Homework 1

21.03.2024

This homework aims to cultivate your ability to conceptualize problems in terms of matrices and vectors, reflecting MATLAB's approach to data organization. You'll discover that complex tasks can frequently be streamlined into a few lines of code by leveraging the right functions and organizing data effectively. Additionally, this task is intended to enhance your proficiency in utilizing help resources to familiarize yourself with unfamiliar functions, which are highlighted in **bold** for your convenience. Remember, spaces are not permitted in the naming of script files.

Homework must be submitted in Google Classroom by 17:00 on 25.03.2024.

What to turn in: At the end of the homework, you will be asked for a MATLAB file (m. file) and a document (*.doc or *.pdf). Copy your MATLAB code into this document in an organized manner. If a question asks you to plot or display something to the screen, also include the plot and screen output your code generates. When submitting these two files to Google Classroom, you must upload the zip file named as:

"HW1_SametORAN_150103014.zip".

1. **Scalar variables.** Make the following variables
   a. $a = 10$
   b. $b = 2.5 \times 10^{23}$
   c. $c = 2 + 3i$, where i is the square root of -1
   d. $d = e^{j2\pi/3}$, where j is the square root of -1 and e is Euler's number[1] (use **exp**, **pi**)

2. **Vector variables.** Make the following variables
   a. $aVec = \begin{bmatrix} 3.14 & 15 & 9 & 26 \end{bmatrix}$
   b. $bVec = \begin{bmatrix} 2.71 \\ 8 \\ 28 \\ 182 \end{bmatrix}$
   c. $cVec = \begin{bmatrix} 5 & 4.8 & \cdots & -4.8 & -5 \end{bmatrix}$ (all the numbers from 5 to -5 in increments of -0.2)
   d. $dVec = \begin{bmatrix} 10^0 & 10^{0.01} & \cdots & 10^{0.99} & 10^1 \end{bmatrix}$ (Logarithmically spaced numbers between 1 and 10, use **logspace**, make sure you get the length right!)
   e. $eVec = Hello$ ($eVec$ is a string, which is a vector of characters)

3. **Matrix variables.** Make the following variables

   a. $aMat = \begin{bmatrix} 2 & \cdots & 2 \\ \vdots & \ddots & \vdots \\ 2 & \cdots & 2 \end{bmatrix}$ a 9x9 matrix full of 2's (use **ones** or **zeros**)

   b. $bMat = \begin{bmatrix} 1 & 0 & \cdots & & 0 \\ 0 & \ddots & 0 & \ddots & \\ \vdots & 0 & 5 & 0 & \vdots \\ & \ddots & 0 & \ddots & 0 \\ 0 & & \cdots & 0 & 1 \end{bmatrix}$ a 9x9 matrix of all zeros, but with the values $\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 4 & 3 & 2 & 1 \end{bmatrix}$ on the main diagonal (use **zeros**, **diag**).

   c. $cMat = \begin{bmatrix} 1 & 11 & \cdots & 91 \\ 2 & 12 & \ddots & 92 \\ \vdots & \vdots & \ddots & \vdots \\ 10 & 20 & \cdots & 100 \end{bmatrix}$ a 10x10 matrix where the vector 1:100 runs down the columns (use **reshape**).

   d. $dMat = \begin{bmatrix} NaN & NaN & NaN & NaN \\ NaN & NaN & NaN & NaN \\ NaN & NaN & NaN & NaN \end{bmatrix}$ a 3x4 NaN matrix (use **nan**)

   e. $eMat = \begin{bmatrix} 13 & -1 & 5 \\ -22 & 10 & -87 \end{bmatrix}$

   f. Make $fMat$ be a 5x3 matrix of random integers with values on the range -3 to 3 (First use **rand** and **floor** or **ceil.** Now only use **randi**)

4. **Scalar equations.** Using the variables created in 1, calculate x, y, and z.

