Homework 1 Description
CmpE 362 Spring 2016

Instructor : Fatih Alagoz
Teaching Assistant : Yekta Said Can
Due: 3 March, 23:59, sharp

Homework 1

This homework is designed to teach you to think in terms of matrices and vectors because this is how Matlab organizes data. You will find that complicated operations can often be done with one or two lines of code if you use appropriate functions and have the data stored in an appropriate structure. The other purpose of this homework is to make you comfortable with using **help** to learn about new functions. The names of the functions you'll need to look up are provided in **bold** where needed.

For problems 1-7, write a script called `shortProblems.m` and put all the commands in it. Separate and label different problems using comments.

1. **Scalar variables.** Make the following variables
   a. $a = 10$
   b. $b = 2.5 \times 10^{25}$
   c. $c = 4 + 3i$, where i is the imaginary number
   d. $d = e^{j2\pi/3}$, where j is the imaginary number and e is Euler's number (use **exp**, **pi**)

2. **Vector variables.** Make the following variables
   a. $aVec = \begin{bmatrix} 3.14 & 16 & 19 & 26 \end{bmatrix}$
   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 (use **rand** and **floor** or **ceil**)

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**

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**).

5. **Vector equations.** Using the variables created in 2, solve the equations below, elementwise. For example, in part a, the first element of $xVec$ should just be the function evaluated at the value of the first element of $cVec$: $xVec_1 = \dfrac{1}{\sqrt{2\pi 2.5^2}} e^{-cVec_1^2/(2 \cdot 2.5^2)}$, and similarly for all the other elements so that $xVec$ and $cVec$ have the same size. Use the elementwise operators **.***, **./**, **.^**.

a. $xVec = \frac{1}{\sqrt{2\pi 2.5^2}} e^{-cVec^2/(2 \bullet 2.5^2)}$

b. $yVec = \sqrt{\left(aVec^T\right)^2 + bVec^2}$, $aVec^T$ indicates the transpose of $aVec$

c. $zVec = \log_{10}(1/dVec)$, remember that $\log_{10}$ is the log base 10, so use **log10**

6. **Matrix equations.** Using the variables created in 2 and 3, solve the equations below. Use matrix operators.

   a. $xMat = (aVec \bullet bVec) \bullet aMat^2$

   b. $yMat = bVec \bullet aVec$, note that this is *not* the same as $aVec \bullet bVec$

   c. $zMat = |cMat|(aMat \bullet bMat)^T$, where $|cMat|$ is the determinant of $cMat$, and T again indicates the transpose (use **det**).

7. 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**).

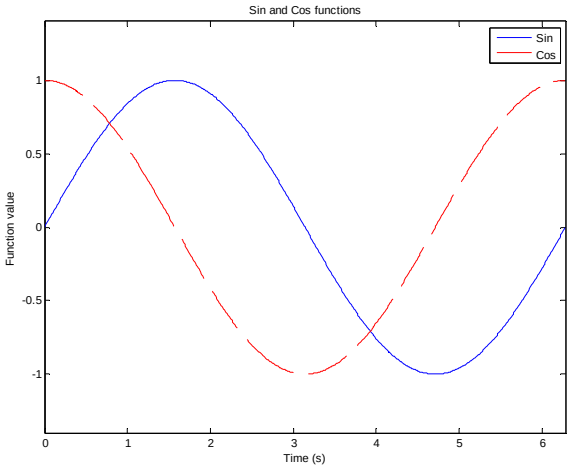
   c. Replace the top row of $eMat$ with $[1 \quad 1 \quad 1]$.

   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 other 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**).

8. **Plotting multiple lines and colors.** In class we covered 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:



9. **Manipulating variables.** Write a script to read in some grades, curve them, and display the overall grade. To do this, you'll need to download the file `classGrades.mat` off the class website and put it in the same folder as your script.
    a. Open a script and name it `calculateGrades.m`. Write all the following commands in this script.
    b. Load the `classGrades` file using **load**. This file contains a single variable called *namesAndGrades*
    c. To see how *namesAndGrades* is structured, display the first 5 rows on your screen. The first column contains the students 'names', they're just the integers from 1 to 15. The remaining 7 columns contain each student's score (on a scale from 0 to 5) on each of 7 assignments. There are also some NaNs which indicate that a particular student was absent on that day and didn't do the assignment.
    d. We only care about the grades, so extract the submatrix containing all the rows but only columns 2 through 8 and name this matrix *grades* (to make this work on any size matrix, don't hard-code the 8, but rather use **end** or **size(namesAndGrades,2)**).
    e. Calculate the mean score on each assignment. The result should be a 1x7 vector containing the mean grade on each assignment.
        i. First, do this using **mean**. Display the mean grades calculated this way. Notice that the NaNs that were in the grades matrix cause some of the mean grades to be NaN as well.
        ii. To fix this problem, do this again using **nanmean**. This function does exactly what you want it to do, it computes the mean using only the numbers that are not NaN. This means that the absent students are not considered in the calculation, which is what we want. Name this mean vector *meanGrades* and display it on the screen to verify that it has no NaNs
    f. Normalize each assignment so that the mean grade is 3.5 You'll want to divide each column of *grades* by the correct element of *meanGrades* .

