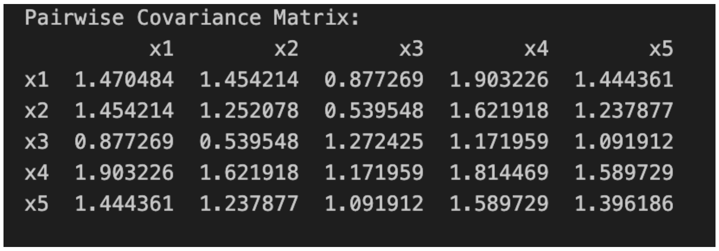
Project 1

1.
    a. Mean: 0.050197904769
       Variance: 0.0103324764
       Skewness: 0.12062572595
       Kurtosis: 0.2300698131702
    b. It seems the data shows mild tails and passes normality tests; Normal distribution would be the better model.
    c. The AICc results indicate that the normal distribution has a slightly lower AICc than the t-distribution, implying a marginally better fit. This minimal difference arises because the t-distribution's degrees of freedom are high, making it closely resemble the normal distribution.
2.
    a.



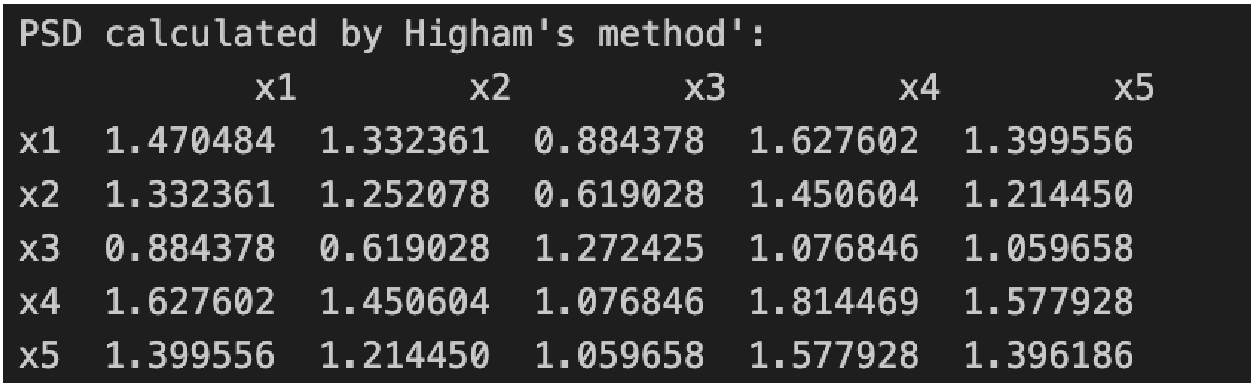
```
Pairwise Covariance Matrix:
          x1        x2        x3        x4        x5
x1  1.470484  1.454214  0.877269  1.903226  1.444361
x2  1.454214  1.252078  0.539548  1.621918  1.237877
x3  0.877269  0.539548  1.272425  1.171959  1.091912
x4  1.903226  1.621918  1.171959  1.814469  1.589729
x5  1.444361  1.237877  1.091912  1.589729  1.396186
```

    b. Eigenvalues of the Covariance Matrix:

	Eigenvalues
0	6.786706
1	0.834434
2	-0.310243
3	0.027978
4	-0.133232

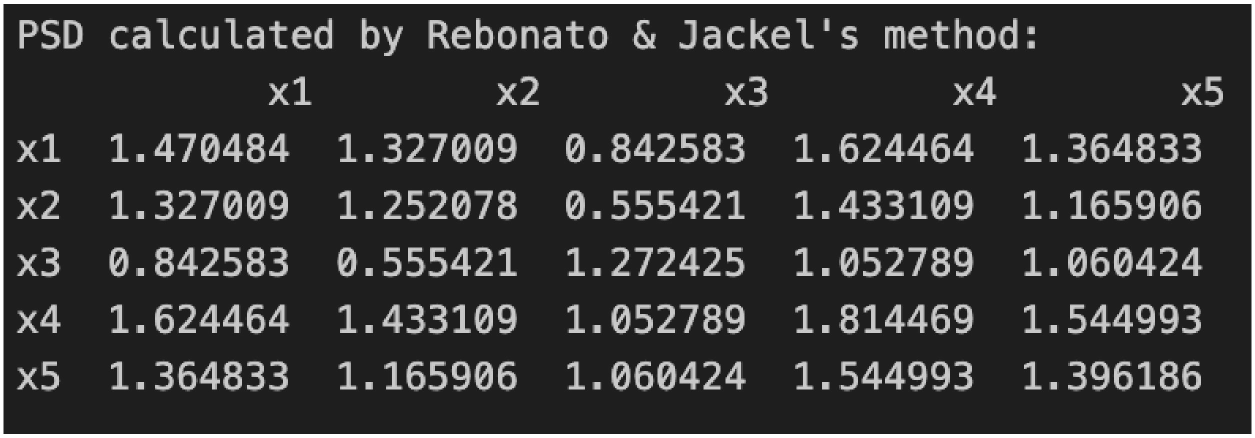
Because not all eigenvalues are greater than zero, the pairwise covariance matrix is NOT PSD.

