Lecture 2 MATLAB basics and Matrix Operations

Common tools: % and ;

- % indicates a comment, not a command to be executed MATLAB ignores comments. % can be placed at the end of an executable line to comment on that line

```
>> weight=input('give weight');     %getting weight
```

- a ; following an assignment will suppress display of the command result. ; signals the end of a single command, and can be used to separate multiple sequential statements on the same line.

```
>> density = 1.25;            diameter = 10.2;
```

Operators

+	Addition
-	Subtraction
*	Multiplication
/	Division
^	Power
'	Complex conjugate transpose (swaps columns and rows)
( )	Specify evaluation order

Order of Operations

( ) ' ^	highest level, first priority
* /	next priority level
+ -	last operations to be performed

```
>>  y = 2;      x = 3 * y^2
```
Ans: 12

```
>> y = 2;       x = (3*y)^2
```
Ans: 36

```
>> y = 2;     x = 3 * y + 2
```
Ans: 8
```
>> z = 3*6+6*2/4
```
Ans: 21
```
>> x = 5^2/2
```
Ans: 12.5

Order of Operations Example

Express the quadratic formula in MATLAB $x = \frac{-B + \sqrt{B^2 - 4AC}}{2A}$

Look at the following expressions. Identify which one will properly express the above equation. For the ones that don't work, specify all the reasons it won't work.

```
>> x = -B + sqrt B^2 - 4*A*C / 2A   % ???
```

```
>> x = -B + (B*B - 4*A*C)^(0.5)/2*A  % ???
```

```
>> x = -B + sqrt(B^2 - 4*A*C)/(2*A)   %???
```

```
>> x = (-B + sqrt(B^2 - 4*A*C))/(2*A)   %???
```

Matrix operations:

MATLAB is short for **MAT**rix **LAB**oratory, and is designed to be a tool for quick and easy manipulation of matrix forms of data. We've seen the matrix before in Lecture 1 as a 2-D array. That is, many pieces of information are stored under a single name. Different pieces of information are then retrieved by pointing to different parts of the matrix by row and column. Here we will learn some basic matrix operations: Adding and Subtracting, Transpose, Multiplication.

Adding matrices

Add two matrices together is just the addition of each of their respective elements.
If A and B are both matrices of the same dimensions (size), then

C = A + B

produces C, where the i^th row and j^th column are just the addition of the elements (numbers) in the i^th row and j^th column of A and B

Let's say that:

$$A = \begin{bmatrix} 1 & 3 & 5 \\ 7 & 9 & 11 \end{bmatrix}, \text{ and } B = \begin{bmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \end{bmatrix}$$

so that the addition is :

$$C = A + B = \begin{bmatrix} 3 & 7 & 11 \\ 15 & 19 & 23 \end{bmatrix}$$

The MATLAB commands to perform these matrix assignments and the addition are:

```
A = [1 3 5 ; 7 9 11]
B = [2 4 6 ; 8 10 12]
C = A + B
```

Rule: A, B, and C must all have the same dimensions

Transpose

Transposing a matrix means swapping rows and columns of a matrix. No matrix dimension restrictions
Some examples:

1-D $A = \begin{bmatrix} 5 & 2 & 9 \end{bmatrix}, \quad A^T = \begin{bmatrix} 5 \\ 2 \\ 9 \end{bmatrix}$ 1x3 becomes ==> 3x1

2-D $B = \begin{bmatrix} 8.1 & -4.5 & -7.6 \\ 3.2 & 3.1 & 3.9 \end{bmatrix}$, $B^T = \begin{bmatrix} 8.1 & 3.2 \\ -4.5 & 3.1 \\ -7.6 & 3.9 \end{bmatrix}$ 2x3 becomes ==> 3x2

