yields the most accurate results.
3. The Rebonato and Jackel method is the simplest and fastest, but it excludes negative eigenvalues.
4. The overlapping approach relies on less data, leading to a smaller covariance.

3.

A.

Accourding the formula I mentioned in the previous questions. We cant get:



```
===== A. Mean Vector and Covariance Matrix =====
Mean Vector:
x1    0.046002
x2    0.099915
dtype: float64

Covariance Matrix:
          x1        x2
x1  0.010162  0.004924
x2  0.004924  0.020284
```

B.

Base on formula:

$$\mu_{X_2|X_1=0.6} = \mu_2 + \frac{\sigma_{12}}{\sigma_{11}}(0.6 - \mu_1)$$

$$\sigma^2_{X_2|X_1=0.6} = \sigma_{22} - \frac{\sigma^2_{12}}{\sigma_{11}}$$

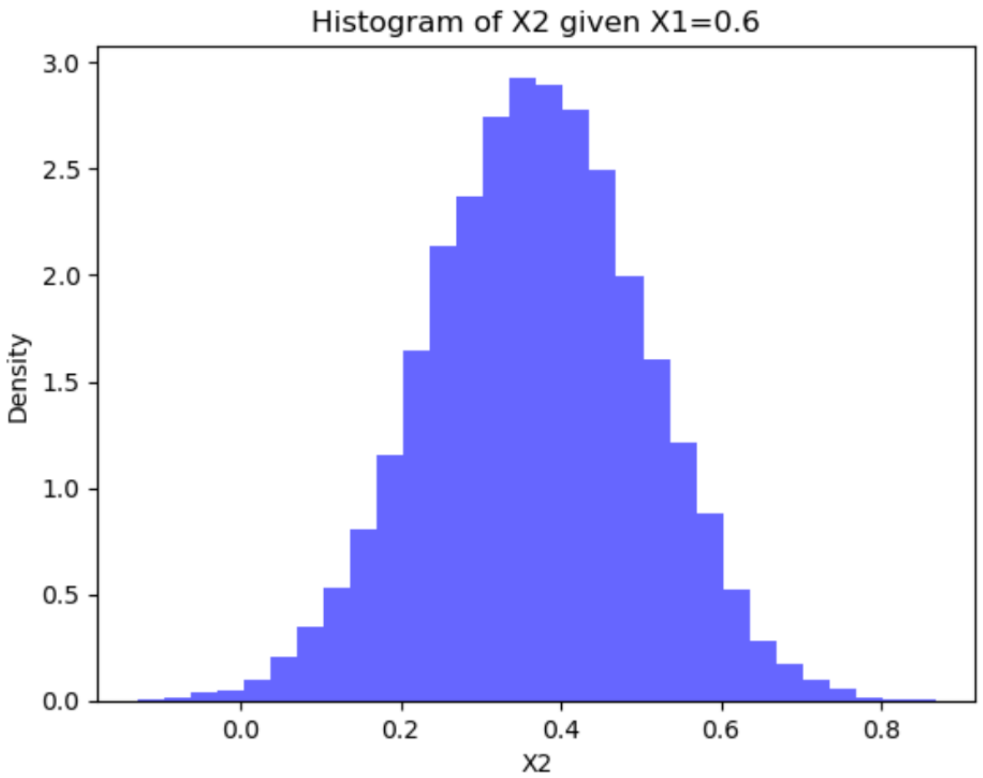
Base on OLS:

