Final Problem Set (300 Points Total)

Question 1 (60 Points)

A square matrix A is formed using the following command:

```
A = eye(3) + flipdim(eye(3),1)
```

Answer the following questions:

(a) Describe what `eye` does. How many arguments does the function take?

(b) Describe what `flipdim` does. How many arguments does the function take?

(c) What is the size of A?

(d) Write a short script that sets the value(s) of A that is (are) greater than 1 to 1. Name this new matrix as B. How many 1's does B have? (Hint: use for and if loops)

(e) Write a short script that sets the value(s) of B that is (are) equal to zero to 1. Name this new matrix as C. How many 1's does C have? (Hint: use for and if loops)

Question 2 (60 Points)

A neuron is a specialized and electrically excitable cell that processes and transmits information throughout the body. Assume that when a neuron is excited to transmit a signal, it takes a value of 1, otherwise 0. We can express electrical states (signals) of multiple neurons using a matrix notation. For instance, nine excited neurons can be represented as $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$.

(a) How many neurons are excited in A generated by the following script containing a nested for loop (a for loop within a for loop):

```
A = zeros(4,4);
for i = 1:length(A(:,1))
    for j = 1:length(A(1,:))
        if(i==j)
            A(i,j) = 1;
        end
    end
end
```

(b) It has been hypothesized that electrical states of a group of neurons are encoded by a "mask" matrix using the following equation:

$$\text{Encoded Signal} = (\text{Mask}) \cdot (\text{Original Signal})$$

The encoded signal from 169 (13×13) neurons can be downloaded from Blackboard. If the mask matrix used to generate the encoded signal is a 13-by-13 magic square (`magic(13)`), compute the original signal.

(c) You can visualize the signal you obtained in part (b) using the function `imagesc`. For example, `imagesc(B)` where `B` is the original signal will display the signal as an image. Run this command. What message does the original signal contain?

Question 3 (100 Points)

The concetration of a drug in the body, C_b (mg/L), can be modeled by the equation:

$$C_b = \frac{D}{V}\left(\frac{k_a}{k_a - k_e}\right)(e^{-k_e t} - e^{-k_a t})$$

where D is the drug dose administered (mg), V is the volume distribution (L), k_a is the absorption rate constant (h^{-1}), k_e is the elimination rate constant (h^{-1}), and t is the time (h) since the drug was administered. For a certain drug, the following parameters are given: $D = 80$ mg, $V = 25$ L, $k_a = 1.6$ h^{-1}, and $k_e = 0.4$ h^{-1}.

(a) Write a MATLAB function with inputs D, V, k_a, k_e, and t that computes C_b. Name the function as `computeC_b`.

(b) A single does is administered at $t = 0$. Write a script that calls the function you wrote for part (a) to plot C_b versus t for 8 hours (label and title the plot as well).

(c) Modify the script from part (b) to plot C_b versus t for 12 hours.

(d) Compute the maximum C_b during the 12-hour interval. At what time does the maximum C_b occur? (Hint: use `max` function)

(e) A medical device was designed to regulate C_b. The device does not allow C_b to go above 1.5 mg/L. Complete the if loop in the snippet below and place it in your script to plot the drug-regulated C_b versus t:

```
for i = 1:length(C_b)
    if(condition)
        statement;
    end

end
```

Question 4 (80 Points)

The ability to generate random variables in certain distributions is essential to any language. One very commonly used distribution, the normal distribution, can be generated from the uniform distribution via the Box-Muller Transform, which is defined as:

$$\begin{aligned} z_1 &= \sqrt{-2\ln x_1}\cdot\cos(2\pi x_2) \\ z_2 &= \sqrt{-2\ln x_1}\cdot\sin(2\pi x_2) \end{aligned}$$

Given that x_1 and x_2 are uniformly distributed between 0 and 1, then z_1 and z_2 will be normally distributed between 0 and 1.

(a) Create a function called `bmt.m` that takes no input arguments and returns the Box-Muller Transform of two random numbers generated using MATLAB's native `rand()` function, which is uniformly distributed. This can be done in two ways, by generating both x_1 and x_2 together or separately (either is fine). The function should return two numbers, z_1 and z_2, in a 1x2 or 2x1 vector. Then, write a short script (should not take more than 15-20 lines) to test your `bmt.m` function called `main.m`. Have your script generate and store 100 generated z_1 and z_2 numbers with `bmt.m` and visualize your distributions with the built-in `hist` function. z_1 and z_2 can be stored either together or separately. You can create a new figure using:

```
hist(params);
figure;
hist(params);
```

(b) In the same script, generate 1000 numbers using the built-in `rand()` function, and create a new histogram of these numbers using the `hist` function again. Your script should now generate 3 total figures - z_1, z_2, and the built-in `rand` result. Briefly (in 1 sentence) note the difference in shape between the distributions.

(c) Now that you have created your very own normal random variable generator, you can compare it to MATLAB's built-in random variable generator, `randn`. At the very end of your script, generate 1000 random variables using `randn`, and visualize the result using `hist`. This will bring you to a final total of 4 figures, the two generated from `bmt.m`, one from `rand()`, and one from `randn()`. Briefly (in 1 sentence) note any similarities or differences in shape between all the distributions.