Problems

1. Write a Matlab function called `Cholesky` that takes as input a (square, symmetric, positive-definite) matrix Σ and returns the Cholesky decomposition L, where $\Sigma = LL^T$. To compute L, the following algorithm should be implemented:

Step 1: Check to see if the matrix is square and determine its size ($k \times k$). If not, issue the error "Matrix must be square".

Step 2: Initialize L to a zero matrix of size $k \times k$.

Step 3: Repeat steps 4-6 for $i = 1, \ldots, k$:

Step 4: Compute the expression

$$l = \Sigma_{ii} - \sum_{n=1}^{i-1} L_{in}^2.$$

Step 5: If $l > 0$ then set $L_{ii} = \sqrt{l}$. If not, issue the error "Matrix must be positive-definite".

Step 6: Now, set

$$L_{ji} = \frac{1}{L_{ii}} \left(\Sigma_{ij} - \sum_{n=1}^{i-1} L_{in} L_{jn} \right).$$

for $j = i + 1, \ldots, k$.

Note, you will need to implement nested for loops for i and j. To make things efficient, however, the expressions in steps 4 and 6 should be vectorized using the `sum` command. You can check your code against the built-in Matlab command `chol`.

Now, in a separate Matlab script, generate a 3×10,000 matrix $Z \sim \mathcal{N}_3(0, I)$ of (independent) standard normal random numbers using the `randn` command. Compute $X = LZ \sim \mathcal{N}_3(0, \Sigma)$, where L is the Cholesky decomposition of the covariance matrix

$$\Sigma = \begin{bmatrix} 1 & 0.9 & 0.95 \\ 0.9 & 1 & 0.855 \\ 0.95 & 0.855 & 1 \end{bmatrix}.$$

With each column in matrix X representing a point in space, produce a 3D scatter plot. If everything is correct, you should have produced a 'cigar-shaped' cluster of points. To ensure that you use the same sequence of pseudo-random numbers as in the tutorial solution, issue the command `rng(0)` at the beginning of your code—this initializes the Matlab seed to zero.

(Note: In implementing the Cholesky decomposition, you were not asked to check whether or not the matrix Σ is symmetric. Can you think of a quick, i.e. vectorized, way of checking?)

2. Write a Matlab program to generate 10,000 (standard) normal random numbers using the rejection method on the exponential density with mean 1 (algorithm provided on page 12 of the notes). To generate the exponential random numbers, use uniform random numbers generated using the Matlab `rand` function and the inverse transform method on the cdf $G(x)$, derived by integrating $G(x) = \int_0^x g(u)\, du$. Remember to use the small simplification that if U is $\mathcal{U}[0, 1)$, then so is $1 - U$.

To start with, write code that does not use vectorization. Then, if you issue the command `rng(0)` at the beginning of your code and retain the ordering inherent in the algorithm on page 12 (i.e., generate Y first, then U), the first 5 numbers generated will be $[-0.4583, 0.0357, 0.0438, 1.9527, 0.4220]$. Now, side by side, produce a histogram of the numbers generated, and, for comparison a histogram of 10,000 normal random numbers generated by using `randn`.

If you have managed to finish this in good time, write a new version of the code that uses logical indexing to vectorize the code as effectively as possible. You can also try plotting the graphs found in Figure 1.3 in the notes.