    c. PSD calculated by Higham's method':



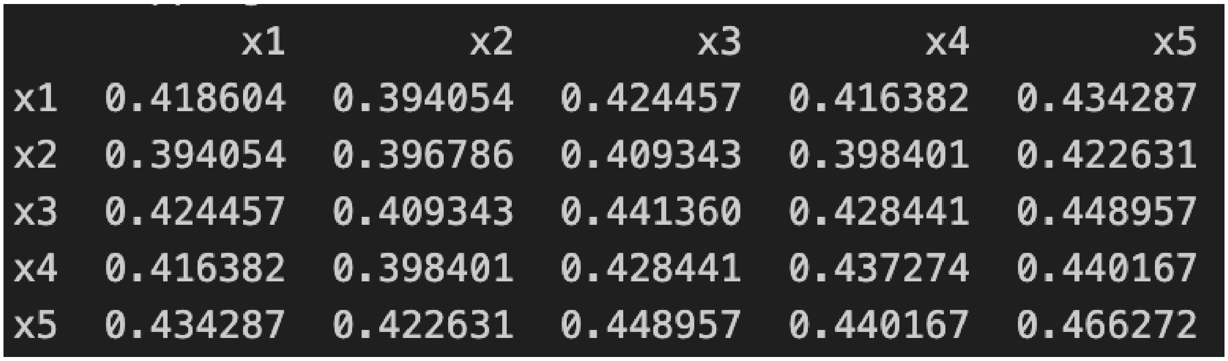
```
PSD calculated by Higham's method':
          x1        x2        x3        x4        x5
x1  1.470484  1.332361  0.884378  1.627602  1.399556
x2  1.332361  1.252078  0.619028  1.450604  1.214450
x3  0.884378  0.619028  1.272425  1.076846  1.059658
x4  1.627602  1.450604  1.076846  1.814469  1.577928
x5  1.399556  1.214450  1.059658  1.577928  1.396186
```

PSD calculated by Rebonato & Jackel's method:



```
PSD calculated by Rebonato & Jackel's method:
          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
```

d.



```
          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. 对比结果:C中结果接近真实情况;D中结果与真实情况差异很大。

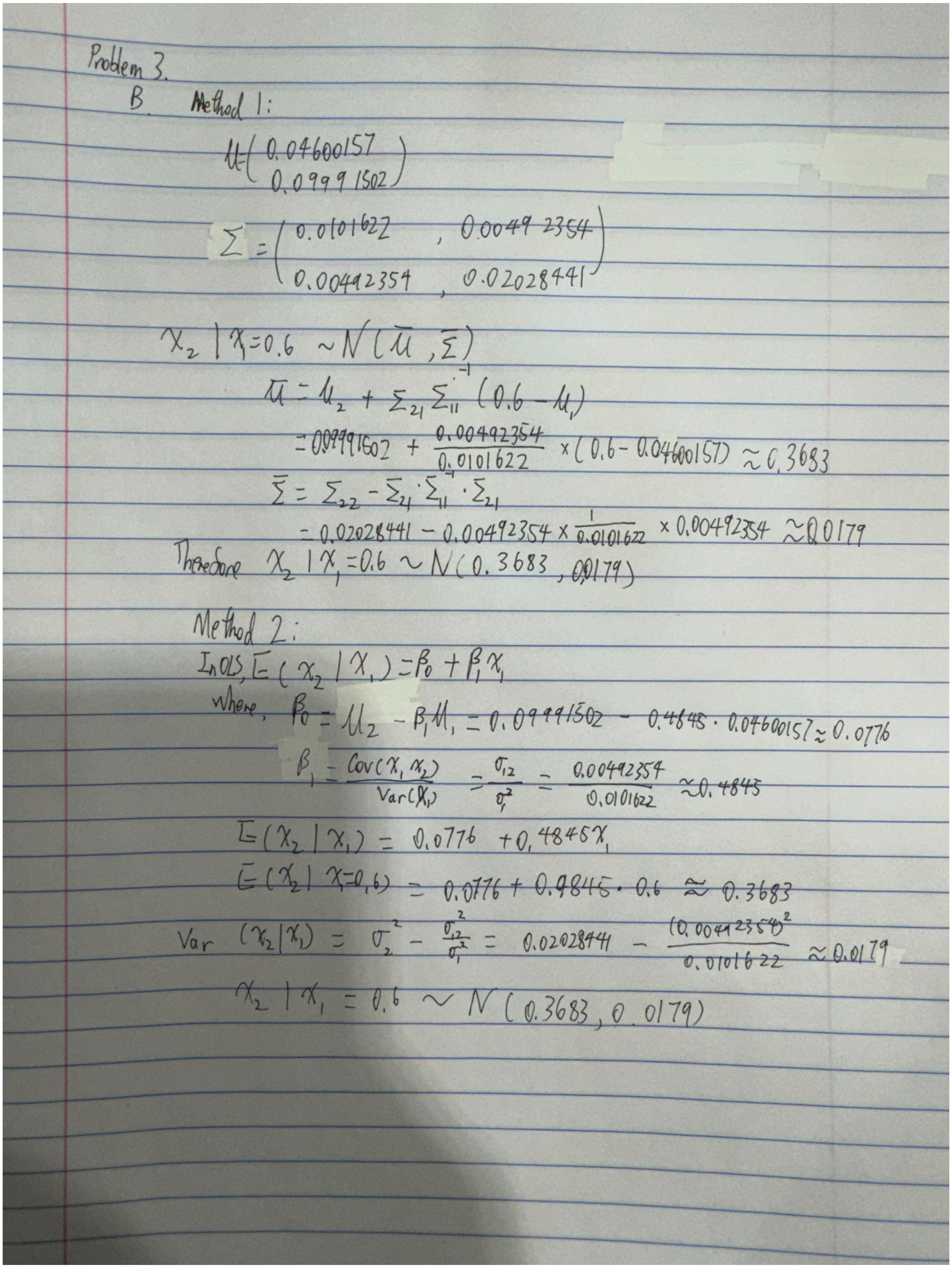
原因:C中最大程度利用已知数据，得到的协方差矩阵结果与真实情况差异并不大。而D中使用overlappling data，筛选后的数据由原始的50条变为8条，损失很多信息，所以计算出的协方差矩阵与真实情况差异很大。Comparison result: C is closer to the realistic situation, and D is not as close. Because C used as much known data as possible and produced a covariance that was not much different from the realistic situation. However, in D, after filtering the overlapping data, the quantity of data changed from 50 to only 8, and this significant loss of data caused the final covariance to be much more different from the realistic situation.