In general

$$B(i, j) = B^T(j, i)$$

In MATLAB, The transpose is indicated by a single quote after the array

```
>> B = [5 3 6 2; 9 8 4 7]
```

$$B = \begin{bmatrix} 5 & 3 & 6 & 2 \\ 9 & 8 & 4 & 7 \end{bmatrix}$$

```
>> B'
```

$$ans = \begin{bmatrix} 5 & 9 \\ 3 & 8 \\ 6 & 4 \\ 2 & 7 \end{bmatrix}$$

Multiplication

Multiplication of matrices is not as simple as addition / subtraction. It is not an element by element multiploication as you might suspect it would be. Rather, matrix multiplication is the result of the dot products of rows in one matrix with columns of another. Consider:

C = A * B

matrix multiplcation gives the ith row and kth column spot in C as the scalar results of the dot product of the ith row in A with the kth column in B. In equation form this looks like:

$$C_{i,k} = \sum_{j=1}^{\text{\# or columns in A}} A_{i,j} * B_{j,k}$$

Let's break this down in a step-by-step example:

Step 1: Dot Product (a 1-row matrix times a 1-column matrix)
The Dot product is the <u>**scalar**</u> result of multiplying one <u>row</u> by one <u>column</u>

$$\underset{1x3}{\begin{bmatrix} 2 & 5 & 3 \end{bmatrix}} * \underset{3x1}{\begin{bmatrix} 6 \\ 8 \\ 7 \end{bmatrix}} = 2*6+5*8+3*7 = 73_{1x1}$$

DOT PRODUCT OF ROW AND COLUMN

Rule:

1) # of elements in the row and column must be the same
2) must be a row times a column, <u>not</u> a column times a row

Step 2: general matrix multiplication is taking a <u>series of dot products</u>
each row in pre-matrix by each column in post-matrix

$$\underset{2x3}{\begin{bmatrix} 1 & 4 & 2 \\ 9 & 3 & 7 \end{bmatrix}} * \underset{3x2}{\begin{bmatrix} 5 & 6 \\ 8 & 12 \\ 10 & 11 \end{bmatrix}} = \begin{bmatrix} 1*5+4*8+2*10 & 1+6*4*12+2*11 \\ 9*5+3*8+7*10 & 9*6+3*12+7*11 \end{bmatrix} = \underset{2x2}{\begin{bmatrix} 57 & 76 \\ 139 & 167 \end{bmatrix}}$$

C(i,k) is the result of the dot product of row i in A with column k in B

Matrix Multiplication Rules:

1) The # of columns in the pre-matrix must equal # of rows in post-matrix
inner matrix dimensions must agree
2) The result of the multiplication will have the outer dimensions
# rows in pre-matrix by # columns in post-matrix

For this example, apply rules

```
>> C = A * B
```

A is nra x nca (# rows in a by # columns in a)
B is nrb x ncb

Rule 1 says:

nca = nrb or else we can't multiply (can't take dot products with
different number of terms in row and column)

Rule 2 says:
C will be nra x ncb

$$\overbrace{nra \ x \ \underbrace{nca * nrb}_{\text{inner dimensions must agree}} \ x \ ncb}^{\text{result C has outer dimensions}}$$

How to perform matrix multiplication in MATLAB???
Easy

```
>> A = [4 5; 2 1];
>> B = [9 1; 6 12];
>> C = A*B
```

if inner matrix dimensions don't match, you'll get an error
example: Let's try to multiply a 2x3 by another 2x3 (rules say we can't do this)

```
>> A = [3 4 1 ; 0 4 9];
>> B = [2 9 5 ; 9 4 5];
>> C = A * B
```

```
MATLAB will tell you:

??? Error using ==> *
Inner matrix dimensions must agree.
```

Since the # of columns in A was not equal to # of rows in B, we can't multiply A * B

IMPORTANT:
Another example: Say we create a 1-D vector x with the following:

```
>> x = [2 4 9 5 2];
```

Now say we want to square each number in x. It would seem natral to do this:

```
>> y = x^2
```

But MATLAB tells us:

```
??? Error using ==> ^
Matrix must be square.
```

Note that y = x^2 is the same as saying y = x*x

MATLAB by default will always interpret any multiplication as a standard dot product type matrix multiplication, thus we can't take a dot product of two row vectors, since rules of matrix multiplication are violated in this case. If we just want to square the numbers in x, we can do this:

```
>> y = x.^2
```