   a. $x = \dfrac{1}{1+e^{(-(a-15)/6)}}$

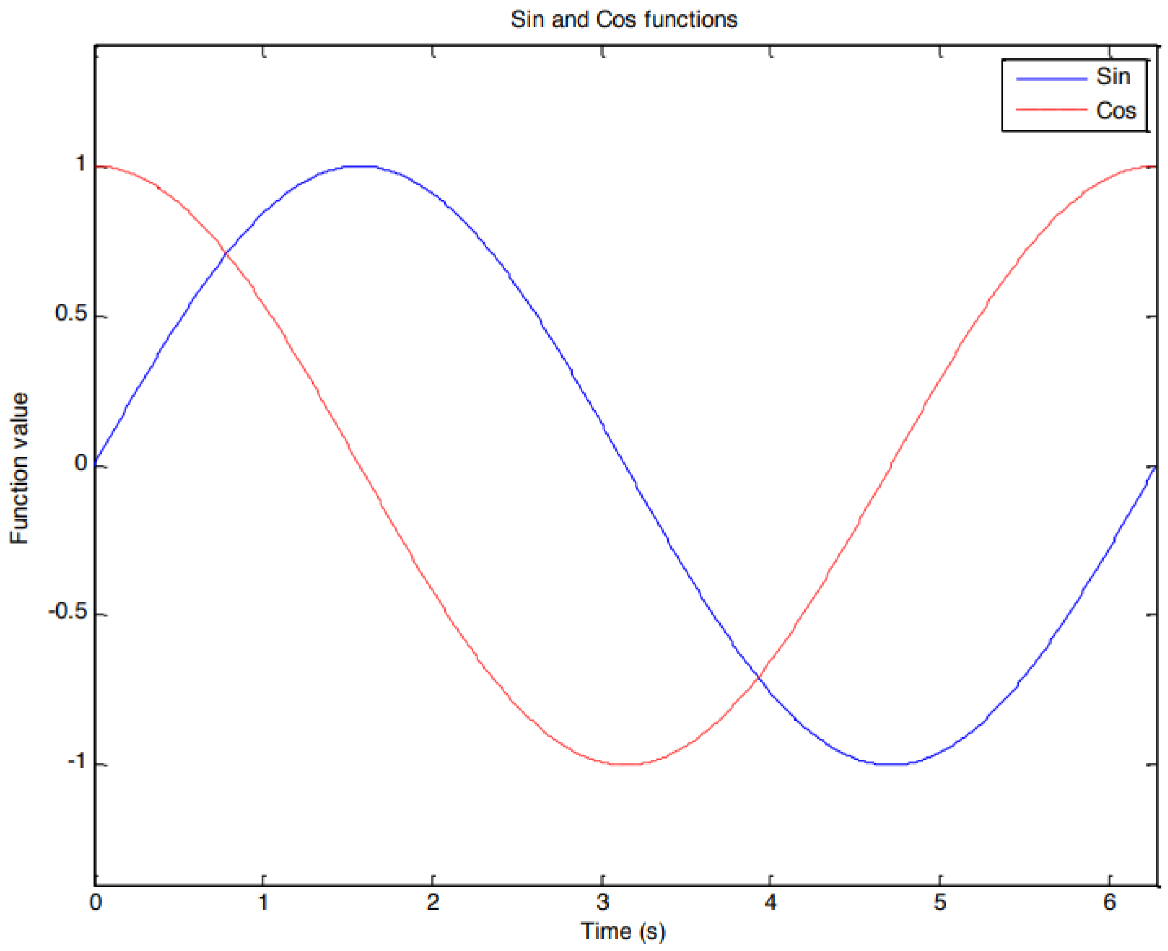
   b. $y = \left(\sqrt{a} + \sqrt[21]{b}\right)^{\pi}$, recall that $\sqrt[g]{h} = h^{1/g}$, and use **sqrt.** You can also use **nthroot** (refer to the MATLAB help to understand the difference between **nthroot** and a fractional power)

   c. $z = \dfrac{\log\left(\Re\left[(c+d)(c-d)\right]\sin(a\pi/3)\right)}{c\bar{c}}$ where $\Re$ indicates the real part of the complex number in brackets, $\bar{c}$ is the complex conjugate of c, and $\log$ is the *natural* log (use **real**, **conj**, **log**).

6. **Common functions and indexing.**
   a. Make $cSum$ the column-wise sum of $cMat$. The answer should be a row vector (use **sum**).
   b. Make $eMean$ the mean across the rows of $eMat$. The answer should be a column (use **mean**).
   c. Replace the top row of $eMat$ with $[1 \quad 1 \quad 1]$.
   d. Make $cSub$ the submatrix of $cMat$ that only contains rows 2 through 9 and columns 2 through 9.
   e. Make the vector $lin = [1 \quad 2 \quad \cdots \quad 20]$ (the integers from 1 to 20), and then make every even value in it negative to get $lin = [1 \quad -2 \quad 3 \quad -4 \quad \cdots \quad -20]$.
   f. Make r a 1x5 vector using **rand**. Find the elements that have values <0.5 and set those values to 0 (use **find**).

7. **Plotting multiple lines and colors.** In class we saw how to plot a single line in the default blue color on a plot. You may have noticed that subsequent plot commands simply replace the existing line. Here, we'll write a script to plot two lines on the same axes.
   a. Open a script and name it `twoLinePlot.m`. Write the following commands in this script.
   b. Make a new figure using **figure**
   c. We'll plot a sine wave and a cosine wave over one period
      i. Make a time vector t from 0 to 2π with enough samples to get smooth lines
      ii. Plot $\sin(t)$
      iii. Type **hold on** to turn on the 'hold' property of the figure. This tells the figure not to discard lines that are already plotted when plotting new ones. Similarly, you can use **hold off** to turn off the hold property.
      iv. Plot $\cos(t)$ using a red dashed line. To specify line color and style, simply add a third argument to your plot command (see the third paragraph of the **plot** help).

      This argument is a string specifying the line properties as described in the help file. For example, the string 'k:' specifies a black dotted line.
   d. Now, we'll add labels to the plot
      i. Label the x axis using **xlabel**
      ii. Label the y axis using **ylabel**
      iii. Give the figure a title using **title**
      iv. Create a legend to describe the two lines you have plotted by using **legend** and passing to it the two strings 'Sin' and 'Cos'.
   e. If you run the script now, you'll see that the x axis goes from 0 to 7 and y goes from -1 to 1. To make this look nicer, we'll manually specify the x and y limits. Use **xlim** to set the x axis to be from 0 to 2π and use **ylim** to set the y axis to be from -1.4 to 1.4.
   f. Run the script to verify that everything runs right. You should see something like this:


Sin and Cos functions
Sin
Cos
Function value
Time (s)
1
0.5
0
-0.5
-1
0
1
2
3
4
5
6