1. Preliminary Work

What do you observe for the last command above (i.e. print(np.dot(U[:,0], U[:,1]))? Can you formally prove that this is the result you would expect for the specific structure in the matrix B?

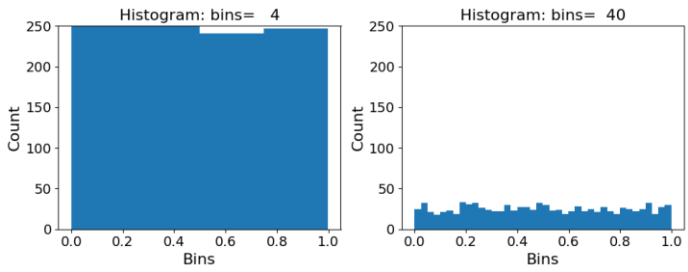
The last command print(np.dot(U[:,0], U[:,1])) computes the dot product between the first two eigenvectors of matrix B.

```
Eigenvalues (D):
[13.11766327  3.44324229  1.43909444]

Eigenvectors (U):
[[-0.19527095 -0.74608451  0.63657455]
 [-0.59990492 -0.42261825 -0.67934373]
 [-0.77587586  0.5145403   0.3650547 ]]

Dot product of the first and second eigenvectors:
-2.7755575615628914e-16
```

The dot product is very close to 0 (possibly with some small numerical error like 1e-16 or similar).

1. By definition: $Bv_1 = \lambda_1 v_1$ and $Bv_2 = \lambda_2 v_2$
2. Take the dot product: $v_1^T(Bv_2) = v_1^T(\lambda_2 v_2) = \lambda_2(v_1^T v_2)$
3. Since B is symmetric ($B = B^T$), we also have: $v_1^T(Bv_2) = (B^T v_1)^T v_2 = (Bv_1)^T v_2 = (\lambda_1 v_1)^T v_2 = \lambda_1(v_1^T v_2)$
4. Therefore: $\lambda_2(v_1^T v_2) = \lambda_1(v_1^T v_2)$
5. This gives us: $(\lambda_2 - \lambda_1)(v_1^T v_2) = 0$
6. Since $\lambda_1 \neq \lambda_2$ (distinct eigenvalues), we must have $\lambda_2 - \lambda_1 \neq 0$
7. Therefore: $v_1^T v_2 = 0$

2. Random Numbers and Uni-variate Densities



Though the data is from a uniform distribution, the histogram does not appear flat. Why?

-Even though the random numbers are drawn from a uniform distribution, the histogram is based on a finite sample (1000 numbers in this case). A perfectly flat histogram would **only** be achieved with an infinite number of samples. With a limited number of samples, there will always be some natural variation in the number of data points that fall into each bin. This is due to random chance. Some bins will happen to get slightly more numbers, and others slightly fewer, leading to the visible fluctuations in the histogram.

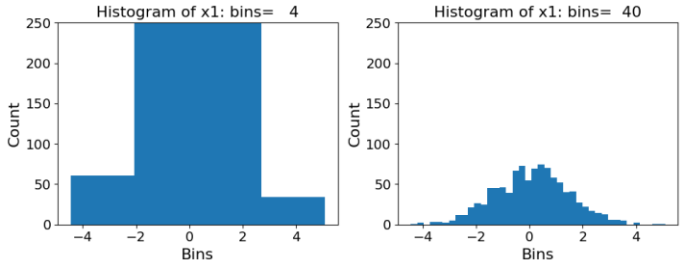
Every time you run it, the histogram looks slightly different? Why?

This is because new random numbers are generated every time the code is executed. The np.random.rand() function produces a different sequence of random numbers each time it's called (unless a random seed is set for same randomness each time). Since the histogram is a representation of the distribution of these specific random numbers, a different set of random numbers will result in a slightly different histogram.

How do the above observations change (if so how) if you had started with more data?

If the number of data points is increased (e.g., generate 10,000 or 100,000 uniform random numbers), the histogram will tend to look "flatter" and closer to the theoretical uniform distribution. With more samples, the effect of random fluctuations in each bin becomes less pronounced relative to the total number of data points. Similarly, while the histogram will still change slightly each time with more data, the differences between runs will become less significant. The overall shape will be more consistently uniform.

Let us now add and subtract some uniform random numbers:



What do you observe?

When this code is run and plot the histogram of x1, a non uniform historgram is obeserved. Instead, it will tend to be bell-shaped, resembling an almost normal (Gaussian) distribution. The values in x1 will be centered around zero, and values further away from zero will occur less frequently.

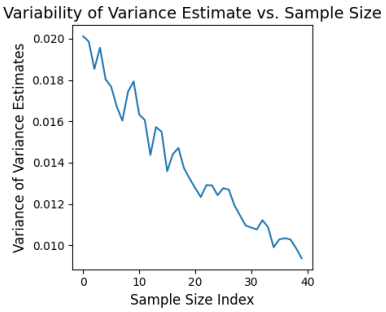
How does the resulting histogram change when you change the number of uniform random numbers you add and subtract?

