QRM report-Jiaxin zhao(jz474)

Question 1:

A

for question 1 data set:

Mean: 0.05019795790476916,

Variance: 0.010332476407479581,

Skewness: 0.1204447119194402,

Kurtosis: 0.2229270674503816

B

The skewness and kurtosis of a normal distribution are close to zero. In the data for Problem 1, our calculations show a skewness of about 0.12 and a kurtosis of about 0.22—both quite near zero—indicating that a normal distribution is more appropriate. According to the Central Limit Theorem, when the sample size exceeds 30, real-world data commonly approximates a normal distribution. Conversely, if the sample size is under 30 or if the variance is unknown and we need to perform mean estimation or hypothesis testing, a t-distribution is often used. However, in this problem the mean and variance are already known, and the sample size is 1000, so the normal distribution provides a better fit.

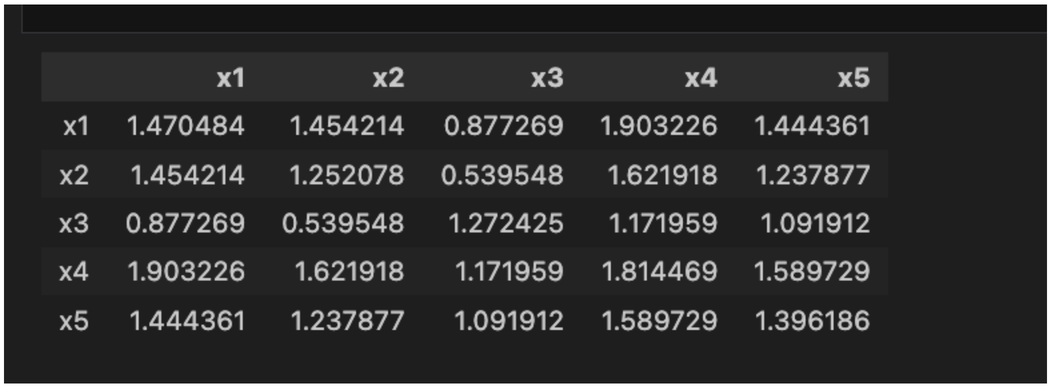
C

By plotting the probability density functions (PDFs) of both the fitted normal and t distributions, we see that the two curves nearly coincide, indicating that both distributions fit the data fairly well. There is no marked skewness or heavy-tailed behavior that would give the t distribution a clear advantage. However, based on the AICc criterion—which favors smaller values—the normal distribution's AIC is about -1731.5867, compared to -1731.4184 for the t distribution. Since lower is better for AICc, the normal distribution does indeed provide a better fit.

Question 2:

A

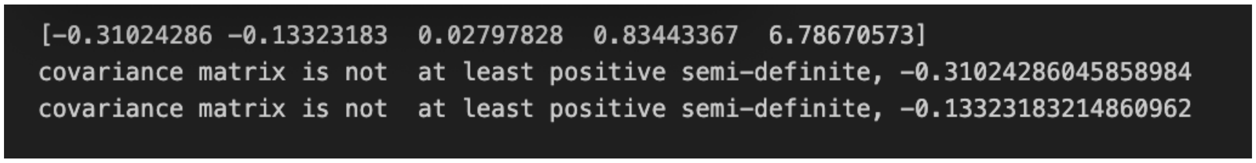
this is the screenshot of the pairwise covariance matrix



	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 .

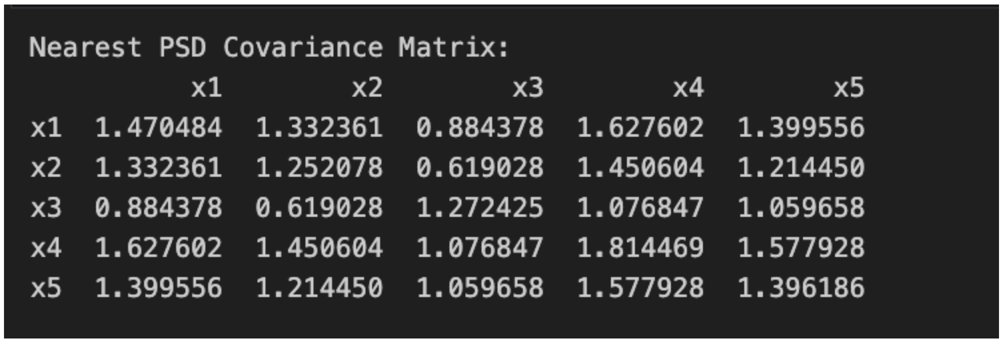
the eigenvalue of the matrix is following:



```
[-0.31024286 -0.13323183  0.02797828  0.83443367  6.78670573]
covariance matrix is not  at least positive semi-definite, -0.31024286045858984
covariance matrix is not  at least positive semi-definite, -0.13323183214860962
```

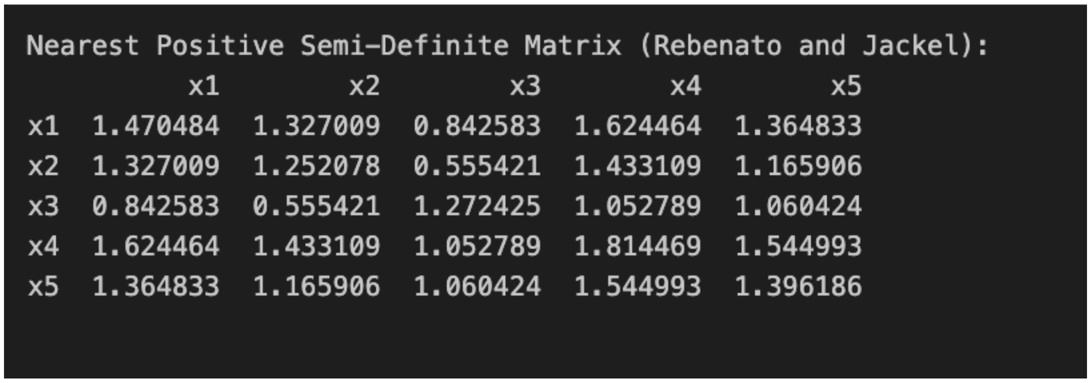
if all of the eigenvalues are greater or equal than zero, which indicates that the matrix is at least positive semidefinite. But you can see that not all of the eigenvalues are greater than zero; some are negative. Therefore, the matrix is not at least positive semidefinite.

The nearest positive semi-definite matrix using Higham's method



```
Nearest PSD Covariance Matrix:
          x1        x2        x3        x4        x5
x1  1.470484  1.332361  0.884378  1.627602  1.399556
x2  1.332361  1.252078  0.619028  1.450604  1.214450
x3  0.884378  0.619028  1.272425  1.076847  1.059658
x4  1.627602  1.450604  1.076847  1.814469  1.577928
x5  1.399556  1.214450  1.059658  1.577928  1.396186
```

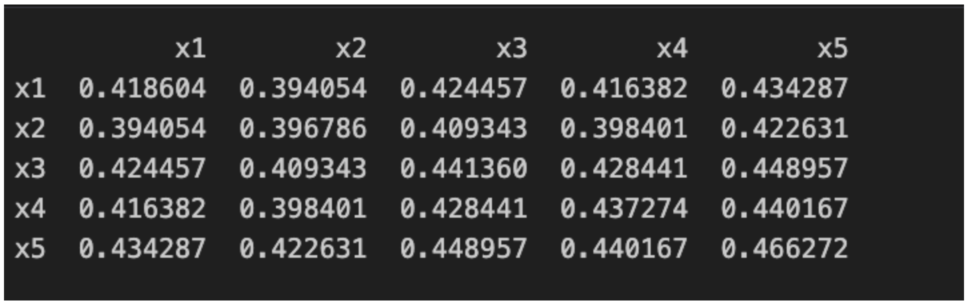
The nearest positive semi-definite matrix using Rebenato and Jackel:



```
Nearest Positive Semi-Definite Matrix (Rebenato and Jackel):
          x1        x2        x3        x4        x5
x1  1.470484  1.327009  0.842583  1.624464  1.364833
x2  1.327009  1.252078  0.555421  1.433109  1.165906
x3  0.842583  0.555421  1.272425  1.052789  1.060424
x4  1.624464  1.433109  1.052789  1.814469  1.544993
x5  1.364833  1.165906  1.060424  1.544993  1.396186
```

