- Problem 1

  A:

  mean is 0.05019795790476916
  variance is 0.010332476407479588
  skewness is 0.1206257259522635
  kurtosis is -2.7699301868297135

  B:

  The normal distribution is more suitable for this dataset as the size of the dataset is 1000 which is quite large. When the sample size is large, the Central Limit Theorem states that the sampling distribution of the mean approaches a normal distribution, regardless of the underlying population distribution.

  C:

  1. AIC (Akaike Information Criterion):
     Normal Distribution: AIC = -1731.586728836508
     T-Distribution: AIC = -1731.4183689195256
     Conclusion: The normal distribution has a slightly lower AIC value, indicating a better balance between model fit and complexity.
  2. Kolmogorov-Smirnov (K-S) Test:
     Normal Distribution:
     Statistic: 0.01275165971875869
     P-value: 0.9962422791034079
     T-Distribution:
     Statistic: 0.012697924916462489
     P-value: 0.9964533691214174
     Conclusion: Both distributions have very high p-values, indicating that the sample distribution is very close to both the normal and t-distributions.

  Final conclusion:

  The result proves my choice in B. Based on the AIC values, the normal distribution is slightly preferred over the t-distribution.

- Problem 2

  A:

  pairwise covariance matrix

	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:

  eigenvalues: [6.78670573 0.83443367 -0.31024286 0.02797828

-0.13323183]
Since not all eigenvalues of the matrix are non-negative, the pairwise covariance matrix is not positive semi-definite.