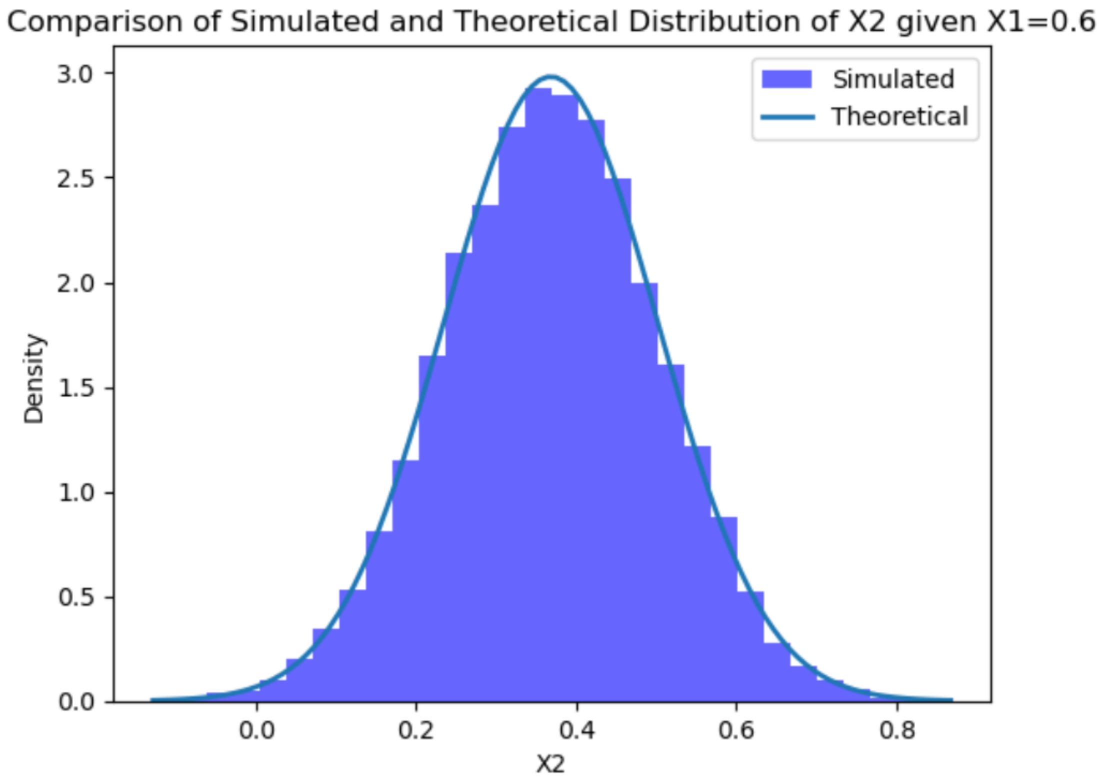
$$\mu_{X_2|X_1=0.6} = \beta_0 + \beta_1 \times 0.6$$

C.

Sample Mean: 0.3687

Sample Variance: 0.0180





4. A. ACF (Autocorrelation Function) Observations:

MA(1), MA(2), MA(3):

