BIOE 198MI Biomedical Data Analysis. Spring Semester 2019
Lab 1b. Matrices, arrays, m-files, I/O, custom functions

A. Scalars, vectors, and matrices versus arrays and the associated syntax

In terms of computing applications, vectors and matrices are similar to 1 and 2-D arrays. While the mathematical properties and very specific for vectors and matrices, Matlab treats them the same for the most part so it makes sense for us to introduce a little matrix math. Vectors like $\mathbf{x} = x\hat{\imath} + y\hat{\jmath}$ include constant unit vectors $\hat{\imath}, \hat{\jmath}$ that indicate direction while scalar variables x,y indicate magnitude. A unit vector has magnitude 1. (Equivalently, we can apply polar coordinates and Matlab's `angle`).

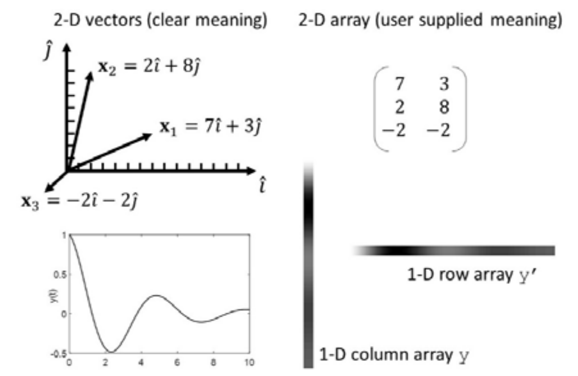
2-D vectors (clear meaning)
2-D array (user supplied meaning)
$\hat{\jmath}$
$\mathbf{x}_2 = 2\hat{\imath} + 8\hat{\jmath}$
$\mathbf{x}_1 = 7\hat{\imath} + 3\hat{\jmath}$
$\mathbf{x}_3 = -2\hat{\imath} - 2\hat{\jmath}$
$\hat{\imath}$
7 3
2 8
−2 −2
1-D row array y′
1-D column array y

Figure 1. Illustration of vectors, matrix, and arrays.

```
>> a=2; %defines scalar at an address
labeled a with value 2.
>> v = [1;2;3;4;5;6] %6x1 column array.
v =
     1
     2
     3
     4
     5
     6
>> u = v' %u is transpose of v.
u =
     1     2     3     4     5     6
or
>> u = 1:6
or
>> u = [1 2 3 4 5 6]

>> b=a*v; %scalar-vector product.  Note that b' = (a*v)′ = a*v'

>> b=v'*v %v is a vector (1-D array) and b is a scalar given by
```
`% inner product v'*v =` $\sum_{j=1}^{N} v_j^2 = 1 + 4 + 9 + 16 + 25 + 36 = 91$

```
>> b = 91

>> B = v*v' %define B as 6x6 matrix given by the outer product v*v'
B =
     1     2     3     4     5     6
     2     4     6     8    10    12
     3     6     9    12    15    18
     4     8    12    16    20    24
     5    10    15    20    25    30
     6    12    18    24    30    36
>> % note vector products do not commute
```

$$\boldsymbol{uv}^t = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} \begin{pmatrix} v_1 & v_2 & v_3 \end{pmatrix} = \begin{pmatrix} u_1v_1 & u_1v_2 & u_1v_3 \\ u_2v_1 & u_2v_2 & u_2v_3 \\ u_3v_1 & u_3v_2 & u_3v_3 \end{pmatrix}$$

See https://www.coursera.org/lecture/matrix-algebra-engineers/inner-and-outer-products-3JZwp

```
>> % type in 3x2 matrix C and 2x3 matrix D
>> C = [1 2;3 4;5 6] %spaces & semicolons indicated different actions
C =
     1     2
     3     4     (3x2 matrix)
     5     6

>> D = [1 3 5;2 4 6]
D =
     1     3     5     (2x3 matrix)
     2     4     6
```