C:

```
Nearest PSD Matrix using Higham's Method:
[[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]]

Nearest PSD Matrix using Rebonato & Jackel's Method:
[[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:
covariance matrix using only 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:
The covariance matrix calculated using only overlapping data shows smaller and more consistent values compared to the nearest positive semi-definite (PSD) matrices obtained using Higham's Method and the Rebonato and Jackel Method. This is because the overlapping data likely has less variability and fewer extreme values. In contrast, the nearest PSD matrices, which are identical in this case, exhibit larger values and greater variability. These matrices are adjusted to be positive semi-definite and reflect higher correlations, as assumed in the generating process with ones on the diagonals and high correlations (close to one) elsewhere. Thus, the covariance matrix using overlapping data represents the actual variability in the non-NaN data, while the nearest PSD matrices ensure positive semi-definiteness and higher correlations.

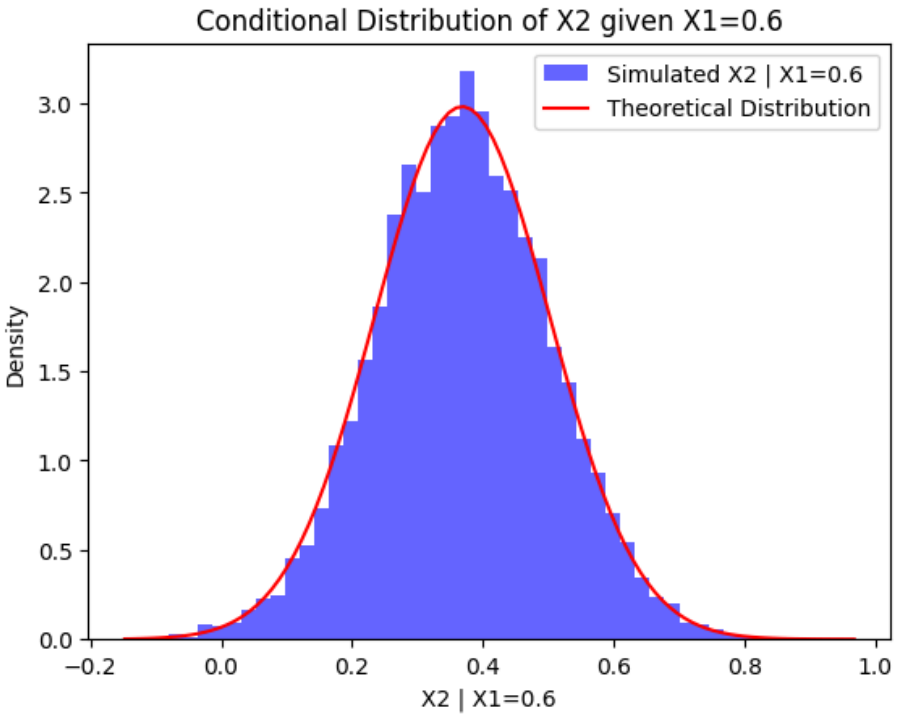
- Problem 3
  A:
  For the multivariate normal distribution:
  Mean: [0.04600157 0.09991502]
  Covariance Matrix:
  [[0.0101622 0.00492354]

[0.00492354 0.02028441]]
B:
To find out the distribution of X2 given X1=0.6, I used the analytical formula method and the regression method.

Analytical Formula:
Conditional Mean of X2 | X1 = 0.6: 0.3683249958609775