The period after the vector x tells MATLAB <u>DO NOT</u> follow multiplication rules, just operate on the individual elements within the matrix. So y = x . ^2 is NOT the same as y = x ^2 to MATLAB

Practice matrix operations on the following examples.
List the size of the resulting matrix first. then perform the operations by hands. Use MATLAB to confirm each of your answers.

$$\begin{bmatrix} 9 & 12 & 5 \end{bmatrix} * \begin{bmatrix} 2 \\ 8 \\ 3 \end{bmatrix} = \begin{bmatrix} \quad & \quad & \quad \\ & & \\ & & \end{bmatrix}$$

$$\begin{bmatrix} 9 & 7 \\ 3 & 6 \\ 7 & 1 \end{bmatrix} \begin{bmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \end{bmatrix} = \begin{bmatrix} \quad & \quad & \quad \\ & & \\ & & \end{bmatrix}$$

$$\begin{bmatrix} 4 & 3 \\ 2 & 1 \end{bmatrix} * \begin{bmatrix} 8 & 7 \\ 9 & 6 \end{bmatrix} = \begin{bmatrix} \quad & \quad \\ & \end{bmatrix}$$

$$\begin{bmatrix} 4 \\ 9 \\ 7 \end{bmatrix} * \begin{bmatrix} 2 & 1 & 8 \end{bmatrix} = \begin{bmatrix} \quad & \quad & \quad \\ & & \\ & & \end{bmatrix}$$

$$\begin{bmatrix} 1 & 4 & 9 \\ 8 & 6 & 4 \end{bmatrix} * \begin{bmatrix} 3 & 0 \\ 9 & 3 \\ 4 & 7 \end{bmatrix} = \begin{bmatrix} \quad & \quad \\ & \end{bmatrix}$$

$$\begin{bmatrix} 4 & 9 & 8 \end{bmatrix}^T * \begin{bmatrix} 2 & 7 & 1 \end{bmatrix} = \begin{bmatrix} \quad & \quad \\ & \end{bmatrix}$$

$$\left(\begin{bmatrix} 4 & 7 \\ 8 & 9 \end{bmatrix} + \begin{bmatrix} 5 & 3 \\ 0 & 1 \end{bmatrix} \right)^T * \begin{bmatrix} 10 & 2 & 7 \\ 6 & 3 & 5 \end{bmatrix} = \begin{bmatrix} \quad & \quad \\ & \end{bmatrix}$$

Simple Help

There are two good ways to get help without any books in front of you. Within the MATLAB environment (the MATLAB window) there is a help button that takes you to a set of manuals. There is also a 'help' command you can issue.

- 1) Online manuals / tutorials through help menu (HTML / PDF). These include everything from 'getting started' tutorials to basic user's guide manuals to detailed manuals on advanced features like designing Graphical User Interfaces (GUIs)
- 2) 'help' command in MATLAB environment. This is an instant reference and explanation of basic commands

try these in MALAB

```
>> help
```
this command breaks down the potential topics into sub-groups

```
>> help 'name of sub group'
```
this lists all the various commands that fall into that sub-group

```
>> help 'command'
```
provides details on a specific command including usage. Usually includes cross reference to related commands and examples.

E.G.
```
>> help plot
>> help sin
>> help sqrt
>> help mean
```

Fundamental Program Structure

Labeling the program using comments
- program title
- student information
- program summary

executable statements
- program input (load command, assignment statements, etc.)
- perform operations needed (sequential execution, loops, etc.)
- display program output (graphs, numbers, output files, etc.)
- intersperse comments to explain program