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

$$\boldsymbol{B} = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

$$\boldsymbol{AB} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}$$

```
>> E = C-D'
E =
     0     0
     0     0     (3x2 – 3x2 = 3x2 matrix)
     0     0

>> F = C'-D
F =
     0     0     0     (2x3 – 2x3 = 2x3 matrix)
     0     0     0

>> % Note that C = D' and C' = D but C-D' ≠ C'-D.  Also,
>> D = [1:3;4:6]  %another method for generating matrices
D =
     1     2     3
     4     5     6

>> E = C*D % matrix product exist only if C is NxM, D is MxP, and E is NxP
E =
     9    12    15
    19    26    33
    29    40    51

>> F = E/D %right divide D into E: F = E/D = C*D/D = C.  Note since
% C and D are rectangular matrices, inv(C) and inv(D) do not exist.
F =
    1.0000    2.0000
    3.0000    4.0000
    5.0000    6.0000

>> G = C\E % left divide of C into E: G = C\E = C\C*D = D
G =
    1.0000    2.0000    3.0000
    4.0000    5.0000    6.0000
```

Default number values in MATLAB are double precision float point where 5 significant figures are displayed. For example, the decimal value for 5/4 is 1.2500.

Exercise: Do the following both analytically and numerically.

1. Find the inner and outer products: `clear all; a = [4;3;2;1];`

2. Let `A = [1 2;3 4];` Find `A^2, A.^2, sqrt(A), A*A', A'*A` and explain differences.

3. Let `b = [4 3 2 1];` Try to compute `a-b, a'-b` and `a-b'` Explain the differences.

4. Complex numbers (vectors in the complex plane), e.g., $z = x + iy$ and its conjugate $z^* = x - iy$.

Let `a=1+i; b=2-2*i;` Also let `c = (2*a').';` and `d = conj(b/2);` Explain why `a = d` and `b = c`

5. Take the square root of the product of a and b. The ambiguity is intentional!

6. Scalars are special cases of vectors just as vectors are special cases of matrices. Scalars have just one element. Yet the right and left divide operations still work. Before entering these into Matlab, predict the results of computing `8/2, 2\8` and `8\2`. Hint, which number is in the denominator?

B. Scripts (m-files) and Data Display

If $y_n = f_n(x)$, find y_n from x for $n = 1, 2$. First, open an m-file using the editor, then type or copy and paste the following (being careful to note that some ASCII characters do not transfer accurately). Give your m-file a name by clicking SAVE AS.

```
%
clear all; close all;
x = linspace(0,1,11);
y1 = x.^2; y2=sin(2*pi*x/x(11)); plot(x,y1,x,y2)                    %(1)
str1=['Maximum value in y1 ' num2str(max(y1))]; disp(str1);
%%
z=max(y1); fprintf('Maximum value in y1 is %1.2f\n' ,z)             %(2)
str2 = ['Maximum value in y2 is ' num2str(max(y2)) ' volts.']; ...   %(3)
    disp(str2)
%
```

This code generates two functions `y1` and `y2` and…

1. Finds the maximum value of `y1`, converts the numerical value to a string for display using `num2str`, define string `str1`, and display the result using `disp`.

2. Displays the same value using the `fprintf` command. Notice the different representation: `1.00` from `%1.2f/n` syntax, where `%` is a control character, `1.2f` prints a FP number with one digit to the left and two to the right, and `/n` imposes a CR because `fprintf` does not supply one.

3. Repeat for `y2` using `disp`. What do you expect for the maximum value? Why is the computed value less than your expected number? Hint, look at the plot. Note that **...** continues the function on the next line, which in this case is completely unnecessary! Do you see why?

Summary:

- m-files are a great way to develop and store code.
- There are two (rather cumbersome) ways of outputting results to the display.
- The use of `%%` generates a segment that can be executed independently. It only works in a m-file, not in the command window.