3.

a.

mu1 = 0.046002
mu2 = 0.099915
Covariance matrix:
[0.0101622 , 0.00492354],
[0.00492354, 0.02028441]

b.



Problem 3.

B. Method 1:

$$\mu = \begin{pmatrix} 0.04600157 \\ 0.0999\,1502 \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 0.0101622 & , 0.0049\,2354 \\ 0.00492354 & , 0.02028441 \end{pmatrix}$$

$$x_2 \mid x_1 = 0.6 \sim N(\bar{\mu}, \bar{\Sigma})$$

$$\bar{\mu} = \mu_2 + \Sigma_{21} \Sigma_{11}^{-1} (0.6 - \mu_1)$$

$$= 0.09991502 + \frac{0.00492354}{0.0101622} \times (0.6 - 0.04600157) \approx 0.3683$$

$$\bar{\Sigma} = \Sigma_{22} - \Sigma_{21} \cdot \Sigma_{11}^{-1} \cdot \Sigma_{21}$$

$$= 0.02028441 - 0.00492354 \times \frac{1}{0.0101622} \times 0.00492354 \approx 0.0179$$

Therefore $x_2 \mid x_1 = 0.6 \sim N(0.3683, 0.0179)$

Method 2:

In OLS, $E(x_2 \mid x_1) = \beta_0 + \beta_1 x_1$

where, $\beta_0 = \mu_2 - \beta_1 \mu_1 = 0.09991502 - 0.4845 \cdot 0.04600157 \approx 0.0776$

$$\beta_1 = \frac{Cov(x_1, x_2)}{Var(x_1)} = \frac{\sigma_{12}}{\sigma_1^2} = \frac{0.00492354}{0.0101622} \approx 0.4845$$

$$E(x_2 \mid x_1) = 0.0776 + 0.4845 x_1$$

$$E(x_2 \mid x_1 = 0.6) = 0.0776 + 0.4845 \cdot 0.6 \approx 0.3683$$

$$Var(x_2 \mid x_1) = \sigma_2^2 - \frac{\sigma_{12}^2}{\sigma_1^2} = 0.02028441 - \frac{(0.00492354)^2}{0.0101622} \approx 0.0179$$

$$x_2 \mid x_1 = 0.6 \sim N(0.3683, 0.0179)$$