C

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
```

D

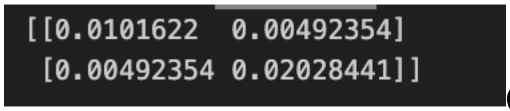
First, we can observe that the covariance values in the matrix from step C are generally larger than those in step D. This is because, compared to the data used in step C, step D applies the overlap method, which results in fewer available data points, leading to generally smaller covariance values. Additionally, the reduction in sample size makes the covariance estimation less stable, which may affect the positive semidefiniteness of the matrix.

Also, In step C, we use Higham method and the Rebenato and Jackel method to modify the original covariance matrix by truncating or raising any negative eigenvalues, thereby producing a matrix whose eigenvalues are all nonnegative and hence numerically (semi)positive-definite. In contrast, the covariance matrix obtained in step D is calculated using only the truly overlapping portion of the data—a straightforward sample covariance that does not artificially adjust negative eigenvalues and thus directly reflects the covariance structure present in the partially overlapping observations. As a result, the modified matrix in step C has no negative eigenvalues yet may not strictly adhere to the original data's covariance structure, whereas the purely overlapping-data-based estimate in step D remains more faithful to the actual observations but can lead to negative eigenvalues or fail to maintain definiteness if the overlap is insufficient or the data is noisy.

Question 3:

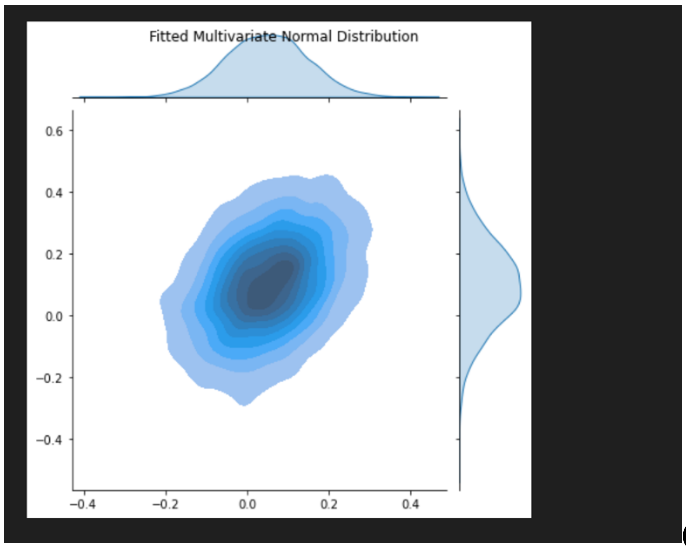
A.

By calculating the mean values from Question 3's data: the mean of x1 is **0.04600157**, and the mean of x2 is **0.09991502**, along with the covariance matrix (screenshot attached). Additionally, generate **10,000 random samples** from this distribution and visualize the results.



```
[[0.0101622  0.00492354]
 [0.00492354 0.02028441]]
```

(this is the covariance matrix)



(this is the virtualization of distribution)

B.

Method 1:

According to the following formula: x1=0.6

$$X_2|X_1 = x_1 \sim \mathcal{N}(\mu_{2|1}, \sigma^2_{2|1})$$

where:

$$\mu_{2|1} = \mu_2 + \frac{\sigma_{12}}{\sigma_{11}}(X_1 - \mu_1)$$

$$\sigma^2_{2|1} = \sigma_{22} - \frac{\sigma^2_{12}}{\sigma_{11}}$$

Method 1 - Conditional Mean of X2 | X1=0.6: 0.3683249958609772

Method 1 - Conditional Variance of X2 | X1=0.6: 0.01789896964508753

Method 2:

According to the following formula: x1=0.6

X2=β0+β1X1+ϵ

$$\beta_1 = \frac{\sigma_{12}}{\sigma_{11}}$$

$$\beta_0 = \mu_2 - \beta_1\mu_1 \qquad \mu_{2|1} = \beta_0 + \beta_1 X_1 \qquad \sigma^2_{2|1} = \sigma_{22} - \beta_1\sigma_{12}$$

Method 2 (OLS) - Conditional Mean of X2 | X1=0.6: 0.3683249958609772

Method 2 (OLS) - Conditional Variance of X2 | X1=0.6: 0.01789896964508753

C

According to the following formula: x1=0.6

$$\Sigma = LL^T \qquad L = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \qquad \Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{12} & \sigma_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} L_{11} & L_{21} \\ 0 & L_{22} \end{bmatrix}$$

$$Z \sim \mathcal{N}(0, I)$$

- $L_{11}^2 = \sigma_{11} \Rightarrow L_{11} = \sqrt{\sigma_{11}}$
- $L_{21}L_{11} = \sigma_{12} \Rightarrow L_{21} = \frac{\sigma_{12}}{\sqrt{\sigma_{11}}}$
- $L_{22}^2 = \sigma_{22} - L_{21}^2 \Rightarrow L_{22} = \sqrt{\sigma_{22} - \frac{\sigma_{12}^2}{\sigma_{11}}}$

$$X_1 = L_{11}Z_1 + \mu_1$$

$$X_2 = L_{21}Z_1 + L_{22}Z_2 + \mu_2$$

By analyzing the simulated conditional mean and variance of X2, we can get the following information:

Simulated Conditional Mean of X2 | X1=0.6: 0.36737469966036296

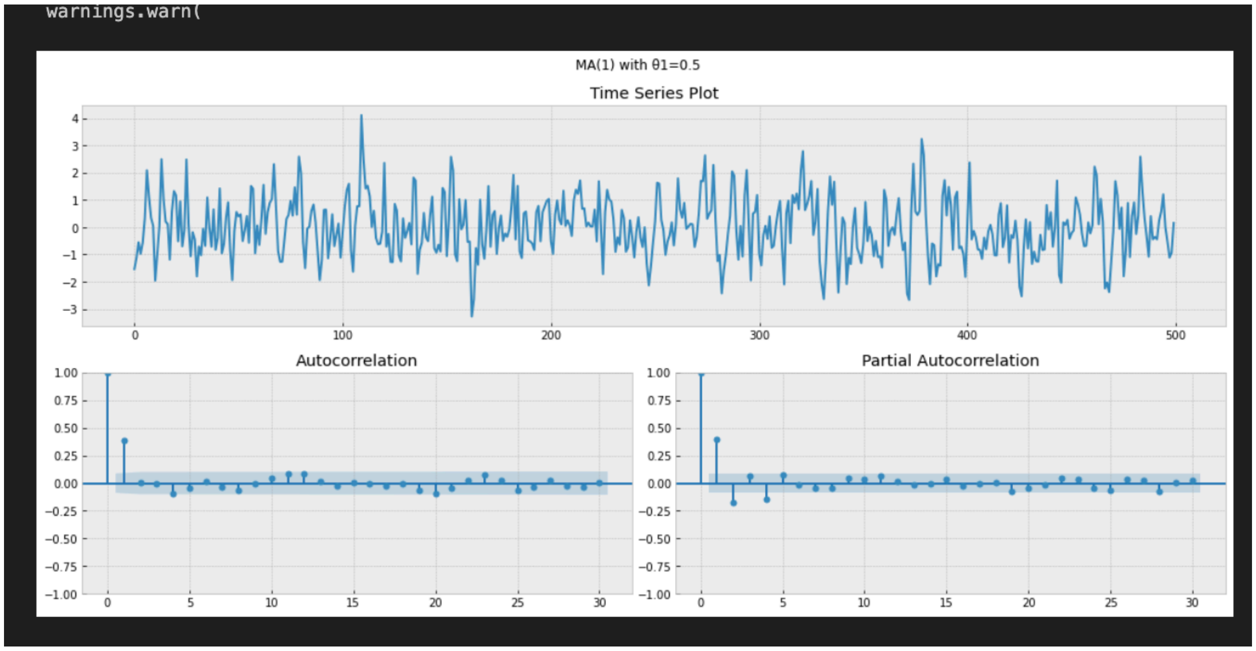
Simulated Conditional Variance of X2 | X1=0.6: 0.017968616403230032

we observe that since Z~N(0,I) is randomly generated, the results vary with each simulation. However, both the mean and variance remain very close to the values obtained using the two methods in **Problem B**, confirming the consistency and accuracy of the simulation.

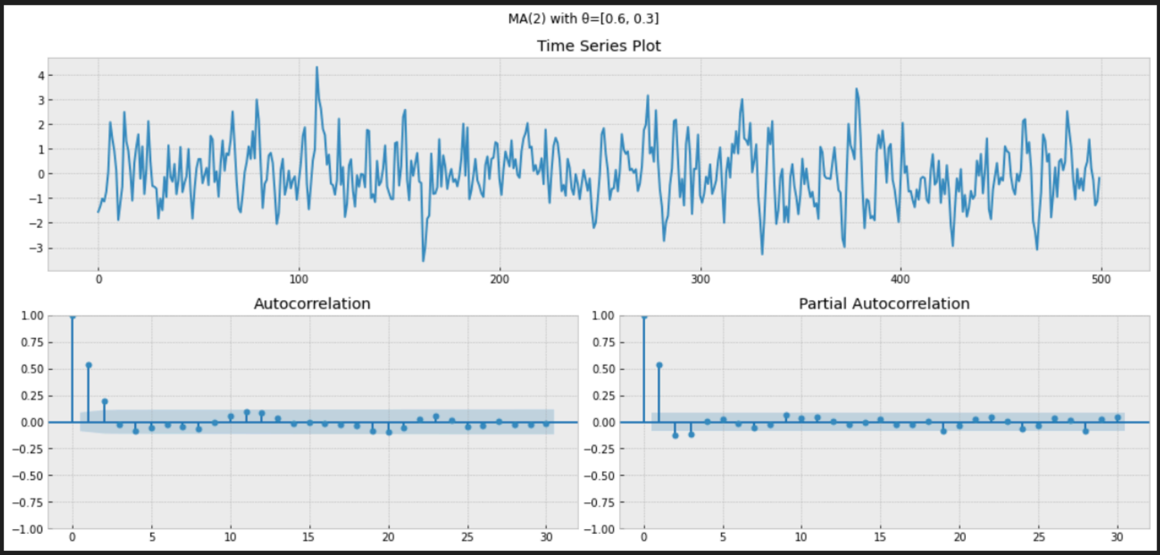
Question 4:

A

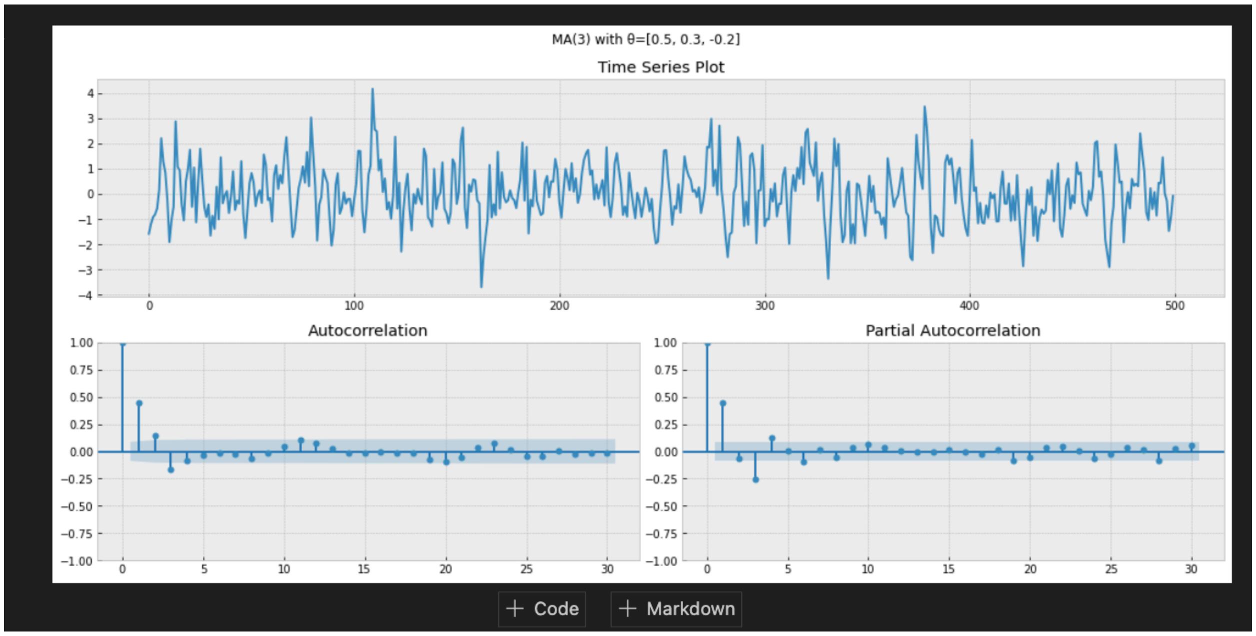
Here is the simulation of MA(1), MA(2), and MA(3):