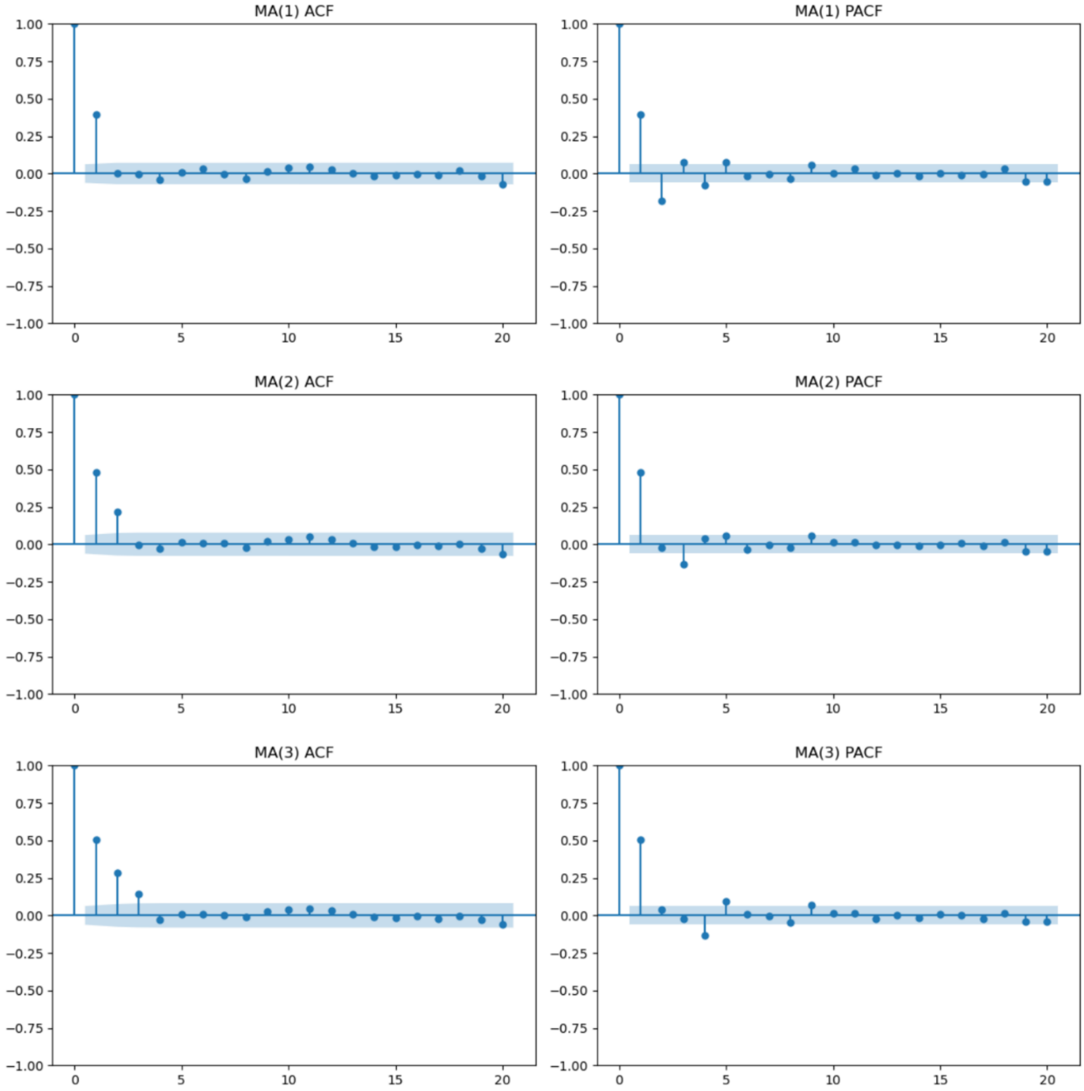
ACF: Cuts off after lag 1, 2, and 3 respectively.

PACF: Tails off gradually without sharp cutoffs.

AR(1), AR(2), AR(3):

PACF: Cuts off after lag 1, 2, and 3 respectively.