Conditional Variance of X2 | X1 = 0.6: 0.017898969645087522

Regression:
Conditional Mean of X2 | X1 = 0.6: 0.3683249958609775
Conditional Variance of X2 | X1 = 0.6: 0.017898969645087522

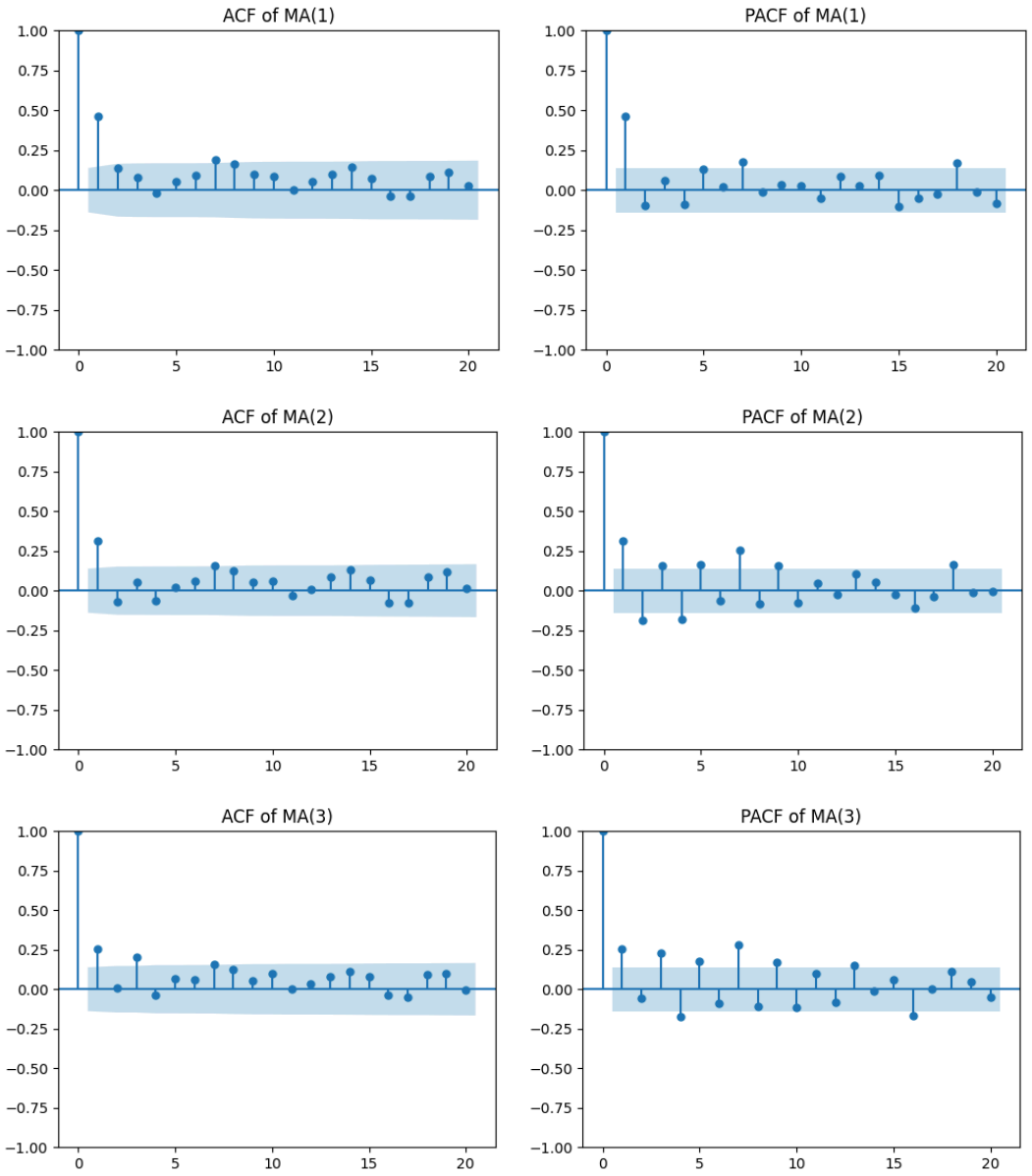
Conclusion: the expected value of X2 when X1 is 0.6 is approximately 0.3683, with a variance of 0.0179.
C:



The simulated histogram closely matches the theoretical PDF, confirming that the Cholesky-based sampling correctly reproduces the conditional distribution, thereby validating that the distribution derived in Part B is correct.

- Problem 4

  A:
  To make sure the figures below are the same every time it runs, I set random seed = 0.
  Simulate MA(1),MA(2), MA(3) processes:
  MA(1) parameters: 0.5
  MA(2) parameters: 0.5, -0.3
  MA(3) parameters: 0.5, -0.3, 0.2



B:

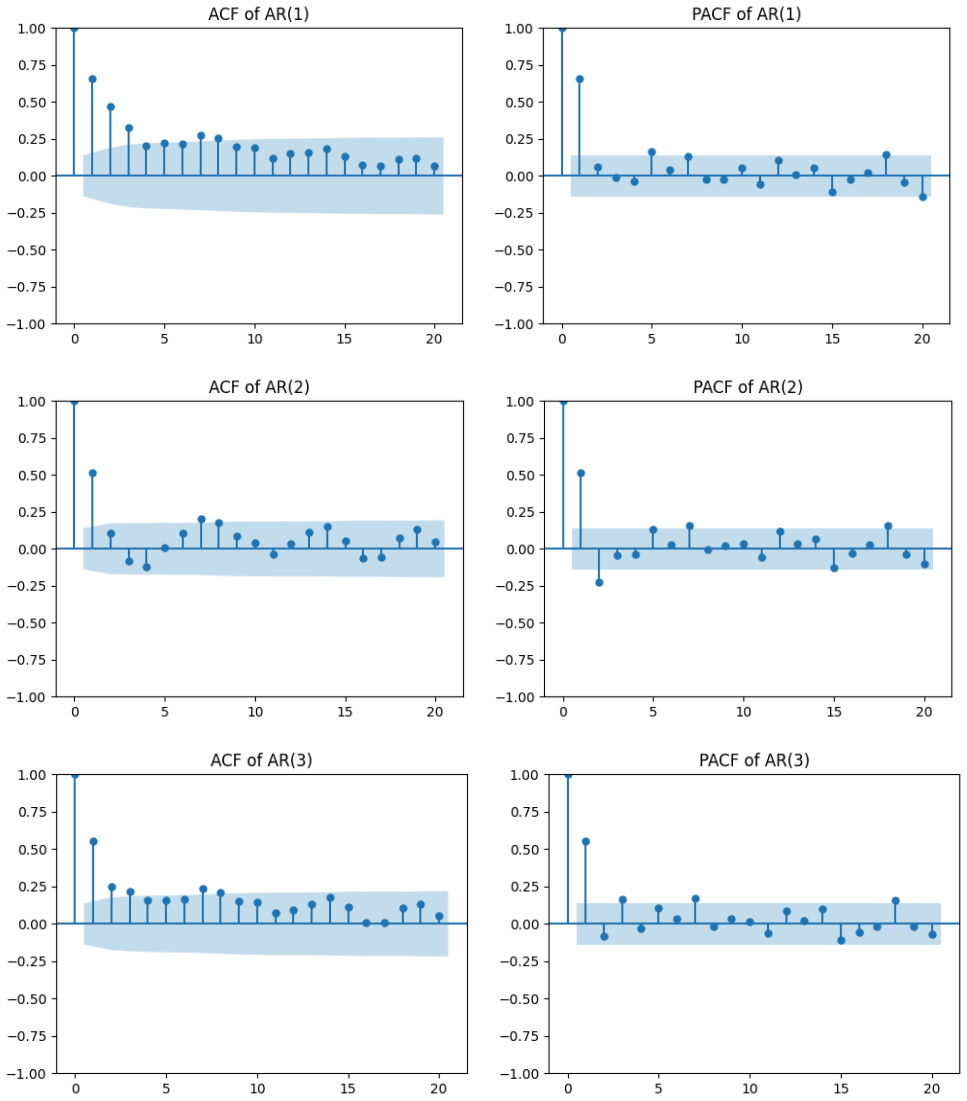
To make sure the figures below are the same every time it runs, I set random seed = 0.

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

AR(1) parameters: 0.6

AR(2) parameters: 0.6, -0.3

AR(3) parameters: 0.6, -0.3, 0.2

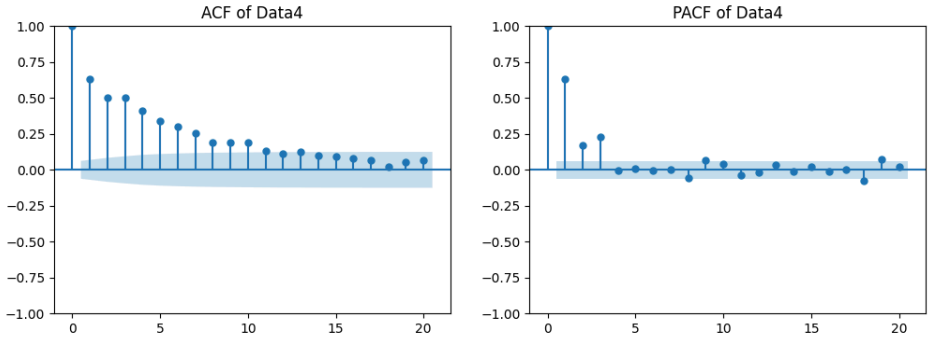


C:
The data4 is best modeled using an AR(3) process.

The ADF test has a p-value of 1.2934437199813811e-13 which is less than 0.05, so it confirms that the data is stationary, meaning no differencing is required.