C. Saving and Loading Data

```
%%                            %SECTION 1
clear all; close all;
a = [1 2 3 4 5];              %generate a row vector (1-D array)
dlmwrite('temp.txt',a,'\t'); %save a to disk as text file w/tab delimiters
whos                          %see what is in the workspace
%%                            %SECTION 2
b = load('temp.txt'); whos;  %read text file and convert to DPFP values
c = b.^2;                     %c is the square of each element of vector b
save('tempx.mat','b','c'); clear all;   %save b and c and clear workspace
load('tempx.mat'); b,c        %read the saved mat file and display b and c
%
```

In this m-file:

- The first section generates row vector a and writes the 1-D array into a txt file. Look for it in the **current folder** section. This type of write uses delimiters so the file can be read by other programs, like Excel. This particular delimiter is a tab as specified by `'/t'`. Use `','` to apply a comma delimiter.
- The second section loads the variables from a mat-file into an array. In this case, it is loading into memory data from an ASCII file into a double-precision floating-point array. DPFP data type allows mathematical operations on b to generate `c`. We then store those values as strings in a mat-file. You can choose to save the entire work space using `save('FILENAME.mat')`. Finally, we cleared the workspace and reloaded data arrays b and c.

D. Keyboard Input, FOR loops, and IF statements

```
%% SECTION 1
close all;
a=[1;2;3;4;5;6]; b=a'*a; %inner product
c=0;
for j=1:6   %inner product using for loop
    c=c+j^2;
end
b,c
%% SECTION 2
clear all; close all; clc
%Place an IF statement within a FOR loop.
for j=1:100     %display end of line text only when j=100
    if (j > 99)
        disp(['End of the line! ' num2str(j)]);
    end
end
%
%%  SECTION 3
%FOR loop within an IF statement: Request user input integer to calculate m!
m = input('Enter an integer: ');    %input value from keyboard.
fac = 1;                          %next line, m <= 21 to remain accurate
if (m >= 0 && m-floor(m) == 0 && m <= 21) %check three conditions
    for j=2:m           %FOR lo22op computes m! if input qualifies
        fac = fac*j;
    end
    disp(['The factorial of ' num2str(m) ' is ' num2str(fac)]);
    disp(['Matlab function gives ' num2str(factorial(m))]);
else
    disp(['Cannot compute the factorial of ' num2str(m)]);
end
```

The first section applies a simple for loop to compute an inner product by hand.

The second section looks at 100 numbers and tells you when it reaches the last number, i.e., 100. We needed to add an if statement.

The third section computes the factorial of a positive integer but only for outputs within the range of accuracy for double precision floating-point numbers (15 digits).

E. Filling a 2-D Array with Different 1-D Arrays

```
%
clear all; close all;
x = -10:0.01:10; X0=length(x); X2=ceil(X0/2); Y=zeros(X0);%parameters
                                  %Look for these variables in Workspace
for j=1:X0      % j is the address; x(j) is the value at that address
    m=x(j);     % slope for each row starts negative and increases
    Y(j,:)=m*x; % zero intercept on linear function
end
imagesc(Y); colormap gray; axis square %create grayscale image of result
figure; subplot(3,1,1);plot(Y(1,1:2000));     %first row or Y
subplot(3,1,2);plot(Y(X2,1:2000))            %second row of Y
subplot(3,1,3);plot(Y(X0,1:2000))            %third row of Y
%
% Explore via min(min(Y)) and Y(1001,1001)  Y(1,1)
```

How do we build a 2-D array from a 1-D array? This section applies a FOR loop to fill each row of matrix Y with a linear function `m*x` where slope $-10 \le m \le 10$ steps from negative to positive as we index through rows. Note that `Y(j,:)` addresses all elements in the jth row, since we fix the row first and then fill all columns in that row of the 2-D array. We also introduce the `subplot` for making arrays of plots. In this case, we make a column vector of three plots. The third element in the argument addresses each plot element. I used the second line of code to define X0 and X2 so, when those parameters change, only that single line changes.

```
% Also consider z=3.14 and apply ceil, floor, round, and fix.
```