```
warnings.warn(
```



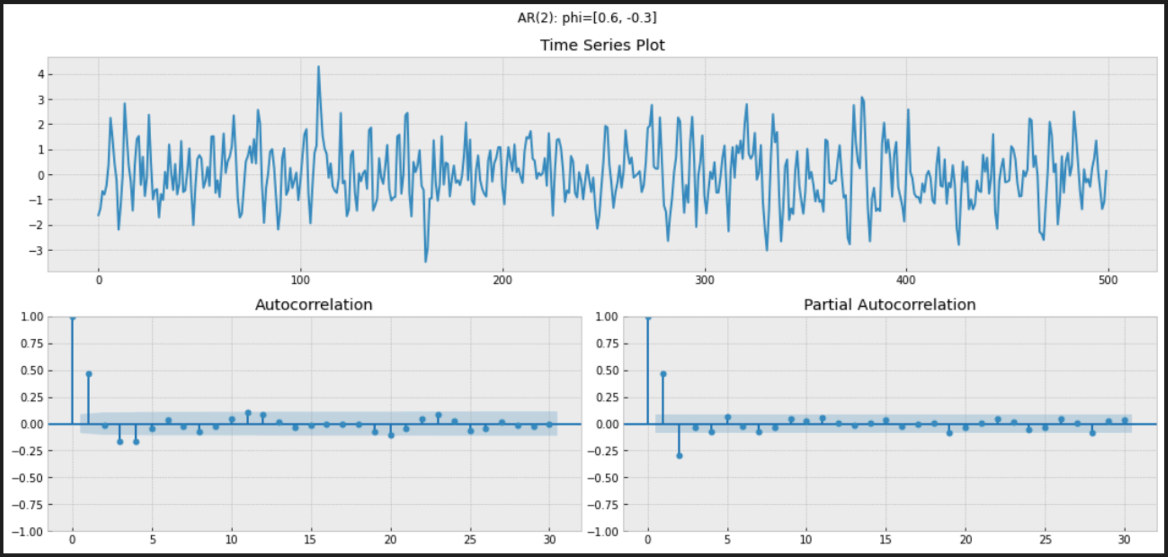
```
/Users/zhaojiaxin/opt/anaconda3/lib/python3.9/site-packages/statsmodels/graphics/tsaplots.py:348: FutureWarning
  warnings.warn(
```



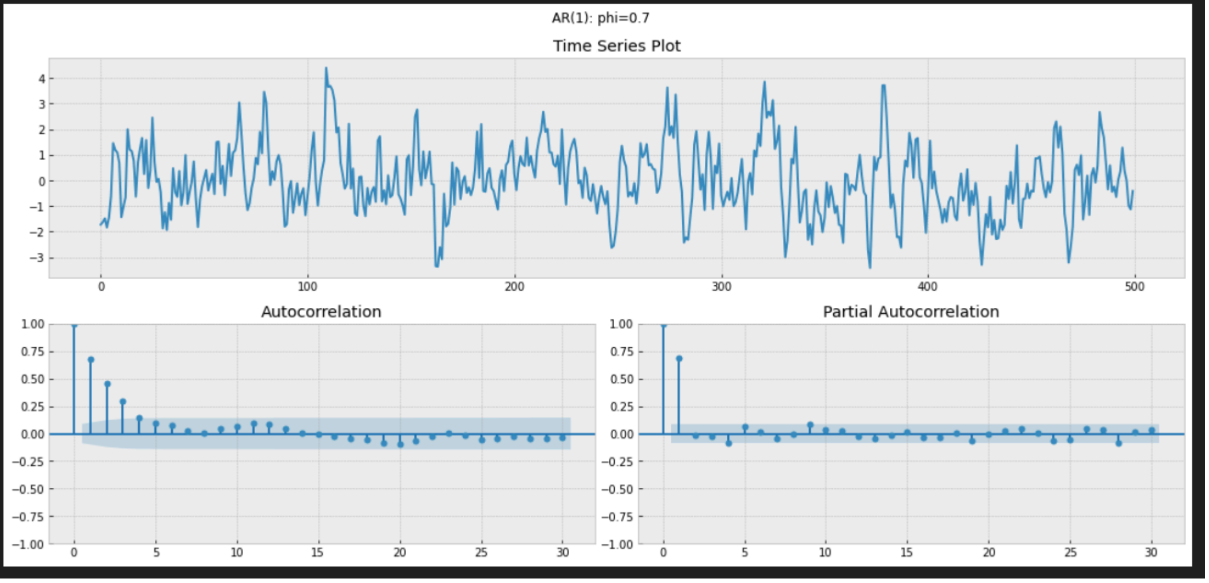
We can see from the plot above, The ACF of an MA(q) process cuts off at lag q, whereas the PACF exhibits a gradual decay.

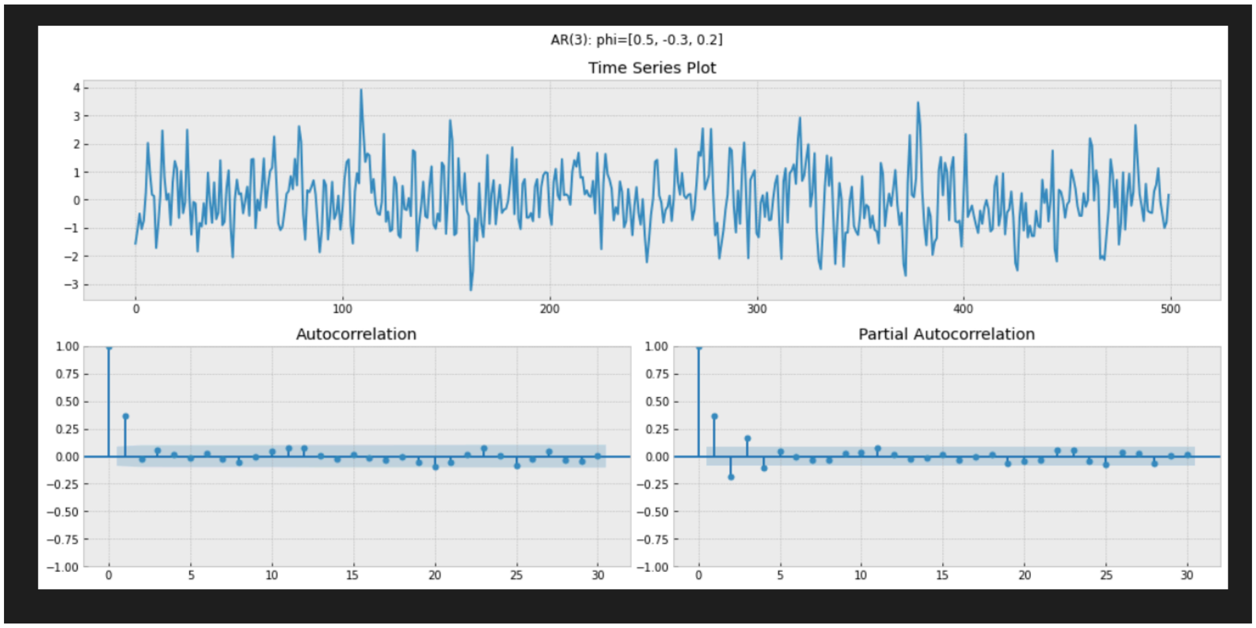
B