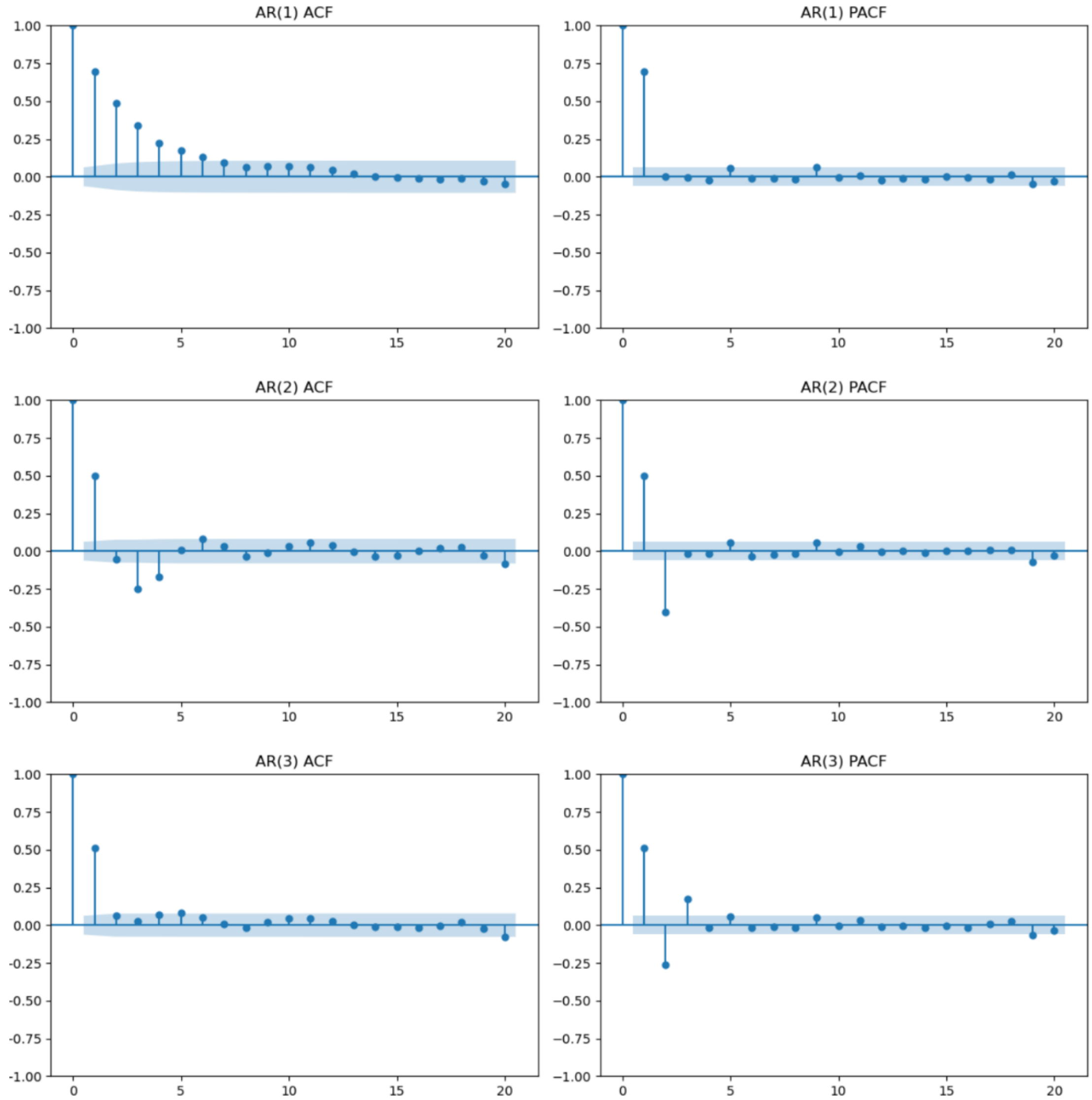
ACF: Tails off gradually, showing exponential decay.



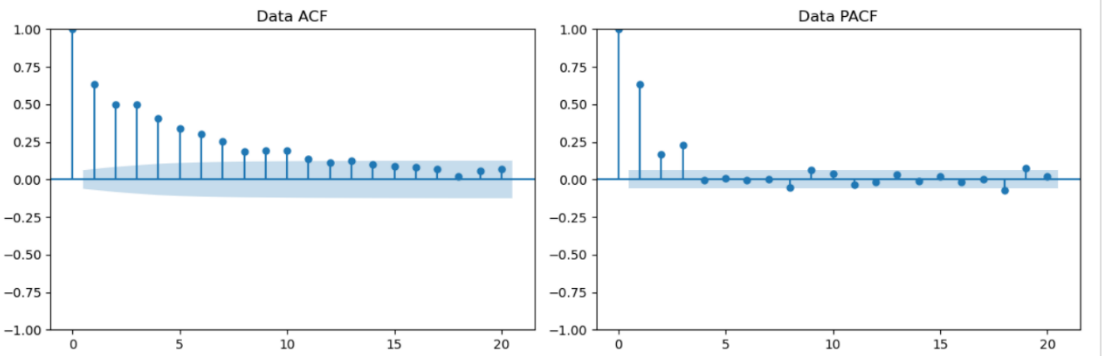
B:

ACF: Gradually decays, indicating persistent autocorrelation.

PACF: Cuts off sharply after lag 3.



C.