In the code, adding and subtracting 12 uniform random numbers (np.sum(np.random.rand(12,1))). If changed the number of uniform random numbers being summed (e.g., change 12 to a larger number like 20 or 30), the resulting histogram will become even more bell-shaped and closer to a true normal distribution. If very small number (like 2 or 3) is selected, the bell shape will be less pronounced.

Is there a theory that explains your observation?

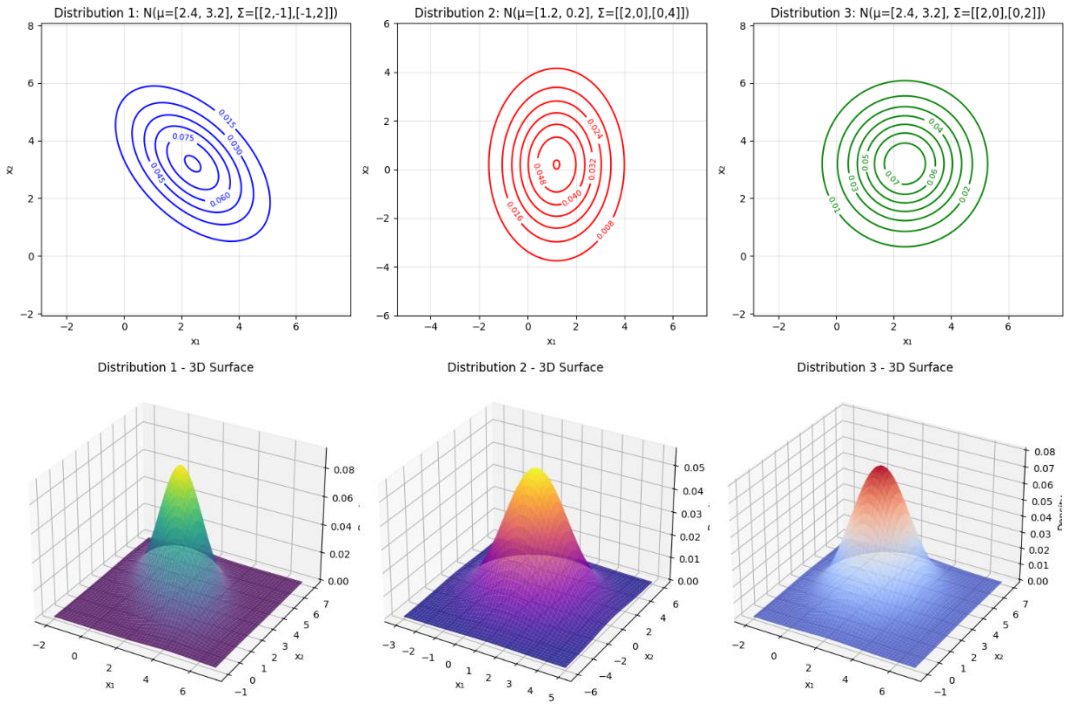
Yes, there is a fundamental theorem in probability and statistics that explains this observation, which is The Central Limit Theorem (CLT).

3. Uncertainty in Estimation



- It is anticipated that a downward trend will be observed in the plot. As the sample size is increased along the x-axis, the variance of the variance estimates, depicted on the y-axis, should decrease.
- This visually demonstrates that with a larger volume of data, the variability inherent in the parameter estimate (specifically, the sample variance) is reduced, thereby leading to an estimate that is considered more reliable.
- It is generally established that the utilization of larger sample sizes contributes to estimates of model parameters that are characterized by greater precision.