```
  warnings.warn(
```



```
  warnings.warn(
```

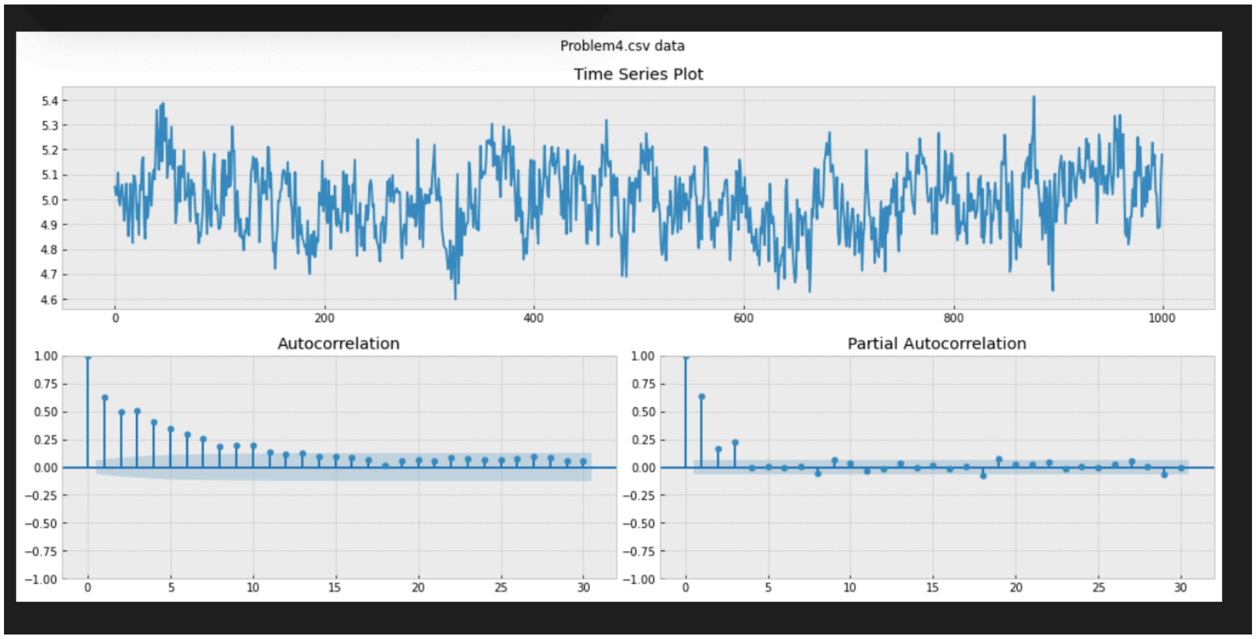




We can see from the plot above that the PACF of an AR(p) process usually cuts off at lag p (or decays significantly after p), while the ACF often exhibits a trailing pattern.

C

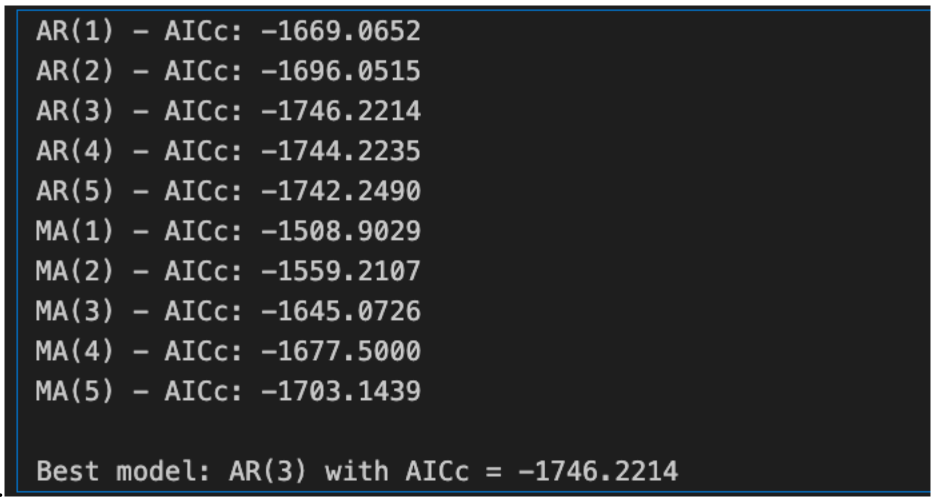
The following is the time series plot , ACF ,PACF plot of problem 4 :



After analyzing the ACF (Autocorrelation Function) and PACF (Partial Autocorrelation Function) of the problem4.csv data, I chose AR(3) (an autoregressive model of order 3) as the appropriate model. PACF cuts off, while ACF tails off: From the plots, we can see that PACF quickly declines to zero after lag 3, while ACF gradually decays. This is usually a characteristic of an AR(p) process. In an AR(p) model, PACF truncates after a certain lag, while ACF decays slowly. Here, PACF significantly decreases after lag 3, so AR(3) is a reasonable choice. Although MA(10) may also be an option, using a high-order MA model may bring problems. Too many parameters increase the risk of overfitting: MA(q) requires estimating q moving average parameters (plus the noise term), and MA(10) may cause the model to overfit the data, affecting generalization ability. High-order MA models may be suitable for short-term high-frequency fluctuations, but they are not necessarily the best choice. Considering the patterns in ACF and PACF plots, AR(3) is a reasonable choice because it captures the data's correlation structure while avoiding overfitting.

D

By fitting AR(1,2,3,4,5) and MA(1,2,3,4,5,10) models using the data from Problem 4, we obtained the following results：



```
AR(1) - AICc: -1669.0652
AR(2) - AICc: -1696.0515
AR(3) - AICc: -1746.2214
AR(4) - AICc: -1744.2235
AR(5) - AICc: -1742.2490
MA(1) - AICc: -1508.9029
MA(2) - AICc: -1559.2107
MA(3) - AICc: -1645.0726
MA(4) - AICc: -1677.5000
MA(5) - AICc: -1703.1439