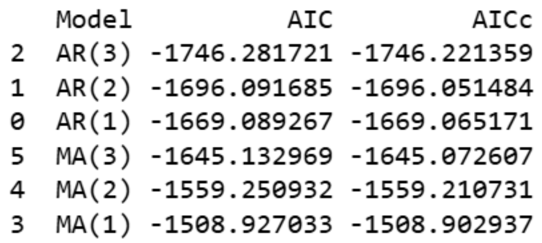


Based on ACF and PACF:

PACF cuts off at lag 3 and ACF tails off.

Best Fit: The data is best modeled by an AR(3) process.

D.



```
   Model          AIC          AICc
2  AR(3) -1746.281721 -1746.221359
1  AR(2) -1696.091685 -1696.051484
0  AR(1) -1669.089267 -1669.065171
5  MA(3) -1645.132969 -1645.072607
4  MA(2) -1559.250932 -1559.210731
3  MA(1) -1508.927033 -1508.902937
```

Models fitted: AR(1), AR(2), AR(3), MA(1), MA(2), MA(3).

AR(3) has the lowest AICc, indicating the best model fit.

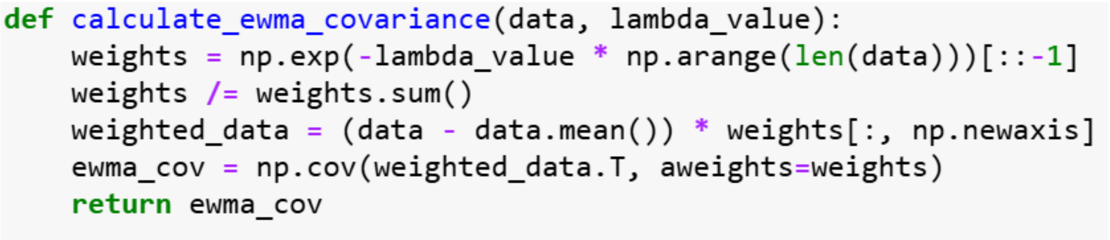
Final Conclusion:

Best Model: AR(3)

Reason: It balances model complexity and accuracy, confirmed by the lowest AICc value.

5

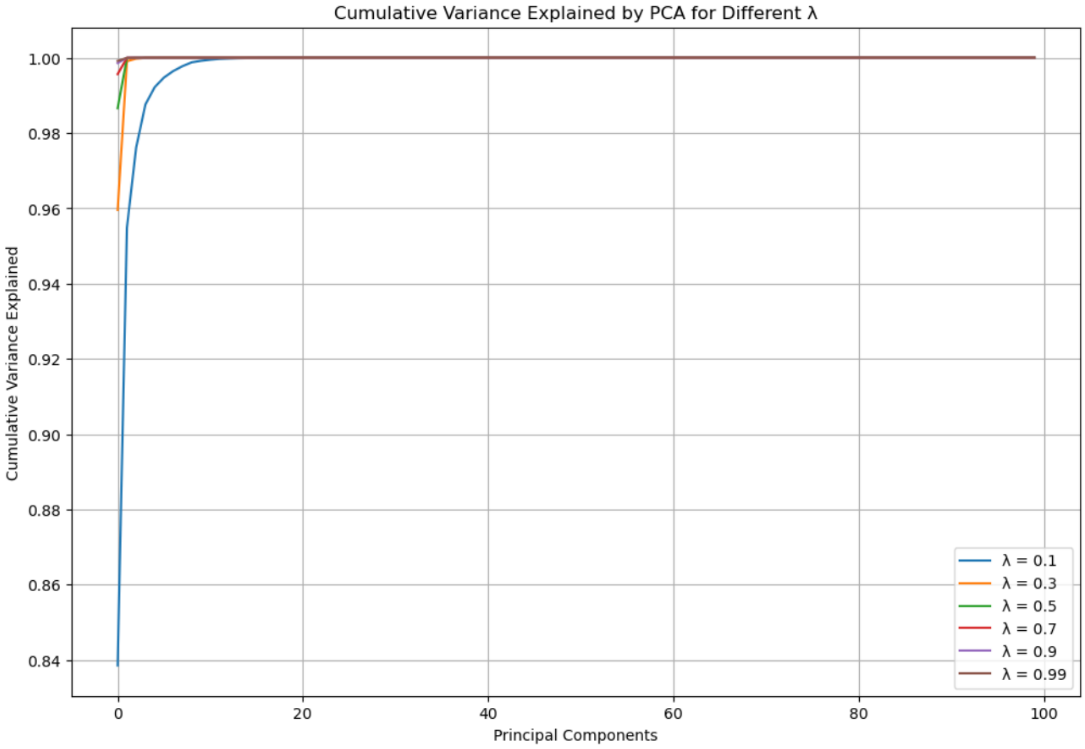
A.



```
def calculate_ewma_covariance(data, lambda_value):
    weights = np.exp(-lambda_value * np.arange(len(data)))[::-1]
    weights /= weights.sum()
    weighted_data = (data - data.mean()) * weights[:, np.newaxis]
    ewma_cov = np.cov(weighted_data.T, aweights=weights)
    return ewma_cov
```