4. Bi-variate Gaussian Distribution

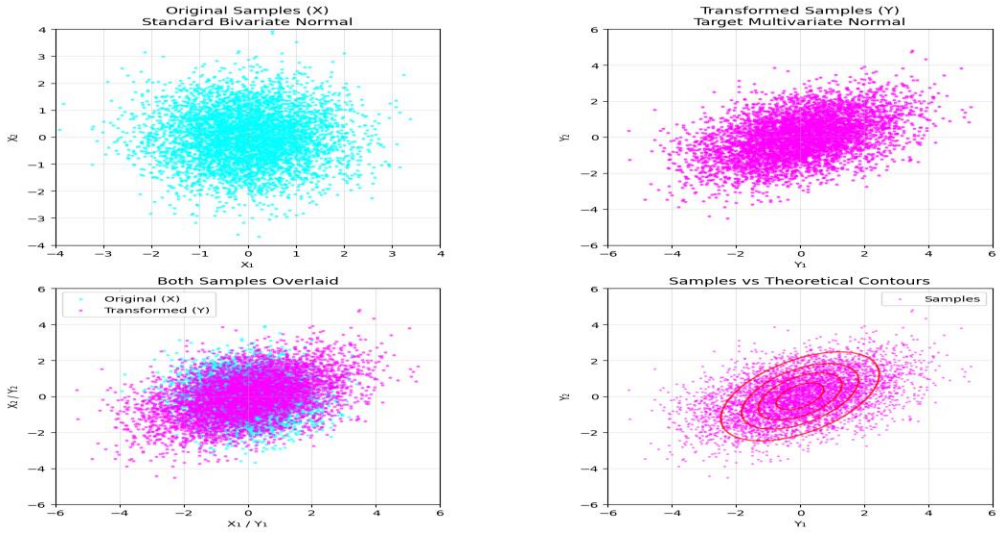


Analysis of Covariance Matrices:
==========
Distribution 1: C_1 = [[2, -1], [-1, 2]]
Eigenvalues: [3. 1.]
Determinant: 2.9999999999999996
Effect: Negative correlation creates diagonal elliptical contours

Distribution 2: C_2 = [[2, 0], [0, 4]]
Eigenvalues: [2. 4.]
Determinant: 7.999999999999998
Effect: Diagonal matrix creates axis-aligned ellipses, more spread in y-direction

Distribution 3: C_3 = [[2, 0], [0, 2]]
Eigenvalues: [2. 2.]
Determinant: 4.0
Effect: Identity scaled by 2 creates circular contours
==========
SHAPE COMPARISON:
Distribution 1: Tilted ellipses (negative correlation)
Distribution 2: Vertical ellipses (independent variables, different variances)
Distribution 3: Circles (independent variables, equal variances)

5. Sampling from a multi-variate Gaussian



MATHEMATICAL EXPLANATION

WHY DOES THIS WORK?

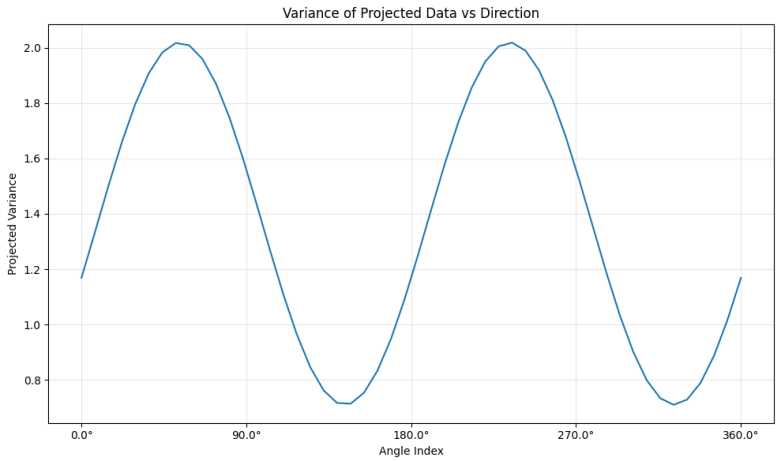
1. CHOLESKY DECOMPOSITION:
   - Any positive definite matrix C can be decomposed as C = A @ A.T
   - A is a lower triangular matrix called the Cholesky factor
   - This is unique and computationally efficient

2. TRANSFORMATION PROPERTY:
- If X ~ N(0, I) (standard normal)
- Then Y = X @ A ~ N(0, A @ A.T) = N(0, C)
- This preserves the multivariate normal distribution

3. LINEAR TRANSFORMATION THEOREM:
- If X is multivariate normal with mean μ and covariance Σ
- Then Y = AX + b is multivariate normal with:
  * Mean: Aμ + b
  * Covariance: A Σ A.T

4. OUR CASE:
- X ~ N(0, I) → Y = XA ~ N(0, AA.T) = N(0, C)
- We successfully transformed standard normal to our target distribution

6. Distribution of Projections



What are the maxima and minima of the resulting plot? The plot will show:

Maximum variance = largest eigenvalue of the covariance matrix = 2.0402
Minimum variance = smallest eigenvalue of the covariance matrix = 0.7164
The plot oscillates sinusoidally between these values

Relationship between eigenvalues/eigenvectors and maxima/minima? Key relationships:

Maximum projected variance = largest eigenvalue (λ_1)
Minimum projected variance = smallest eigenvalue (λ_2)
Direction of maximum variance = first eigenvector Direction of minimum variance = second eigenvector

Can you analytically confirm this for the 2D problem? Yes! For 2D data, the projected variance follows: $Var(\theta) = \lambda_1 \cos^2(\theta-\phi_1) + \lambda_2 \sin^2(\theta-\phi_1)$ Where:

λ_1, λ_2 are the eigenvalues φ_1 is the angle of the first eigenvector This explains the sinusoidal shape of the variance plot

- Physical Interpretation:

Principal Component Analysis (PCA) finds the directions of maximum variance The first principal component points in the direction where data has maximum spread The second principal component is perpendicular and captures remaining variance This is fundamental to dimensionality reduction and data compression techniques