Best model: AR(3) with AICc = -1746.2214
```

.

comparing their AICc values, AR(3) is determined to be the best-fitting model, as it has the lowest AICc (-1746.2214). This further validates our selection from Question C, confirming that the data exhibits an AR(3) structure, meaning that the current values are best explained by the past three observations.

Question 5:

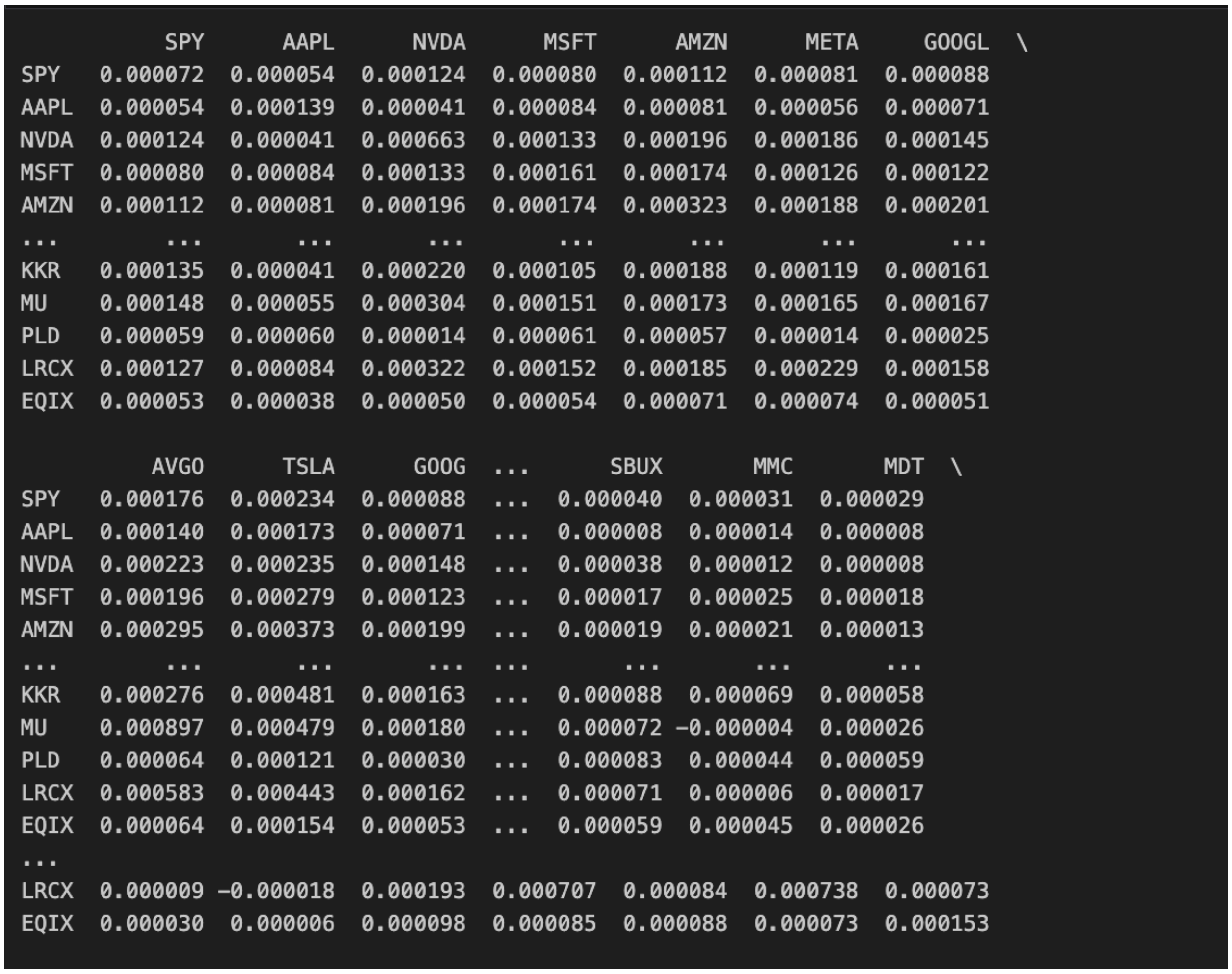
A：

My code uses the following formulas to calculate the Exponentially Weighted Covariance Matrix， I have also tested it using the data from the testdata folder, and it is correct.

$$w_t^* = \frac{w_t}{\sum_{i=1}^{T} w_i} \qquad \mu_i = \sum_{t=1}^{T} w_t^* R_{t,i} \qquad \text{EWCov}(i,j) = \sum_{t=1}^{T} w_t^* (R_{t,i} - \mu_i)(R_{t,j} - \mu_j)$$

$$\Sigma_{i,j} = \sum_{t=1}^{T} w_t^* (R_{t,i} - \mu_i)(R_{t,j} - \mu_j)$$

The following is the screenshot of the Exponentially Weighted Covariance Matrix for question 5.

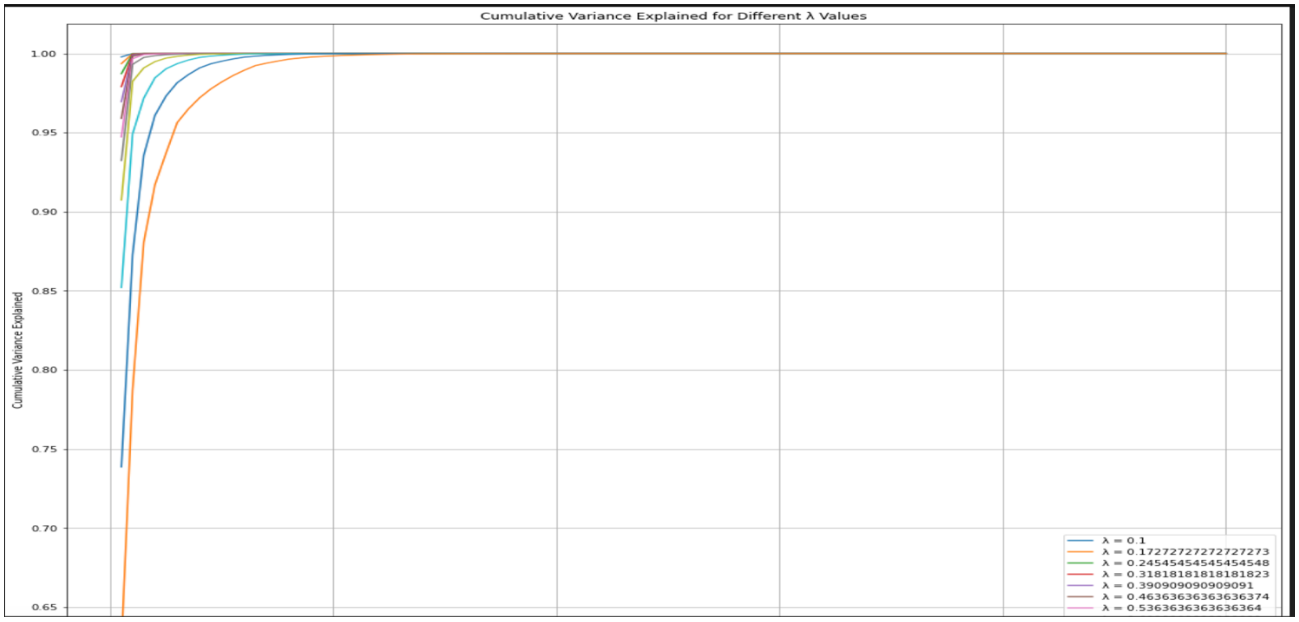


```
           SPY      AAPL      NVDA      MSFT      AMZN      META     GOOGL  \
SPY   0.000072  0.000054  0.000124  0.000080  0.000112  0.000081  0.000088
AAPL  0.000054  0.000139  0.000041  0.000084  0.000081  0.000056  0.000071
NVDA  0.000124  0.000041  0.000663  0.000133  0.000196  0.000186  0.000145
MSFT  0.000080  0.000084  0.000133  0.000161  0.000174  0.000126  0.000122
AMZN  0.000112  0.000081  0.000196  0.000174  0.000323  0.000188  0.000201
...        ...       ...       ...       ...       ...       ...       ...
KKR   0.000135  0.000041  0.000220  0.000105  0.000188  0.000119  0.000161
MU    0.000148  0.000055  0.000304  0.000151  0.000173  0.000165  0.000167
PLD   0.000059  0.000060  0.000014  0.000061  0.000057  0.000014  0.000025
LRCX  0.000127  0.000084  0.000322  0.000152  0.000185  0.000229  0.000158
EQIX  0.000053  0.000038  0.000050  0.000054  0.000071  0.000074  0.000051