1. np.exp(-lambda_value * np.arange(len(data)))[::-1] generates weights that decay over time, giving more importance to recent data.
2. weights /= weights.sum() ensures the weights sum to 1 for numerical stability.
3. (data - data.mean()) * weights[:, np.newaxis] subtracts the mean and applies weights to highlight recent data's influence.
4. np.cov(weighted_data.T, aweights=weights) computes the weighted covariance matrix, reflecting time-dependent relationships.
5. Outputs the EWMA covariance matrix for use in volatility modeling or risk analysis.

B. We set λ to following numbers to draw the image: 0.1, 0.3, 0.5, 0.7, 0.9, 0.99.



C. As λ increases from 0.1 to 0.99, the maximum eigenvalue of the covariance matrix grows from 0.0000 to 0.0050, with the minimum eigenvalue always close to zero.

When λ is small, the covariance matrix is almost zero, reflecting that the model overly relies on the latest data and ignores historical fluctuations.

As λ increases, the model starts to take more historical information into account, and the volatility gradually appears and the covariance matrix becomes more stable.

This suggests that λ selection is crucial in risk management and can balance the sensitivity to short-term and long-term volatility.

6.

A. We use Cholesky decomposition to factor the covariance matrix into a lower triangular matrix LLL. The simulation of 10,000 multivariate normal samples is done using the formula:

$$X = Z \cdot L^T$$

Where:

ZZZ is a random matrix from the standard normal distribution (dimensions: 10,000 × 500).

LLL is the lower triangular matrix obtained from Cholesky decomposition.

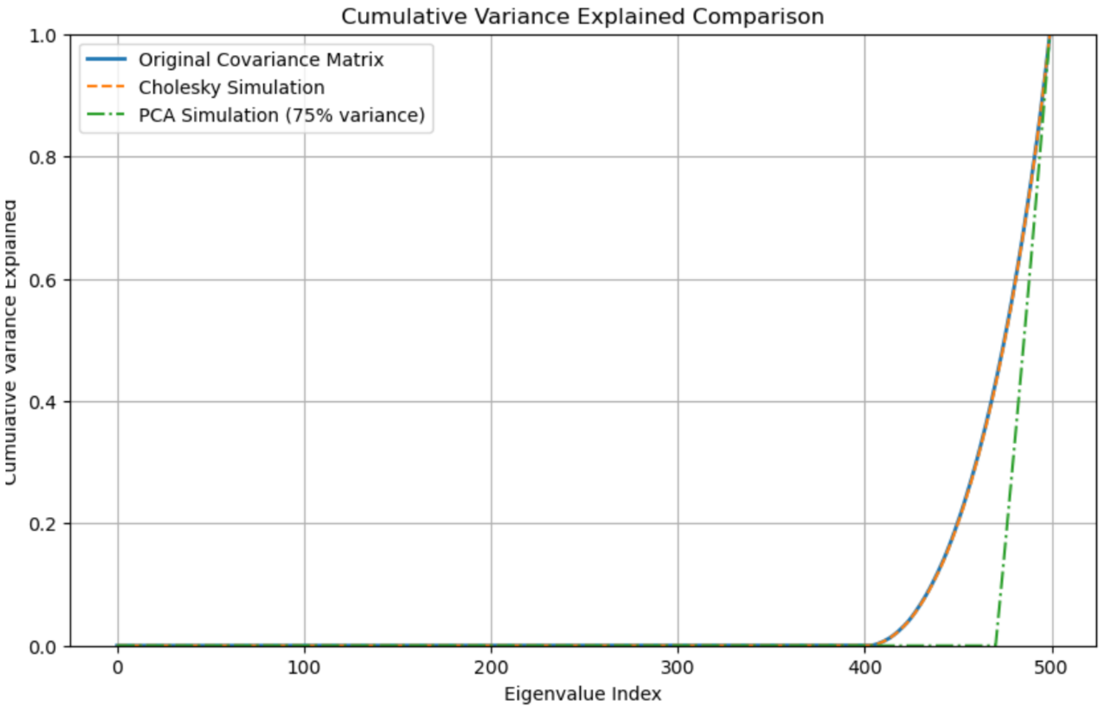
B. We apply Principal Component Analysis (PCA) to reduce dimensionality, retaining components that explain 75% of the total variance. The samples are then reconstructed in the original space using inverse transformation.