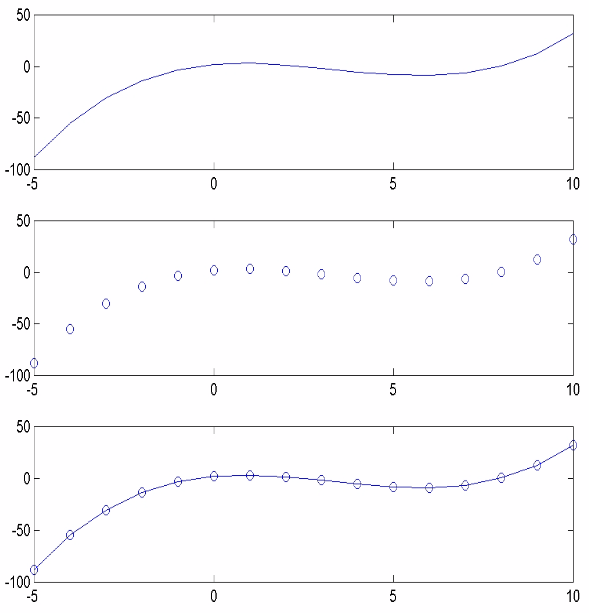
```
>> % Example program #1
>> % K. Gurley, CGN 3421, 1/11/99
>> % This program does very little
>> %
>> %   INPUT SECTION
>> %   enter x vector, then calc. y and z
>> x=[1, 2, 4, 8];   %input constants
>> %
>> %   PERFORM OPERATIONS NEEDED
>> %create a scalar y as a function of x(2) and x(3)
>> y= (x(2) + x(3))^2   %leave off ; to show result
>> z= x.^2 + y;          %create z vector from x and y
>> %
>> %   DISPLAY OUTPUT GRAPH
>> plot(x,z)
>> %end of program
```

Creating data / example algorithm

vectors filled with data can be created using similar notation as that in the FOR - END loop structure.

e.g. I want to plot the function $y = 2 + 3x - 2x^2 + 0.2x^3$ over the range x = [-5, 10]

```
>> %Example program #2
>> %K. Gurley, CGN 3421, 1/13/99
>> %program to plot a function over a range
>> %
>> %create a vector of x values
>> %start at -5, and go to 10 in steps of .2
>> x = -5:.2:10
>> %now evaluate the equation for each x
>> NDP = length(x);      % length tells me how many pieces of data
>> %                    % are in the vector x
>> for i=1:NDP
>>      y(i) = 2 + 3*x(i) - 2*x(i)^2 + .2*x(i)^3;
>> end
>> %now plot the results as x vs. y
>> subplot(311)
>> plot(x,y)
>> %plot again using dots instead of lines
>> help plot
>> subplot(312)
>> plot(x,y,'o')
>> %plot using both lines and dots
>> subplot(313)
>> plot(x,y,'o-')
```



Vector operations (do we have to use the for - end loop?)

In the previous example, y was created using a **for - end** loop to create each y value individually.
A nice MATLAB feature is called '**vector operations**', which allows a single command to operate upon every element within an array.

e.g. create a vector x, and square each element

```
>> x = 1:5                         %creates x=[1 2 3 4 5]
>> %option #1, square each element individually
>> for i=1:5;
>>     x(i) = x(i)^2;
>> end
>> x                               %shows result
>> %option #2, use vector operation all at once
>> x = 1:5                         %reset x=[1 2 3 4 5]
>> x = x^2
>> %produces an error, trying to square the
>> %entire vector x=[1 2 3 4 5] *  [1 2 3 4 5]
>> %which we'll learn later is not valid
>> %try this instead
>> x = x.^2;
>> %the . means to operate on each element
>> %individually, not on the vector as a whole
>> x                               %shows result
```

Vector operations continued:

Now let's apply vector operation to the function we used earlier in this lecture

$y = 2 + 3x - 2x^2 + 0.2x^3$

here is the previous code less the comments

```
>> x = -5:.2:10
>> NDP = length(x);                %tells me how long x is
>> for i=1:NDP
>>     y(i)=2 + 3*x(i) - 2*x(i)^2 + .2*x(i)^3;
>> end
>> plot(x,y,x,y,'o')
```

Let's replace the blue code with a vector operation
The old code is replaced with the red code below

```
>> x = -5:.2:10
>> y = 2 + 3*x - 2*x.^2 + .2*x.^3;
>> plot(x,y,x,y,'o')
```