        i. Make a matrix called *meanMatrix* such that it is the same size as *grades* , and each row has the values *meanGrades* . Do this by taking the outer product of a 15x1 vector of ones and the vector *meanGrades* , which is a row (use **ones**, *****). Display *meanMatrix* to verify that it looks the way you want.
        ii. To calculate the curved grades, do the following: $curvedGrades = 3.5\left(grades / meanMatrix\right)$. Keep in mind that you want to do the division elementwise.
        iii. Compute and display the mean of *curvedGrades* to verify that they're all 3.5 (**nanmean**).

iv. Because we divided by the mean and multiplied by 3.5, it's possible that some grades that were initially close to 5 are now larger than 5. To fix this, find all the elements in $curvedGrades$ that are greater than 5 and set them to 5. Use **find**

g. Calculate the total grade for each student and assign letter grades

i. To calculate the $totalGrade$ vector, which will contain the numerical grade for each student, you want to take the mean of $curvedGrades$ across the columns (use **nanmean**, see help for how to specify the dimension). Also, we only want to end up with numbers from 1 to 5, so calculate the ceiling of the $totalGrade$ vector (use **ceil**).

ii. Make a string called $letters$ that contains the letter grades in increasing order: FDCBA

iii. Make the final letter grades vector $letterGrades$ by using $totalGrade$ (which after the ceil operation should only contain values between 1 and 5) to index into $letters$.

iv. Finally, display the following using **disp**: Grades: $letterGrades$

h. Run the script to verify that it works. You should get an output like this

```
>> calculateGrades
ans =
    1.0000    2.5064    3.6529    2.4617    3.3022    2.5189    0.0963    4.6502
    2.0000    2.1586    3.2324    3.4737    0.2378    2.4480    0.4194    1.9951
    3.0000    4.9878       NaN    4.8637    1.7439    4.3852    4.8740    0.2370
    4.0000    4.0580    1.9914    1.6388    2.2567    1.7657    3.2567    1.7119
    5.0000    2.4283    3.7491    4.1890       NaN    2.2472    1.1562    3.6798
meanGrades =
       NaN       NaN    2.8361       NaN    2.8540    1.6481       NaN
meanGrades =
    2.9690    2.9445    2.8361    2.4879    2.8540    1.6481    2.5677
meanMatrix =
    2.9690    2.9445    2.8361    2.4879    2.8540    1.6481    2.5677
    2.9690    2.9445    2.8361    2.4879    2.8540    1.6481    2.5677
    2.9690    2.9445    2.8361    2.4879    2.8540    1.6481    2.5677
    2.9690    2.9445    2.8361    2.4879    2.8540    1.6481    2.5677
    2.9690    2.9445    2.8361    2.4879    2.8540    1.6481    2.5677
    2.9690    2.9445    2.8361    2.4879    2.8540    1.6481    2.5677
    2.9690    2.9445    2.8361    2.4879    2.8540    1.6481    2.5677
    2.9690    2.9445    2.8361    2.4879    2.8540    1.6481    2.5677
    2.9690    2.9445    2.8361    2.4879    2.8540    1.6481    2.5677
    2.9690    2.9445    2.8361    2.4879    2.8540    1.6481    2.5677
    2.9690    2.9445    2.8361    2.4879    2.8540    1.6481    2.5677
    2.9690    2.9445    2.8361    2.4879    2.8540    1.6481    2.5677
    2.9690    2.9445    2.8361    2.4879    2.8540    1.6481    2.5677
    2.9690    2.9445    2.8361    2.4879    2.8540    1.6481    2.5677
    2.9690    2.9445    2.8361    2.4879    2.8540    1.6481    2.5677
ans =
    3.5000    3.5000    3.5000    3.5000    3.5000    3.5000    3.5000
Grades: BCBBBACCBCCCCAB
```

For problems 10-17, write a script called `signalAndNoise.m` and put all the commands in it. Separate and label different problems using comments.