F. Creating Custom Plotting Function

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Create a simple script that generates a one-cycle sine and cosine waves.  Use a
custom function called myplot to plot each with clear features.  Note that you should
place the myplot function in your Documents > Matlab directory or define a path.
%
t=1:1000;x=2*sin(2*pi*t/1000);myplot(t,x)
hold on; y=2*cos(2*pi*t/1000);myplot(t,y)
%
%
%In a separate m-file labeled myplot we have the following
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function myplot(x,y) %place this function in the same folder
%
plot(x,y,'linewidth',2) %use figure to avoid over-writing last plot
ax = gca; %current axes
ax.FontSize = 20;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Assignment:

1. Hill functions $f(X)$ model the action of activator X (input) as it engages the promotor of the gene within cellular DNA that ultimately produces protein Y (output). In its simplest form, the Hill function for activator $X \rightarrow Y$ is

$$f(X) = \frac{X^n}{K^n + X^n} = \frac{1}{1+\left(\frac{X}{K}\right)^{-n}} \quad . \qquad (1)$$

There are two parameters. K is the activation coefficient that describes the affinity between X and the promotor (among other factors). At K=X, we find Y=$Y_{\max}/2$. Parameter n is the Hill coefficient that describes the sensitivity of Y concentration to changes in X concentration, $\Delta Y/\Delta X$.

(a) Generate a plot of Y protein concentration as a function of scaled translational activator concentration input X/K for Hill coefficient values n = 0.25, 0.5, 1.0, 2.0, 4.0. Compare those 5 plots with a step function. You can compute a step function by selecting a sixth value of n appropriately.

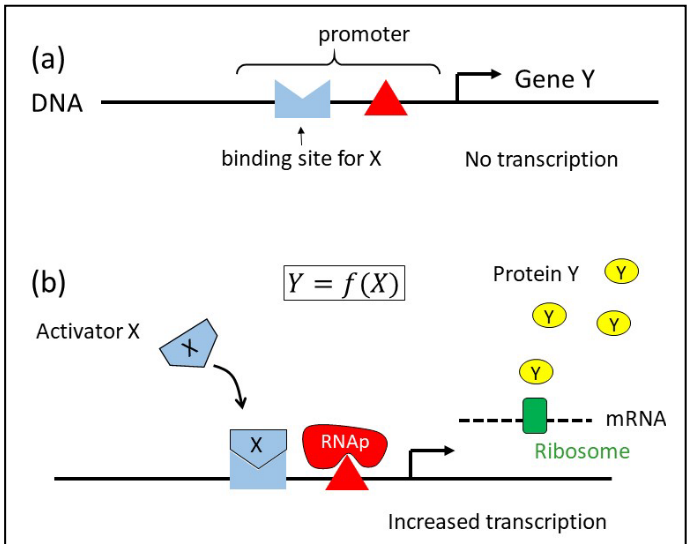

Figure 2. Illustration of transcriptional regulation of a gene. *Transcription* is the synthesis of mRNA from a DNA template whereas *translation* is the synthesis of protein from an mRNA template. Diagram (a) describes an inactive gene. Diagram (b) describes the actions of activator X to increase the affinity for RNA polymerase (RNAp) to the promotor site that then synthesizes messenger mRNA from the genetic code of gene Y. During translation, a ribosome works with mRNA to synthesize protein Y. The process is described by Y= f(X).

(b) Some promoters are naturally activated. In those cases, action is required to turn these genes off. The translational repressor $X \dashv Y$ is also modeled by the Hill equation where exponent $-n$ is changed to $+n$. Generate a plot of Y protein concentration as a function of scaled repressor concentration input X/K for Hill coefficient values n = 1.0, 2.0, 4.0. Compare those 3 plots with a step function.

(c) Write a lab report BRIEFLY describing the biology of transcriptional regulation. Also, tell me what the Hill equations describe. Are there other models? Include the Hill equations and two plots showing translational activation and translational repression curves. The activator plot should have 6 curves, each should be labeled and there needs to be a figure caption clearly explaining. The repressor plot should have 4 curves.

Hint: Start with n=1 in the activator equation and generate one plot. Then find a way to index through the entire range of n.