C. Frobenius Norm Formula:

$$\|A - B\|_F = \sqrt{\sum_{i,j}(A_{ij} - B_{ij})^2}$$

According to the formula, I notice that the Cholesky method shows a smaller Frobenius norm, indicating that the simulated data closely matches the original covariance matrix.
The PCA method has a larger Frobenius norm because dimensionality reduction causes some information loss.

D.



Original Covariance Matrix: The cumulative variance curve smoothly increases, eventually approaching 1.
Cholesky Simulation: The curve closely overlaps with the original, indicating high accuracy.
PCA Simulation: The curve plateaus after explaining 75% of the variance, reflecting the loss of variance from omitted components.

E.
Frobenius norm differences:
Cholesky method: 0.021325282569965392
This is a very small value, indicating that the covariance matrix generated by the Cholesky method is very close to the original covariance matrix.
This is expected because Cholesky decomposition fully retains the original covariance structure without losing any information.

PCA method: 5.184857190502035

This value is significantly larger, suggesting that the covariance matrix from the PCA simulation differs more from the original matrix.
This result makes sense because PCA involves dimensionality reduction, retaining only 75% of the variance, which leads to information loss and affects the accuracy of the reconstructed covariance matrix.

Simulation execution times (seconds):
Cholesky method: 0.13988947868347168
The execution time is relatively longer because Cholesky decomposition requires complex matrix factorization and large matrix operations, especially with high-dimensional data (e.g., a 500 × 500 matrix), which increases computational complexity.

PCA method: 0.05664825439453125

The execution time is shorter because PCA performs dimensionality reduction, which reduces the computational workload. After retaining only 75% of the variance, the matrix size becomes smaller, resulting in faster calculations.

F.
Cholesky Decomposition
Advantages:

- High Accuracy: Minimal Frobenius norm difference (0.0213), accurately preserves the original covariance structure.
- Full Covariance Retention: Maintains all variable relationships without information loss.

Disadvantages:

- Computationally Intensive: Slower (0.1399 seconds) due to complex matrix operations.
- Less Efficient for Large Datasets: Performance decreases with high-dimensional data.

PCA (Principal Component Analysis)
Advantages:

- Fast Computation: Faster (0.0566 seconds) due to dimensionality reduction.
- Efficient for Large Datasets: Reduces computational load while capturing key variance.

Disadvantages:

- Information Loss: Larger Frobenius norm difference (5.1849) due to reduced variance.
- Incomplete Covariance Reconstruction: Doesn't fully retain original data structure.

Conclusion:

- Cholesky: Best for small to medium datasets requiring high accuracy.
- PCA: Suitable for large datasets where speed is prioritized, with acceptable variance loss.