10. Let x is vector of real numbers (-100:100)

    plot y1 = sinx, y2=sin50x, y3=50sinx, y4= sinx+50,y5= sin(x+50),y6= 50sin50x, y7= x*sinx, y8=sinx/x

    "Use 4x2 subplot to fit all subfigures belong to a single figure" (Hint: write help for SUBPLOT in MATLAB)

11. Let x is vector of real numbers (-20:20)

    Plot y1 = sinx, y2=sin50x, y3=50sinx, y4= sinx+50,y5= sin(x+50),y6= 50sin50x, y7= x*sinx, y8=sinx/x , y9= y1+y2+y3+y4+y5+y6+y7+y8

    "Use 5x2 subplot to fit all subfigures belong to a single figure"

12. *randn* generates zero-mean, unit variance Gaussian distributed random number in $(-\infty,\infty)$. Generate 41 random numbers following Gaussian distributed random numbers, call this as vector z.

    Plot y10= z, y11 = z+x , y12= z+sinx, y13= z sinx, y14=xsinz, y15= sin(x+z), y16= zsin50x, y17=sin(x+50z) y18=sinx/z, y19= y11+y12+y13+y14+y15+y16+y17+y18

    "Use 5x2 subplot to fit all subfigures belong to a single figure"

13. *rand* generates uniformly distributed random number in [0,1] . Generate 41 random numbers following uniformly distributed random numbers.
    Plot y20= z, y21 = z+x , y22= z+sinx, y23= z sinx, y24=xsinz, y25= sin(x+z), y26= zsin50x, y27=sin(x+50z) y28=sinx/z, y29= y21+y22+y23+y24+y25+y26+y27+y28

    "Use 5x2 subplot to fit all subfigures belong to a single figure"

14. Starting with z (0,1) Gaussian(Normal) Random variable. (Use help menu for "hist")
    a. Generate 10000 random variables with mean 0, variance 1; call it r1 vector
    b. Generate 10000 random variables with mean 0, variance 4; call it r2 vector
    c. Generate 10000 random variables with mean 0, variance 16; call it r3 vector
    d. Generate 10000 random variables with mean 0, variance 256; call it r4 vector

    Plot hist(r1), hist(r2), hist(r3),hist(r4) on the same figure for comparison purposes

15. Starting with z (0,1) Gaussian Random variable. (Use help menu for "hist")
    a. Generate 10000 random variables with mean 10, variance 1; call it r6 vector
    b. Generate 10000 random variables with mean 20, variance 4; call it r7 vector
    c. Generate 10000 random variables with mean -10, variance 1; call it r8 vector
    d. Generate 10000 random variables with mean -20, variance 4; call it r9 vector

    Plot hist(r6), hist(r7), hist(r8),hist(r9) on the same figure for comparison purposes

16. Starting with z (0,1) uniformly distributed random variable.
    a. Generate 10000 random variables with mean 0, variance 1; call it r11 vector
    b. Generate 10000 random variables with mean 0, variance 4; call it r21 vector
    c. Generate 10000 random variables with mean 0, variance 16; call it r31 vector
    d. Generate 10000 random variables with mean 0, variance 256; call it r41 vector

    Plot hist(r11), hist(r21), hist(r31),hist(r41) on the same figure for comparison purposes

17. Starting with z (0,1) uniformly distributed random variable. (Use help menu for "hist")
    a. Generate 10000 random variables with mean 10, variance 1; call it r61 vector
    b. Generate 10000 random variables with mean 20, variance 4; call it r71 vector
    c. Generate 10000 random variables with mean -10, variance 1; call it r81 vector
    d. Generate 10000 random variables with mean -20, variance 4; call it r91 vector

    Plot hist(r61), hist(r71), hist(r81),hist(r91) on the same figure for comparison purposes

18. Briefly describe what you have learnt from the above plots (plots from Questions 10-17).

19. Briefly describe what you have learnt about MATLAB. What were the challenges that you faced? What are the differences (advantages and disadvantages) between MATLAB and the other programming languages you have learned so far?

Submission and Grading

Prepare a report includes your code , explanations and comments of your code for each question. Figures would be in the report also. Add the answer to the 18th and 19th questions to your report.
Compress the report and code files. Name it as "YourNumber CmpE362 HW1.zip"(or rar, or 7z etc.). Send the file to yektasaid.can@gmail.com before the deadline. Subject of the mail would be CmpE362 HW1.

Notes

Deadline is strict. Do not send after deadline. When copying is detected, both parties will get zero.