c.

Problem 3

C:

$$x = Lz + \mu$$

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} + \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}$$

$$x_1 = L_{11} z_1 + \mu_1 \quad , \quad z_1 = \frac{x_1 - \mu_1}{L_{11}}$$

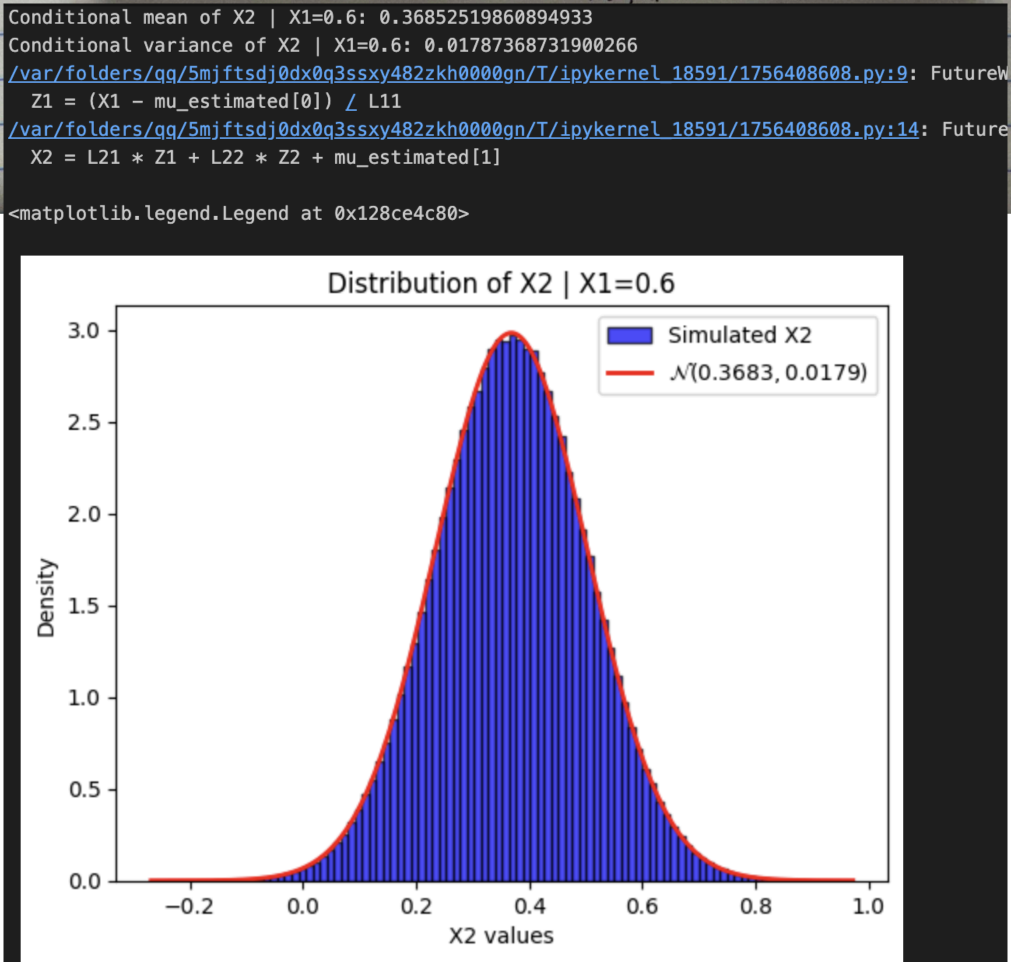
$$x_2 = L_{21} z_1 + L_{22} z_2 + \mu_2 = L_{21} \frac{x_1 - \mu_1}{L_{11}} + L_{22} z_2 + \mu_2 \text{ , where } z_2 \sim N(0,1)$$

From Python, $L = \begin{bmatrix} 0.10080772 & 0 \\ 0.04884093 & 0.13378703 \end{bmatrix}$

given $x_1 = 0.6$ $\quad \mu_1 = 0.04600157$, $\mu_2 = 0.09991502$, $L_{11} = 0.10080772$, $L_{21} = 0.04884093$

$L_{22} = 0.13378703$

After generating 100000 standard normal distribution sample



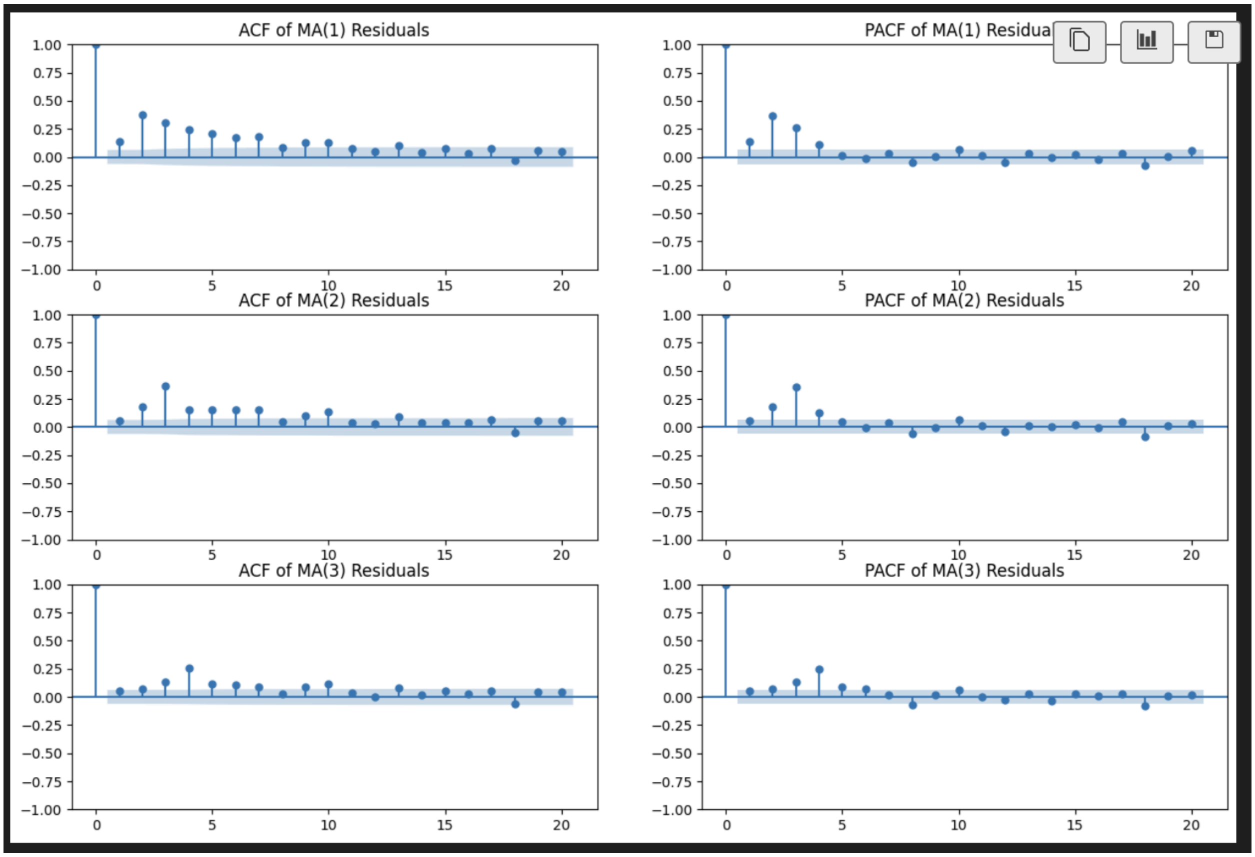
```
Conditional mean of X2 | X1=0.6: 0.36852519860894933
Conditional variance of X2 | X1=0.6: 0.01787368731900266
/var/folders/qq/5mjftsdj0dx0q3ssxy482zkh0000gn/T/ipykernel_18591/1756408608.py:9: FutureW
  Z1 = (X1 - mu_estimated[0]) / L11
/var/folders/qq/5mjftsdj0dx0q3ssxy482zkh0000gn/T/ipykernel_18591/1756408608.py:14: Future
  X2 = L21 * Z1 + L22 * Z2 + mu_estimated[1]

<matplotlib.legend.Legend at 0x128ce4c80>
```

After generating 100000 standard normal distribution samples, we found that the conditional mean is 0.36852519, and the conditional variance is 0.0178736.