          AVGO      TSLA      GOOG  ...      SBUX       MMC       MDT  \
SPY   0.000176  0.000234  0.000088  ...  0.000040  0.000031  0.000029
AAPL  0.000140  0.000173  0.000071  ...  0.000008  0.000014  0.000008
NVDA  0.000223  0.000235  0.000148  ...  0.000038  0.000012  0.000008
MSFT  0.000196  0.000279  0.000123  ...  0.000017  0.000025  0.000018
AMZN  0.000295  0.000373  0.000199  ...  0.000019  0.000021  0.000013
...        ...       ...       ...  ...       ...       ...       ...
KKR   0.000276  0.000481  0.000163  ...  0.000088  0.000069  0.000058
MU    0.000897  0.000479  0.000180  ...  0.000072 -0.000004  0.000026
PLD   0.000064  0.000121  0.000030  ...  0.000083  0.000044  0.000059
LRCX  0.000583  0.000443  0.000162  ...  0.000071  0.000006  0.000017
EQIX  0.000064  0.000154  0.000053  ...  0.000059  0.000045  0.000026
...
LRCX  0.000009 -0.000018  0.000193  0.000707  0.000084  0.000738  0.000073
EQIX  0.000030  0.000006  0.000098  0.000085  0.000088  0.000073  0.000153
```

B

By varying different values of λ, we use PCA and plot the cumulative variance explained/ The visualization result is as follows:



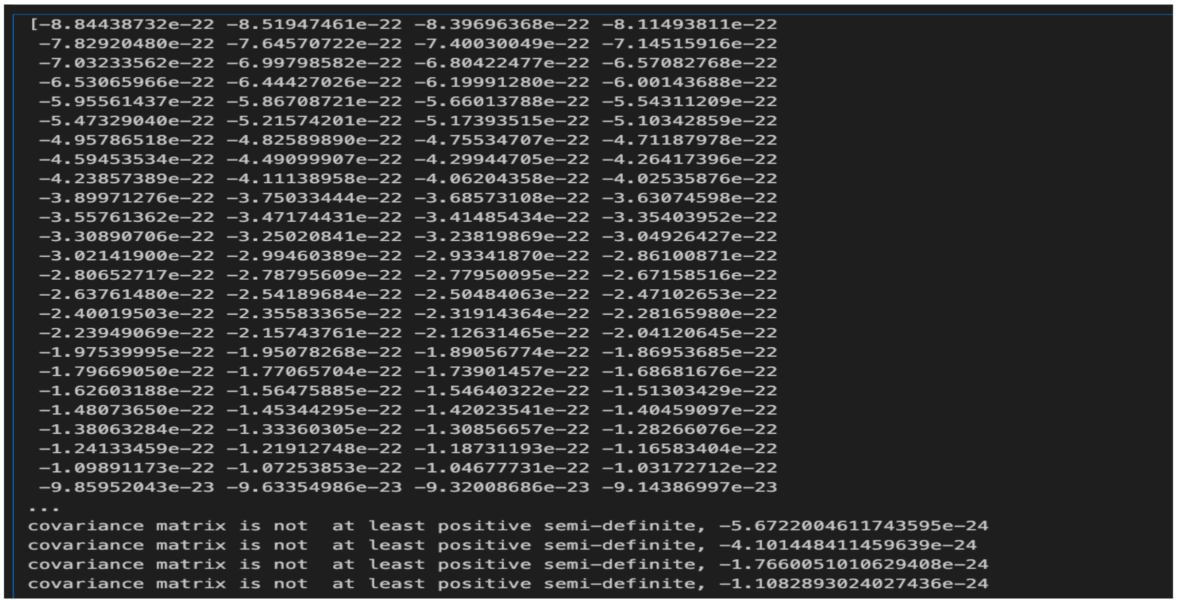
C

From the graph, we can see that higher λ values (close to 1.0) result in cumulative variance reaching 1.0 more quickly , requiring fewer principal components to capture most of the variance. This is because the covariance matrix is more structured, with a few dominant eigenvalues, reflecting strong long-term correlations in the data. For lower λ values (close to 0.1), variance is spread across more principal components, and the cumulative variance increases more slowly, requiring more components to explain the same proportion of variance. This indicates that the covariance matrix places more emphasis on recent fluctuations, leading to a more evenly distributed eigenvalue structure. Overall, higher λ concentrates variance in fewer principal components, while lower λ distributes variance more evenly, requiring more components to capture the overall variance in the data. In other words, High λ leads to more concentrated eigenvalues, with a few principal components dominating. Low λ leads to a more evenly distributed eigenvalue structure, indicating that more factors influence the data.

Question 6:

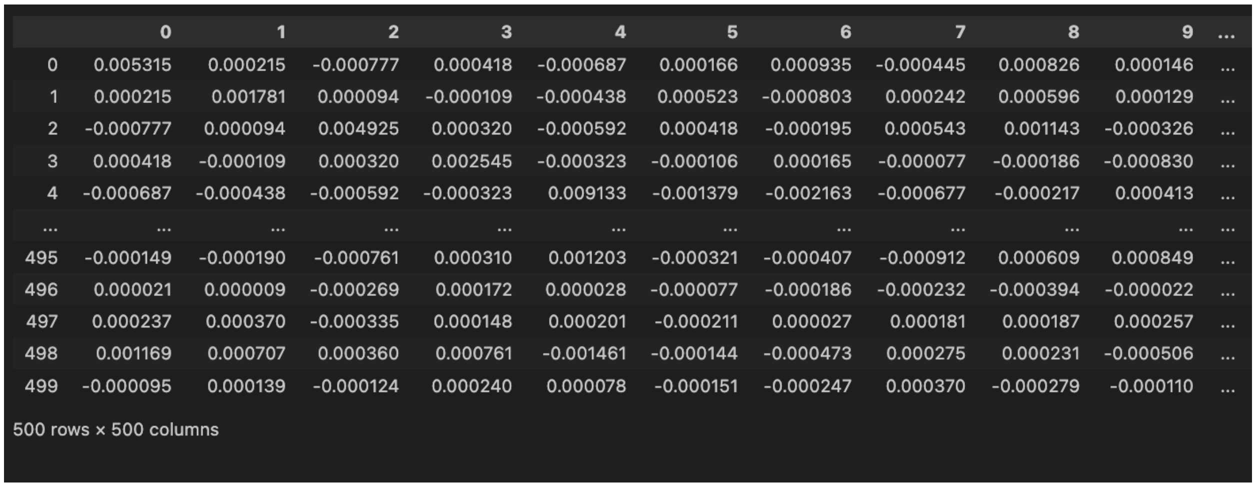
A :

By determining whether the data in Problem 6 is positive definite, we found that the data in Problem 6 is not positive definite, the following is the screenshot of the eigenvalues and partial result:



```
[-8.84438732e-22 -8.51947461e-22 -8.39696368e-22 -8.11493811e-22
 -7.82920480e-22 -7.64570722e-22 -7.40030049e-22 -7.14515916e-22
 -7.03233562e-22 -6.99798582e-22 -6.80422477e-22 -6.57082768e-22
 -6.53065966e-22 -6.44427026e-22 -6.19991280e-22 -6.00143688e-22
 -5.95561437e-22 -5.86708721e-22 -5.66013788e-22 -5.54311209e-22
 -5.47329040e-22 -5.21574201e-22 -5.17393515e-22 -5.10342859e-22
 -4.95786518e-22 -4.82589890e-22 -4.75534707e-22 -4.71187978e-22
 -4.59453534e-22 -4.49099907e-22 -4.29944705e-22 -4.26417396e-22
 -4.23857389e-22 -4.11138958e-22 -4.06204358e-22 -4.02535876e-22
 -3.89971276e-22 -3.75033444e-22 -3.68573108e-22 -3.63074598e-22
 -3.55761362e-22 -3.47174431e-22 -3.41485434e-22 -3.35403952e-22
 -3.30890706e-22 -3.25020841e-22 -3.23819869e-22 -3.04926427e-22
 -3.02141900e-22 -2.99460389e-22 -2.93341870e-22 -2.86100871e-22
 -2.80652717e-22 -2.78795609e-22 -2.77950095e-22 -2.67158516e-22
 -2.63761480e-22 -2.54189684e-22 -2.50484063e-22 -2.47102653e-22
 -2.40019503e-22 -2.35583365e-22 -2.31914364e-22 -2.28165980e-22
 -2.23949069e-22 -2.15743761e-22 -2.12631465e-22 -2.04120645e-22
 -1.97539995e-22 -1.95078268e-22 -1.89056774e-22 -1.86953685e-22
 -1.79669050e-22 -1.77065704e-22 -1.73901457e-22 -1.68681676e-22
 -1.62603188e-22 -1.56475885e-22 -1.54640322e-22 -1.51303429e-22
 -1.48073650e-22 -1.45344295e-22 -1.42023541e-22 -1.40459097e-22
 -1.38063284e-22 -1.33360305e-22 -1.30856657e-22 -1.28266076e-22
 -1.24133459e-22 -1.21912748e-22 -1.18731193e-22 -1.16583404e-22
 -1.09891173e-22 -1.07253853e-22 -1.04677731e-22 -1.03172712e-22
 -9.85952043e-23 -9.63354986e-23 -9.32008686e-23 -9.14386997e-23
...
covariance matrix is not  at least positive semi-definite, -5.6722004611743595e-24
covariance matrix is not  at least positive semi-definite, -4.101448411459639e-24
covariance matrix is not  at least positive semi-definite, -1.7660051010629408e-24
covariance matrix is not  at least positive semi-definite, -1.1082893024027436e-24
```

Then using the Higham method find the nearest positive definite, the partial result are following:



	0	1	2	3	4	5	6	7	8	9	...
0	0.005315	0.000215	-0.000777	0.000418	-0.000687	0.000166	0.000935	-0.000445	0.000826	0.000146	...
1	0.000215	0.001781	0.000094	-0.000109	-0.000438	0.000523	-0.000803	0.000242	0.000596	0.000129	...
2	-0.000777	0.000094	0.004925	0.000320	-0.000592	0.000418	-0.000195	0.000543	0.001143	-0.000326	...
3	0.000418	-0.000109	0.000320	0.002545	-0.000323	-0.000106	0.000165	-0.000077	-0.000186	-0.000830	...
4	-0.000687	-0.000438	-0.000592	-0.000323	0.009133	-0.001379	-0.002163	-0.000677	-0.000217	0.000413	...
...	...	...	...	...	...	...	...	...	...	...	...
495	-0.000149	-0.000190	-0.000761	0.000310	0.001203	-0.000321	-0.000407	-0.000912	0.000609	0.000849	...
496	0.000021	0.000009	-0.000269	0.000172	0.000028	-0.000077	-0.000186	-0.000232	-0.000394	-0.000022	...
497	0.000237	0.000370	-0.000335	0.000148	0.000201	-0.000211	0.000027	0.000181	0.000187	0.000257	...
498	0.001169	0.000707	0.000360	0.000761	-0.001461	-0.000144	-0.000473	0.000275	0.000231	-0.000506	...
499	-0.000095	0.000139	-0.000124	0.000240	0.000078	-0.000151	-0.000247	0.000370	-0.000279	-0.000110	...

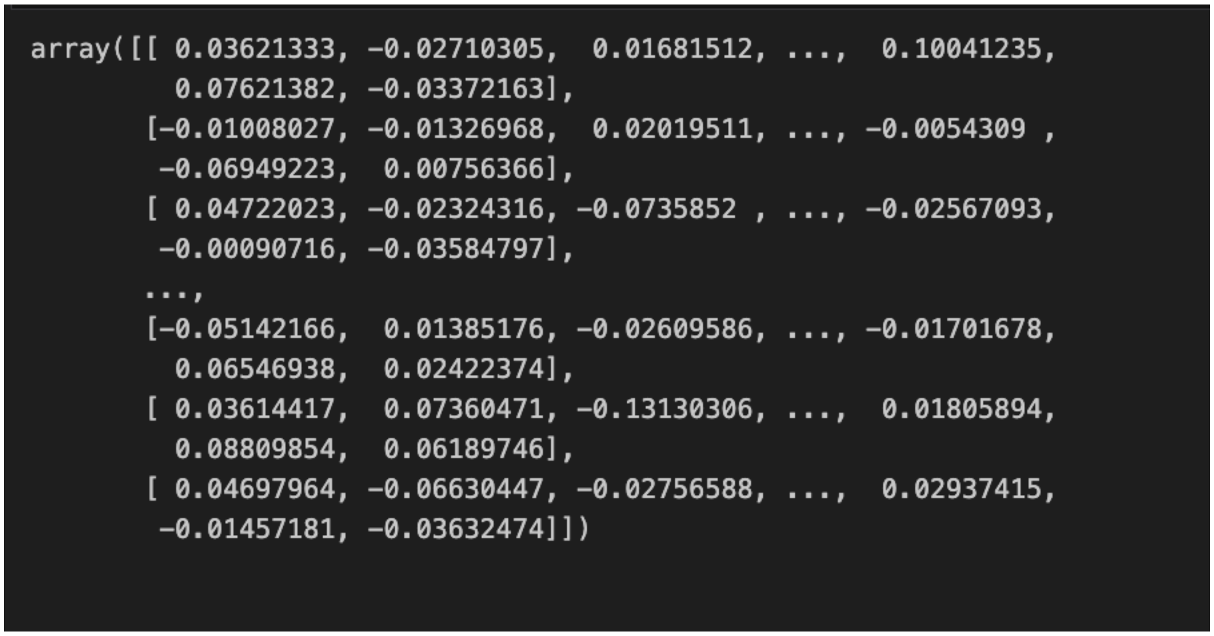
500 rows × 500 columns

Then uisng this formula to get the 10,000 draws:

$$X = ZL^T, \quad Z \sim \mathcal{N}(0, I), \quad \Sigma = LL^T$$

Also, using “time model” to calulate the time cost of the simulation.

The result are followings:



```
array([[ 0.03621333, -0.02710305,  0.01681512, ...,  0.10041235,
         0.07621382, -0.03372163],
       [-0.01008027, -0.01326968,  0.02019511, ..., -0.0054309 ,
        -0.06949223,  0.00756366],
       [ 0.04722023, -0.02324316, -0.0735852 , ..., -0.02567093,
        -0.00090716, -0.03584797],
       ...,
       [-0.05142166,  0.01385176, -0.02609586, ..., -0.01701678,
         0.06546938,  0.02422374],
       [ 0.03614417,  0.07360471, -0.13130306, ...,  0.01805894,
         0.08809854,  0.06189746],
       [ 0.04697964, -0.06630447, -0.02756588, ...,  0.02937415,
        -0.01457181, -0.03632474]])
```

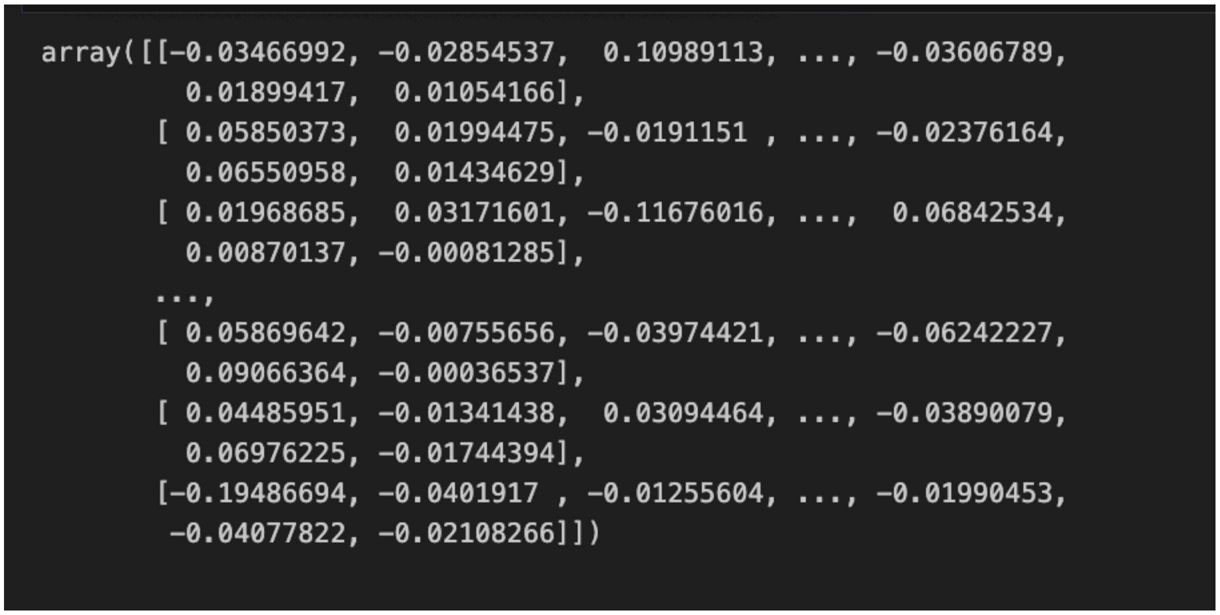
B

we select the smallest k such that the cumulative explained variance Ck satisfies Ck≥0.75. This ensures that the first k principal components capture at least 75% of the total variance in the data. To generate random samples that conform to the reduced covariance matrix, we use the top k eigenvectors and eigenvalues. generate a standard normal k-dimensional random vector Z~N(0,Ik) by using this formula:

$$X = Z\Lambda_k^{1/2} Q_k^T, \quad Z \sim \mathcal{N}(0, I_k)$$

Also, using “time model” to calulate the time cost of the simulation.

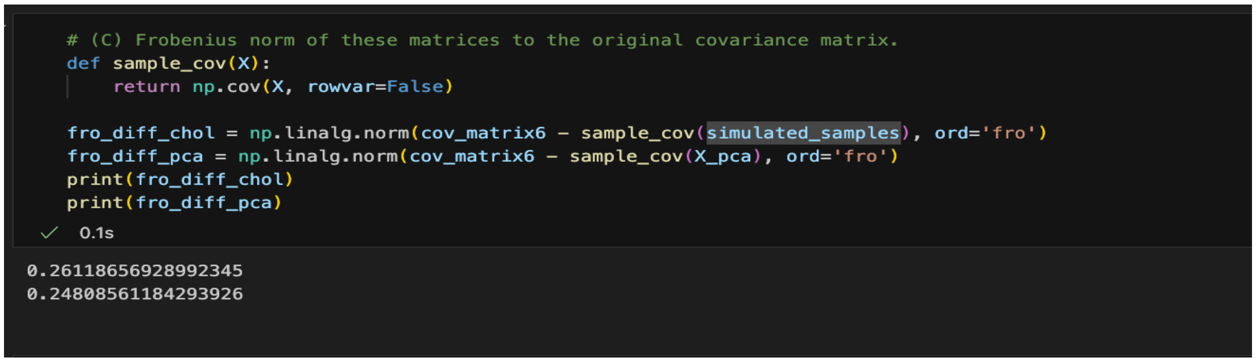
The following are the results:



```
array([[-0.03466992, -0.02854537,  0.10989113, ..., -0.03606789,
         0.01899417,  0.01054166],
       [ 0.05850373,  0.01994475, -0.0191151 , ..., -0.02376164,
         0.06550958,  0.01434629],
       [ 0.01968685,  0.03171601, -0.11676016, ...,  0.06842534,
         0.00870137, -0.00081285],
       ...,
       [ 0.05869642, -0.00755656, -0.03974421, ..., -0.06242227,
         0.09066364, -0.00036537],
       [ 0.04485951, -0.01341438,  0.03094464, ..., -0.03890079,
         0.06976225, -0.01744394],
       [-0.19486694, -0.0401917 , -0.01255604, ..., -0.01990453,
        -0.04077822, -0.02108266]])
```

C

By using np.linalg.norm(A, ord='fro') function to calculation the Frobenius, the followings are the result:



```
# (C) Frobenius norm of these matrices to the original covariance matrix.
def sample_cov(X):
    return np.cov(X, rowvar=False)

fro_diff_chol = np.linalg.norm(cov_matrix6 - sample_cov(simulated_samples), ord='fro')
fro_diff_pca = np.linalg.norm(cov_matrix6 - sample_cov(X_pca), ord='fro')
print(fro_diff_chol)
print(fro_diff_pca)
```

✓ 0.1s

```
0.26118656928992345
0.24808561184293926
```

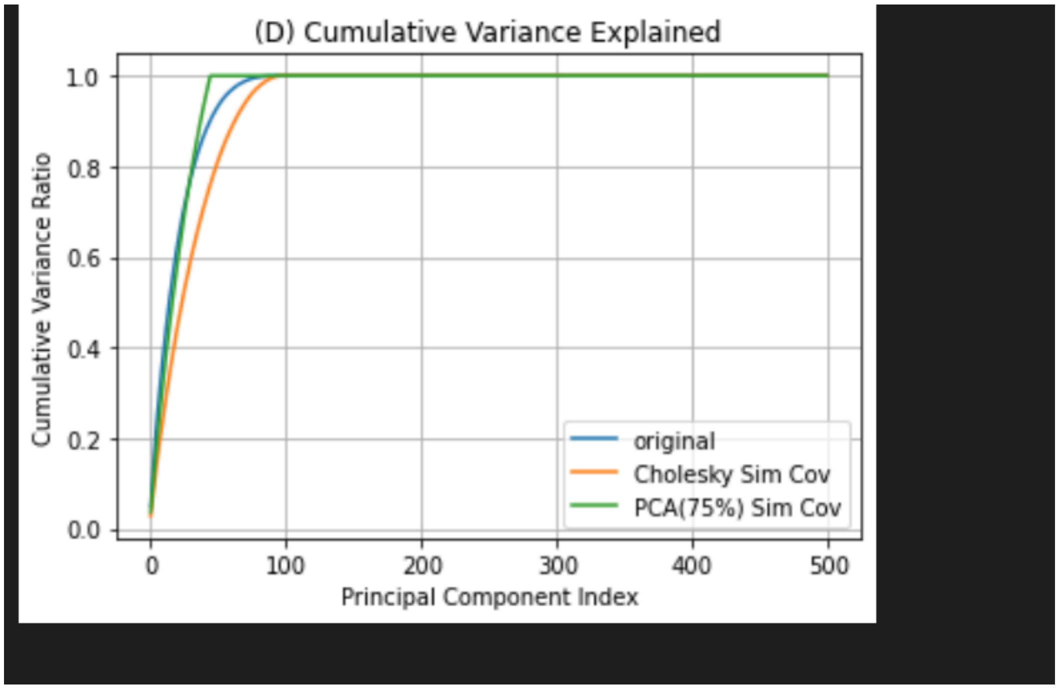
Cholesky: 0.2611, PCA: 0.2481, Purely based on the Frobenius error, the covariance matrix of the samples generated by the PCA method has a slightly smaller error than that of the Cholesky method, which may indicate that the PCA-approximated covariance matrix is closer to the original matrix in some aspects.

D

By using this formula to get the cumulative variance explained:

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{N} \lambda_i}$$

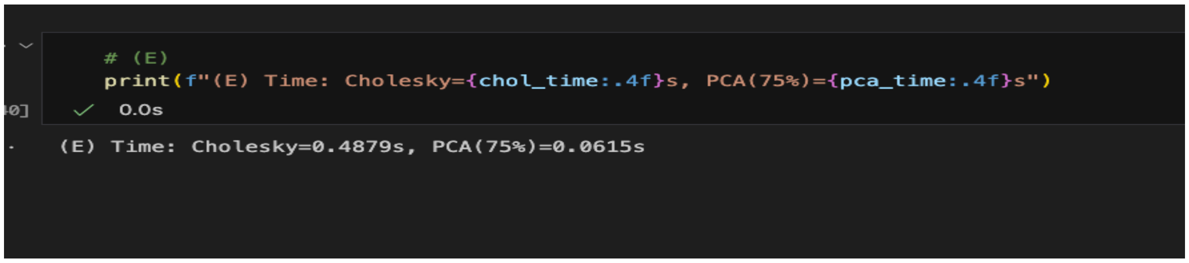
The following is the screenshot of the result:



The PCA method (green) accumulates variance more quickly in the first few principal components, indicating that PCA prioritizes preserving the features with the highest variance, allowing it to capture the majority of covariance information with fewer principal components. It retains the most significant eigenvalues while disregarding smaller ones. The Cholesky method, due to the adjustment of the covariance matrix during the positive-definiteness correction, results in a slower increase in cumulative variance explained compared to the PCA method.

E

I previously mentioned using the “time” module to measure the simulation time. The following screenshot shows the results.



```
# (E)
print(f"(E) Time: Cholesky={chol_time:.4f}s, PCA(75%)={pca_time:.4f}s")
```

```
0.0s
```

```
(E) Time: Cholesky=0.4879s, PCA(75%)=0.0615s
```

We can see that the PCA method is approximately 7.93 times faster than the Cholesky method.

F

Firstly, the PCA method retains only the major eigenvalues (75% cumulative variance) while ignoring smaller ones, thereby reducing the data dimension. This approach sacrifices some covariance information but still preserves the overall data structure. On the other hand, since the original covariance matrix is not positive semi-definite, the Cholesky method requires matrix correction before use. As a result, the generated samples conform to the corrected matrix rather than the original one.

Secondly, the PCA method is significantly faster than the Cholesky method (PCA = 0.0615s vs. Cholesky = 0.4879s). Since PCA only uses the top principal components, the computational cost is significantly reduced, making it more advantageous in high-dimensional data scenarios. Also, the Frobenius norm of pca is 0.248 is less than Cholesky Root method (0.261).

However, the PCA method's cumulative variance contribution grows faster in the first few principal components, indicating that PCA can capture most of the covariance information with fewer components while discarding some low-variance features. In contrast, the Cholesky method (if the original matrix is positive definite) can fully preserve the covariance information. However, in this case, since the matrix requires correction, the samples generated by Cholesky actually conform to the corrected covariance matrix rather than the original one.

To sum up, PCA is fast and efficient for high-dimensional data, making it ideal for dimensionality reduction, though it sacrifices some covariance information. Cholesky fully preserves covariance but is computationally expensive. If the covariance matrix is not positive definite, Cholesky requires correction, meaning the generated samples align with the corrected matrix rather than the original.