The PACF shows significant spikes at Lags 1, 2, and 3, indicating strong dependence on the first three lagged values, while the PACF drops to zero at Lag = 4, suggesting no significant influence beyond the third lag. The ACF exhibits a gradual decay rather than an immediate cut-off, which is typical of an autoregressive process rather than a moving average process. Since there is no evidence of moving average behavior, a pure AR(3) model is appropriate and avoids overfitting.

Given the PACF cut-off at lag 3 and the gradual decay in ACF, the AR(3) model is the most appropriate choice.



D:
I defined several candidate models that are close to my initial predictions and fitted them to the data. For each model, I calculated the AICc to evaluate their performance. The AICc values were then compared to identify the best-fitting model, which is the one with the lowest AICc.

AICc values for different models:
AR(3): -1746.22
ARMA(3,1): -1744.23
AR(4): -1744.22
ARMA(3,2): -1742.26
AR(5): -1742.25
ARMA(2,2): -1739.96
ARMA(2,1): -1725.41
ARMA(1,1): -1723.41
AR(2): -1696.05
AR(1): -1669.07
MA(3): -1645.07
MA(2): -1559.21
MA(1): -1508.90

Best fitting model: AR(3) with AICc = -1746.22

- Problem 5

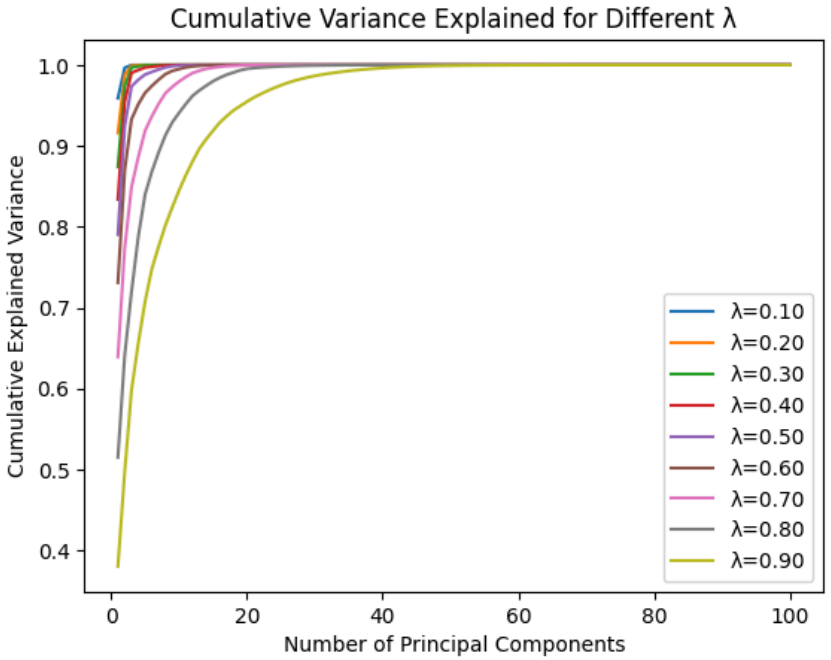
  A:
  The routine ewma_cov_matrix calculates an exponentially weighted moving average covariance matrix by computing weights based on a decay factor (lambda_val), determining weighted mean returns, and scaling deviations to produce the final covariance matrix. To validate its accuracy, the results were compared with the pandas.ewm.cov method, yielding a MAE of 5.8748931986707075e-05. This small error, within a reasonable range, confirms that the routine is correct and produces expected results.