With the result and the graph, distribution is proved.

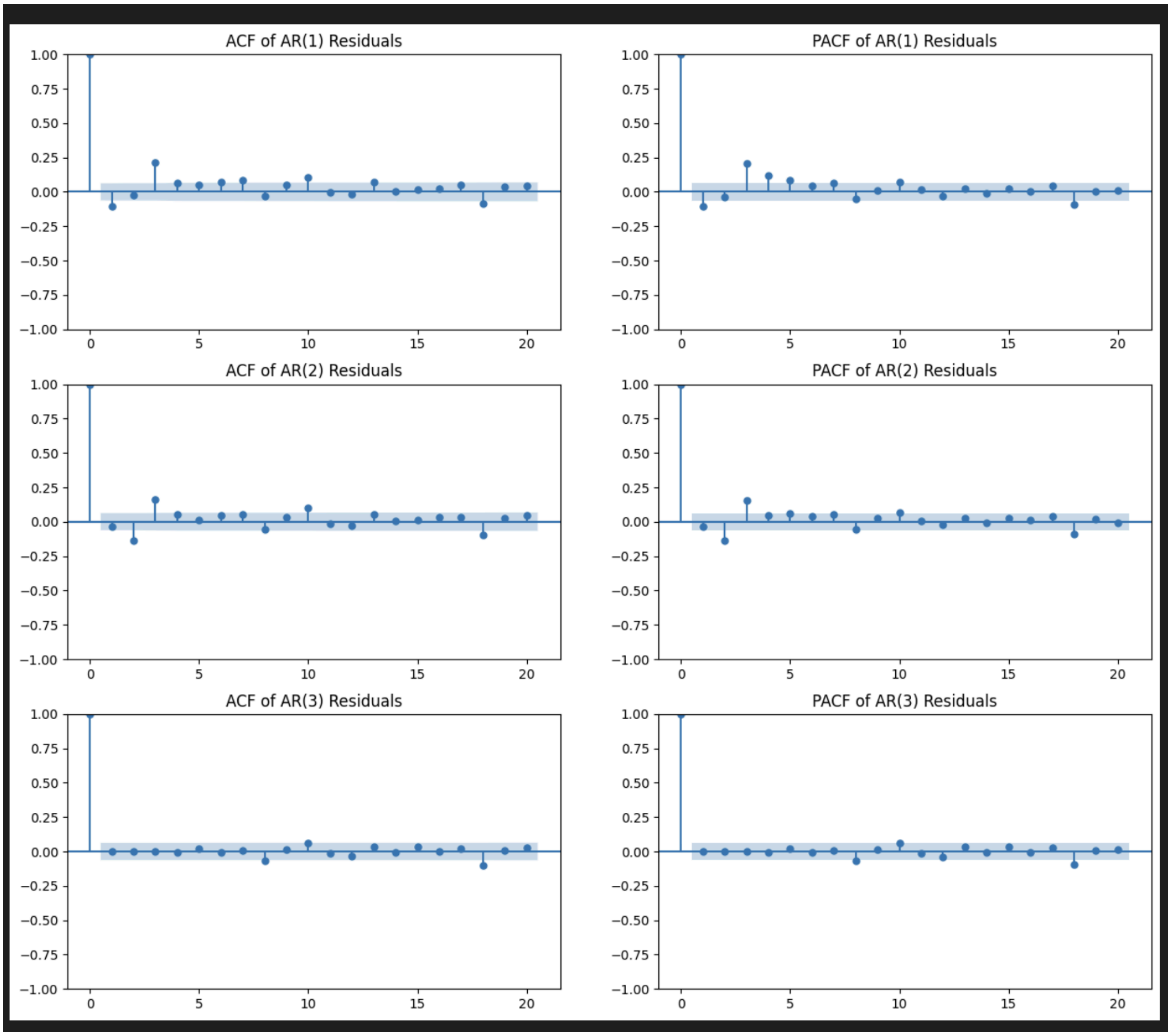
4.

a.



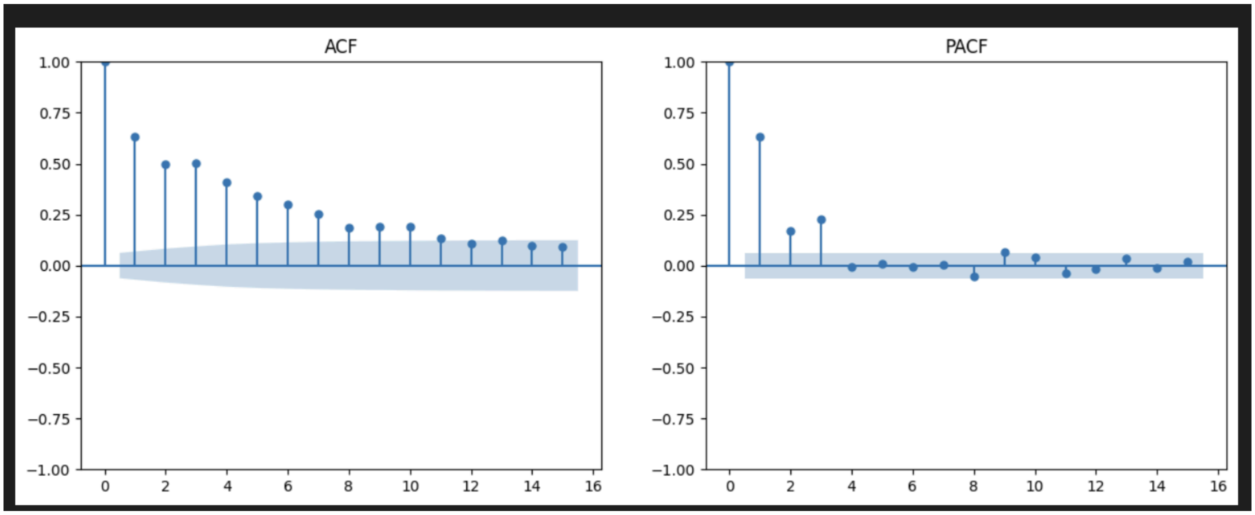
The ACF and PACF plots of the residuals for the MA(1), MA(2), and MA(3) models help evaluate how well each model captures the data's structure. Ideally, the residuals should resemble white noise, with ACF and PACF values falling within the confidence bounds, indicating no significant autocorrelations. For the MA(1) model, significant spikes in the residuals would suggest underfitting, as the model fails to capture all dependencies. The MA(2) model may show fewer significant spikes, indicating a better fit compared to MA(1). For the MA(3) model, if the ACF and PACF of the residuals exhibit no significant spikes and fall entirely within the confidence bounds, it implies that the MA(3) model adequately captures the data structure and is likely the best fit among the three. Comparing these residuals allows for determining the most appropriate model for the data.

b.



The ACF and PACF plots of the residuals for the AR(1), AR(2), and AR(3) models assess how well each model fits the data. Ideally, the residuals should exhibit no significant autocorrelations, with all values falling within the confidence bounds, indicating that the model sufficiently captures the data's structure. For the AR(1) model, if the residuals' ACF or PACF show significant spikes outside the confidence bounds, it suggests underfitting, where the model does not fully explain the data. The AR(2) model may show improved residual behavior, with fewer significant spikes, indicating a better fit. For the AR(3) model, if the ACF and PACF residuals show no significant spikes and resemble white noise, it implies that the AR(3) model provides the best fit among the three. Comparing the residuals across these models helps determine the most suitable one for the data.

c.



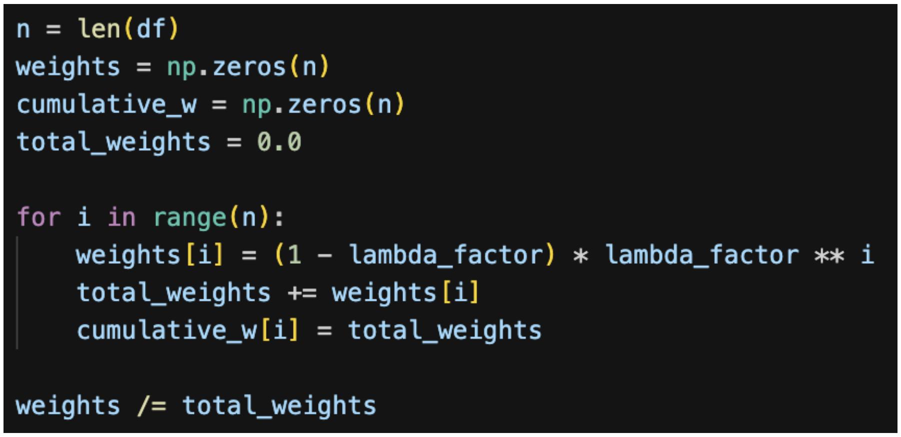
Based on the gradual decay in the ACF and the sharp cutoff in the PACF after lag 1, the data is best modeled using an AR(1) process. This model suggests that the current value depends on its immediately preceding value, and no higher lags are needed.

d.

After calculating all the AICcs, AR(3) has the smallest AICc, which means it is the best fit.