```
Manual EWMA Covariance Matrix:
 [[ 1.38773153e-04 -2.07928163e-06  8.60182115e-05 ...  8.08341038e-05
   5.89540072e-05  3.87773331e-05]
 [-2.07928163e-06  5.72367208e-04  1.07296400e-04 ...  3.76716661e-05
   2.47231926e-04  8.07845344e-05]
 [ 8.60182115e-05  1.07296400e-04  1.50577614e-04 ...  7.45132602e-05
   1.42691964e-04  7.10351155e-05]
 ...
 [ 8.08341038e-05  3.76716661e-05  7.45132602e-05 ...  2.53849570e-04
   1.37564641e-04  9.78001838e-05]
 [ 5.89540072e-05  2.47231926e-04  1.42691964e-04 ...  1.37564641e-04
   6.41327036e-04  1.09131803e-04]
 [ 3.87773331e-05  8.07845344e-05  7.10351155e-05 ...  9.78001838e-05
   1.09131803e-04  1.45760988e-04]]


Pandas EWMA Covariance Matrix:
 [[ 1.43202083e-04  3.77976962e-05  2.22125996e-05 ...  3.66049156e-05
   3.93741639e-05 -1.92835938e-06]
 [-2.14564168e-06  6.38189586e-05  2.20538527e-05 ...  9.33568165e-05
   1.97621883e-05  1.53624735e-05]
 [ 8.87634736e-05  4.03785205e-05  1.61339187e-05 ...  7.44276882e-05
   7.85684582e-05 -3.23973854e-06]
 ...
 [ 8.34139157e-05  9.63985228e-05  5.08849478e-05 ...  7.01733719e-05
   2.44667605e-05  6.23132705e-05]
 [ 6.08355180e-05  1.19827285e-04  6.62265902e-07 ...  7.52178226e-05
   3.07173554e-05  4.56498678e-05]
 [ 4.00149076e-05  7.51361673e-05  3.87084855e-05 ...  6.49733204e-05
   5.36588831e-05  3.50848708e-05]]