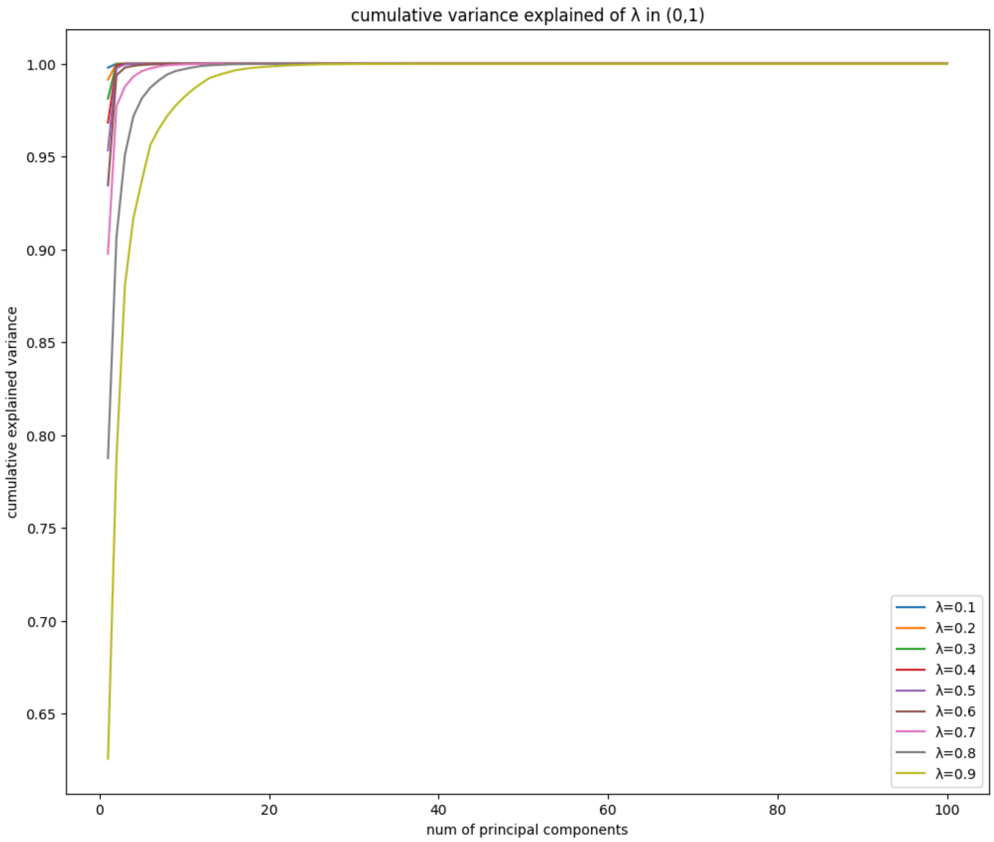
5.

a.



```
n = len(df)
weights = np.zeros(n)
cumulative_w = np.zeros(n)
total_weights = 0.0

for i in range(n):
    weights[i] = (1 - lambda_factor) * lambda_factor ** i
    total_weights += weights[i]
    cumulative_w[i] = total_weights

weights /= total_weights
```

b.



c.

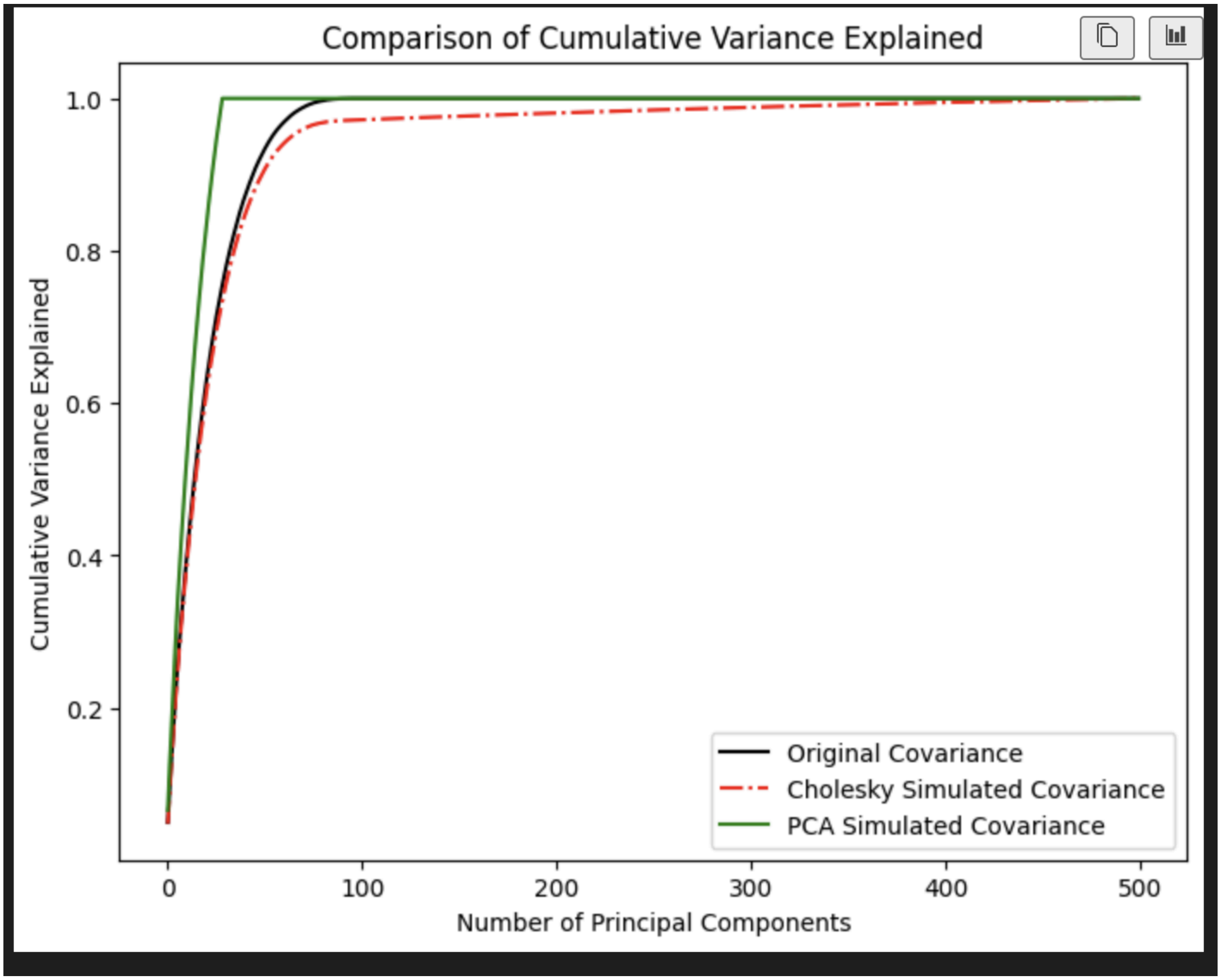
In an exponentially weighted covariance matrix, the parameter λ emphasizes recent observations relative to past data. Lower λ values give more weight to newer information, capturing short-term swings and adapting quickly to recent shifts. In contrast, higher λ values allocate more influence to older data, producing a smoother covariance estimate that reflects long-term trends. From the PCA findings and their cumulative variance explained, smaller λ values yield a more dynamic eigenvalue profile, showcasing dominant principal components shaped by recent movements. Conversely, larger λ values spread the eigenvalues more evenly, revealing the significance of enduring historical patterns. This demonstrates the balance between focusing on near-term volatility and preserving a broader historical perspective in covariance matrix estimation.

6.

a. Cholesky_simulation:
Frobenius Norm Difference 1.4441344036190884e-06
Execution Time: 0.074136 s
Pca:
Frobenius Norm Difference 5.789022296050302e-06
Execution Time: 0.028307 s

The Cholesky simulation achieved a Frobenius norm difference of 1.444×10^{-6}, indicating an almost perfect reconstruction of the original covariance matrix, while the PCA simulation had a slightly higher Frobenius norm difference of 5.789×10^{-6}, meaning it deviated more from the original structure. However, PCA was significantly faster, with an execution time of 0.028 seconds, compared to 0.074 seconds for Cholesky. This suggests a clear trade-off: Cholesky is more accurate but computationally heavier, whereas PCA is faster but sacrifices some accuracy due to dimensionality reduction. If preserving the exact covariance structure is critical (e.g., in finance, risk modeling, or precise simulations), Cholesky is the better choice. If speed and efficiency matter more (e.g., in data science and machine learning, where slight covariance deviations are acceptable), then PCA provides a computational advantage.

b.



The graph demonstrates that the Cholesky simulation (red line) closely follows the original covariance structure (black line), indicating its ability to accurately reproduce the full covariance matrix. In contrast, the PCA simulation (green line) captures most of the variance more quickly with fewer components but diverges slightly from the original as more components are added, reflecting its focus on dimensionality reduction. This confirms that Cholesky is ideal for preserving the exact covariance structure, while PCA is better suited for tasks requiring computational efficiency and emphasis on the most significant variance components.

c. time cost by Cholesky Root method: 0.0741 s
time cost by PCA method: 0.0283 s

The Cholesky Root method took 0.0741 seconds, while the PCA method was faster, taking only 0.0283 seconds. This highlights the computational efficiency of PCA, which leverages dimensionality reduction to reduce the complexity of the simulation. However, the Cholesky method's slightly longer time is justified by its higher accuracy in preserving the original covariance structure. The trade-off here is between speed and accuracy: PCA is advantageous when efficiency is critical, while Cholesky is preferred for precise covariance replication.

d. Advantages

The Cholesky Root method excels in accuracy, as it fully preserves the covariance structure of the original data. This makes it ideal for applications where exact replication of the covariance matrix is critical, such as in financial modeling, risk simulations, and scientific computations. By maintaining the integrity of the data's covariance, it ensures highly reliable results. On the other hand, the PCA method is highly efficient, as it reduces the computational burden by focusing only on the components that explain the most variance. This not only makes PCA faster but also simplifies high-dimensional problems by reducing dimensionality, making it particularly useful for exploratory data analysis, feature extraction, and scenarios where computational resources are limited.

Disadvantages

The Cholesky Root method, while accurate, is computationally intensive, especially for large covariance matrices, making it less suitable for high-dimensional datasets or real-time applications. Its inability to reduce dimensionality also means it does not offer the simplification benefits that PCA provides. Conversely, the PCA method sacrifices some accuracy by approximating the covariance structure, particularly for higher-order components. This can lead to deviations from the original covariance matrix, making it less reliable for tasks that require precise covariance preservation. Furthermore, PCA's reliance on dimensionality reduction may discard potentially relevant information that resides in the lower-variance components, which could be important in certain contexts.