Mean Absolute Error between manual and Pandas EWMA covariance 5.8748931986707075e-05
```

B: To make sure the figures below are the same every time it runs, I set random seed = 42.



C:

A higher λ smooths out short-term fluctuations and prioritizes long-term stability in the covariance matrix. A lower λ reacts more strongly to recent changes, making it more useful for short-term trading strategies where fast adaptation is needed.

- Problem 6

A:

First I check whether the covariance matrix in data6 is positive definite. The result shows the covariance matrix is not positive definite. To address the issue of near-zero eigenvalues and ensure the matrix is strictly positive definite, I added a small positive value (0.000001) to the diagonal elements (Tikhonov regularization). This adjustment ensures the covariance matrix is positive definite, allowing for the successful application of the Cholesky decomposition method in subsequent steps.

To make sure the figures below are the same every time it runs, I set random seed = 0.

Simulation of 10000 draws using the Cholesky Root method:

```
array([[ 0.03621673, -0.01008121,  0.04722467, ..., -0.0514265 ,
         0.03614757,  0.04698406],
       [-0.02711125, -0.01327327, -0.02325044, ...,  0.01385646,
         0.07362495, -0.06632391],
       [ 0.01681887,  0.02019737, -0.07359091, ..., -0.02610066,
        -0.13131827, -0.02756533],
       ...,
       [ 0.10649973,  0.00221888, -0.02444146, ..., -0.02126386,
         0.02097008,  0.04094775],
       [ 0.10086256, -0.06498925, -0.01422733, ...,  0.07864218,
         0.13312361, -0.00410319],
       [-0.0217295 ,  0.00361992, -0.03766752, ...,  0.02571732,
         0.06879675, -0.03136218]], shape=(500, 10000))
```

B:

To make sure the figures below are the same every time it runs, I set random seed = 0.

Simulation of 10000 draws using the Cholesky Root method:

```
array([[ 0.02697585, -0.09144258,  0.05281014, ...,  0.0898822 ,
        -0.03135391,  0.06659652],
       [-0.00319415,  0.04444234, -0.06837276, ..., -0.00151892,
        -0.01338989, -0.01553911],
       [-0.04891228, -0.05017948, -0.06606618, ...,  0.1148471 ,
        -0.01026722,  0.06154846],
       ...,
       [-0.02915892, -0.04452223, -0.00116936, ...,  0.00864026,
        -0.00392733, -0.02307916],
       [ 0.04572002, -0.01053006, -0.034926  , ...,  0.03000636,
        -0.05388574,  0.04996218],
       [ 0.00183123,  0.00529478, -0.00317145, ..., -0.01533678,
        -0.02225911, -0.04163302]], shape=(500, 10000))
```

C:

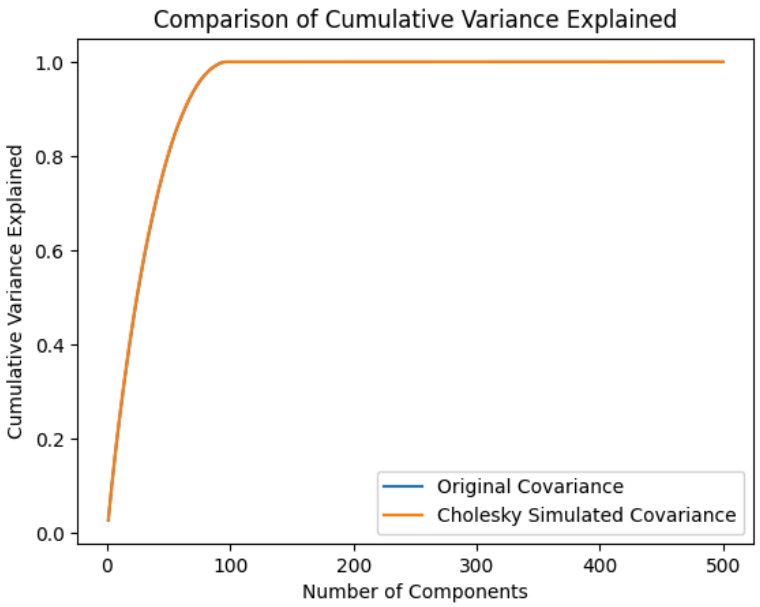
Cholesky Frobenius Norm: 0.021191270918006117
PCA Frobenius Norm: 0.08323375645414555

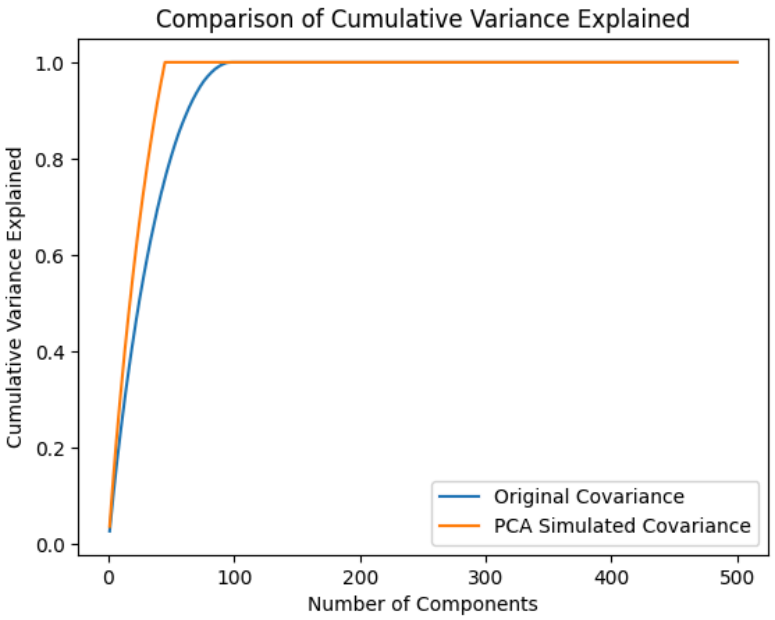
The Cholesky simulation has a lower Frobenius norm (0.021) compared to the PCA simulation (0.083). This indicates that the Cholesky method better

preserves the original covariance structure than PCA. PCA-based simulation, while still capturing the main variance structure, introduces some distortion due to dimensionality reduction.

D:
To compare cumulative variance explained by each eigenvalue of the 2 simulated covariance matrix, I plotted the cumulative variance explained for two simulations and the input matrix. Additionally, I computed the maximum absolute deviation and MSE.

Results:
Max Deviation (Cholesky): 0.003517809112811743
Max Deviation (PCA): 0.24098398688211017
MSE (Cholesky): 5.34979444558622e-07
MSE (PCA): 0.0034181845250586913





The results show significant differences in performance between the Cholesky and PCA methods. The Cholesky method demonstrates much smaller deviations and errors compared to PCA. This is also indicated by the plots. The plot of cumulative variance explained for the Cholesky method aligns closely with the input matrix, indicating high accuracy in preserving the variance structure. In contrast, the PCA method shows a noticeable discrepancy, suggesting less accuracy in capturing the original variance distribution.

In conclusion: the Cholesky method more accurately preserves the variance structure of the input matrix. In contrast, the PCA method shows a larger discrepancy, indicating that it may not capture the variance distribution as effectively.

E:
Cholesky simulation time: 0.1088249683380127
PCA simulation time: 0.02804708480834961

The PCA simulation is faster than the Cholesky simulation, with PCA completing in 0.028 seconds compared to Cholesky's 0.109 seconds. This indicates that PCA is more computationally efficient for this task.

F:
Cholesky decomposition preserves the covariance structure of the original data more accurately, leading to a lower Frobenius norm. However, it requires the covariance matrix to be positive definite and can be computationally expensive for large matrices. This is reflected in its longer runtime compared to PCA-based simulation.

PCA-based simulation reduces dimensionality by retaining only the most

significant principal components, making it computationally efficient with a shorter runtime. This efficiency comes at the cost of reduced accuracy in covariance preservation, as shown by its higher Frobenius norm.